Task 2 Report

Data Storage Paradigms, IV1351

Jonathan Nilsson Cullgert jonnc@kth.se

23 November 2023

1 Introduction

In this task the conceptual model is meant to be transformed in to a logical/physical model so that a database can be created. That is the new model that is created will be logical with enough physical aspects enabling it to be turned into a database. Some changes were made to the conceptual model so that it will work when it is turned into a database but generally changes are avoided if they were not necessary. All the information provided by Soundgood Music school is supposed to be find in the new model as well as in the database. The model must be created in crow foot notation, we used IE notation.

I have worked on this task with Simon Malmqvist Dichy and Andrej Miladinovic.

2 Literature Study

I watched the lecture on normalisation and the lecture on the logical and physical model. I also read the tips and tricks pdf for task 2.

I have learned how to convert a conceptual into a logical model as well as to normalise the model. I have also learned to how to create an actual database from the model as well as how to import data into that database.

3 Method

The conceptual model was transformed into a logical model by following an eleven step procedure. All entities and attributes with cardinality higher than one were turned into entities in the new model. All other attributes with cardinality of 1 were left in their entities. A naming scheme was also implemented so that all names are in lower case letter and every space is indicated by having an underscore. All attributes were given a type like varchar, date, etc. No attribute was given the char attribute, only varchar. Some attributes were given new cardinalities since they had the wrong ones in the conceptual model.

The next step was to identify the super entities and give them primary keys. After the primary keys were assigned to the super entities the one-to-one relations and one-to-many relations were handled. When handling these relations it is important to evaluate if an entity is important without the super entity and in that case create a new primary key for that entity, otherwise the primare key from the other entity is kept.

The many-to-many relation was handled by creating cross-reference tables in which the primary key in that table is the foreign key from the two entities being connected. If this table represents an entity which should be used attributes should be added but in this task no such cross-reference tables were made.

The final part was to create a database from the model. In Astah which the model was created in, it is possible to export a SQL script which creates a database of the model. The model in Astah was made using the IE crow foot notation. When the database was created all the tables were filled with data, the data was generated using a website.

4 Result

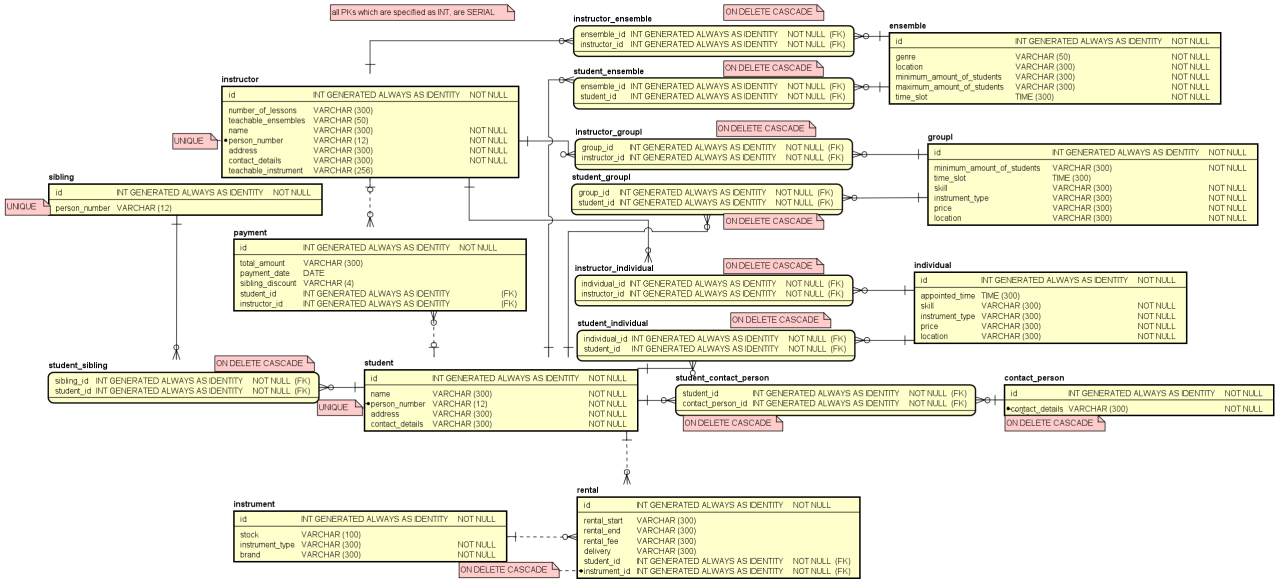

Figure 1: The logical model created from the conceptual model in task 1

The git repository in which the sql code for creating the database and filling the tables is located: https://gits-15.sys.kth.se/jonnc/DatabaseIV1351

In this model some attributes were made into entities since their cardinality were higher then one, they were after some time remade into attributes since their cardinality were wrongly interpreted in the conceptual model. An example of this is the stock attribute in instrument, it was interpreted as a having multiple values since the stock could vary (1,5,2,7) etc. It was then realized that the cardinality will be one since the value of stock can be changed but it will always hold one value.

The reason that some attributes like personal_number do not have the type char instead of varchar is because of uncertainty. It is not clearly provided in the instructions from the customer if the personal number is supposed 12 or 10 characters, therefore it is varchar(12), it can then be both 10 and 12. It should obviously be changed if clear instructions are given from the customer. The same could be said for many attributes that are varchar(300). The length is undefined from the customer and therefore it is just given some arbitrary length.

The super identities in this model is student and instructor since they can exist without other entities, they are not reliant on other entities for themselves to exist.

All the notes that are "ON DELETE CASCADE" is data that can be removed without it interfering with the database. As an example if an individual lesson is removed you can remove that lesson from a student and an instructor without it creating any problems, since that lesson has not yet happened an no information will be needed from it.

4.1 Complementation

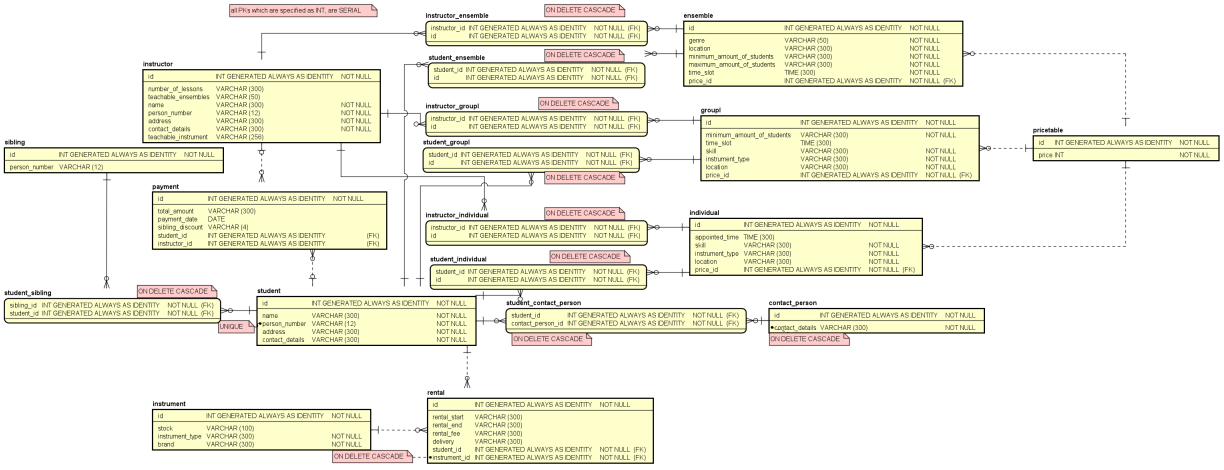

Figure 2: The fixed logical model

5 Discussion

The naming convention is correctly followed, all the names are in lowercase and every space is marked by an underscore. All the cross-reference tables are named by the primary keys of the two entities that connect to the table. All the foreign keys are also named correctly from the entity they come from.

The crow foot notation (IE notation) is correctly followed and comments are also added correctly. Unique and not null values are marked correctly, some data types might not be correct but that is due to limited information from the customer.

All the relevant tables exist in this model and with these tables it is possible to extract all the data that is required from the soundgood music school and no information that is necessary is lost.

The primary keys in the tables are well chosen since they are either in super entities or the information that they contain is necessary to keep even if they are being used. As an example in the different types of lessons, they contain super keys that are unique. If the student is deleted from the database the lesson will still exist which means that it will be possible to still pay a teacher, even after the student quits. This also allows data about certain properties to be collected even if some data relating to that table is deleted. Like payments, if a student is deleted you can still collect how many lessons has been taught by a teacher. This is true for all the super keys in the model.

In the project description certain operations are described that needs to be able to be done. It is possible to see when instructors are available since they are connected to a certain amount of lessons, all the time during the day when those lessons are not happening, the teacher is available to instruct. It is possible to see if a student has a sibling and, which student is their sibling since the personal number is connected to the sibling and that same personal number is connected to name, contact detail, address etc. Since every lesson has a unique id it is possible to change the price for every lesson, so if the price increases for a certain type of lesson, all the lessons of that type can have an increased price after the price change. For renting it is possible to if an instrument is in stock, its personal id and its type. This means that every instrument will be able to be rented if they are in stock and it is possible to see which student that has rented which instrument and that way it is possible to see what the rent cost will be.

All the datatypes that are not unclear by the customer are also correct. This means for example that skill level will be, beginner, intermediate or advanced whilst time slot will be stored as a time of day.

5.1 Complementation

In the first version of this task the price for all lessons were stored in their separate tables, meaning that ensemble, group lessons and individual lessons had price as an attribute. This causes problems since a lessons could be booked by multiple students and that would duplicate the price data. It also causes the problem that new prices can not be added until a lesson with that price has been taught.

To solve this issue the price was moved to its own table. In this table the price is an

attribute and it has a primary key which is an attribute in the different lesson entities. This means that the price will not be duplicated since it will be stored separately and identified with a specific id. This also means that if a new cost of a lesson is being introduced that price can be added to the new price entity and be given a specific id without it needing to be connected to a lesson.

6 Comments About the Course

I spent roughly 14-16 hours on this task.