# Task 2 Logical and Physical Model

**IV1351 Data Storage Paradigms**

Klara Lindemalm, klindema@kth.se

Fall 2022

## Introduction

For this part of the project the task was to implement a logical model with physical attributes with the conceptual model as a guideline that was built in the previous task for the Sound Good Music School. Additionally, the task was to set up the actual database with created insert data to fit the model. The collaborator for this project was Rasmuss Per Brodin.

## Literature Study

For the concept of normalisation the live lecture was attended in KTH Kista held by Paris Carbone and additional reading through chapter 14 and 15.1 to 15.2 in the main book (*Fundamentals of Database Systems (7th Edition recommended, 6th edition is also fine) by Ramez Elmasri and Shamkant B. Navathe)* was read through for deeper understanding. Additionally we glimpsed though chapters 7 to 7.5 in the additional course book *(Database System Concepts (7th Edition) by Silberschatz Abraham, Henry Korth, Sudarshan S).*

Followed up by video lectures on logical and physical model divided into three parts providing a deep introduction to the log-phys model that is designed in this task of the project. Additional reading through chapter 9.1 in the main book was done along with a glance through section 7.9 and 7.10 in the additional book to further understand the model. For more help along the way with the actual implantation process we attended the tutorial occasions on site receiving hand on help from the teachers and instructors. In order to understand inheritance in this model we read through section 9.2.1 more specifically page 298 in the main book.

For the set up of the database we received help from the final tutorial session to export the model created, to a database in SQL language. The project description section of the task on the canvas page also provided tips on how to create the insert data from online generator tool https://generatedata.com/ along with the file *tips-and-tricks-task3.pdf.*

## Method

For the procedure to set up the data base, the initial part of the task was to set up the logical model with physical attributes. The software used for the initial part of the task was the diagram editor Astah Professionals and a ER diagram with IE notation and a logical model type, was created.

For the first part of the model, a table was created for each entity from the conceptual model. In these tables we added all attributed with a cardinality of at most one value, as columns. A new table for each entity with a grater cardinality than one was inserted as it's own table in the model. Type domain was specified for all columns such as varchar, int and int always generated as identity. The last mentioned has to be generated as it's not a default type in the software Astah, but a standard in SQL. Column constrains where considered, initially only if the columns where unique and/or not null. Not null was added to each column from the editor directly but the uniqueness had to be mentioned with a note if not a primary key such as social security number.

The table was evaluated to determine what entities where strong, meaning not dependant on any other entities and can stand on their own without relations. Identifiable columns where added as primary attributes. Additionally the relations where listed between entities one-to-one or one-to-many similar to the conceptual model, relations are also defined as identifiable or non identifiable. When listing the relation the primary key in the strong relation is set to a foreign key in the weak end. This key is either identifiable (set to primary key in the weak end) or non-identifiable if a surrogate key is better suited on it's own.

Cross reference tables where added to the entities with a many-to-many relationship in the conceptual model. The primary key for these relations is either a combination of foreign keys listed as primary keys with a identifiable relationship or a surrogate key. Column constraints where further evaluated to determine what would happened if certain operations where preformed such as delete. These where added as notes in the model on delete cascade, on delete set null or on delete no action.

Lastly for a review of the logical model with physical attributes we evaluated if the model was normalised and that all planed operations could be preformed. By normalised we intend to reduce data redundancy and find the best fit place for all data to be stored in the database, for easy access and appropriate placing.

To create the database and convert the logical model to SQL code we let Astah write the SQL code for the database, converting the logical model with physical attributes to SQL code of the actual database. This was done with the following steps from the Astah Software.

1. Tools

2. ER-diagram
3. Export sql...

To write the insert data we used the online generating tool and inserted the tables with attributes to convert to SQL code to insert data in the database. All code was tested using PostgresSQL and Docker Desktop managed from the terminal window. In order to set up the model we used the commands received from lab one to test that all data was available and modifiable.

For the higher grade part we added an inheritance relationship to the entities that could inherit attributes from parent entity. Insert data was altered and converted accordingly to fit the model with inheritance.

## Result

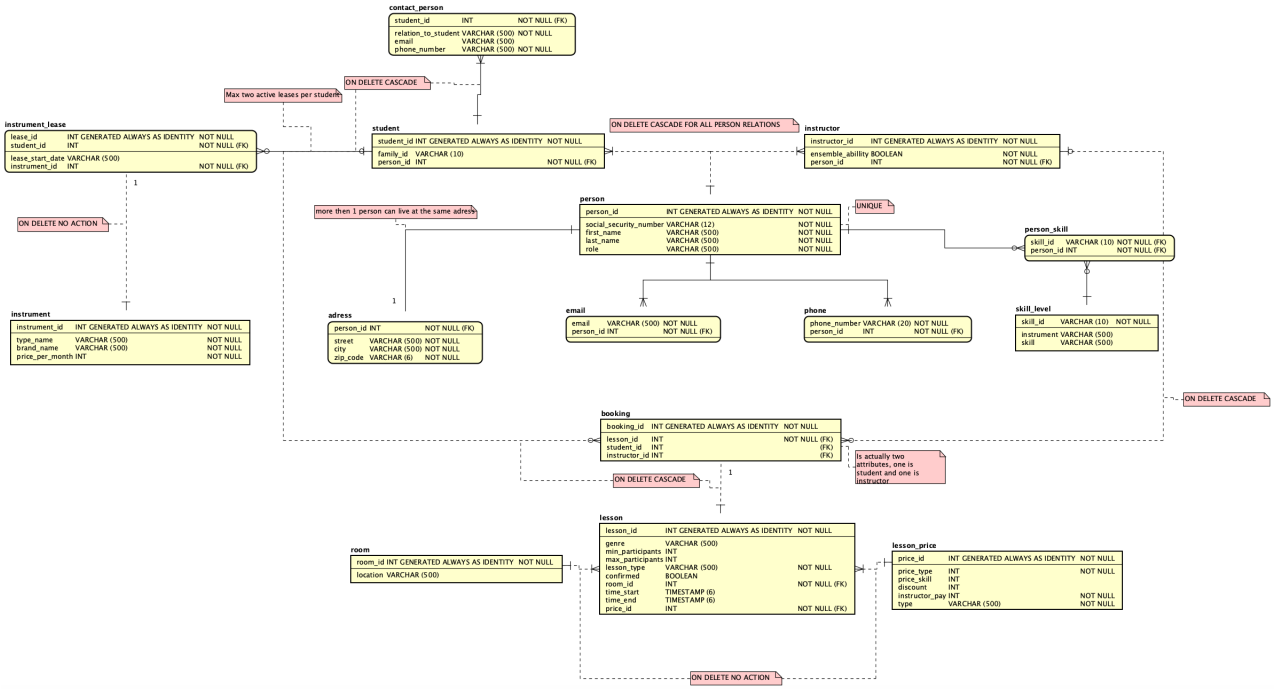


Figure 1: Logical model of the Sound Good Music School

The model is very similar to the conceptual model apart from email, phone and skill_level set to their own table. We also applied a cross reference table to skill_level being the person_skill to cross reference students and instructors with the same skill set for different instruments. Most commonly used typed for the columns in this model are varchar, int and int always generated as identity. Link to and insert data and database sql can be found here, both for the model with and without inheritance and the corresponding insert data: *Click this*
If not possible to open the link type:
https://github.com/KlaraLindemalm/Soundgood-Music-School-Task-2/tree/main

## Discussion

When implementing the model we stumbled upon a hand full of obstacles that are explained below, however when these where evaluated and some what solved we found that we had a model that met all the criteria of the project description of the Soundgood Music school. All information is available and can be accessed with SQL commands. There are however a few things things that where altered during the process that we did not find a solution that we where completely satisfied with, these are also explained.

One of these things that we tried to implement was the usage of enum types, witch can have a set of named values. Our idea was to have the skill levels set to an enum type, one for beginner, intermediate and advanced. However we did not manage to implement these types and instead inserted the value in text. This could of course cause trouble when typed incorrectly. However this could be solved by the administrators or when the the application is designed.

Redundant data exists, as the idea for the initial model was to only use person_id as the identifiable attribute for instructor and student, with a role column identifying if the registered person is a student or an instructor. However Astah did not manage to represent the two person_id's as different columns when related to lesson and only represented one column for person_id. Therefore we decided to add personid as a primary key for the student table and instructor_id for the instructor table with the person_id inherited with a non-identifiable relationship. From that solution provided the role column provided redundant data as the same information as the information is available if the student is assigned a student or an instructor id.

Furthermore when taking about redundant data the usage of cross reference table could have been used for both the phone and email. However as we designed the model we decided that duplicates of this sort wasn't going to cause that much storage waste. We reasoned that most students would not have the same email address or phone-number as another student and if so there would only be a few duplicate values in the table and not enough to create a cross reference table to relate two students to the same. The same reasoning was applied to the cross reference table available in the model person_skill table where it is very likely that several students have the same skill set as another. Therefore redundant values of this sort are avoided with the usage of the cross reference table.

### Inheritance

Just like the conceptual model, inheritance was implemented from person to both instructor and student. The same relation without inheritance between the same entities was non identifying where all entities has their own primary key. When creating the database and altering insert data for the instructor

and student the type for the primary key INT GENERERATED ALWAYS AS IDENTITY gave us problems. No primary key was generated when creating an instructor or student so we had to first create a person and then assigning the corresponding person_id to be either a student or instructor with identifiable attributes. However this generated duplicate values in the person table when the student and instructor was added to the respective table and the primary key was no longer unique in the person table. Therefore the type was set to INT which is definitely not ideal as now the task to maintain unique primary keys for all persons is placed on the user. However with this implementation the inheritance relationship could be fulfilled and student and instructors could be created directly with attributes inherited from person.

A drawback of this solution with inheritance is that no instructor would be able to take classes as each person has a role. Adding a duplicate row for a student registration is neither possible as the social_security_number has a unique constraint meaning that no other person can have the same person number.

The inheritance relationship helps maintain a normalized database with less redundant data as the inherited class only maps to the parental class without duplicating values in different tables. However with the usage of inheritance the search for information is the database becomes less efficient as it has to relate back to the parent table when fetching the information, making complexity increase and speed reduce compared to the implementation without inheritance.