Project Report

Data Storage Paradigms, IV1351

02.Jan.2025

Project members:
[Yuhui Gan, yuhuig@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

In Task 2, I transitioned the conceptual model created in Task 1 into a logical and physical model for the Soundgood Music School. This step bridges the gap between high-level design and the actual implementation of a relational database. The objective was to design a database that aligns with the school's operational requirements and supports efficient data storage and retrieval. This involved refining the conceptual model into a logical schema, normalizing it to the third normal form (3NF) to eliminate redundancy, and incorporating necessary physical aspects for database implementation.

I used standard conventions, such as naming surrogate primary keys using the format `<table_name>_id` for clarity, and consistency and to avoid duplicated meaning.

Throughout the task, I worked independently, and I believe the resulting database model provides a solid foundation for implementing the Soundgood Music School's data management system and supports future tasks in the project.

2 Literature Study

To prepare for Task 2, I watched lectures by Leif Lindbäck which provided guidance in creating a logical and physical model from the already existing conceptual model. To complete the task, I particularly looked into Normalization Principles: First Normal Form (1NF) requires that all columns in a table have atomic values—meaning each column contains only one value per row.

In Second Normal Form (2NF) all non-key attributes must depend entirely on the primary key. This eliminates partial dependencies, where a non-key attribute is related only to part of a composite primary key.

The third Normal Form (3NF) eliminates transitive dependencies, where non-key attributes depend on other non-key attributes instead of the primary key.

I learned that ALLOW, PROHIBIT, and CASCADE are terms often used to describe actions defined for foreign key constraints when changes are made to the referenced table. With delete (CASCADE), changes to a parent record automatically propagate to related child records. And ALLOW (SET NULL) sets the foreign key in the child table to NULL when the referenced parent record is deleted or updated. Prohibit (NO ACTION) prohibits any changes to a parent record if there are related child records in the child table. The Tips and Tricks for Task 2 document provided additional insights into practical considerations, such as naming conventions for surrogate primary keys, avoiding derived attributes, and ensuring all necessary operations described in the Soundgood Music School's business requirements could be performed. The emphasis was on creating a database that stores all relevant data, avoiding reliance on external application logic, and maintaining scalability and data integrity.

3 Method

To create the logical and physical model for Task 2, I started by translating the conceptual model from Task 1 into tables. The diagram editor being used in task 2 is Astah Professional. Each entity became a table, with weak entities including foreign keys referencing their parent tables. I added columns for attributes, assigning appropriate data types based on their nature, VARCHAR was used for most of the attributes and INTEGER for IDs so they can be SERIAL. For multi-valued attributes, I created separate tables or used junction tables to handle many-to-many relationships.

I assigned surrogate keys as primary keys for strong entities to ensure clarity and consistency. For weak entities and junction tables, I used composite or surrogate keys as needed. Relationships between tables were represented using foreign keys with clear naming conventions, like `<table_name>_id`.

Next, I normalized the model to 3NF to remove redundancy and ensure data integrity. I verified that attributes depended only on primary keys and eliminated partial and transitive dependencies. I also defined constraints, including NOT NULL for required fields and UNIQUE for identifiers, and specified actions for foreign keys, such as ON DELETE CASCADE where appropriate.

4 Result

Due to new knowledge gained from watching the recommended videos, the tables have been heavily modified compared to how they looked in Task 1. Entities have been removed or created to ensure the database adheres to 3NF.

To achieve this, and inspired by the tables `person_email` and `person_phone` discussed in Leif's lectures, new tables such as `preference` and `schedule` have been introduced. In the model, a student queries the school to check if the `available_spot` field is an integer greater than zero. If so, the student's personal data is recorded in the `person` table, along with staff data, since anyone can potentially become a student.

The student then checks the `schedule` table to find a course they are interested in and makes booking request(s). While a student can make multiple requests, only one course is booked at a time to maintain the database in 3NF. The instructor reviews their availability, and the `administrative_staff` verify if the individual is a valid student and whether the instructor has time before deciding whether to enroll the student.

All courses are grouped, with the school responsible for setting the schedule for group classes. Additionally, the school could provide a list of available time slots for instructors, allowing students to choose suitable slots. Most new tables have been designed to avoid multiple values in a single row. For example, siblings are stored in separate rows but share the same `student_id` as the primary key. The enrollment table is then used to determine if any discounts apply.

Many tables, such as `invoice` and `salary`, are now determined by relationships between multiple tables. The database structure is clearer than in Task 1, thanks to the proper use of primary keys and the accurate application of crow's foot notation introduced in the recommended videos.

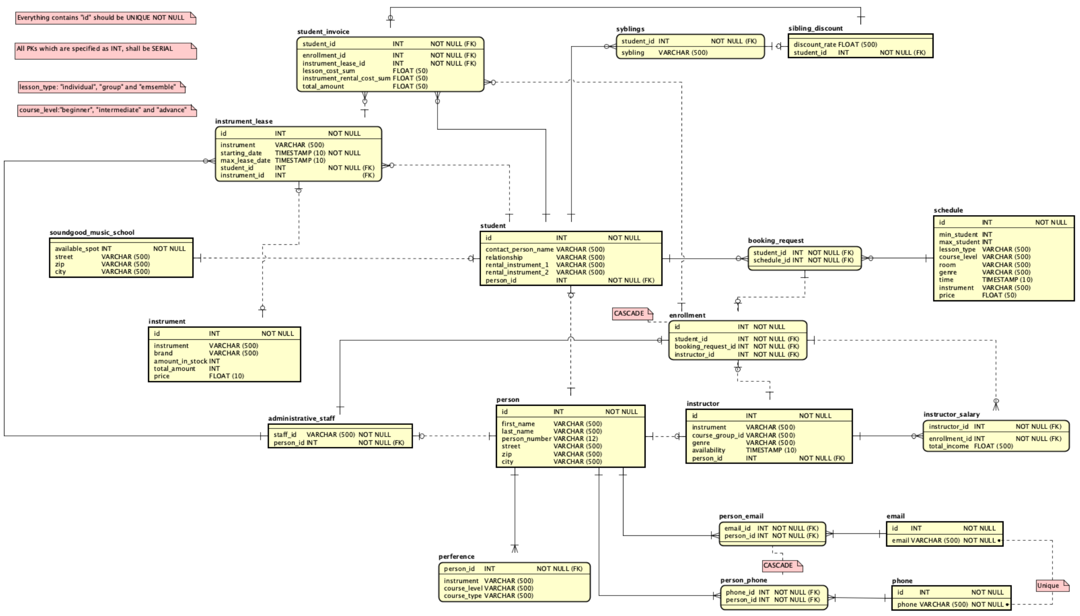
Everything contains "id" should be UNIQUE NOT NULL
All PKs which are specified as INT, shall be SERIAL
lesson_type: "individual", "group" and "emsemble"
course_level:"beginner", "intermediate" and "advance"
student_invoice
student_id INT NOT NULL (FK)
enrollment_id INT NOT NULL (FK)
instrument_lease_id INT NOT NULL (FK)
lesson_cost_sum FLOAT (50)
instrument_rental_cost_sum FLOAT (50)
total_amount FLOAT (50)
syblings
student_id INT NOT NULL (FK)
sybling VARCHAR (500)
sibling_discount
discount_rate FLOAT (500)
student_id INT NOT NULL (FK)
instrument_lease
id INT NOT NULL
instrument VARCHAR (500)
starting_date TIMESTAMP (10) NOT NULL
max_lease_date TIMESTAMP (10)
student_id INT NOT NULL (FK)
instrument_id INT (FK)
soundgood_music_school
available_spot INT NOT NULL
street VARCHAR (500)
zip VARCHAR (500)
city VARCHAR (500)
student
id INT NOT NULL
contact_person_name VARCHAR (500)
relationship VARCHAR (500)
rental_instrument_1 VARCHAR (500)
rental_instrument_2 VARCHAR (500)
person_id INT NOT NULL (FK)
booking_request
student_id INT NOT NULL (FK)
schedule_id INT NOT NULL (FK)
schedule
id INT NOT NULL
min_student INT
max_student INT
lesson_type VARCHAR (500)
course_level VARCHAR (500)
room VARCHAR (500)
genre VARCHAR (500)
time TIMESTAMP (10)
instrument VARCHAR (500)
price FLOAT (50)
CASCADE
enrollment
id INT NOT NULL
student_id INT NOT NULL (FK)
booking_request_id INT NOT NULL (FK)
instructor_id INT NOT NULL (FK)
instrument
id INT NOT NULL
instrument VARCHAR (500)
brand VARCHAR (500)
amount_in_stock INT
total_amount INT
price FLOAT (10)
administrative_staff
staff_id VARCHAR (500) NOT NULL
person_id INT NOT NULL (FK)
person
id INT NOT NULL
first_name VARCHAR (500)
last_name VARCHAR (500)
person_number VARCHAR (12)
street VARCHAR (500)
zip VARCHAR (500)
city VARCHAR (500)
instructor
id INT NOT NULL
instrument VARCHAR (500)
course_group_id VARCHAR (500)
genre VARCHAR (500)
availability TIMESTAMP (10)
person_id INT NOT NULL (FK)
instructor_salary
instructor_id INT NOT NULL (FK)
enrollment_id INT NOT NULL (FK)
total_income FLOAT (500)
perference
person_id INT NOT NULL (FK)
instrument VARCHAR (500)
course_level VARCHAR (500)
course_type VARCHAR (500)
person_email
email_id INT NOT NULL (FK)
person_id INT NOT NULL (FK)
email
id INT NOT NULL
email VARCHAR (500) NOT NULL
CASCADE
person_phone
phone_id INT NOT NULL (FK)
person_id INT NOT NULL (FK)
phone
id INT NOT NULL
phone VARCHAR (500) NOT NULL
Unique

Figure 1: Updated ER Diagram in 3NF

5 Discussion

I believe the logical and physical model I designed effectively meets the requirements of the Soundgood Music School. The model adheres to the third normal form (3NF), since I actively looked into is there are multiple values under attributes, if there are any non-key attributes depending on composite primary key, and no non-key attribute depended on another non-key attribute, except for the zip and city, that would be over normalization. My attributes and tables are logical enough and named after their key function. Primary and foreign keys are named consistent and descriptive, except I am unsure about if all strong attributes or equivalents' primary key should be "id" instead of the surrogate primary keys as when it is a foreign key referring back to the original table.

Constraints, including NOT NULL for required fields and UNIQUE for identifiers were included in my diagram so that data is more reliable. The model also addresses complex requirements, such as handling price changes by storing historical prices with timestamps instead of overwriting them. This design ensures accurate billing and instructor payments based on the data stored at the time students were enrolled in the lesson. Multi-valued attributes are appropriately managed through separate tables to support many-to-many relationships which surprised me as there's a crow's foot relation labeled "many to many", but I supposed that it is because the database needed to be in 3NF.