Project Report

Data Storage Paradigms, IV1351

2024/11

Group 77 Project members:
[Sam Serbouti, serbouti@kth.se]
[Danni Noren, dannin@kth.se]

1 Introduction

This report presents a conceptual model for the Soundgood music school database using ER diagram in Astah Professional. Soundgood is a music school where they offer lesson based services and the students pay for each lesson they receive and the instructors are paid for the lessons they provide to the students. The conceptual model should respect the following requirements:

- **Lessons:** Soundgood offers individual and group lessons.The group lesson and ensembles unlike individual lessons have a fixed schedule. Moreover they require a minimum applicants and has specific number of spots available. Ensembles have a genre and different instruments can participate in the same lesson. Group and individual lessons are provided for various instruments and different skill levels.
- **Student:** Student's personal information(personal number, name, address and contact details) is to be presented along with a contact information of a contact person. In addition wether if the student has a sibling/siblings that take lesson in the same school as them, discount is offered for siblings who take lesson at the same time.
- **Instructor:** The personal information of the instructors is to be presented. Instructors have group lesson and ensembles scheduled and can teach individual lessons depending on their availability. Each instructor can teach a specific type of instruments and the ability to teach specific ensembles may vary depending on the instructor.

- **Instrument rentals:** The students have the opportunity to rent up to two instruments from a wide selection of instruments and various brands from the school for up to 12 months.
- **Payment system:** Different type of lessons have different prices and the prices for advanced level is different from the beginner and intermediate level. The discount for siblings in to be taken into account for student payments. Moreover the pricing scheme may vary over time.In addition students are also charged for the rented instruments monthly. Both Students and instructors are billed monthly for the lessons they have participated in during the previous month.

2 Literature Study

To complete this task, we have read chapters 1, 2 and 3 of the 7th edition of *Fundamentals of Database Systems* by Ramez Elmasri and Shamkant Navathe. They respectively dealt with:

1.
   - the properties of DBMSs (they are self-describing, insulate programs with data, abstract data, support multiviews and process multiuser transactions) and their advantages to file-processing
   - the basic steps of designing a database and the people using DBMSs and the people creating and mainting them
   - the history of databases, and how they come from file processing
2.
   - what data models are
   - the three schema architecture
   - data languages and interfaces
   - the database system environment (notably the component modules), its utilities
   - centralized DBMS architecture and two/three/n-tier client/server architecture
3.
   - tmportance of high-level conceptual data models
   - Entities, attributes, relationships and weak entities in ER diagramming
   - UML class diagrams and comparison to ER model

Understanding these concepts allowed us to have a big picture on DBMSs. We could have a limited but general understanding of what would come after this conceptual design stage and this was crucial since it was our first time designing a database.

Chapter 4 of *A First Course in Object-Oriented Development, A Hands-On Approach* by Leif Lindbäck and his six tutorials on UML, domain model, IE notation and conceptual models gave us technical insight on *how* to build an ER diagram in Astah professional. Most importantly, his notes on common mistakes gave us bounds when we were designing the CM.

3 Method

To design the database required by the Soundgood school we used the entity–relationship (ER) model with IE notation, created using Astah Professional diagram editor.

To create the conceptual model the steps below were followed:

- **Noun identification:** The description of the Soundgood school was read thoroughly and the nouns for the requirements were extracted from the text to create entities.
- **Category list:** Used the category list provided in Leif Lindbäck's book [1] to discover more entities for the conceptual model.
- **Removing unnecessary entities:** In this step the redundant and unnecessary entities were removed.
- **Finding attributes:** essential properties for each entity was identified(attributes can be string, boolean, number and time) and entities without attributes were removed.
- **Finding associations:** The relationship between entities was found.
- And lastly specifying cardinality, constraints (as notes on the diagram) and determining if the attribute is allowed to be without value.

4 Result

The ER conceptual model in Figure 1 represents the domain model of the Soundgood music school. The main purpose of this model is managing the student's and instructor's personal information, various types of lessons and their schedules, instrument rentals and the financial transactions for the lessons provided. A more detailed description of the model is presented below.

Contact and personal information management: Since personal information is needed for several entities, a 'Person' entity was created as a superclass to subclasses 'Applicant', 'Student', 'Instructor', 'ContactPerson' and 'Sibling' to avoid redundancy. These subclasses inherit the attributes of the 'Person' entity. An 'Address' entity was created separately which has the atomic attributes city, zip code and street. 'Address' is in relation with the 'Person' entity and address is an composite attribute of 'Person'.

Student: An applicant entity was created with attributes skill level(indicating the applicant's current level), desired instrument (the instrument the applicant is interested in learning), and a boolean attribute, terminated, which indicates whether the applicant becomes a student or if they remain an applicant. If the applicant becomes a student,

[1] *A First Course in Object-Oriented Development* Chapter 4 Section 2 Step 2 page 26

they are assigned a unique studentID (a primary key attribute). Having these two entities could be a waste of space: we thought it could prove useful in the case of students who come and go of the school (it avoids having to recreate their profile) but if the customer deems that there is little chance this happens, we can easily take it out. Each student will also have a ContactPersonID which is in relation to the 'ContactPerson' entity and each student will have a distinctive contact person. If the student is no longer active, their 'terminated' attribute will be updated.

Instructor: The 'Instructor' entity also includes a unique InstructorID (key attribute), along with the instruments they teach and the ensembles they are able to conduct (multiple different instruments are played during ensembles and the instructor would presumably need to be familiar with the instruments played in the ensemble lesson). We did not consider the scenario where an Instructor becomes a Student. But a separate entity captures the Instructor's schedule, linked to a 'TimeSlot' entity where the specific date and time of the availability is recorded.

Lessons: In the 'Lesson' entity, each lesson has a unique lessonID (key attribute). The 'Lesson' entity serves as a superclass to subclass 'Group' which inherits its properties [2]. The 'Lesson' entity tells of its level (beginner, intermediate and advanced) and tracks minimum and maximum students needed to schedule and conduct group lesson and ensembles. Based on the instructor's schedule and the instruments they teach, a time for the lesson is scheduled. The two boolean attributes, 'Scheduled' and 'Given' determines whether the lesson was scheduled and whether it was conducted [3]. 'Group' lessons inherit the same properties from 'Lesson' however in group lessons we are also interested in the number of the participants. Ensembles, a subclass of 'Group', includes an additional attribute to record the genre of the ensemble lessons.

Instrument rental: The 'Rental' entity tracks instrument rentals with a key attribute, rentalID, and additional attributes such as studentID (to record the specific student renting the instrument) and instrumentID(to keep track of the specific instrument rented). Attributes rentStartDate and rentEndDate ensure the time restriction on rentals are enforced, while monthlyRentFee records the fee that need to be paid for the rented instruments each month. A student can have have several rentals in effect but a rental is for only one instrument at a time[4]. The school's rules on rentals will be translated into a set of integrity constraints later in the DB design process.

Lesson pricing and payment: The customer wants flexibility in its pricing scheme,

[2]To avoid redundancy and a spider-in-the-web, an Individual Lesson subclass to Lesson entity was removed, however that entity could be beneficial if the customer want to keep records of the instruments taught in individual lessons. In our model if the minStudent and maxStudennt are both set to 1, the lesson is an individual lesson

[3]the latter is important for instructor payment, as they are paid based on the lessons taught and not just scheduled

[4]However an instrument is linked to several rentals over time

which is why each lesson has its unique price (identified by priceID) which is determined by the lesson and the skill level (as beginner\intermediate and advanced have different prices) in accordance to a 'PricingScheme' entity[5]. To calculate student payment, each student has receipts every months, which update the total lesson fee for the current month based on the number of lessons attended and the rental fee for the rented instruments. If the student has a sibling that currently studies at the school both the student and the sibling receive a discount with is recorded in this entity (the activity of the sibling can be determined from the previousMonthLessonFee).

Instructor payment: The 'InstructorReceipt" entity determines the instructors salaries. The 'Lesson' entity keeps records of the lessons conducted which then would be accounted in the attribute 'currentNumberOfLessonsGiven' and since the price for each lesson is specified, the salary to the instructors is recorded in CurrentMonthSalary.

[5]That way, the customer can always access the cost of each lesson, and can update its scheme

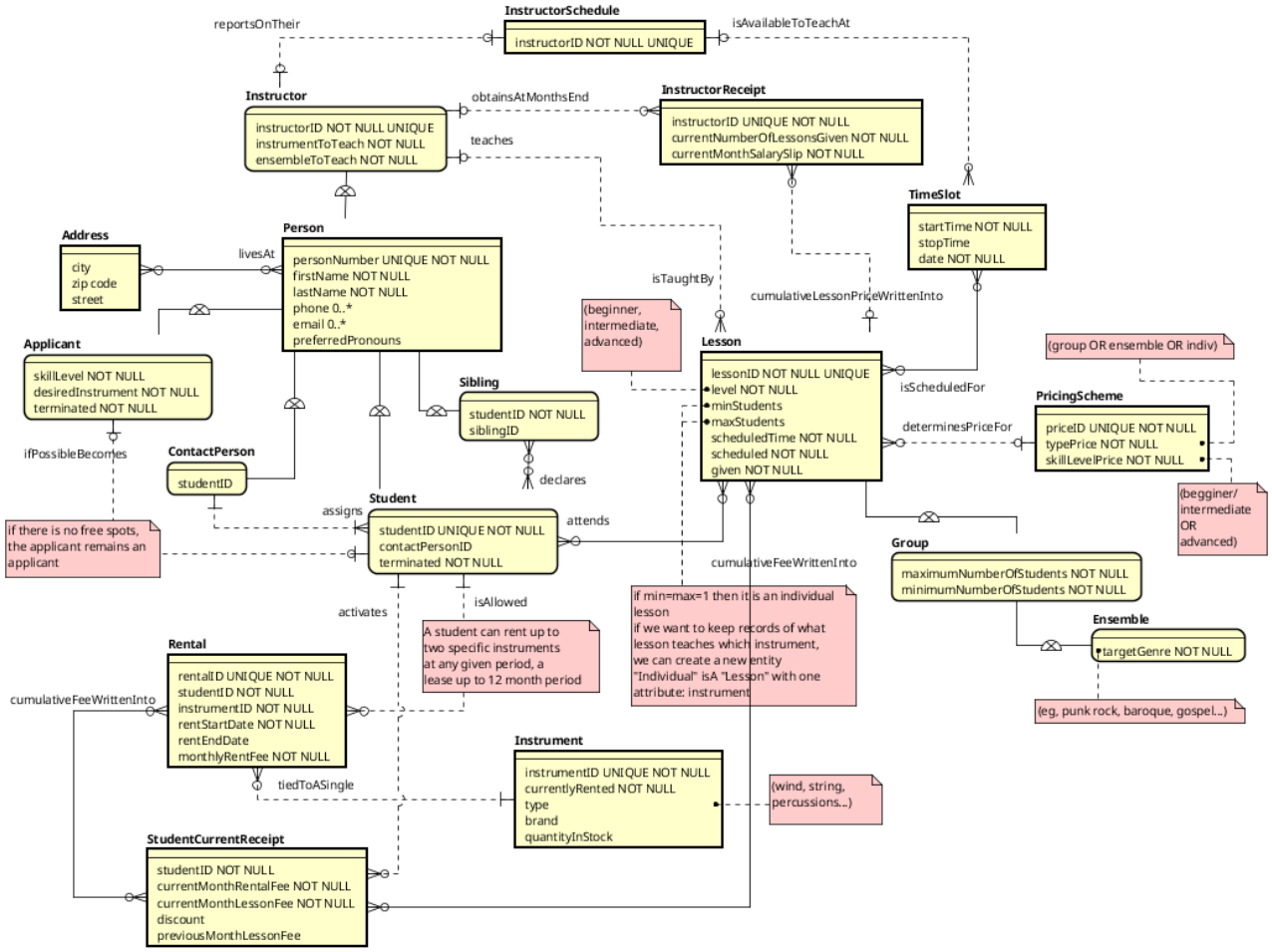

Figure 1: Screenshot of the ER diagram in Astah Professional

5 Discussion

Overall, we found it difficult to find the right balance between a naive and a programmatic model since we could not foresee what the next step would look like (ie how our model would be used for the logical design). We finally took the decision of making too much rather than too little, use self-describing attributes to ease comprehension and always have attributes that meant something "in the real world" but we are aware that our model is **programmatic**.

Concerning the difference between the *appointed* individual lessons and the *schedules* group lessons. We ended up deciding that the two events are the same:

- Let's say a jazz ensemble is scheduled for Friday evenings at the beginning of the semester: several instances of `Lesson` marked as not given will be created, each with their scheduled attribute set a the right `TimeSlot` (same startTime and

stopTime but different date). As the year goes by, the lessons given are marked as given=True while the pending ones remain given=False.
- Imagine a trumpet student contacts the administration staff for a lesson in the coming week with their teacher. The staff checks the instructor's schedule and finds a free time slot. An instance for `Lesson` at that time slot will be scheduled and marked as given=False until it takes place.

As you can see, both events act the same way in our model even if the are different mechanisms in real life.

The way we created the entity `Instrument` is to make the event of searching for a free-to-rent instrument easier: the students can check the complete or instruments or only the ones that are not currently rented (this can be decided when the application program is implemented, based on customer needs).

About personal details, further conversation about customer requirements can be provided. But we decided to make the attributes for the address optional (ie can be NULL and 0:N relations) since students can be homeless or have several registered addresses. And phone and emails can also be NULL if the student doesn't want to share their private information.

5.1 What we could have taken out

1. For the **receipts**: most of the attributes can be deduced from attributes of other entities, (e.g., the price of a lesson is cumulatively written into the current month salary slip of an instructor and the current month lesson fee of a student).
2. For the **discount** and knowing if a student's sibling was active during the previous month: we have taken steps to know this directly in the CM but maybe it will be concepts implemented during the logical design of the database.
3. The **instructor's schedule** makes sense in the real world and in the CM, but was perhaps unnecessary and wouldn't act as a good foundation for logical design.
4. If the customer finds it useless, the **applicant** entity and its relation to student can be easily scratched away.

5.2 Self assessment

Criteria	Notes	Discussion
Does the CM contain all information needed by Soundgood?	We believe the CM contains all the data required by Soundgood. However, to ensure all requirements are met, the CM should be evaluated with the Soundgood music school.	If the school wants to know the instrument for each individual lessons, we can add a `Individual` lesson subclass with one attribute: `instrument`.

Criteria	**Notes**	**Discussion**
Information is reasonably accessible	Only a few hops are required to collect information from the major entities since we avoided islands and spider-in-the-web.	Collecting information from the instructor to schedule lessons might be more tricky.
Reasonable number of entities	We believe a reasonable number of entities have been included to present the data required from Soundgood. One additional entity that could enhance the CM's presentation of reality is an individual lessons entity.	Based on the needs for the logical design stage, we could have deleted some entities (see section 5.1). During stage 3 of the design, we took out the `Individual` lesson because it held no attribute.
Attributes' correctness cardinality uniqueness and nullity	We believe all crucial data attributes required in the text are present and the cardinality is correctly defined. The attributes that require a value for every entity in the set are marked as NOT NULL, while attributes that may not be available are allowed to be NULL.	Regarding NULL attributes for example in the 'Rental' entity the 'rentEndDate' may not be set yet and in the 'StudentCurrentReceipt' entity the student might not be applicable for discount.
Reasonable number of relations; Relations' cardinality and name		Except for inheritance, all relations have names and cardinality has been studied.
Are naming conventions followed? Are all names sufficiently explanatory?	Naming conventions are followed in the CM consistency, and names of the entities and attributes were chosen to make the CM easy to understand for a third party.	Entities follow Java classes conventions. Relations follow Java methods conventions. Attributes follow Java variables conventions. If another programming language is used, we can easily change the convention.
Is the notation (UML or crow's foot) correctly followed?	We used IE notation for this task.	

Criteria	Notes	Discussion
Are there any aspects of the CM that raises concern and can be improved?	The CM may be too programmatic.	This may take away from simplicity of the CM and make it harder to understand for a third party.

Table 1: Self-assessment based on the task's criteria

6 Comments About the Course

For this task, we joined all lectures given by Paris Carbone and spent a minimum of 15 hours on literature study.

Reading chapter 2 of Fundamentals of Database Systems proved to be rather unnecessary and time consuming for this task. Leif Lindbäck's exercises however were very helpful for knowing what *not* to do, though we were a bit lost on drawing associations in the ER diagrams using Astah.

In total, we estimate at 20 hours, the number of hours spent on this task.