Project Report

Data Storage Paradigms, IV1351

8 November 2024

Project members:
[Kundan Krishna Sai Raghav Vengala, vengala@kth.se]
[Sreenija Veladri , sreenija@kth.se]
[Sai Santhoshi Srinithya Darbha, darbha@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

The task involves understanding the theory of the conceptual model and implementing it in a real life example - Soundgood music school. The theoretical understanding of the conceptual model is obtained via lecture videos and the textbook.

The application of this knowledge results in creating a conceptual database model that stores the information about students, instructors and instruments in a music school and facilitating options of taking lessons as individuals, groups or ensembles, the lessons are categorized based on 3 levels of difficulty(beginner, intermediate and advance). The payment for students, instructors and even rental fee is calculated per month.Another key feature of this model is that it offers discount for students whose siblings also take classes. This model ensures that the data is stored efficiently.

We also implement inheritance in order to discuss the pros and cons of inheritance.

2 Literature Study

Before delving into the development of the conceptual database model the prerequisite knowledge is obtained form the below sources.

- *Recorded lecture on IE notation*: This lecture provided the knowledge required to implement the IE notation in the database model. This notation is used to create the ERD(Entity relation Diagrams). This provides an overview on the components of the conceptual model - entities, attributes, relations(identifying and non identifying) and inheritance

- *Recorded lecture on Conceptual model*: This lecture describes the actual model and the method involved in creating the database with the usage of an example. The creation of a conceptual model involves 5 steps- noun identification, category list, removing unnecessary entities, find attributes, find relations; those of which will be discussed in detail in the **Method section**

- *Chapter 3 in Fundamentals of Database Systems*: This chapter cover the fundamentals of the conceptual model. It discusses the basic concepts of this model- entities, attributes, value sets(constraints of values a set can take); about relationships- their type, degree and constraints ; about constraints -cardinality rations and participation constraints(compulsory or can be null)

- *Chapter 4 of object oriented programming-Lief Lindback*: This was particular chapter was used for understanding the analysis of Database model , the relevant information particularly used for this assignment was for the creation of a category list(from page 26) of the book.

3 Method

The aim of the task was to create a conceptual model for the Soundgood music business. The conceptual model was designed using the Astah software and the IE notation(crows feet) was used to model and draw up the schematic.

The following steps were followed as described in the Conceptual model lecture.

- Noun identification: The first step was to identify the nouns in Soundgood music business description. The nouns identified either became entities or attributes.

- Category list: The next step was to draw up a category list, this is to find classes that are not nouns in the requirements as mentioned in *Object Oriented Programming book.*

- Removing unnecessary entities: After the identification of all nouns and obtaining the category list it was observed that some entities aren't needed and those are removed

- Find attributes: Attributes fro each entities are found , cardinality is determined and set , check is a certain attribute can be null or should it be unique.
- Find relations: In this step we form relations between entities and declare them (can be of different sorts- identifying, non-identifying, many to many or subtype(for inheritance))

4 Result

The solution to the conceptual model is presented in Figure 1.

- The solution was started out by identifying the different entities and the attributes that are needed to define them.
- Inheritance was implement on 3 entities - Person, Payments and Lessons- the subclasses inherit all attributes mentioned in the superclasses. The subclasses have specific attributes relevant only to that particular subclass.
  1. Person - has subclasses Student and Instructor
  2. Payment - has subclasses Student payment and Instructor payment
  3. Lesson - has subclasses individual lessons, group lessons and Ensembles
- Once the inheritance was established, the relations were set to meet the requirements of the task. Identification of strong entities and weak entities were done to determine wether the relations were identifying or non identifying.
- The cardinality was set for relations and attributes, and certain attributes were set to be UNIQUE or NOT NULL
- In the model, the enrolled students could take lessons, instructors were assigned lessons, a flexible pricing scheme is present to determine the cost per lesson which is then totaled at the end of the month, instructors are paid monthly, instruments can be rented by students (maximum of 2) whose rent is calculated monthly. If a particular student has siblings who are also enrolled then a discount is applied to the payment.
- Student id and Instructor id have been included in Student payment and Instructor payment respectively in order to determine the lessons the have taken and obtain the lesson id.
- The model has been designed to meet the described business transactions.
- The relation specific results have been pointed out in the Figure 1.

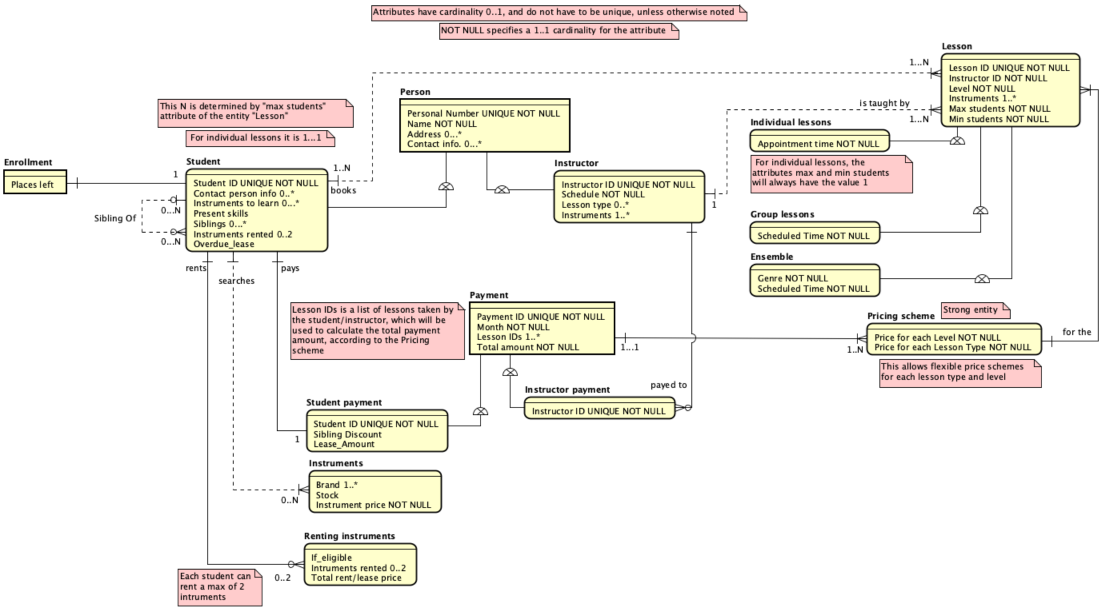

Attributes have cardinality 0..1, and do not have to be unique, unless otherwise noted
NOT NULL specifies a 1..1 cardinality for the attribute
This N is determined by "max students" attribute of the entity "Lesson"
For individual lessons it is 1...1
Enrollment
Places left
Student
Student ID UNIQUE NOT NULL
Contact person info 0..*
Instruments to learn 0...*
Present skills
Siblings 0...*
Instruments rented 0..2
Overdue_lease
Sibling Of
0...N
0...N
1
1..N
books
Person
Personal Number UNIQUE NOT NULL
Name NOT NULL
Address 0...*
Contact info. 0...*
Instructor
Instructor ID UNIQUE NOT NULL
Schedule NOT NULL
Lesson type 0..*
Instruments 1..*
1
Lesson
Lesson ID UNIQUE NOT NULL
Instructor ID NOT NULL
Level NOT NULL
Instruments 1..*
Max students NOT NULL
Min students NOT NULL
1...N
is taught by
1...N
Individual lessons
Appointment time NOT NULL
For individual lessons, the attributes max and min students will always have the value 1
Group lessons
Scheduled Time NOT NULL
Ensemble
Genre NOT NULL
Scheduled Time NOT NULL
rents
searches
pays
Lesson IDs is a list of lessons taken by the student/instructor, which will be used to calculate the total payment amount, according to the Pricing scheme
Payment
Payment ID UNIQUE NOT NULL
Month NOT NULL
Lesson IDs 1..*
Total amount NOT NULL
1...1
Pricing scheme
Strong entity
Price for each Level NOT NULL
Price for each Lesson Type NOT NULL
1..N
for the
This allows flexible price schemes for each lesson type and level
Instructor payment
Instructor ID UNIQUE NOT NULL
payed to
Student payment
Student ID UNIQUE NOT NULL
Sibling Discount
Lease_Amount
1
Instruments
Brand 1..*
Stock
Instrument price NOT NULL
0..N
Renting instruments
If_eligible
Intruments rented 0..2
Total rent/lease price
0..2
Each student can rent a max of 2 intruments

Figure 1: Conceptual model- using inheritance

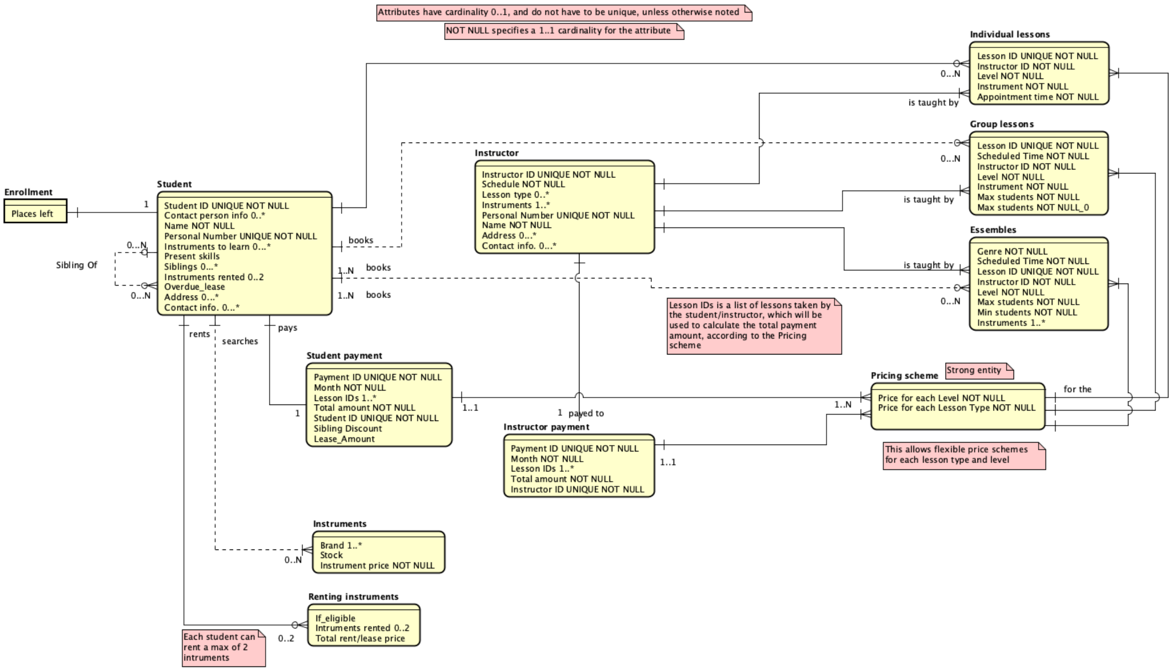
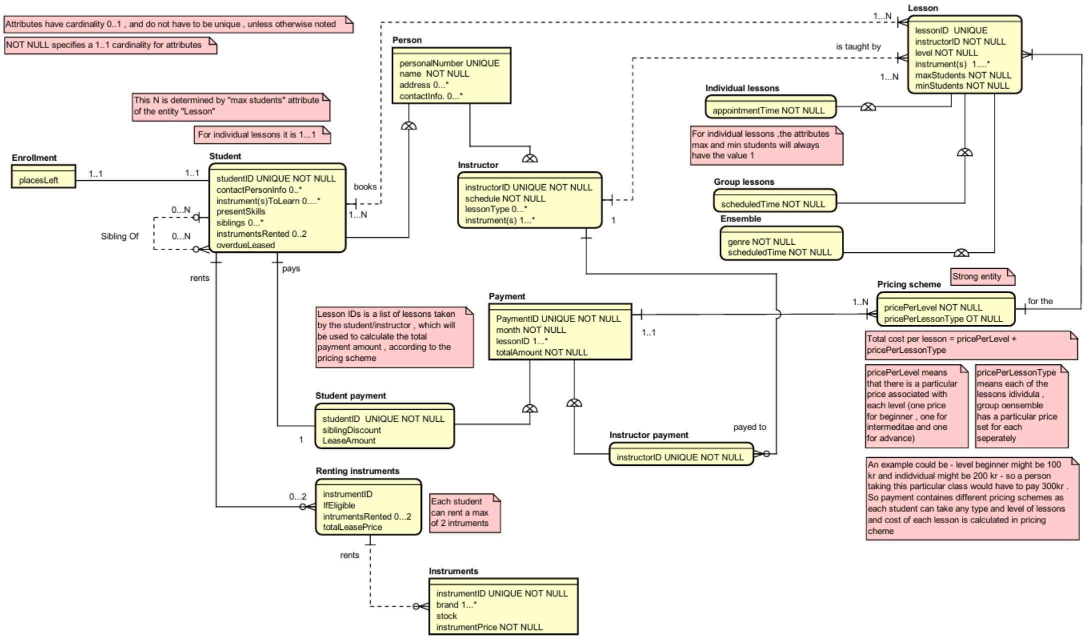


Figure 2: Conceptual model- without inheritance

Improvements

- The pricing scheme entity is used to calculate the price per lesson based on the its level and type and the total payment is sum of all the pricing schemes for that particular month (sum of prices for all the classes the student has taken that month).
- The notation has been corrected to camelCase.
- The relation that connects student to instrument wasn't needed. However a relation between instruments and renting instruments was established which links a specific instrument to details about its rental.


Attributes have cardinality 0..1 , and do not have to be unique , unless otherwise noted

NOT NULL specifies a 1..1 cardinality for attributes

This N is determined by "max students" attribute of the entity "Lesson"

For individual lessons it is 1...1

For individual lessons ,the attributes max and min students will always have the value 1

Strong entity

Lesson IDs is a list of lessons taken by the student/instructor , which will be used to calculate the total payment amount , according to the pricing scheme

Total cost per lesson = pricePerLevel + pricePerLessonType

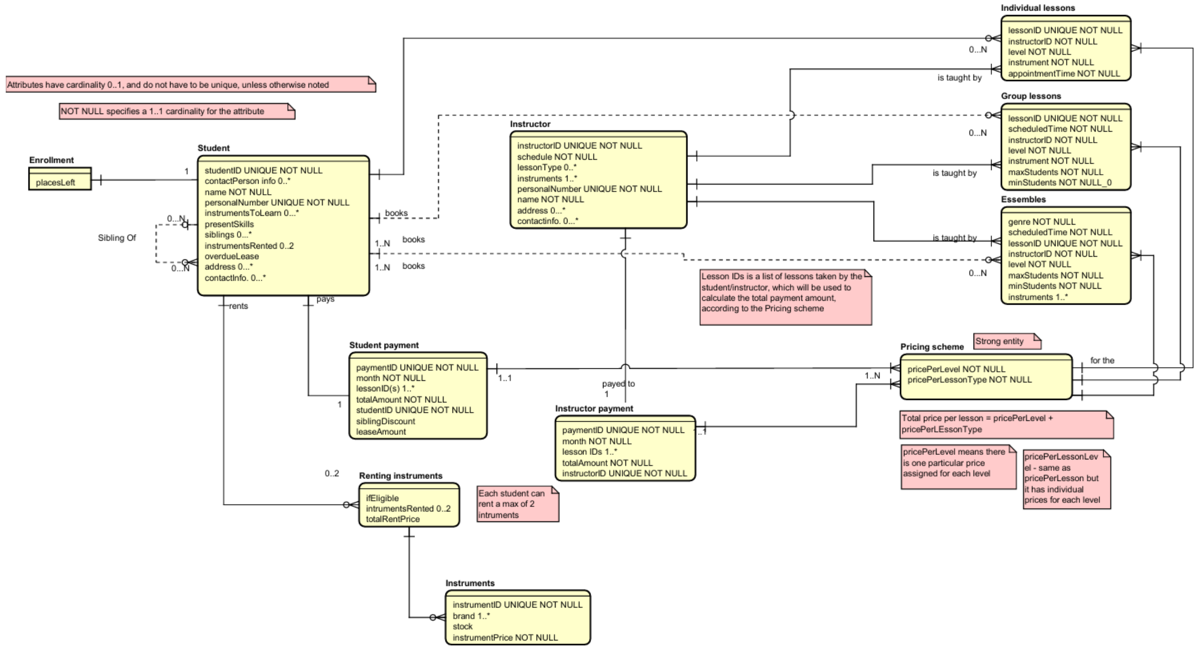
pricePerLevel means that there is a particular price associated with each level (one price for beginner , one for intermeditae and one for advance)

pricePerLessonType means each of the lessons idividula , group oensemble has a particular price set for each seperately

An example could be - level beginner might be 100 kr and indidvidual might be 200 kr - so a person taking this particular class would have to pay 300kr . So payment containes different pricing schemes as each student can take any type and level of lessons and cost of each lesson is calculated in pricing cheme

Each student can rent a max of 2 intruments

Figure 3: Improved Conceptual model- inheritance



Attributes have cardinality 0..1, and do not have to be unique, unless otherwise noted

NOT NULL specifies a 1..1 cardinality for the attribute

Lesson IDs is a list of lessons taken by the student/instructor, which will be used to calculate the total payment amount, according to the Pricing scheme

Strong entity

Total price per lesson = pricePerLevel + pricePerLEssonType

pricePerLevel means there is one particular price assigned for each level

pricePerLessonLevel - same as pricePerLesson but it has individual prices for each level

Each student can rent a max of 2 intruments

Figure 4: Improved Conceptual model- without inheritance

5 Discussion

The Conceptual model has covered the required information to create a Database system for the SoundGood music- covering the lesson types; payment types; rental information, payment; student, instructor and instrument information; a flexible pricing schematic.

The general naming convention has been followed and the cardinality of each relation is specifies close to the edges of the relational lines for more clarity. Cardinality for attributes have also been mentioned. It has been ensured that all the attributes have a mentioned or default cardinality.

From the given information about all the relevant entities are present in the conceptual model.

It has been ensured that the IE notation(crows feet) is followed and the business rules and constraints have been mentioned in the form of notes in the Figure 1.

While checking for unnecessary attributes - Example : In price schematic it was observed that ”price for each lesson” wasn’t needed so it was removed ,this helped ensure that entities didn’t contain and unnecessary attributes.

A comment is made where the differentiation between a strong and weak entity in an identifying relation is not prominent/obvious in the ER diagram using IE notation

Inheritance was used for in 3 places(People, Payment, Lessons) as it was observed that using one superclass to define all common attributes simplified the model and made it more coherent. This also helped ensure that all the required attributes for those entities were included. This inheritance also allows for flexible changes without having to change attributes in all the entities.

For the model that was implemented without inheritance the advantage would be the each entity has a list of all it's specific attributes making it easier to run checks in-case there are irrelevant attributes. Also in cases where inheritance isn't needed in creates unnecessary complications.