Project Report

Data Storage Paradigms, IV1351

Teoman Köylüoglu and Teoman@kth.se

Date: 2023-11-09

1 Introduction

The purpose of this task is to create a conceptual model with the purpose of modeling a database. The database will contain data representing a "Soundgood" music school. The description of the contents of the database is given in a comprehensive business overview which will be used to the create the conceptual model.
Data relating to personal information as well as of instructors, students, persons, instrument, level of lessons and different types of lessons, discounts (for siblings), payment and rental of instruments will be comprehensively modeled in an ER Diagram.

There are different types of lessons; solo, group and ensemble, as well as different types of instruments that the student may sign up for. Ensembles are a specification of a specific genre and group lessons are made for a specific instrument, both have a minimum and maximum number of students. Payment for the instructor and student is also done lesson-wise. Instructors are assigned to group/ensembles and will be available for solo lessons. The price scheme consists of the different prices related to levels and type of lessons, and an invoice is made for the the student's accumulated lessons over the course of the previous month.

A user interface provides administrative staff the possibility of handling bookings and payments etc. Finally, any information about statistics related to students of the sort may be done via manually querying the database.

This project is a joint collaboration with students Julius Larsson and Victor Naumburg.

2 Literature Study

For the purpose of this project, in relation with the course material two sources were mainly used for the gathering of information. The first being the main textbook of this course, fundamentals of database systems. For this seminar, reading chapters one through four were sufficient. Chapter one gave a summary and explanation of what a database is and its integral functions. I learned about the historical context and the different kinds of users that operate on and behind the scene of a database. It did however provide little use in the actual implementation of this section of the project.

Chapter two described different data models used in database systems, the schema and different languages used with the database. Again, these provided little help for completing the task, but were of support in providing beneficial context in the grand scheme of this operation and its end goals.

Chapter three covered conceptual modeling which in combination with the lectures provided by lecturer Leif Lindbäck were essential for the understanding of the material needed in the project the methodology that was to be used throughout the conceptual modeling. Leif and the book provided clear instructions and the different ways of explaining the contents were complementary for my understanding of the material. One clear example that stands out is Leif's discussion on the methodology of identifying entities.

The book - as well as - Leif both mention noun identification as a necessary step. However Leif's inclusion of a "category list" and its following provided a good way of breaking down the problem via stimulating creativity, thus enabling the architect to find entities that otherwise would've gone hidden.
Moreover, chapter three gave in-depth insight on weaker entity types and different types of attributes e.g key, multivalued, composite and derived attributes wherein composite ones that normally ought to become their own entity when encountered which proved to be rather useful when realizing the CM.

Lastly, chapter four in combination with the small segment of inheritance mentioned by Leif provided the necessary knowledge on how to tackle inheritance which would be used to pursue a higher grade in the task.

3 Method

To solve the task a clear methodology has been implemented for this conceptual model. To begin with, Astah, a diagram editor was used to realize the CM. It allows for the use of an ER diagram and IE notation (crow foot) which was the convention used for displaying symbols and relations.

To meet the requirements in the description of the music school, Leif's method was implemented. The first step was to identify all of the nouns that could be found throughout the description. Proceedingly the use of a category list provided a different approach for finding hidden entities within the description. Thereafter, excessive entities were removed via turning such entities into appropriate attributes and lastly associations be-

tween relevant associations were added.
Since a higher grade was sought, the use of inheritance needed to be employed. To do this, finding out which entities had attributes in common were a good way of seeing where inheritance ought to be used.
To check whether the diagram met the requirements, constant consultation and reviewing the description was necessary to see all entities, attributes and relations satisfy the music school's operation.

4 Result

To meet the requirements of the task, each involving party in the operation were handled as their separate entities and attributes of those entities were incorporated.
For the requirements specified in section 1.1 of the project; When a student signs up for a lesson they should submit their contact details, the instrument they want to learn and their present skill.

To meet this requirement a entity for a person containing the attributes for contact details were incorporated into the entity. But seeing as not every person is a student, a Student class inherited the attributes of its superclass (Person). A relation between the superclass "Lesson" class and the student was made to signify the student's signing up for the lesson. Since the conceptual model considers the storing of data, the associations are an abstract way of showing the act a student perform within the organization. The actual "submitting" of details is handled by the simple fact that the student has those attributes, and their relation to said lesson containing said difficulty they'd participate at.

As for the requirement of "present skill", an entity "InstrumentCompetence" with two attributes noting an instrument level and type, was created and associated to the superclass "Person". This is because not only does a student have a competence level, but as mentioned in the following section 1.2, an instructor has to be available for a specific instrument as well noting a competency. Seeing as inheritance is sought when possible, relating competency to "Person" is appropriate as student and instructor will inherit those traits.

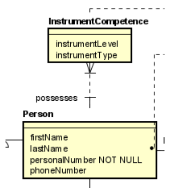

Figure 1: Subdiagram of figure 7, Included to illustrate inheritance of instrument competence

There are three different types of lessons; group, ensemble and individual lessons. Both group lessons and ensembles have a minimum and maximum number of participants, this requirement is ensured via minimum/maximum attributes in their respective subclasses. Each lesson is to be considered a specification of a general concept that is lesson, which has three attributes; A type, level and time of which it takes place. Let this Lesson be a superclass to the subclasses that are the different types that have been specified. Furthermore, ensembles are said to be of a specific genre thus the ensemble subclass should have one more "genre" attribute. Group lessons and individual lessons are specified to be of a specific type of instrument, therefore let those two subclasses contain an attribute of that nature.

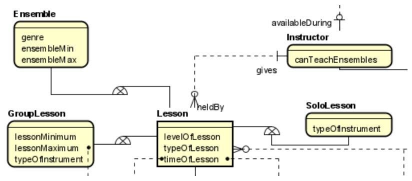

Figure 2: Subdiagram of figure 7, Included to illustrate types of lessons.

Administrative staff is said to be able to make bookings for which it must be known when an instructor is available and for which instrument, the latter has already been dealt with in the beginning of this section. The former (time available) can be seen as a separate entity representing an availability for the instructor. Appropriate attributes are integers representing a time frame; start, end and a specific date for which the booking of the lesson ought to be. Thereafter with an association drawn from the Instructor entity to the TimeAvailable entity, information related to the availability of the instructor becomes apparent.

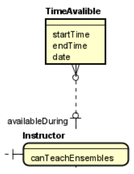

Figure 3: Subdiagram of figure 7, Included to illustrate time availability of instructor.

A student is able to take any number of lessons for an arbitrary number of instruments, this cardinality constraint is upheld via the crow foot notation denoted between the association of Student and Lesson. For this diagram however, inheritance was used and thus cardinality constraints are denoted via comments relating to the assocation in question. Such an example can be seen in figure 7, and its opposite (non-inheritance) is seen in figure 8.

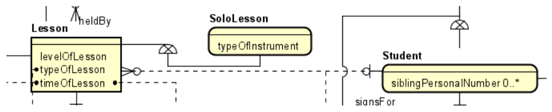

Figure 4: Subdiagram of figure 7, included to illustrate student's signings up.

Students whom are under the age of 18 need also have a contact person to whom they are related to. Therefore a separate entity noting a contact person (inheriting Person superclass) with a single attribute noting their personal relation is associated to the student entity.

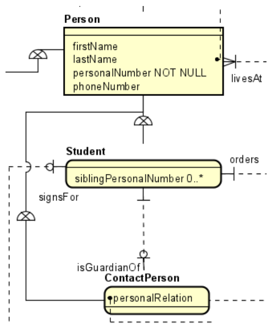

Figure 5: Subdiagram of figure 7, Included to illustrate contact person to student relationship.

Discounts are to be available for siblings whom have attended lessons during the same month. To see who's a sibling of who, each student need to store information about their sibling's personal number, thus it becomes an attribute of the Student entity. When two students whom are siblings their personal numbers will point to each other and since time of lesson already is stored for each instance of a lesson it's possible to see if a sibling has taken a lesson for a specific date. Discounts may then be appropriately incorporated into the invoice accumulated over the course of the month since the sibling's attribute is interrelated to the pricing scheme. As for instructor payment it's derived the same way the student's fee are without the added sibling discount. It's therefore just another instance of the Price entity as previously pointed out in the beginning of the section.

Students should also be able to rent instruments; An Instrument entity is created holding a single attribute containing the unique number for a specific instrument. A wide selection of instruments are available, this is noted by another entity representing the stock of instruments. It has four attributes, a type of instrument seeing as a wide selection of instruments ought to be available. A brand, and maximum stock attribute were added as the instruments has different brands and exist in different quantities. One student may only rent a maximum of two instruments at any given time, thus the cardinality from the student's side is one, and the cardinality of the Rental entity is a ”zero to many” but a constraint via comment limits the rental to two instruments per student as required by the description. Payment is ensured monthly in the same manner as for instructor and student.

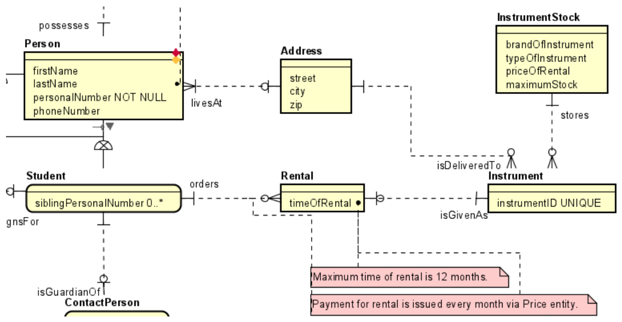

Figure 6: Subdiagram of figure 7, included to illustrate rental of instrument.

To further clarify why this works; Consider the Soundgood's purchase of ten Gibson guitars of a single type. When a student issues a rental then an instrument of that type (Gibson ES-137) with its unique ID is fetched from the stock with a specific rental price connected to it. Each instance of this specific association is a rented instrument, there are ten possibly active associations of this type (maximum stock) and each instance of this association is registered to a specific time of rental thereby ensuring payment through other associations. Delivery is ensured via the instrument's association to the student's address.

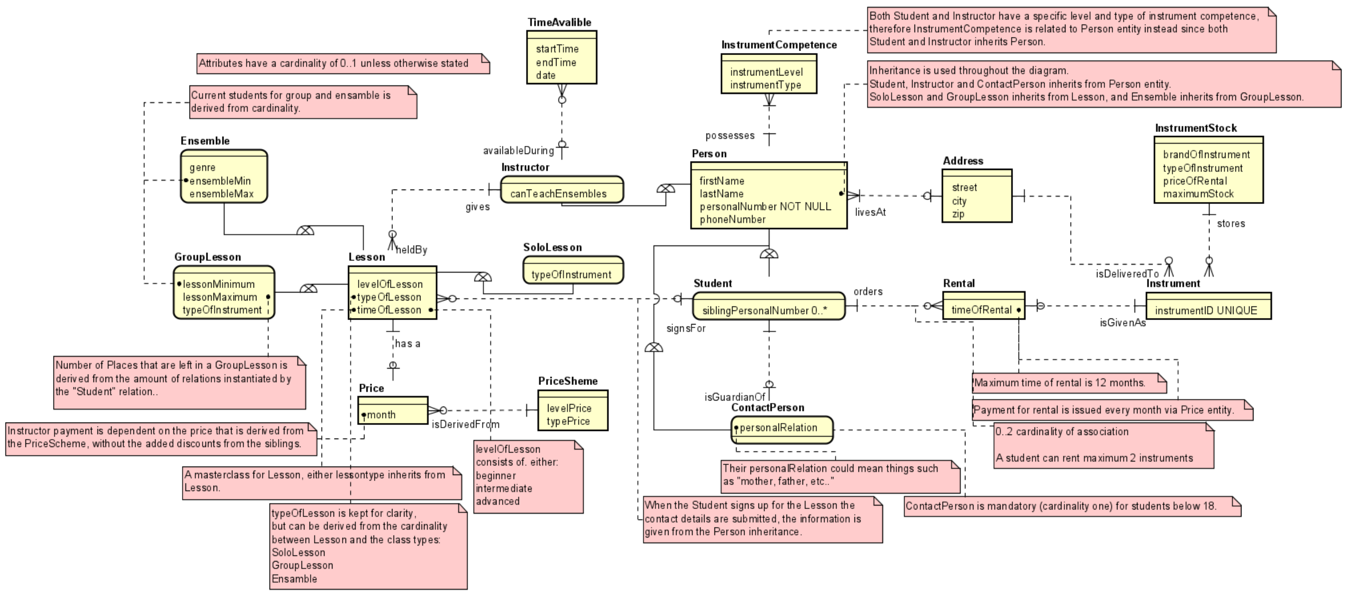
TimeAvalible
startTime
endTime
date
InstrumentCompetence
instrumentLevel
instrumentType
Both Student and Instructor have a specific level and type of instrument competence, therefore InstrumentCompetence is related to Person entity instead since both Student and Instructor inherits Person.
Attributes have a cardinality of 0..1 unless otherwise stated
Inheritance is used throughout the diagram.
Student, Instructor and ContactPerson inherits from Person entity.
SoloLesson and GroupLesson inherits from Lesson, and Ensemble inherits from GroupLesson.
Current students for group and ensamble is derived from cardinality.
InstrumentStock
brandOfInstrument
typeOfInstrument
priceOfRental
maximumStock
possesses
availableDuring
Ensemble
genre
ensembleMin
ensembleMax
Instructor
canTeachEnsembles
Person
firstName
lastName
personalNumber NOT NULL
phoneNumber
Address
street
city
zip
gives
livesAt
stores
heldBy
SoloLesson
typeOfInstrument
GroupLesson
lessonMinimum
lessonMaximum
typeOfInstrument
Lesson
levelOfLesson
typeOfLesson
timeOfLesson
Student
siblingPersonalNumber 0..*
orders
Rental
timeOfRental
isDeliveredTo
Instrument
instrumentID UNIQUE
isGivenAs
signsFor
has a
Number of Places that are left in a GroupLesson is derived from the amount of relations instantiated by the "Student" relation..
Price
month
PriceSheme
levelPrice
typePrice
isDerivedFrom
isGuardianOf
ContactPerson
personalRelation
Maximum time of rental is 12 months.
Payment for rental is issued every month via Price entity.
0..2 cardinality of association
A student can rent maximum 2 instruments
Instructor payment is dependent on the price that is derived from the PriceScheme, without the added discounts from the siblings.
levelOfLesson consists of. either:
beginner
intermediate
advanced
A masterclass for Lesson, either lessontype inherits from Lesson.
Their personalRelation could mean things such as "mother, father, etc.."
typeOfLesson is kept for clarity, but can be derived from the cardinality between Lesson and the class types:
SoloLesson
GroupLesson
Ensamble
When the Student signs up for the Lesson the contact details are submitted, the information is given from the Person inheritance.
ContactPerson is mandatory (cardinality one) for students below 18.

Figure 7: Complete inheritance diagram.

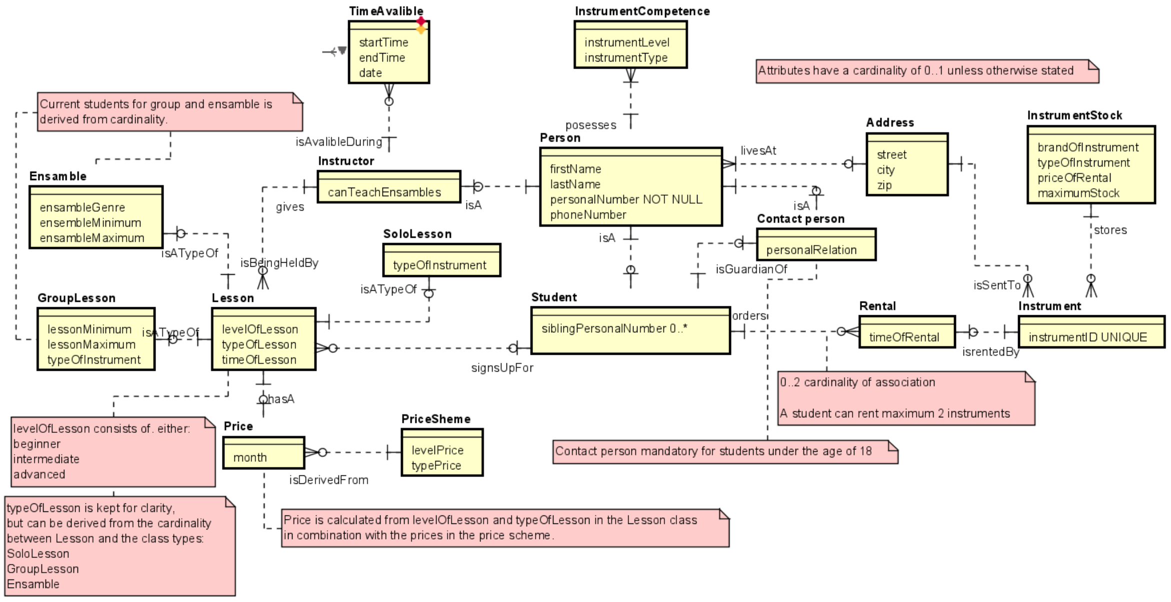

Figure 8: Complete non-inheritance diagram.

5 Discussion

This section will discuss the aforementioned results. To begin with, creating a conceptual model has truly proven to be a iterative process. In the beginning of the project, entities and their attributes were rigorously evaluated to see whether or not specific attributes fit better as relations instead. Discussions with regards to multiple instruments being played by a student were prevalent during early stages. This ever so seemingly issue arose as a result of the different lessons being spread out as three different attributes, in which it was thought to be unclear which level of lesson a student was associated with. Considering the fact that conceptual models and databases at this point was a relatively new subject, it took some thought before it was realized that each instance of the association will specify a relation between a specific level and instrument for each instance of a lesson. Grouping the different difficulties into one attribute were also concluded to be of benefit seeing as remaining attributes of difficulty would be initiated to null which is a waste of attributes and storage.

Furthermore there's a constraint from the company that discounts are distributed whence two siblings have attended a lesson during the same month. Therefore, on the front end, some query on the said month is to be made, as well as at some point in the future a schema for the different times a lesson has been taken place - that encapsulates all types of lesson - ought to happen. As a result, the question on whether to include "month" as an attribute of time would have been redundant was raised. At this point in the CM, an attribute for "time" was in place already to measure the times of lessons, and in such a case why would storing a month be necessary?
But as evident from the resulting diagram, having "month" as an attribute of an accumulated "Price" entity was deemed more appropriate for the task at hand. The "timeOfLesson" attribute inside the "Lesson" entity would timestamp each lesson which would then be used when accumulating the total sum of each month.

5.1 Inheritance

Inheritance, as mentioned in the introductory section, was implemented throughout the conceptual model. Quite quickly after completing the noun-identification it was realized to be a sophisticated solution to relating entities of similar characteristic. Instead of needing to bother naming each association and specifying cardinality using crow foot notation, inheritance solves these issues with a simple line. It does need to be taken into consideration however, that for cardinalities not specified inside relations, must be made clear with a comment.
Having inheritance with superclasses and subclasses in the conceptual model further emphasized the very fact it's real-world modeling being done. Inheritance is the use of generalization and specification; Instructors are at the very essence a specification of a person, therefore in light of real-world hierarchies being modeled, inheritance becomes

overtly appropriate.

It is however, not without its inconvenience. Inheritance is not really necessary, and it does not, at least in this case, make the conceptual model any better by large. For example, the ”Student” entity holds only a single local variable and is thus of not that great significance to perhaps warrant its own entity. Instead its attribute could have been merged with the superclass (person) and have its local attribute hold a null variable in cases where the superclass (person) is not a member of the subclass (student). If a maxim of the conceptual model is indeed the reduction of unnecessary entities then this would seem a rather appropriate thing to do. There is no direct relationship between the student and the person except that they are of similar character, thus even the ”isA” association in the non-inheritance version of the CM becomes questionable.

Similar considerations have to be made when considering that deletion of superclasses will automatically lead to the deletion of its respective subclasses. Having dependency relationships in a company where revenue between students and their rental information are interrelated, may or not be desirable depending on its usage and purposes. Anyhow, inheritance promotes readability and accurately depicts real-world hierarchies which would be of interest in settings where stakeholders, laymen or persons of interest to whom pitching the conceptual model may lead to financial gain or benefit.

5.2 Source of Errors

Some source of errors to be considered during the mandatory part of the task was the error of communication. It is extensively highlighted throughout the project that extensive communication between client and designer is essential for the success of the CM. The task description then itself becomes a culprit for this error as it’s not always semantically clear what is intended in the description.
One particular thing that comes to mind, was that group lesson equally as individual lessons had to have a ”type of instrument” as attribute. It wasn’t always clear from the instructions if what was considered as ”group lesson” was a part of the general term that is ”lesson” or if ”lesson” itself only denoted individual lessons. For example the related subsection starts with saying: ”there are individual lessons and group lessons” and then proceeds to use ”lesson” as a general term making it unclear if what applies to lesson applies equally to both types or not, or additionally ensembles as well. This lead to a situation wherein ensembles, wrongly so, were considered as a specification of the group lesson subclass leading it to become a subclass of the subclass ”GroupLesson”.

Such encounters of semantical misinterpretations were common throughout the task and is something which normally would have been discussed with the client, or in this case something which could have been discussed with the lecturer instead. Something to perhaps consider more often for the coming tasks.

6 Sources

Elmasri, Navathe, Fundamentals of Database Systems, Seventh edition, 2016
Lindbäck, Leif, Föreläsning, Conceptual Model recordings
Lindbäck, Leif, Object Oriented Development, 2023
Lecture one in the course IV1351