Seminar 1

Axel Månson Lokrantz

2022-11-23

Introduction

The task for this project was to create a conceptual model for a fictional music school called Sound Good Music. A conceptual model is an abstract representation on how tasks should be carried out, in this representation a database. The database must not store more data than what is specified in the school's detailed description. This means, student fees and instructor payments should be summed up and sent to Sound Good's financial system. Any other financial purpose like taxes, bookkeeping and bank contacts should be excluded. For higher grades the report must also contain a relevant discussion about the above mandatory part of the task and use inheritance in at least one place of the diagram. The diagram was created together with Daniel Dahlberg in UML modeling tool Astah at the royal institute of technology in Stockholm.

Literature Study

To prepare for this seminar I watched all the conceptual model lectures on canvas, read the tips and tricks pdf and skimmed through the corresponding chapters in the course literature.

Method

To create the conceptual model we used a modelling method similar the one you use for creating a domain model. The main difference between a CM and a DM is that the CM uses entities instead of classes and focus more on data and relations. There can be no entities without an attribute and all attributes must be carefully selected with their corresponding cardinality.

Noun identification means to identify all the nouns in the detailed business description. Each noun was created as an entity in the diagram

Category list A category list can be helpful to find entities that are less obvious. It is better to have an open mind during this step as it is often easier to remove entities than to come up with new ones. It does not matter if an entity is an attribute or not, the main purpose in this step is to find as many entities as possible.

Remove unnecessary entities In this step we removed all the unnecessary entities by first dividing them into categories. One category was every entity describing the rental process, another one was all entities related to the lessons. We also removed the entities which were vague and better described elsewhere in the diagram.

Find all attributes With the complete list of all entities, we specified their relevant attributes and cardinality. It is important to note that an attribute is not part of an entity, like house and window for example. An attribute can be viewed as a property which describes the entity. In our diagram, unless specified, the cardinality was set to `1..1` (`NOT NULL` and at most 1) by default.

Find all relations A CM uses relations between entities. Actions are normally not of interest when creating the diagram since the action seldom include data. Each relation should also have association which shows how data is joined to give all the desired information.

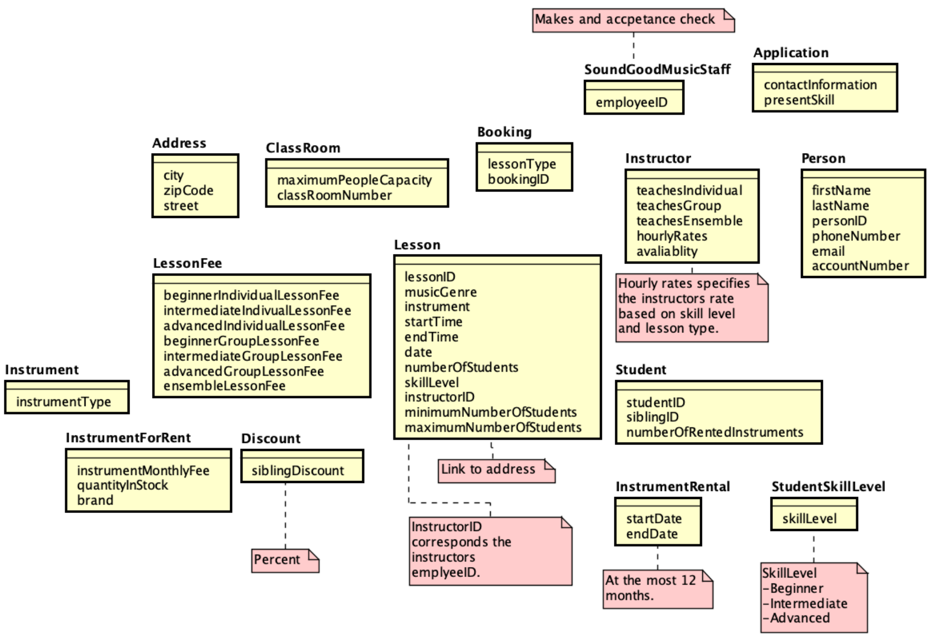

Figure 1: First draft of the CM diagram, attribute cardinality and relations are missing.

Results

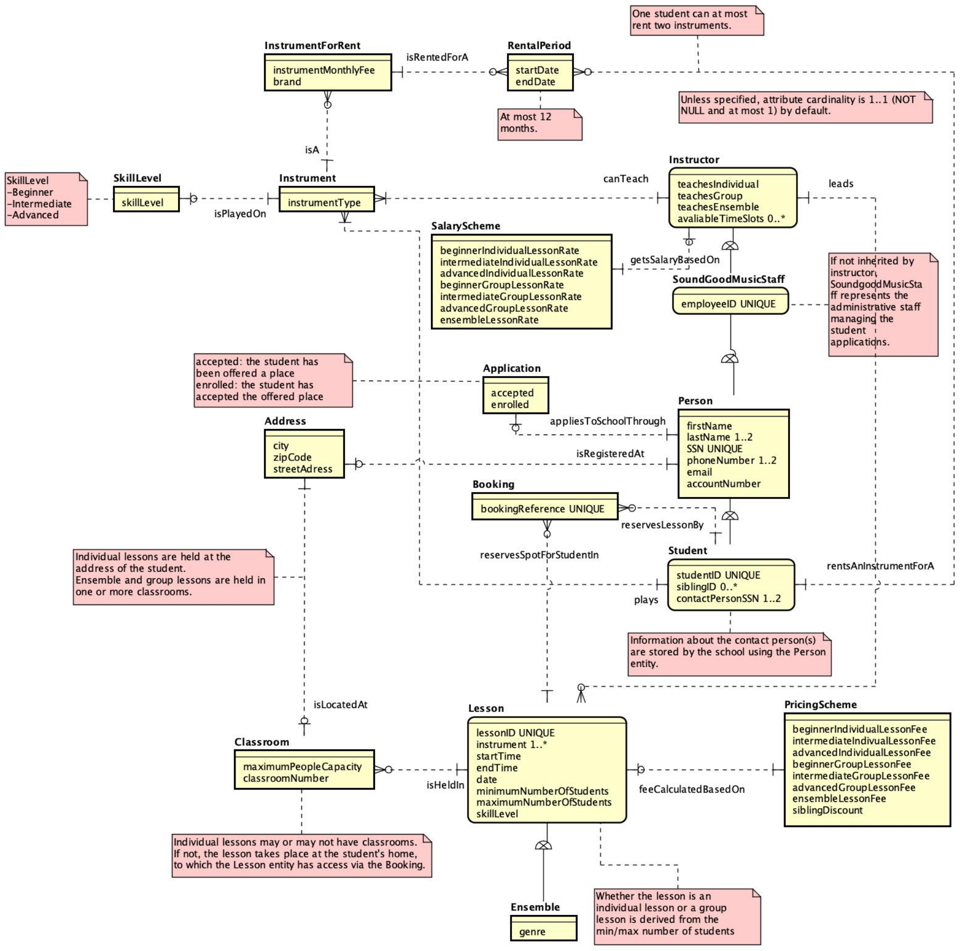
One student can at most rent two instruments.
InstrumentForRent
instrumentMonthlyFee
brand
isRentedForA
RentalPeriod
startDate
endDate
At most 12 months.
Unless specified, attribute cardinality is 1..1 (NOT NULL and at most 1) by default.
isA
SkillLevel -Beginner -Intermediate -Advanced
SkillLevel
skillLevel
isPlayedOn
Instrument
instrumentType
canTeach
Instructor
teachesIndividual
teachesGroup
teachesEnsemble
avaliableTimeSlots 0..*
leads
SalaryScheme
beginnerIndividualLessonRate
intermediateIndividualLessonRate
advancedIndividualLessonRate
beginnerGroupLessonRate
intermediateGroupLessonRate
advancedGroupLessonRate
ensembleLessonRate
getsSalaryBasedOn
SoundGoodMusicStaff
employeeID UNIQUE
If not inherited by instructor, SoundgoodMusicStaff represents the administrative staff managing the student applications.
accepted: the student has been offered a place enrolled: the student has accepted the offered place
Application
accepted
enrolled
Person
appliesToSchoolThrough
firstName
lastName 1..2
SSN UNIQUE
phoneNumber 1..2
email
accountNumber
Address
city
zipCode
streetAdress
isRegisteredAt
Booking
bookingReference UNIQUE
reservesLessonBy
reservesSpotForStudentIn
Student
studentID UNIQUE
siblingID 0..*
contactPersonSSN 1..2
rentsAnInstrumentForA
plays
Individual lessons are held at the address of the student. Ensemble and group lessons are held in one or more classrooms.
Information about the contact person(s) are stored by the school using the Person entity.
isLocatedAt
Lesson
PricingScheme
Classroom
maximumPeopleCapacity
classroomNumber
isHeldIn
lessonID UNIQUE
instrument 1..*
startTime
endTime
date
minimumNumberOfStudents
maximumNumberOfStudents
skillLevel
feeCalculatedBasedOn
beginnerIndividualLessonFee
intermediateIndivualLessonFee
advancedIndividualLessonFee
beginnerGroupLessonFee
intermediateGroupLessonFee
advancedGroupLessonFee
ensembleLessonFee
siblingDiscount
Individual lessons may or may not have classrooms. If not, the lesson takes place at the student's home, to which the Lesson entity has access via the Booking.
Ensemble
genre
Whether the lesson is an individual lesson or a group lesson is derived from the min/max number of students

Figure 2: Final version of the CM diagram.

Above the final version of the CM diagram can be seen. The diagram uses IE notation (crow foot) for relations. One of the business rules was that one student could at most rent two instruments. This rule is implemented in the diagram through the two entities `student` and `rental` where 1 student can rent `0..*` instruments. To clarify the relation we included a note with the proper restriction of `0..2` instruments per students. Another business rule which was tricky to implement was the skill level of a student, which is only allowed to be beginner, intermediate and advanced. For now, we settled on a note describing the rule. Part of the higher-grade criteria was to include at least one inheritance relation in the diagram. An example of this in our diagram is the relation between person and student. Where student inherit all the attributes from person. This means that the student entity not only has a `studentID`, `siblingID`, a `ContactPersonSSN` but also all the attributes inherited by `person` .

Discussion

The CM contains all the vital entities such as `Student`, `Instructor` , `Payments`, and `InstrumentRental` presented in the detailed description. We decided, the best way to handle lesson fees and instructor payments was to describe their corresponding pricing scheme in separate entities. This helped with the overall readability of the diagram. In a previous draft of the diagram our lesson entity contained all types of lessons, individual, group and ensemble. This proved to be problematic since for every lesson we would get two attributes set to `NULL`. To fix the issue, we introduced two new attributes min/max number of students and moved the ensemble attribute to its own entity. This change also helped us to control one of the business rules which was that all types of lessons have different numbers of students. Through `maximumNumberOfStudents` we can derive which type of lesson it is.

In both programming and in a conceptual diagram inheritance is supposed to illustrate a `isA` relation. There are a few differences, however. In a CM inheritance does not break encapsulation and is dynamic which means that we do not have to manually edit the database and change what inherits what. The super entity should always be a generalization of the sub entity and the sub entity should always be a specialized instance of the super entity. All attributes in the super entity must also be relevant for the sub entity. For example, in our diagram we have an entity called `Person` which is considered a super from which the sub entity `SoundGoodMusicStaff` inherits all attributes. It is important to note that all inheritance relations can be interpret to a `0..1` relation. Meaning a person can either be a `SoundGoodMusicStaff` or not. What this translates to in a program like SQL is that entity A may or may not have a foreign key to entity B. The most obvious benefit of such a relation is the ability to introduce common properties without having to write them repeatedly in the diagram. Inheritance also helps us tell all the `isA` relations apart from the rest of the `0..1` relations. One of its drawbacks is that we add another layer of complexity to the diagram. This may hinder the readability when working with a client for example, and since it will be rewritten as a `0..1` relation once implemented why not keep it that way.

One problem with the diagram that we found no real solution to was that our entity `Lesson` owns the attributes `skillLevel` and `instrument` both of which are entities elsewhere in the diagram. We tried to model it differently, with a relation dragged across the diagram from instrument to lesson. This way the skill level of a lesson could be derived from its corresponding instrument. But then, how do you derive the skill level of an ensemble, which has more than one instrument?

Since the detailed business description did not specify where the lessons are held, we assumed that group and ensemble lessons are held in classrooms and individual lessons are held at the address of a student. The address for group or ensemble lesson will be derived from the entity `Classroom` while the address of an individual lesson is derived from student from which the lesson entity has access to via booking.