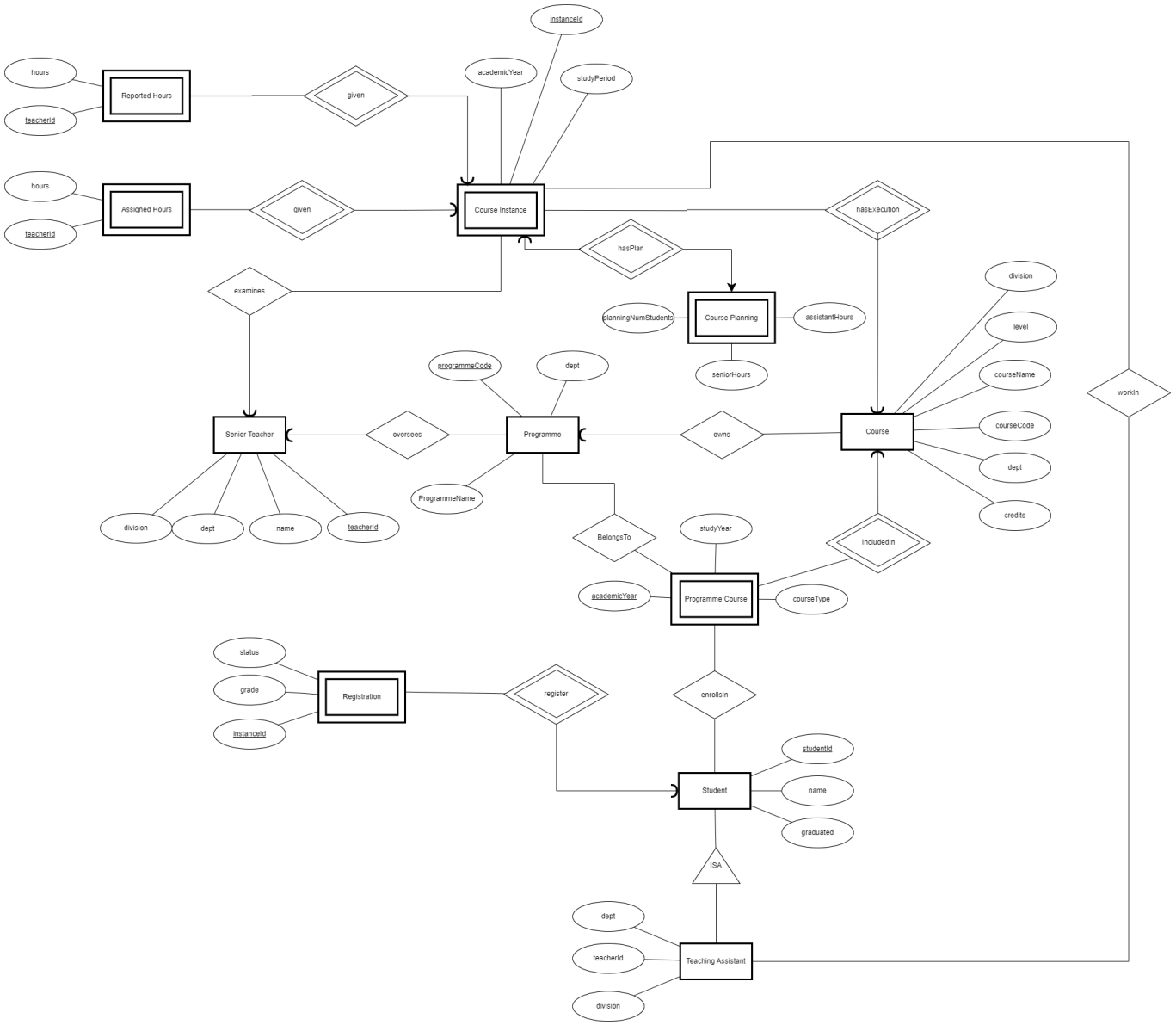
Advanced databases - Assignment 1

José Cutileiro

April 9, 2024

1 Entity-Relationship diagram



Note: I believe the image quality of the diagram is good, but nevertheless, I will add the file 'diagram.png' to ensure it is presentable.

.

2 Primary keys

=== Class ===	=== Primary Key ===
SeniorTeachers	teacherId
Programme	programmeCode
Course	courseCode
Programme Course	courseCode,academicYear
Course Instance	courseCode,instanceId
Course Planning	instanceId
Assigned Hours	teacherId,instanceId
Reported Hours	teacherId,instanceId
Student	studentId
Registration	instanceId,studentId
Teaching Assistant	studentId

2.1 Explanation

Although not always explicitly specified, some primary keys are quite intuitive, especially when dealing directly with IDs or specific codes. An example of this is the **'Student'** entity, whose primary key will intuitively be the 'studentId'. This pattern has been followed in several entities, as noted in the table above.

However, this pattern cannot be applied to all entities. For example, in the case of **'Programme Course'**, it is not feasible to exclusively use the program and course codes to identify the "programme course". This is because it has been mentioned that there may be some discrepancies in this entity from year to year. However, it is important to maintain records of information from previous years. Therefore, I chose to include the academic year in the primary key, which, combined with the course code, will be able to identify any instance of this entity.

When addressing the **"Course Instance,"** it may seem intuitive to believe that the "instanceId" is sufficient to identify this type of entity. However, in my diagram, I highlighted that a "course instance" depends on the existence of a specific course. In other words, it is not possible to have an instance of a course that does not exist. Therefore, I chose to designate this entity as a weak entity, which requires us to include more attributes in the primary key.

In **"Course Planning,"** I assumed that each course instance has, at most, one associated plan. Therefore, the course instance ID is sufficient to identify this plan, without the need to include any other attribute for this purpose besides the instanceId.

Regarding **"Assigned Hours" and "Reported Hours,"** each instructor (whether they are a TA or a senior professor) will have at most one reference to these classes for each course instance they are working on. This does not exclude instructors who teach more than one course. However, to identify these entities, we need the combination of the professor's ID and the course instance ID under consideration.

Now, looking at **"Registration,"** each student can make multiple registrations, with each registration being associated with, at most, one course instance. Therefore, to identify each registration, it is sufficient to use the combination of the student's ID and the course instance ID.

3 Foreign keys and Multiplicity

Regarding the multiplicities of the relationships, I decided not to assume too much to avoid the risk of adding incorrect information to the diagram. Nevertheless, I did make some that I thought made sense given the problem domain.

3.1 oversees: Senior teacher - Programme

I assumed that a program must necessarily be supervised by a senior professor, however, a senior professor may or may not hold the position of director in one or more programs. This is because it is not clear whether a senior professor, when taking on the role of director, is restricted to supervising only one program.

3.2 owns: Programme - Course

I chose to establish that a particular course must necessarily be linked to a program. Additionally, I assumed that a course belongs exclusively to one program, in order to simplify, although in reality this may not be true. As for the minimum number of courses in a program, I chose not to make any assumptions, thus allowing the diagram to represent programs without any courses. Although this may seem nonsensical in certain contexts, I personally consider it the most coherent approach.

3.3 includedIn: Course - Programme course

As mentioned earlier, the "programme course" is a weak entity. When deciding that this entity depended on the course, I considered the identification method. Although its existence also depends on the program, the program code alone would be insufficient to identify it. Therefore, I chose to establish its dependency on the course, rather than the program.

3.4 belongsTo: Programme - Programme course

As I mentioned, the "programme course" should also be related to the program in question. However, I chose not to make assumptions about the multiplicities, as the multiplicity on the side of the "programme course" is already implied by its relationship with the course, and the other follows the same logic discussed in "owns."

3.5 given: Course instance - Reported hours

I also assumed that the entity "reported hours" was weak, since it is not logical to talk about the hours a teacher teaches in a specific course without the existence of that course. The "given" relationship emphasizes this property, and to conclude the identification, we need the ID of the teacher in question. One might question why I didn't choose the teachers as the class upon which the entity "reported hours" depended, but I assume it was a matter of organizing the diagram itself and avoiding creating additional classes to illustrate this dependency.

3.6 given: Course instance - Assigned hours

This follows exactly the same logic as the previous relationship.

3.7 register: Student - Registration

Once again, I assumed that the "registration" entity is dependent on the student, as the existence of a registration implies that a particular student has made it.

3.8 workIn: Teacher assistant - Course Instance

I chose to maintain the multiplicities of this relationship without specific semantic information. Personally, I believe it is relevant to establish a minimum number of courses that a teaching assistant must teach (in this case, 1) to be considered as such. This is because, in principle, it is only appropriate to be recognized as a teacher if one is effectively teaching classes in a specific course instance. However, I chose not to highlight this characteristic in my diagram, keeping the many-to-many relationship intact.

3.9 hasPlan: Course Instance - Course planning

This relationship follows the same idea as the weak entity relationships explained earlier.

3.10 hasExecution: Course instance - Course

This relationship follows the same idea as the weak entity relationships explained earlier.

3.11 examines: Senior teacher - Course instance

I would like to emphasize that I assumed that each course must necessarily have a senior professor as an examiner. Personally, I consider it sensible to establish a "only one" restriction on the side of the senior professor in this relationship. However, I chose not to do so, as this restriction is not explicitly mentioned anywhere in the problem description.

4 Design decisions

4.1 Additional attribute in assigned hours and reported hours

Generally, in diagrams of this type, it's a recommended practice to establish direct relationships between different entities through connections. However, in the specific case of the "Assigned Hours" and "Reported Hours" classes, it seems intuitive to establish a connection with the teacher classes instead of just having a "teacherId" attribute. However, since the "teacherID" can appear in two different classes - both in the "Senior Teacher" class and in the "TA" class - this duplication makes the connection between these classes less direct. Therefore, I chose to keep only one additional attribute, which allows us to deal with this relationship in a simpler way and without cluttering our diagram too much.

4.2 Additional attribute in Registration

While it may seem straightforward to associate "Registration" with "Course instance," it's crucial to note that "Registration" is a weak entity that requires the student's ID for its identification. Furthermore, to properly identify a registration, it's necessary to consider not only the student's ID but also the ID of the course instance that the student is attending. This is because a student can enroll in several different courses. However, creating a weak entity that depends on two entities for its identification seems somewhat unnatural. For this reason, I chose to simply add the course instance ID as an additional attribute in the "Registration" entity.

4.3 Not adding a Teacher class:

I considered adding a new class for teachers because the "Reported Hours" and "Assigned Hours" classes refer to a "teacherId," which can be either a Teaching Assistant (TA) or a Senior Teacher. My initial idea was to have a "Teacher" class and then two subclasses ("TA" and "Senior Teacher"). However, a TA is also a student, which could make the diagram confusing by having the TA as a subclass of two classes. Therefore, I chose to add just one new attribute to the "TA" subclass, which would be the "teacherID," and have the TA subclass a regular student.