

**Course on Database Design**

**Carlo Batini**

**University of Milano Bicocca, Italy**

**Part 2 – Entity Relationship model**

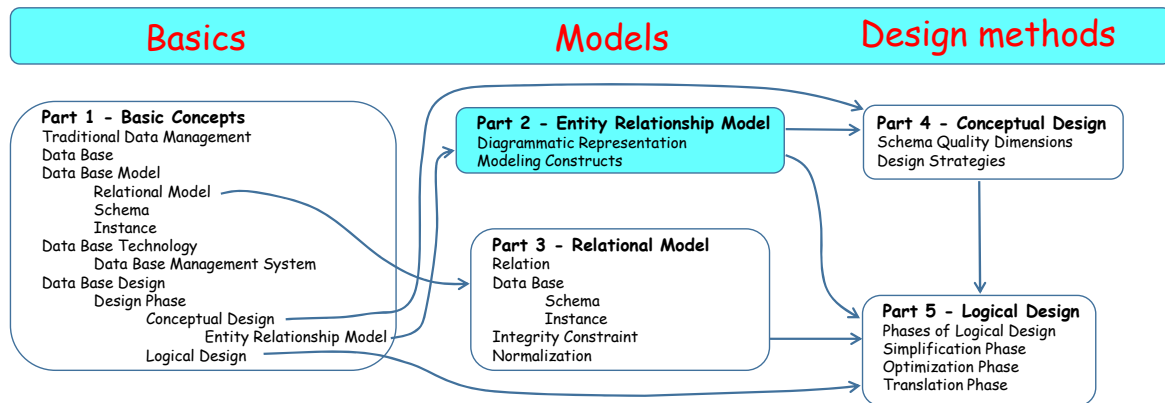


© Carlo Batini, 2015

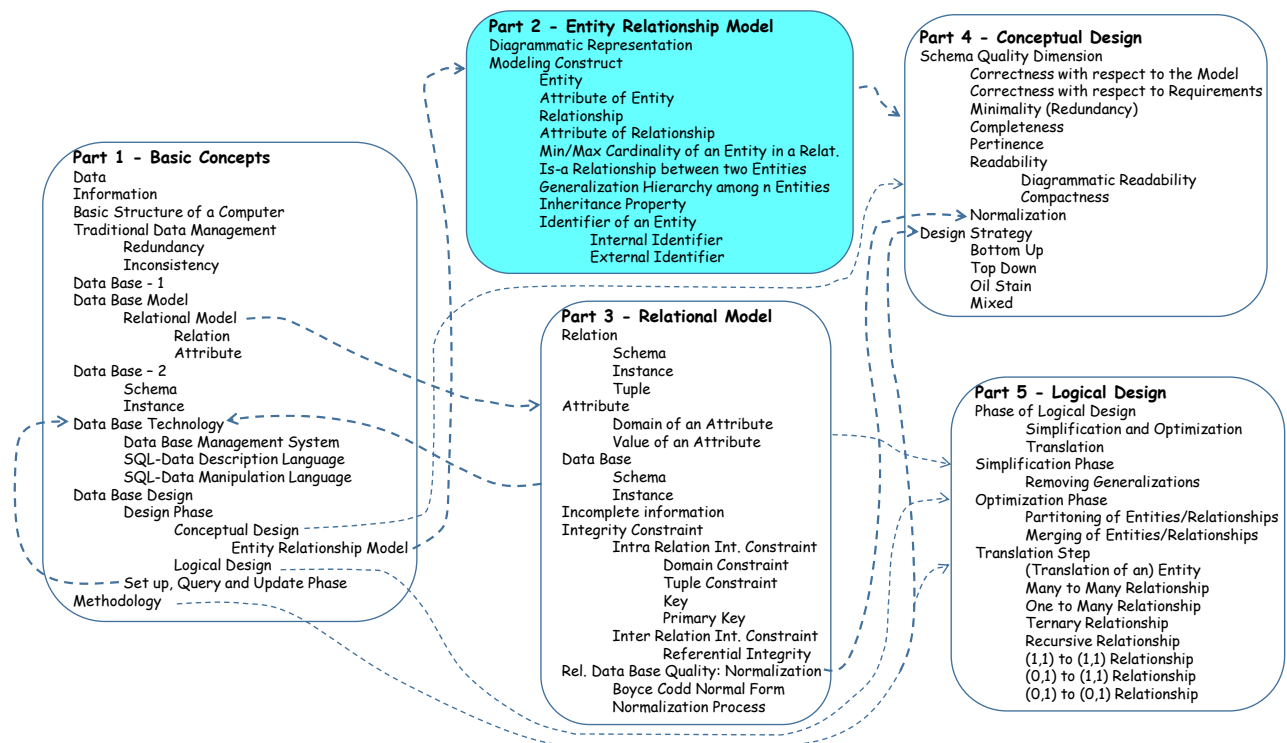
This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

## Part 2 – Lesson 1 - Introduction to the Entity Relationship model

Part 2 of the course focuses on the Entity Relationship Model. You see in these figures at two different levels of detail where we are now in the course, and what follows.

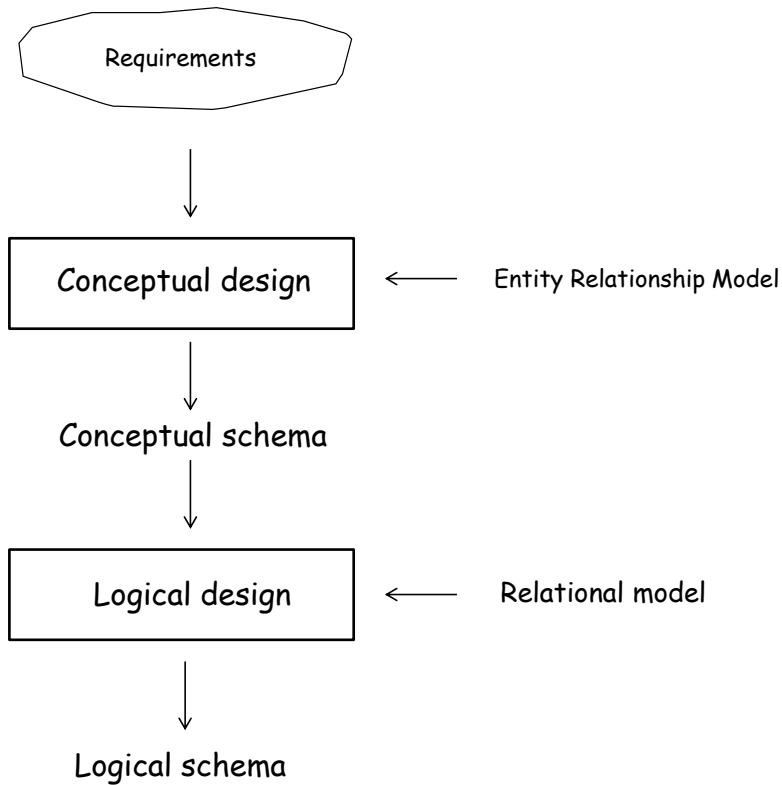


High level conceptual map



Low level conceptual MAP

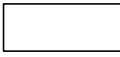

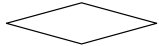
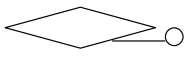

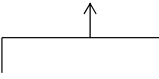

In Part 1 we have seen that the database design activity is divided into two phases.



### Conceptual Design and Logical Design

In this second part of the course we focus on the Entity Relationship model, used in Conceptual design.

I recall that a data model is a set of constructs capable of describing the data requirements of an application. The Entity Relationship model is a conceptual model, namely a model closer to the human being than the relational model. A nice characteristic of the Entity Relationship (ER) model is that its constructs have a diagrammatic representation, that makes the model more readable and easily understandable (see next figure).

Construct of the ER model	Diagrammatic representation
Entity	
Attribute of entity	
Relationship	
Attribute of relationship	
Is-a hierarchy	
Generalization hierarchy	
Identifier	

Diagrammatic representation of the ER Model

## Part 2 – Lesson 2 – Entity and attribute of entity

### Entity

Let us start with an example.

**Exercise 2.1** - Look at these pictures. How do you refer with a collective name to these pictures?



A set of pictures

1. This is a set of persons
2. This is a set of students
3. This is set of Chinese students.

Try again to find a solution and then look at my solution in the following page.

## Discussion on Exercise 2.1

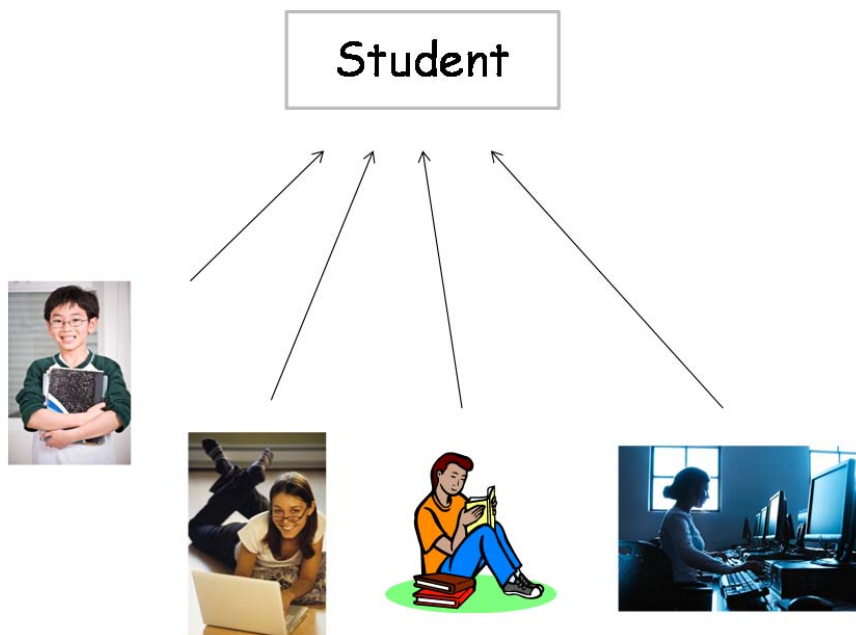
Probably your answer is the second one. The first answer is too generic, while the third answer is wrong, as among pictures, some do not show Chinese people. We say also that this set of pictures represents a class, and we say that such a class is an *entity* in the Entity Relationship model, to which we can assign the name **Student** (see next figure). So, an entity is a class. More precisely.

**Definition** - An *entity* is a class of real world objects that:

1. Are of interest for a database application.
2. Share common properties.
3. Have autonomous existence.

Actually, the above pictures represent real world objects. They correspond to an entity as:

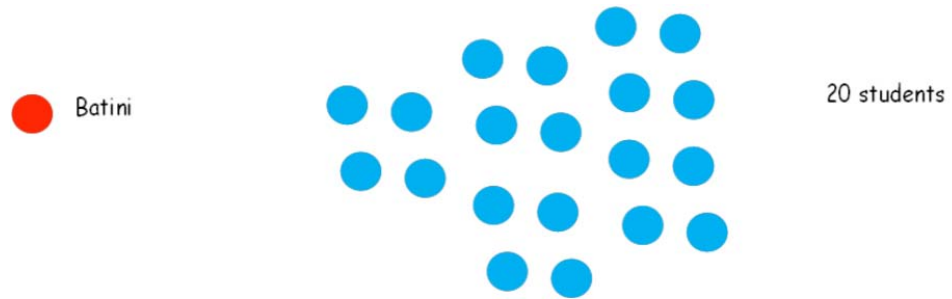
1. in the example we want to use them in a software application that e.g. creates a table of students,
2. they have common properties, e.g. a name and a surname, and
3. they have autonomous existence as all of them are physical persons.



Four student instances and the Entity Student

**Question 2.1** - Assume you are in a classroom, where a professor, say Carlo Batini, and 20 students are present (see the figure). Batini is speaking and the 20 students are listening.

1. How do you represent with an Entity the class made of Batini + the 20 students?
2. How do you represent the class of persons that are listening?



A class with a professor and 20 students

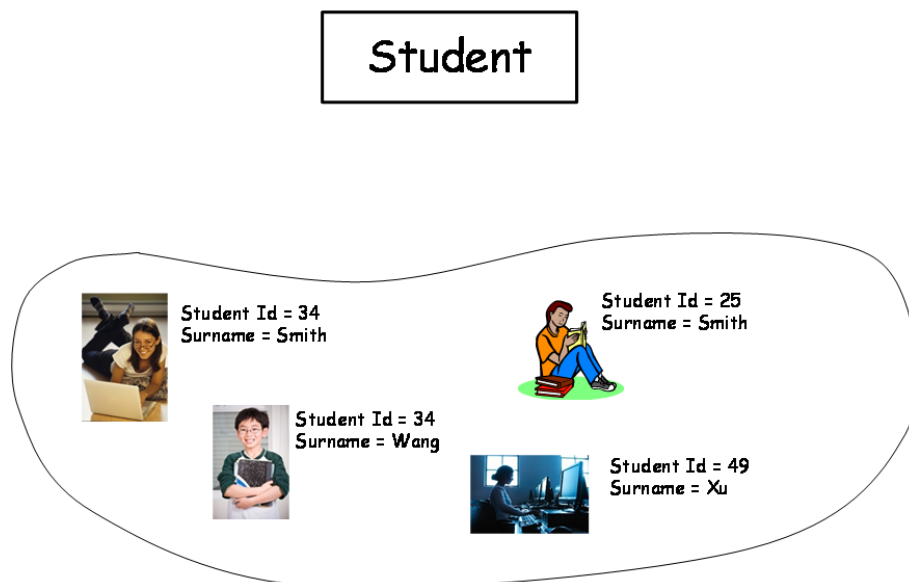


## Discussion on Question 2.1

In the first case we can represent Batini and the 20 students with an entity Person, since being a person is the general property that those individuals have in common; in the second case we can represent the 20 students with an entity Student, since without Batini, they are all students listening to him.

### Attribute of Entity

Let us come back to our previous example. See the following figure, in which we have represented the entity Student using the diagrammatic representation introduced above. We represent also four student instances, each one with two properties associated to them, namely Student Id and Surname.



An Entity and its four instances

Student Id and Surname are examples of *attributes* of the entity Student.

**Definition** - An *attribute* of an entity is an elementary property of all instances of the entity.

### Domain of an Attribute

The values of an attribute range in a **domain**. E.g. for a University that has 20.000 students, the Student Id has as domain [1, ..., 20.000]

## Exercise 2.2

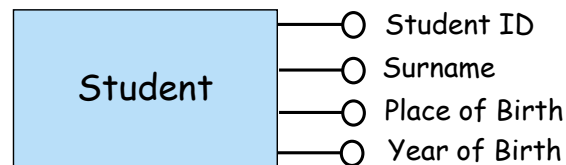
How do you represent the following class of students in the Entity Relationship model?

- Student1 <Id = 41, Surname = Rossi, Place of Birth = Milan, Year of Birth = 1990>
- Student 2 <Id = 37, Surname = Wang, Place of Birth = Harbin, Year of Birth = 1993>
- Student 3 <Id = 53, Surname = Batini, Place of Birth = Rome, Year of Birth = 1987>

A class with four instances of students, four properties, and related values

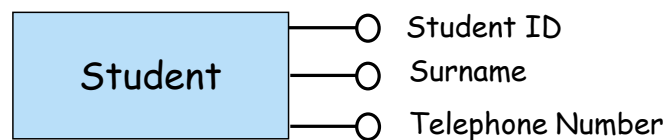
## Answer to Exercise 2.2

We may represent the class with the following Entity and three attributes:



### Minimum and maximum cardinalities of attributes

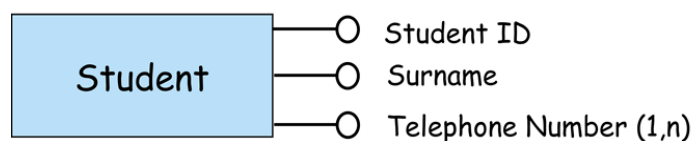
We may add to each attribute in an entity two metadata, that is data on data, called **minimum and maximum cardinalities**. Consider the following schema, made of one entity.



Minimum and maximum cardinalities allow us to represent properties such as the following one:

1. *All students have a telephone, some of them more than one.*

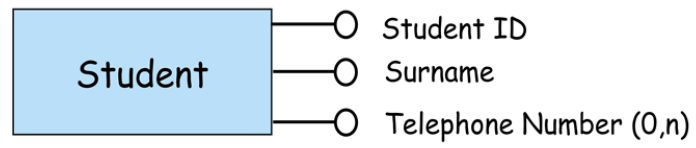
In this case, we have



where 1 in (1,n) represents the property that all students have at least one phone, and n represents the property that some students have more than one (we are not interested to specify which is the maximum value).

2. *Some students do not have a telephone, some of them have more than one.*

In this case we have:

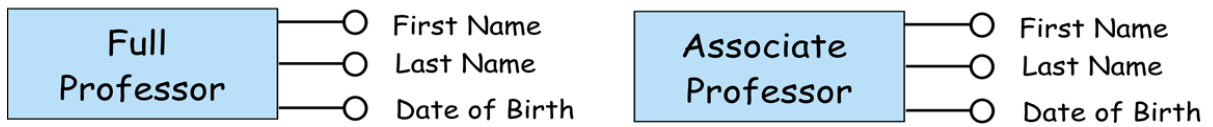


where 0 in (0,n) represents the fact that some student do not possess a phone.

**Exercise 2.3** - Assume again that you are a student in a University. How do you represent Full Professors and Associate Professors and their Last Name, First Name and Date of Birth?

### Solution to Exercise 2.3

A possible schema is shown in the following figure.



Full professors and associate professors

## Part 2 - Lesson 3 - Relationships and attributes of relationships

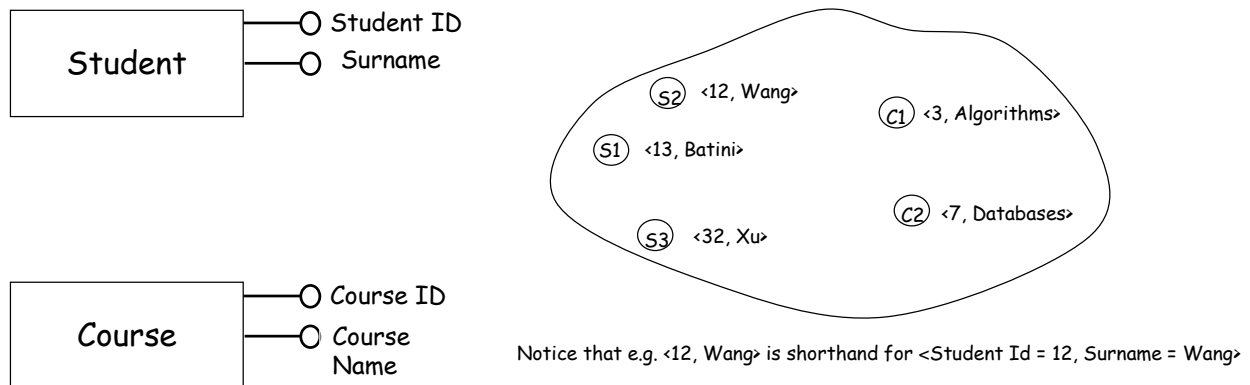
In this lesson, we will introduce other two modeling constructs of the ER model:

1. the relationship and
2. the attribute of relationship.

Again, we start with an example, whose discussion will allow you to understand and conceive yourself these concepts.

### Relationship

Consider two entities Student and Course, and their instances as in the following example.



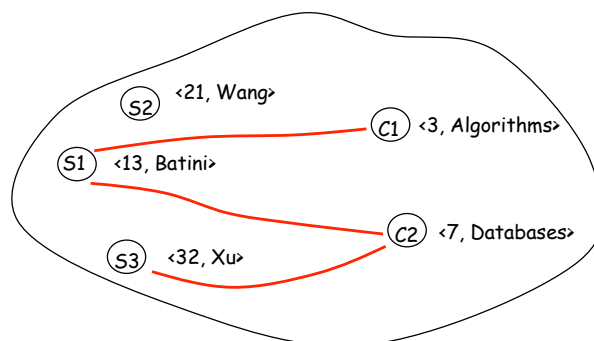
A schema with three student instances and two course instances

We want to represent **in the instance** of the schema the following facts:

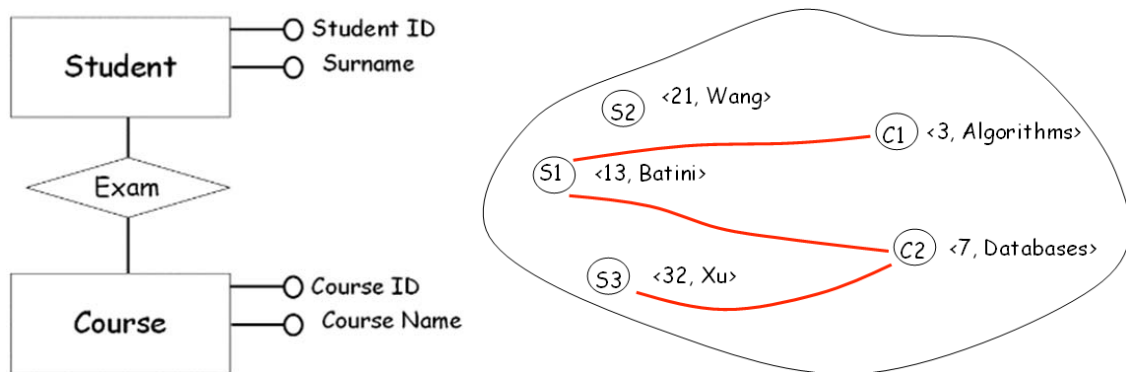
1. Student S1 has passed Course C1 and C2.
2. Student S2 did not pass any Course.
3. Student S3 has passed Course C2.

and we want to represent **in the schema** the corresponding modeling construct.

The new instance is represented in the following, where the above facts are represented with red lines connecting Student and Course instances.



In the new schema we can represent the class made of the three instances of facts relating student and courses, with a new modeling construct called *relationship*, and named Exam.

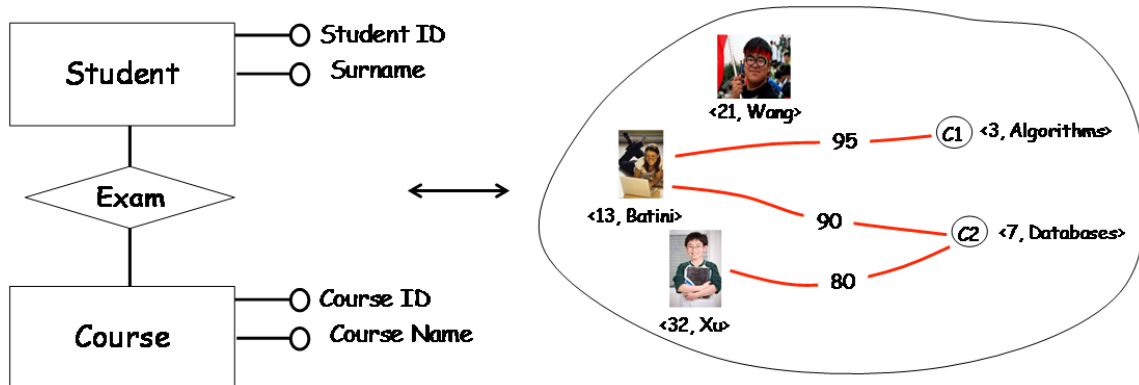


An example of relationship

**Definition** - A *relationship* R between two entities E1 and E2 is a class of instances, each one defined between an instance e1 of entity E1 and an instance E2 of entity E2.

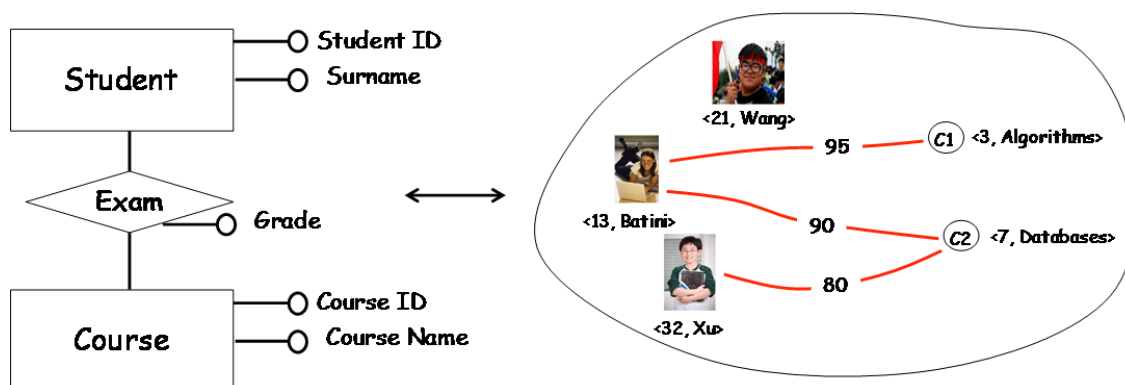
### Attribute of relationship

Now let us introduce attributes of relationships. In the following example (see the figure) you see the two usual entities a. Student and b. Course, respectively with attributes a. Student Id and Surname, and b. Course Id and Course Name, and a relationship Exam defined between them. Looking at the schema instance, we see pictorial instances associated to students, and circles associated to courses. Furthermore, we represent as previously instances of the relationship Exam with lines, that connect pairs of instances of entities. Notice that we have associated to the instances of Exam numerical values (e.g. 80, 90) that represent grades associated to exams, according to Chinese university rules.



ER schema and a corresponding instance with grade values associated to relationship instances

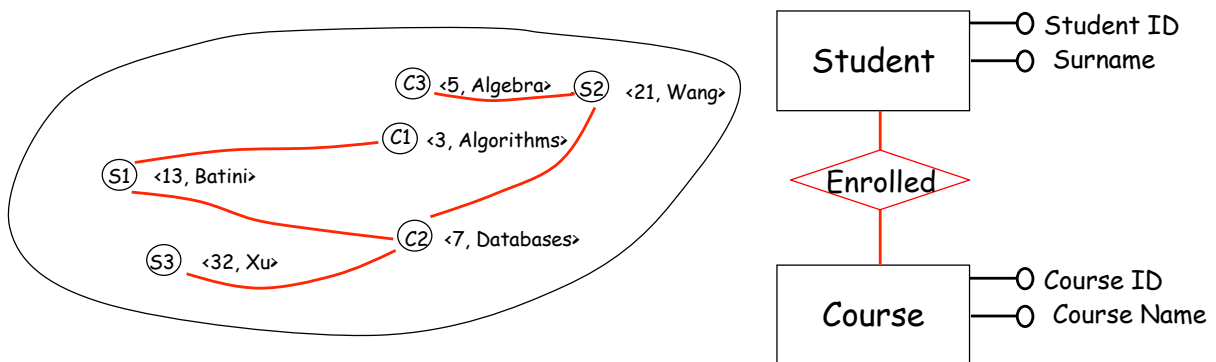
How can we represent such property *Grade* in the ER schema? Well, we can represent Grade in the same way Student Id is associated to the entity Student, namely as an attribute of the relationship Exam (see the following figure).



ER schema with an example of attribute of relationship

Let us see now more complex examples of relationships. Look at the following instance and schema.





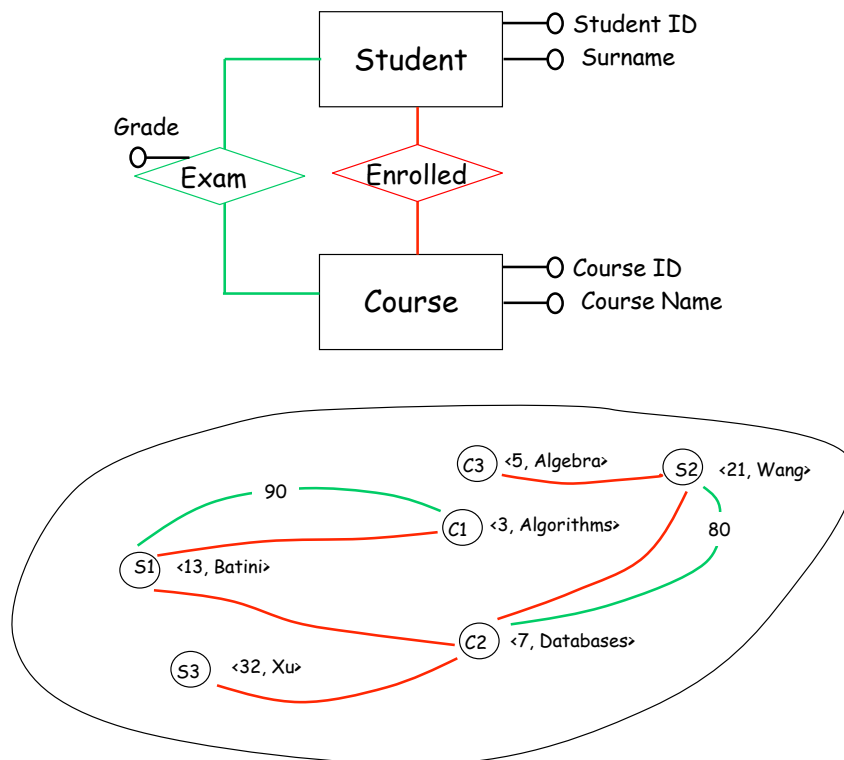
An example of relationship (Enrolled) among two Entities

I have represented in the schema the relationship Enrolled in red, and in the schema instance in red its instances too.

**Question 2.2** - Now assume that I want to represent in the instance, besides courses in which students are enrolled, a different set of relationship instances that refer, as before, to exams that students have passed, with grades. E.g. we may assume that student S1 has passed course C1 with grade 90, and Student S3 has passed course C2 with grades 80. How do you represent the new instances and the corresponding construct in the schema?

## Discussion on Question 2.2

A solution of the exercise is as in the following figure, that represents entities Student and Course with two different relationships Exam (in green) and Enrolled (in red) defined between them.



Two entities with two relationships defined among them and related instances

## Ternary relationship

**Question 2.3** - Assume now that in a University there are:

1. several Departments, each one with an Id, a Name and an Address.
2. several Suppliers of products, each supplier with with an Id, a Name and an Address.
3. several Products, supplied to departments from suppliers, each product with an Id, a Name and a Price.

For instance, supplier Wang supplies pencils to the department with Id = 3 and supplies pens to the department with Id = 5, and supplier Xu supplies DVDs to the department with Id = 4.

For every triple made by: a. a department, b. a supplier and c. a product, we want to represent the quantity of the product supplied by the supplier to the department in year 2014; e.g. in 2014 supplier *Wang* has supplied 300 *pencils* to the department with Id = 3.

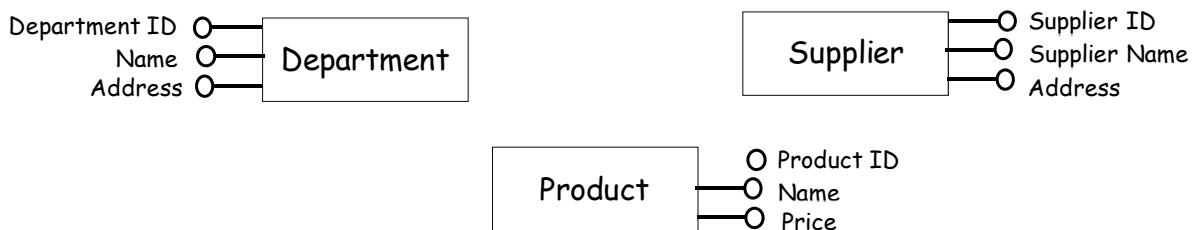
How do you represent such requirements in an ER schema? Which new modeling construct do you need? Are you able to conceive yourselves such modeling construct? Try!

### Discussion on Question 2.3

Let us proceed gradually.

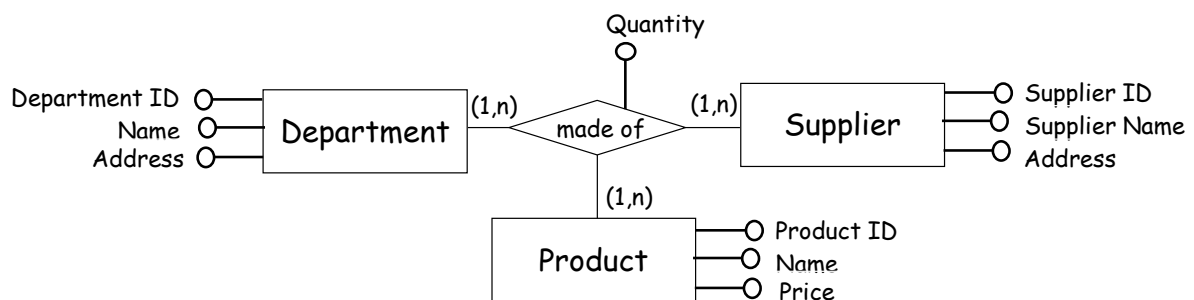
First, how many entities are there in the schema?

The answer is three, Department, Supplier and Product, let us represent the three entities and their attributes.



Now focus on requirements referring to relationship instances, e.g. “supplier Wang supplies pencils to the department with Id = 3 and supplies pens to the department with Id = 5”. Try to conceive the (new) construct that you need in order to represent requirements in the schema.

It is evident that in this case the instances of the relationship connect three entities, Supplier, Department and Product. So, we have discovered relationships between three entities that are called *ternary*, while previous relationships between two entities are called *binary*; see next figure.



An example of ternary relationship

### Recursive relationship

**Question 2.4** - Let us now consider the following requirements, that we represent in a box.

Products sold by a Supplier are made of Parts. Assume that each Part has an Id, a Name and a Price. We want to represent the relationship of Subpart between Parts. So, for each Part we want to represent its (possible) subparts.

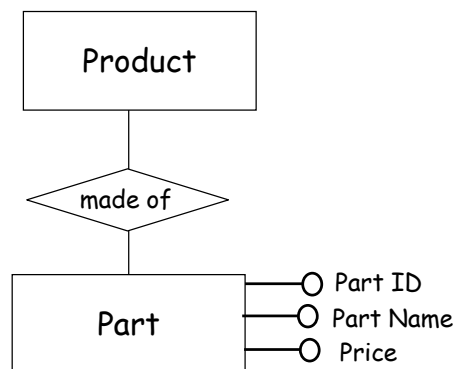
How do you represent them in an ER schema?

### Discussion on Question 2.4

Let us design the schema related to the above requirements with the same process as before.

First discover the entities. How many entities are there in this schema? Try to reply to this question.

There are two entities, Product and Part, or, equivalently, one new entity besides Product.

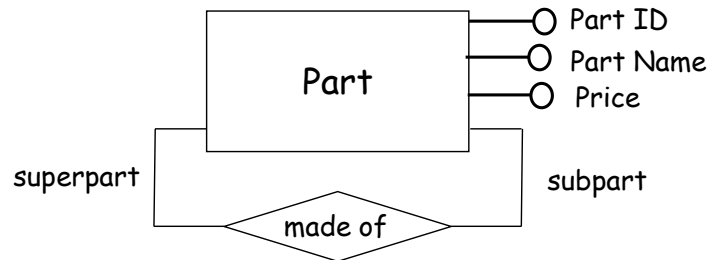


Let us now focus on the entity Part.

**Question 2.5** - How can we represent subparts?

**Clue for Question 2.5** – In order to reply to this question, consider that subparts relate two part instances. So, which is the conclusion? Try to reply yourselves.

**Answer to Question 2.5** - The conclusion is that the relation between parts and their subparts is represented as a relationship between the entity Part and itself, as in the following. Such a relationship is called recursive, since involves recursively two times the same entity.



An example of recursive relationship

Notice that a binary relationship connects always two entities. In this case the two entities are the same. How can we distinguish the first one and the second one? Previously, we did not have this problem, as the entities were different. Now we can represent the first occurrence and the second occurrence of Part associating two tags to the two lines in the relationship; such tags can be named *subpart* and *superpart*. They are also called *roles*.

## Part 2 - Lesson 4 – Exercise on entities and relationships

In this lesson we develop together an exercise in which we apply the concepts seen so far on the ER model.

**Exercise 2.4** - Design a conceptual schema in the Entity Relationship model for the following requirements related to Universities in a given Country.

Professors have an Id and a Date of Birth. Professors work in Universities, and Universities are located in Cities. Universities have a Name and a Dean. Cities have a Name and a Region in which they are located.

I give you a few hints:

1. Find first entities and attributes of entities.
2. Then find relationships and their possible attributes.



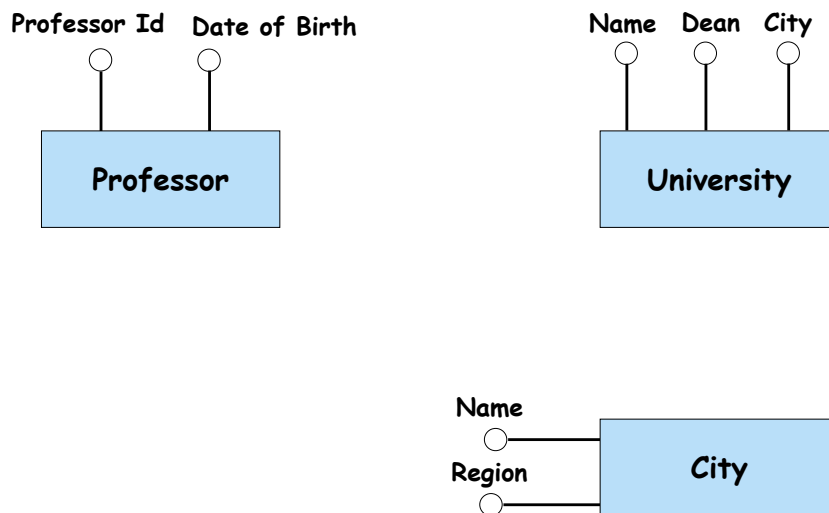
## Discussion on Exercise 2.4

**First step of Exercise 2.4** - Find entities and attributes of entities.

Let us underline in requirements the concepts to be modeled as entities.

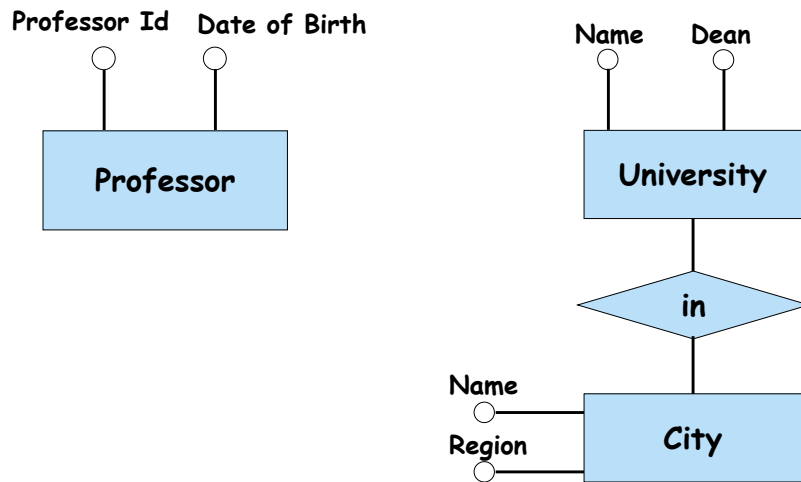
Professors have an Id and a Date of Birth. Professors work in Universities, and Universities are located in Cities. Universities have a Name and a Dean. Cities have a Name and a Region in which they are located.

We choose for them the Entity construct, as all three of them have properties defined in the requirements. Let us identify such properties for University. They are Name, Dean and City where they are located. The properties of City are the Name and the Region. The properties of Professor are Id and Date of Birth. Let us represent such portion of the schema.



First schema with entities and their attributes

Looking at the schema, we see that there is an attribute of University, namely City where Universities are located, that is also an Entity. So we come at the conclusion that the property City is better represented with a relationship *located in* between University and City. We accordingly modify the schema.



Second schema with improved representation of cities

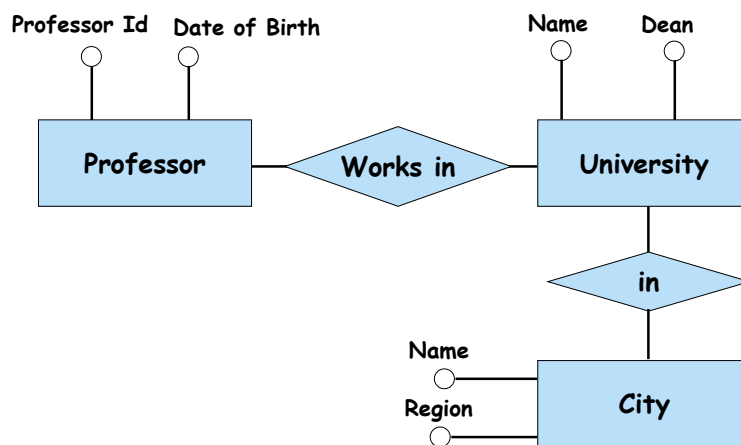
Notice that it was not a mistake to represent City as an attribute of University, but, in the Entity Relationship model, as the name of the model says, we are asked to represent:

1. *objects* of the world in terms of *entities*, and
2. *facts* connecting objects of the world in term of *relationships*.

So, as City is present in the schema as an entity, the property City of universities is better represented in terms of a relationship with the entity City.

**Second step of Exercise 2.4** - Find entities and possible attributes of entities.

We have found a first relationship in the previous step. A second relationship is related to the statement "*Professors work in Universities*". Such statement can be represented in the schema by means of a relationship between entities Professor and University.



## Schema with relationships

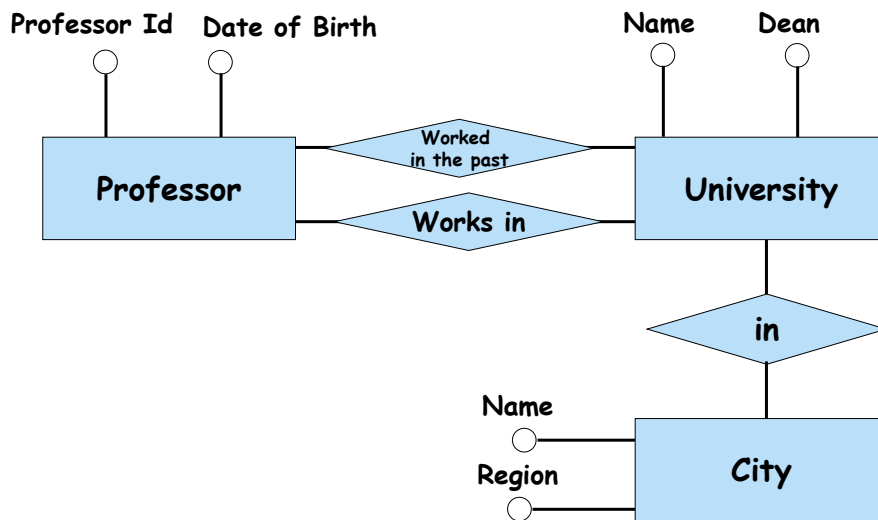
The final schema does not have attributes for relationships.

**Question 2.6** - Let us enrich the previous schema with the following new requirements.

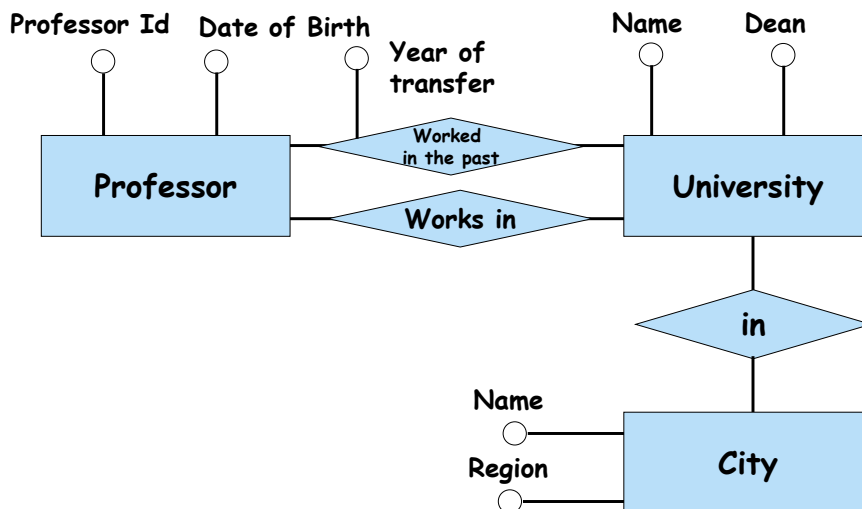
Some professors have worked in the past in other Universities. Represent for universities where they have worked in the past and the year they transferred to another University.

How do you represent such requirements in the schema? Please think yourselves trying to find a solution to this problem.

**Discussion on Question 2.6** - Let us see. The property that professors have worked in the past in other universities relates professors and universities, so it can be represented with another relationship between Professor and University. The name of the relationship can be *In the past* or *Worked in the past*.



The requirement related to the year of transfer is a property of each pair of instances <Professor, University where worked in the past>, so it can be represented as an attribute of the new relationship *Worked in the past*.



The final schema with entities, relationships and their attributes

## Part 2 - Lesson 5 – Is-a hierarchies and Generalizations

In this lesson we will study is-a hierarchies and generalizations.

### Is-a hierarchy

To denote In our life an object we frequently have to choose the more suitable name for such object among a set of names at different levels of abstraction. E.g. if we see the following two trees, and we want to indicate both of them to a friend, we can say:

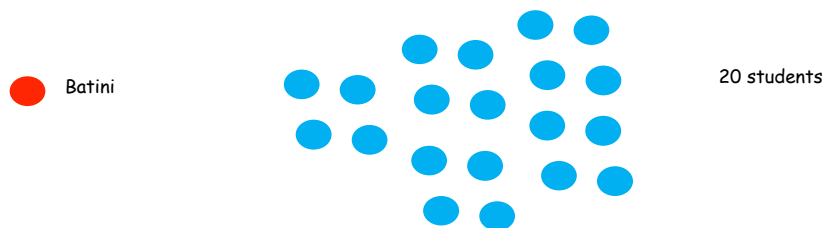


Two trees

- Look at those two trees.
- Look at those two pines.
- Look at that those roman pine and japanese red pine

depending on our knowledge in botanics.

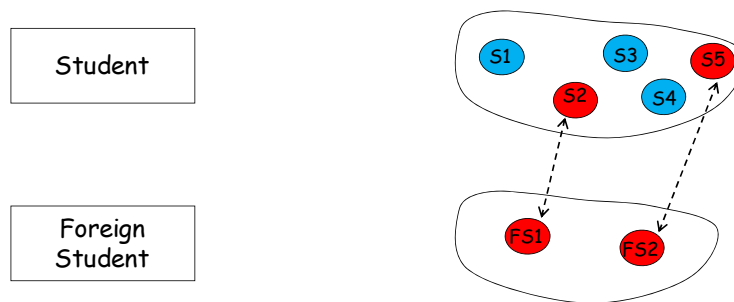
The same happens for classes of objects. In a previous lesson we made the example of Batini and the 20 students.



A class made of a professor and a set of students

When assigning a name to the class made by Batini and the twenty students we choose the term *Person*, while when we had to assign a name to the class of the 20 students, we adopted the term *Student*. We also say in the following that the instance corresponding to the entity *Student* is a subclass (or subset) of the instance corresponding to the entity *Person*.

As a last example consider the following two entities and the corresponding instances. We have colored in red Students that are also Foreign Students.

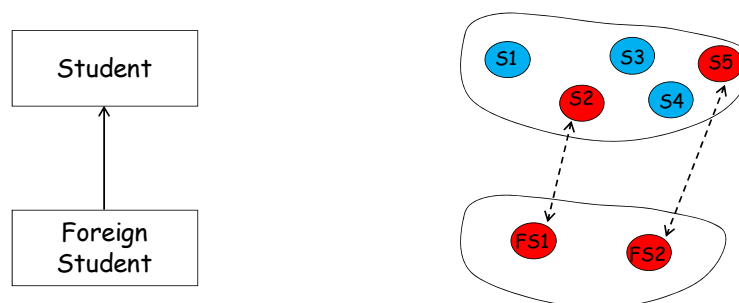


Two entities and corresponding instances:  
the instance of Foreign Student is a subset of the instance of Student

We say that the instance of Foreign Student is a *subclass* of the instance of Student. We also say that entity Foreign Student is-a entity Student, ore more synthetically Foreign Student is-a Student.

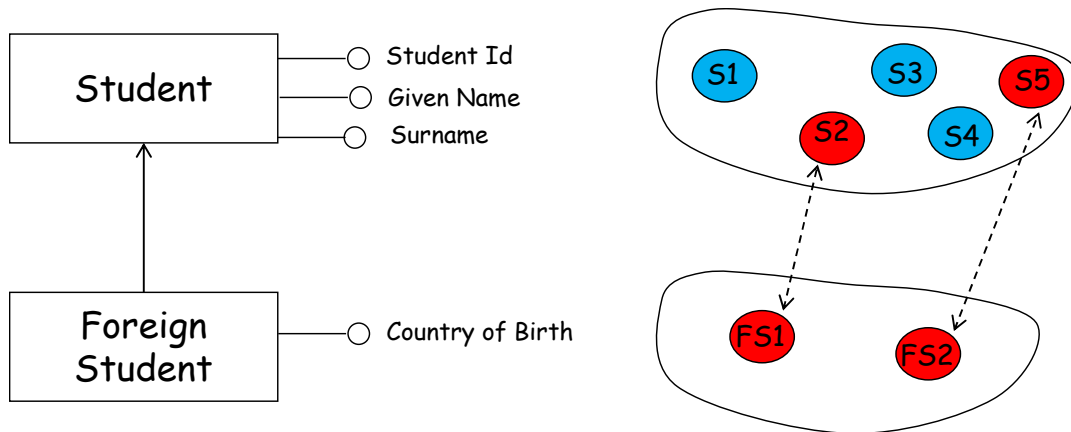
**Definition** - Given two entities E1 and E2, we say that E2 is-a E1 if all instances of E2 are also instances of E1. The construct is also called is-a hierarchy. E1 is said the *parent entity* and E2 isw said the *child entity*.

An is-a hierarchy is represented with an arrow outcoming from the child entity and pointing to the parent entity (see the figure).



An is-a hierarchy between Foreign Student and Student

An important property holds for is-a hierarchies, the inheritance property. Consider the following schema:



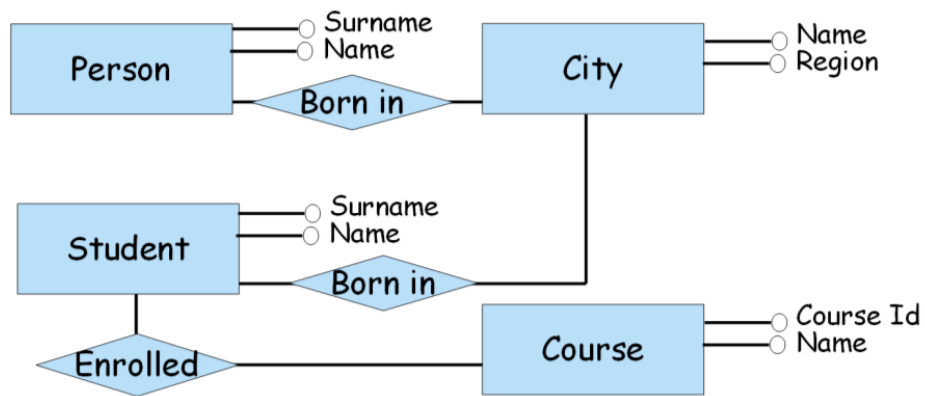
An example of is-a between two entities and the corresponding set of instances

As all the instances of Foreign Students are also instances of Student, all properties of Student (so, in this case, the attributes Student Id, Given Name and Surname) are also properties of Foreign Student. In the schema, there is a property of Foreign Student, the Country of Birth, that is not a property of Student, as it is only defined for the two student instances FS1 and FS2 that correspond to S2 and S5.

When two entities E1 and E2 are such that E2 is-a E1, the following property, called *inheritance property*, holds between E1 and E2: "all properties (namely, all attributes and relationships) of E1 are also properties of E2". The inverse property does not hold.

Notice that we have used the term property, that is more general than the term attribute, including also the relationships in which entities are involved. So, if in the previous schema we add e.g. a relationship to Student corresponding to the University in which students are enrolled, such relationship, due to the inheritance property, is also a relationship of Foreign Student.

**Exercise 2.5** - Let us make an exercise on the inheritance property. Look at the following schema.



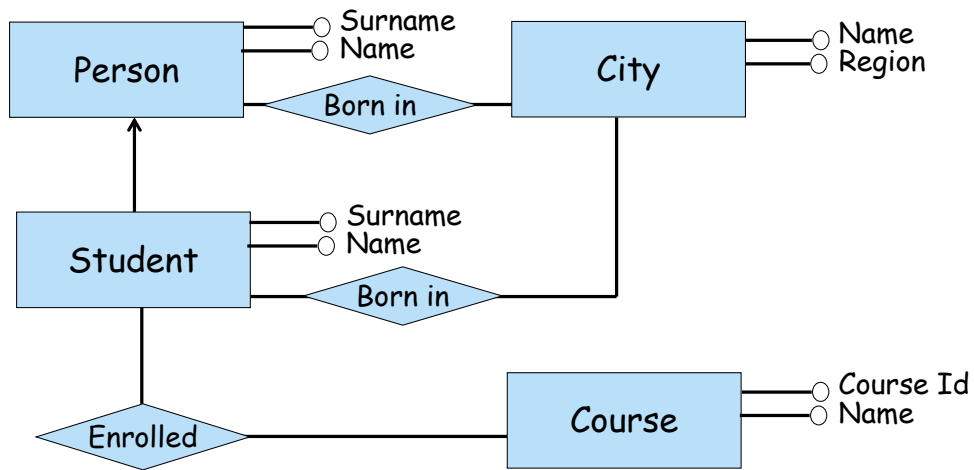
Schema for the exercise 2.10

I ask you to simplify the schema, applying the inheritance property. *Simplify the schema* means dropping from the schema all the properties that can be inferred using the inheritance property, so to have as a result a more compact schema.

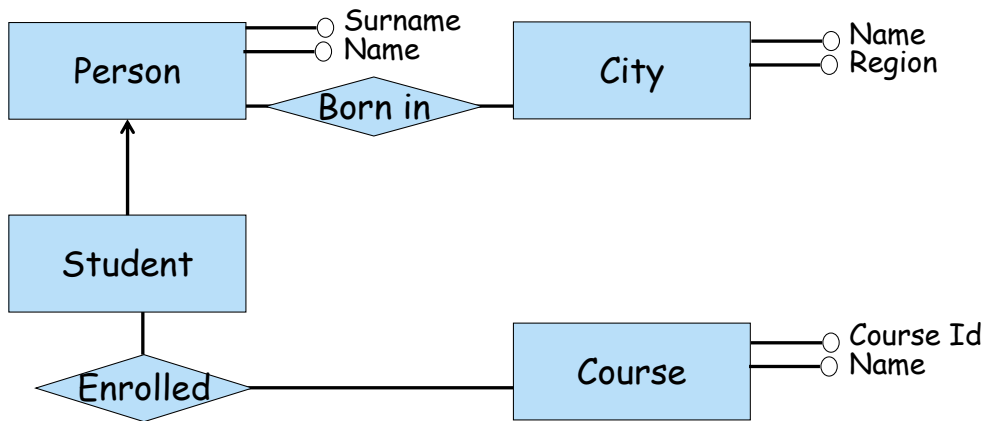


### Solution to Exercise 2.5

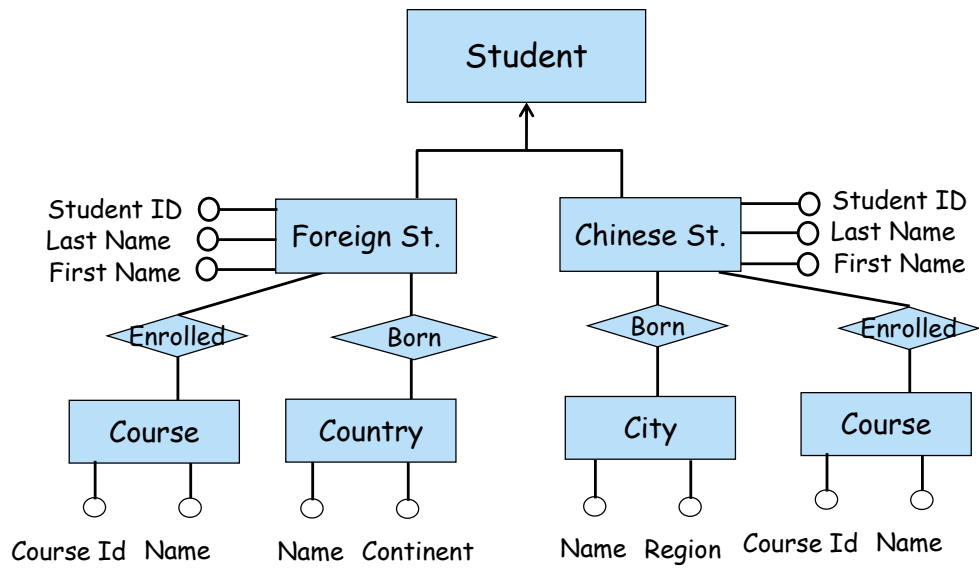
Well, first we add to the schema the is-a hierarchy between Person and Student, obtaining the schema in the following figure.



Then we can cancel from Student attributes Surname and Name. We didn't finish our job: we can also cancel the relationship Born in from Student. I hope you have guessed right!



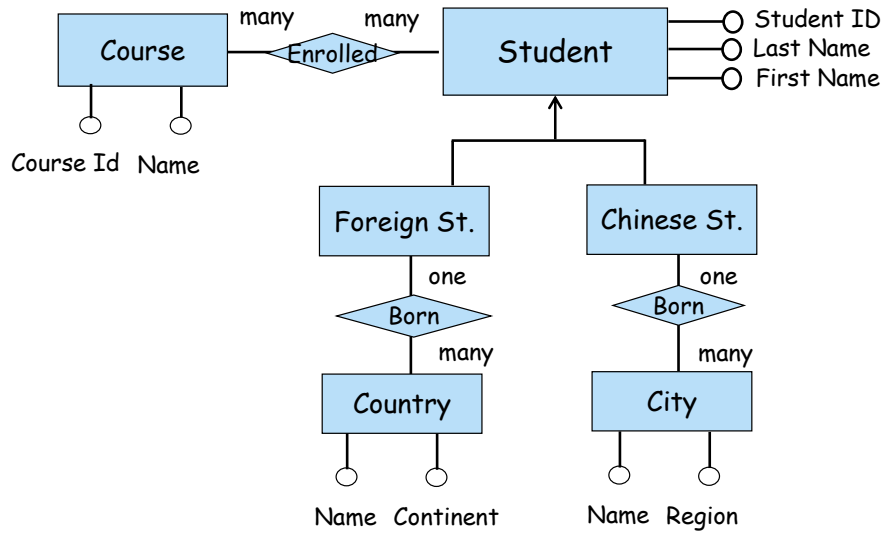
**Exercise 2.6** - As another exercise consider the schema



and apply the inheritance property, leading to a simplified and more compact schema.

## Solution to Exercise 2.6

The new schema is the following.

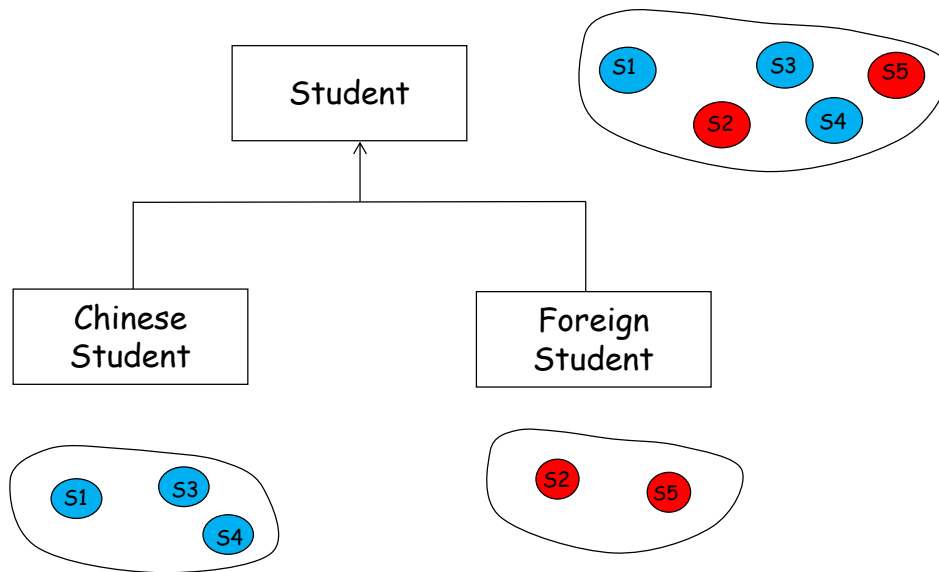


## Generalization

We move now to generalization hierarchies, or, more simply, generalizations. While is-a hierarchies are defined between one parent entity and one child entity, generalizations are defined between one parent entity and more than one child entities. Generalizations, as is-a hierarchies, respect the inheritance property.

More formally, when an entity E and n entities E1, E2, ..., En are that the instances of each entity Ei are a subset of the instances of E, we say that E is a generalization of E1, E2, ..., En. E is said the *parent* entity and E1, E2, ..., En the *child* entities.

The following example shows how we represent generalizations graphically.



Graphical representation of a generalization among entities (with corresponding instances)

You see three entities and their instances. The three entities form a generalization, since the instances of Chinese Student and Foreign Student are subsets of the instances of Student.

With the above definition of generalization we have concluded the lesson.

## Lesson 2.6 - Exercise 2.12 - Design a not-so-simple ER schema

### Exercise 2.7 - Requirements of Exercise 2.7 - first part

Students of a University attend Courses, and pass them. Courses have an ID, a Name and a Year of enrollment. Students have an ID, a Last Name and a First Name. An Exam passed by a Student has a grade and a date.

Courses are taught by Professors. Professors have a Last Name and a First Name. They may be Associate Professors or Full Professors, only for Full Professors we are interested in the City of Birth, with Name and Region. Every Professor belongs to a Department. Departments have a Name and an Address.

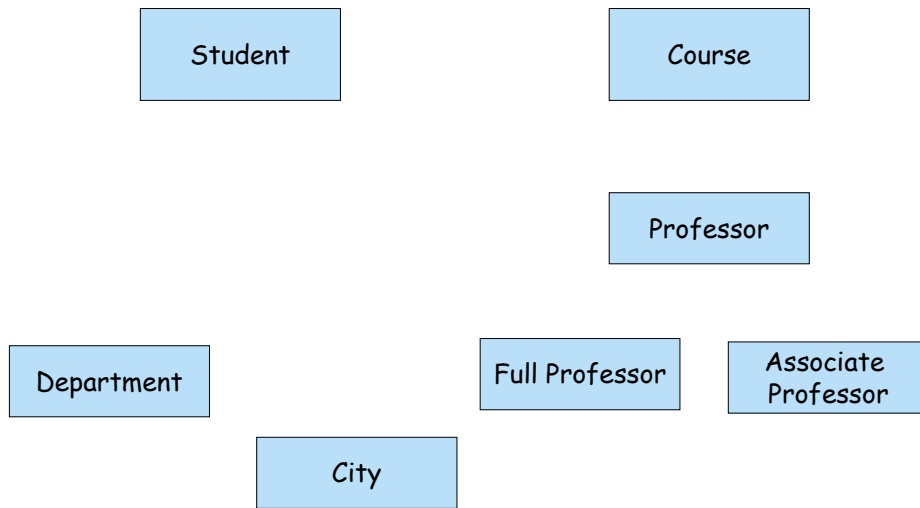
Find and represent entities, relationships, attributes of entities and of relationships, Is-a relationships and generalizations.

#### Hint

1. Find first entities and attributes of entities.
2. Then find Is-a relationships and generalizations.
3. Then find relationships and their possible attributes.
4. Check the schema according to the inheritance property.

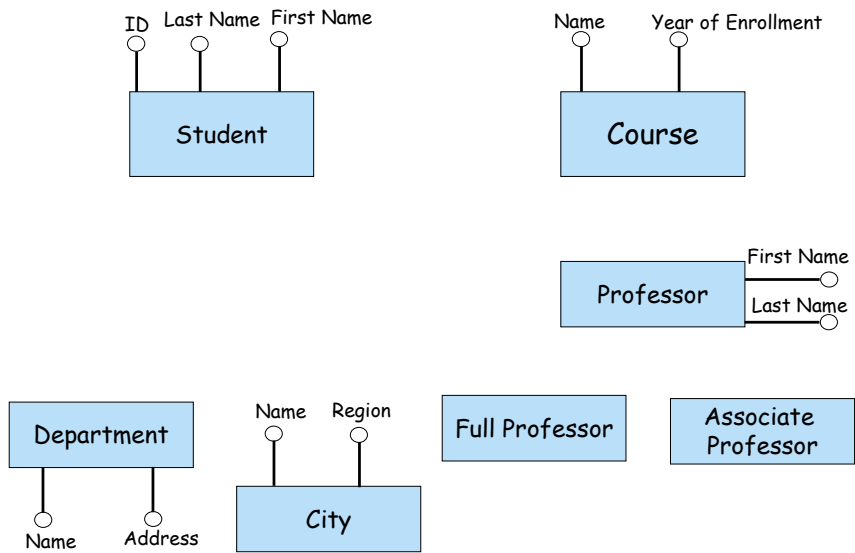
**Discussion on Exercise 2.7 - first part**

**Exercise 2.7 - first step**



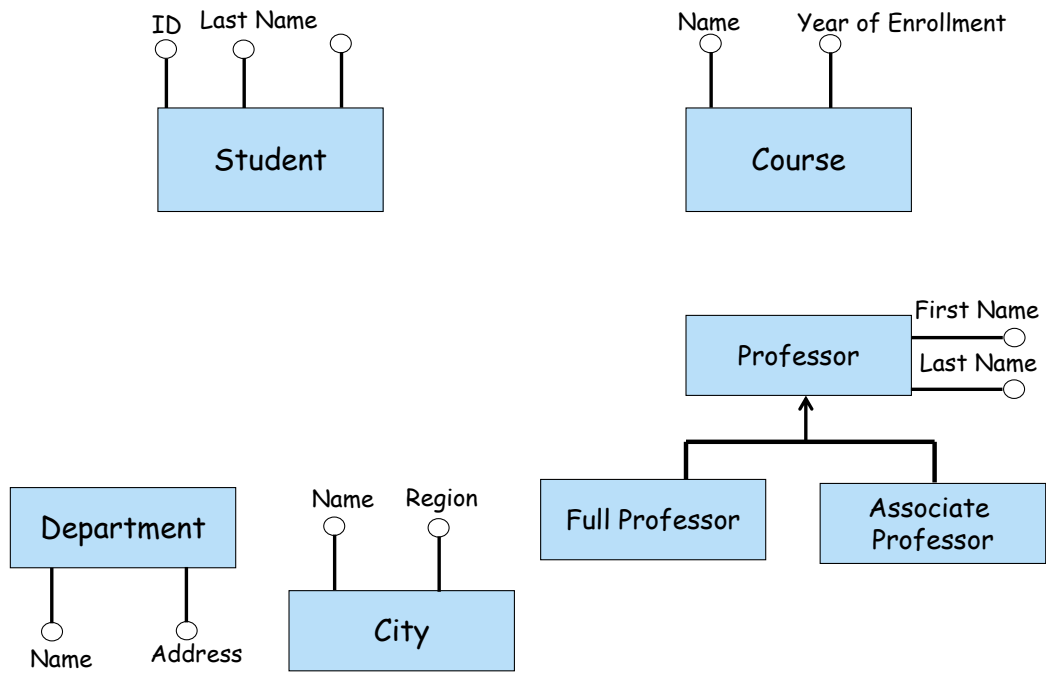
Entities

**Exercise 2.7 – second step**



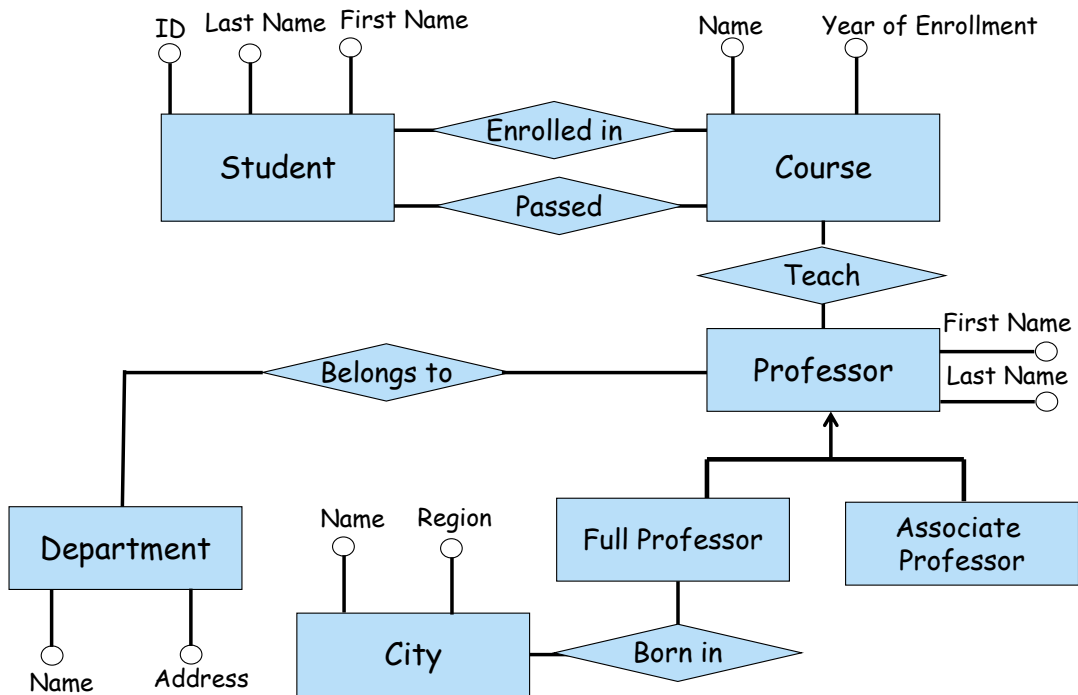
Attributes of Entities

**Exercise 2.7 - Third step**



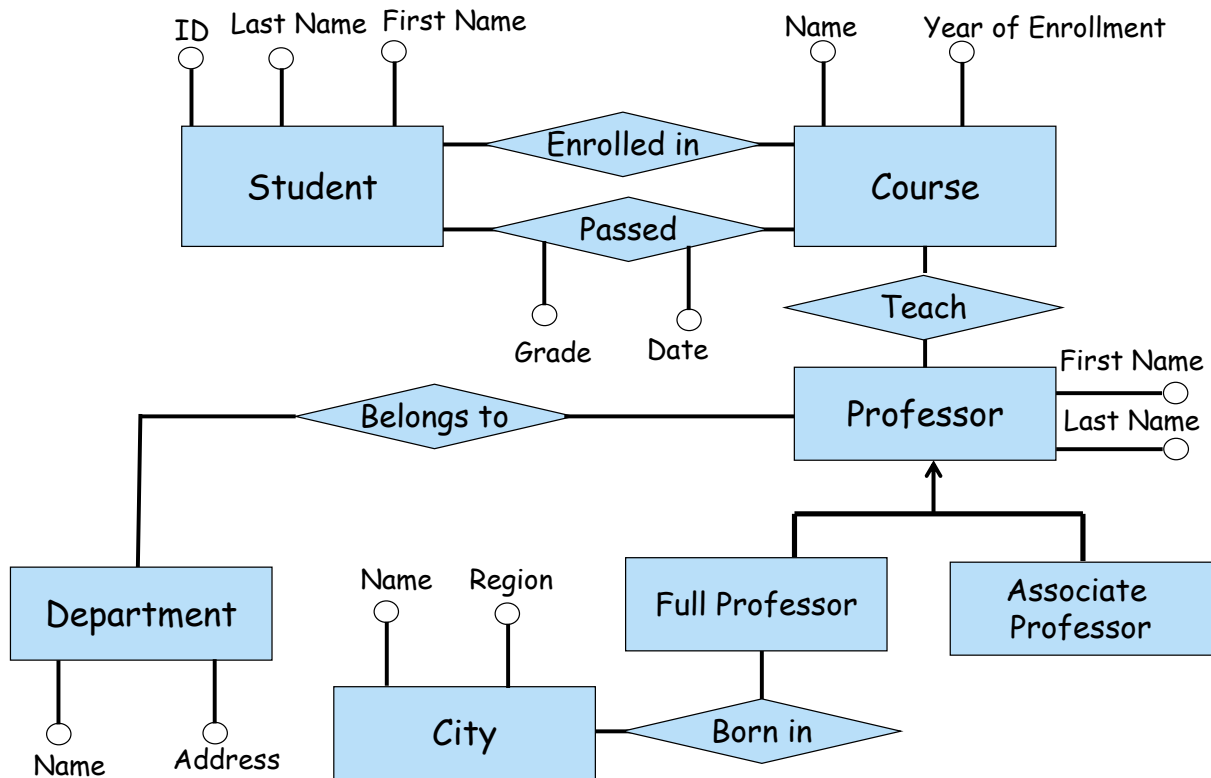
Is-a hierarchies and Generalizations

**Exercise 2.7 - Fourth step**



## Relationships

### Exercise 2.7 - Fifth step



## Attributes of relationships

### Exercise 2.7 - Check the schema according to the inheritance property

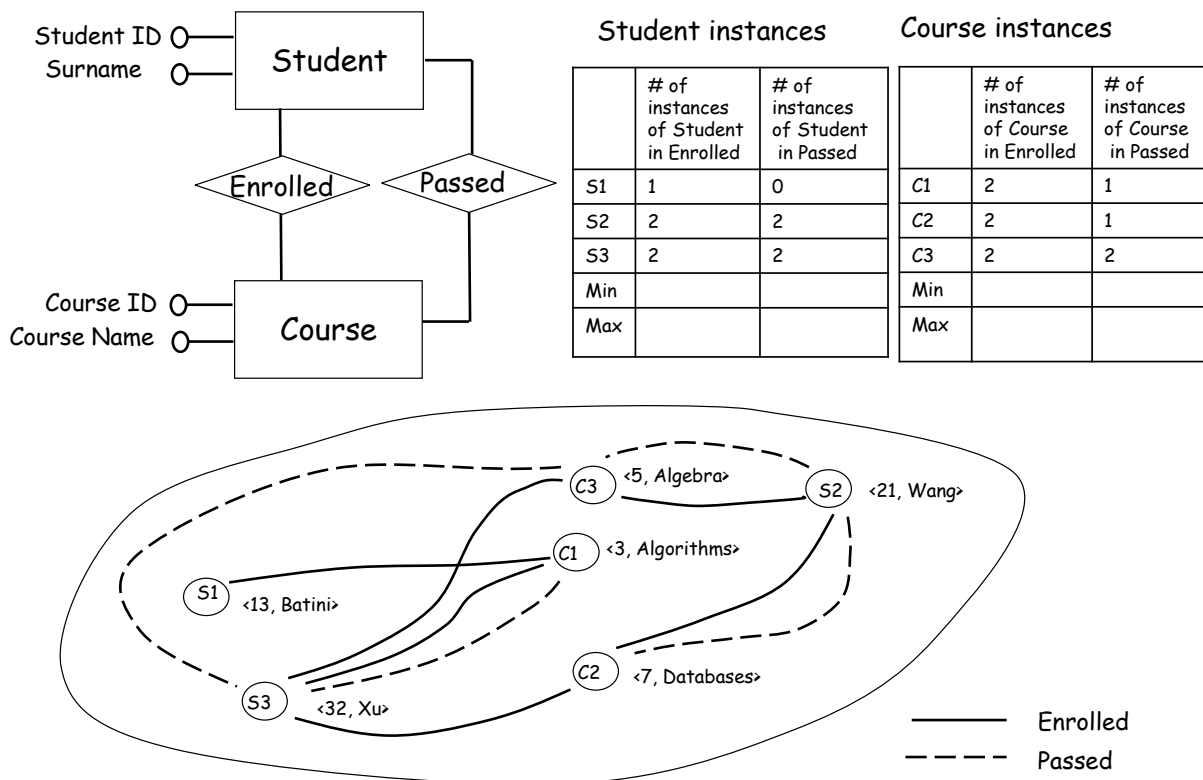
The schema is OK, since the attributes are OK and relationships have been defined after the unique generalization present in the schema, and so the unique relationship related to the Professor generalization has been correctly associated to Full Professor.



## Part 2 – Lesson 7 - Minimum and maximum cardinalities of entities in relationships

We start as always with a motivating example. In the following figure we see:

1. A schema with two entities Student and Course and two relationships defined among them, Enrolled and Passed, with the usual meaning.
2. An instance of the schema with three students and three courses.
3. Two tables called Student instances and Course instances. In the table Student instances we have prefilled for each instance  $S_i$  of Student the number of instances of enrolled and Passed in which  $S_i$  is involved. E.g.  $S_1$  is involved in 2 instances of Enrolled and 0 instances of Passed.
4. For table Course instances we have computed the same values, namely for each course instance the number of instances of Enrolled and Passed in which they appear.



An ER schema, an instance of the schema, and a figure on Student and Course instances involved in relationships

Now we calculate for the two sets of instances of Student and of Course the minimum and maximum of the above values.

### Student instances

	# of instances of Student in Enrolled	# of instances of Student in Passed
S1	1	0
S2	2	2
S3	2	2
Min	1	0
Max	2	2

### Course instances

	# of instances of Course in Enrolled	# of instances of Course in Passed
C1	2	1
C2	2	1
C3	2	2
Min	2	1
Max	2	2

**Definition** - The minimum cardinality of an entity E in a relationship R is the minimum number of instances of R in which an instance of E is involved. We write two possible values for minimum cardinality, 0 or 1, where we write 1 when the minimum cardinality is greater than 0. This means that when the minimum cardinality is equal or greater than 1 we say that “it is not 0”.

According to this definition, in the above schema:

- the minimum cardinality of the entity Student in the relationship Enrolled is 1 and in the relationship Passed is 0. We write 1 and 0 accordingly.
- the minimum cardinality of the entity Course in the relationship Enrolled is 2 and in the relationship Passed is 1. We write 1 and 1 in the two cases.

**Definition** - The maximum cardinality of an entity E in a relationship R is the maximum number of instances of R in which an instance of E is involved. We write two possible values for maximum cardinality, 1 or n , and we write is when the maximum cardinality is greater than 1. This means that when the maximum cardinality is greater than 1 we are not interested to express a specific number.

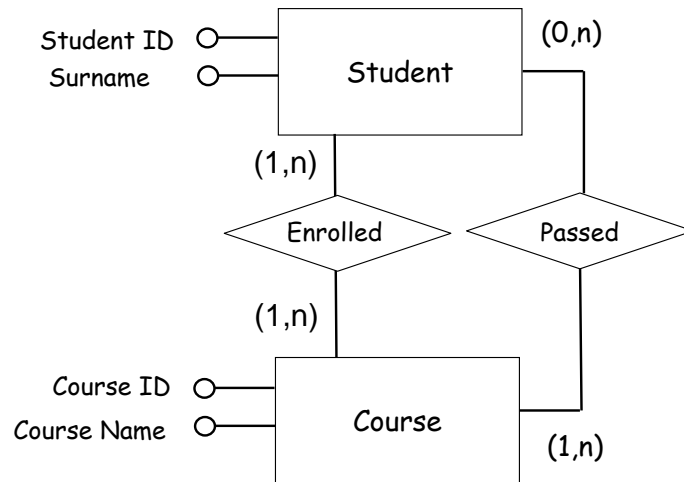
According to this definition, in the schema

- the maximum cardinality of the entity Student in the relationship Enrolled is 2 and in the relationship Passed is 2. Instead of 2 and 2 we write n and n, as specified above.
- the maximum cardinality of the entity Course in the relationship Enrolled is 2 and in the relationship Passed is 2. We write n and n in the two cases.

We represent cardinalities writing

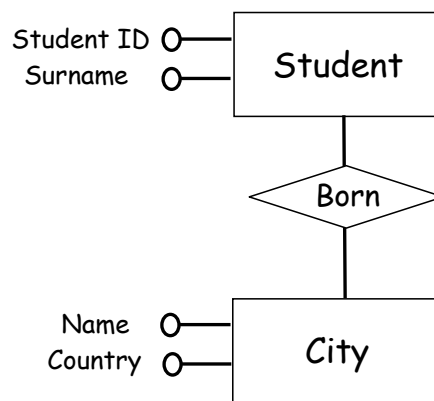
(minumun cardinality, maximum cardinality)
--

close to the corresponding entity. So we have



The schema of the example with minimum and maximum cardinalities

As another example consider the relationship



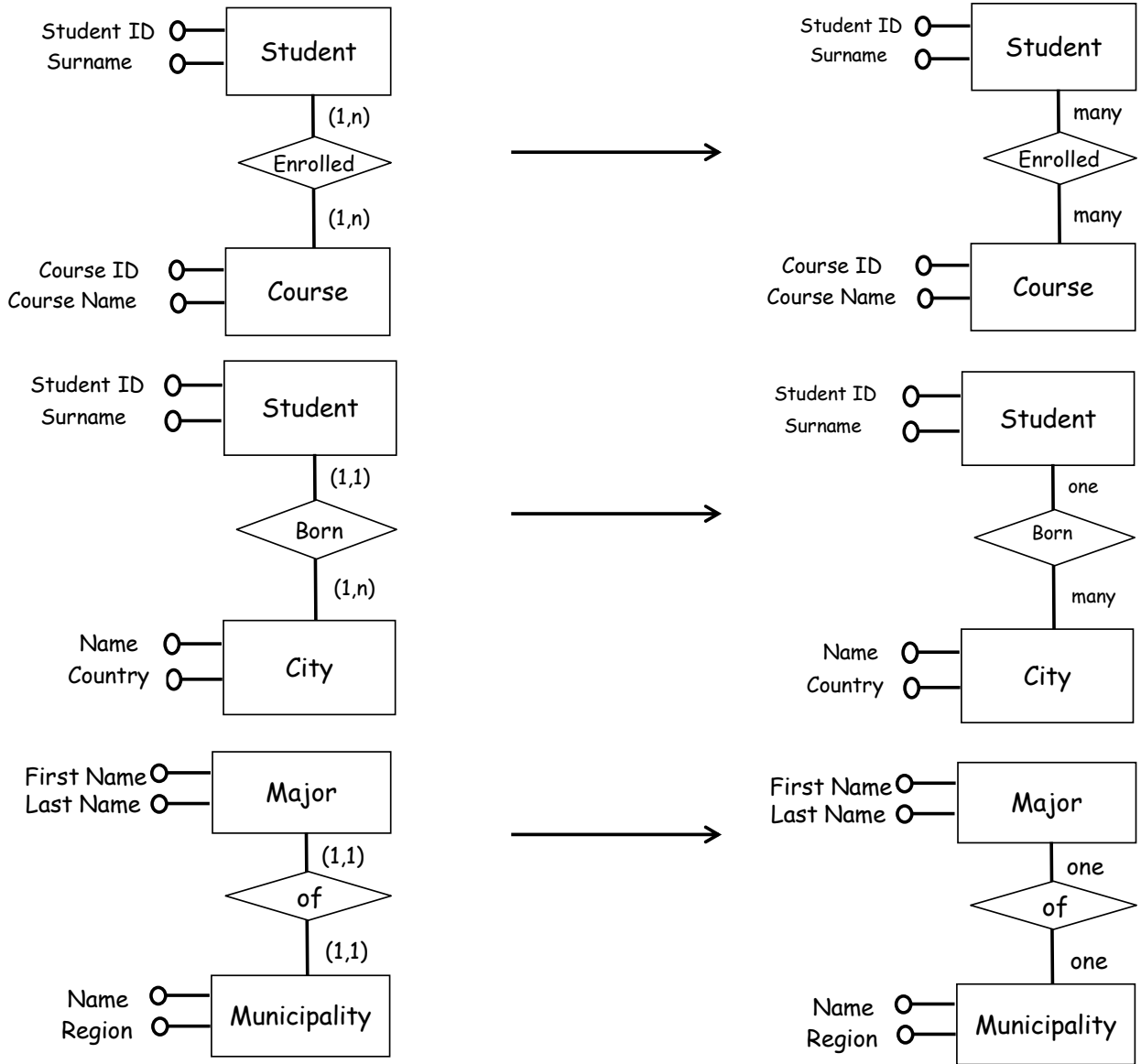
Here we can assign cardinalities without inspecting a generic instance. Student have born in at least (minimum cardinality) one city, and at most (maximum cardinality) in one city. So the cardinalities of Student in Born are (1,1).

If we make the assumption to represent all and only the cities where students are born, in every city at least one student is born, while it may happen in general that in one city several students are born, so cardinalities are (1,n).

### Compact notation for cardinalities

We conclude this lesson on minimum and maximum cardinalities showing a compact notation for them. The compact notation focuses on maximum cardinalities: if the maximum cardinality is one, the compact notation is “one”, if the maximum cardinality is n, the compact notation is “many”.

Let us now consider three typical cases and express them with the compact notation.



Extended notation

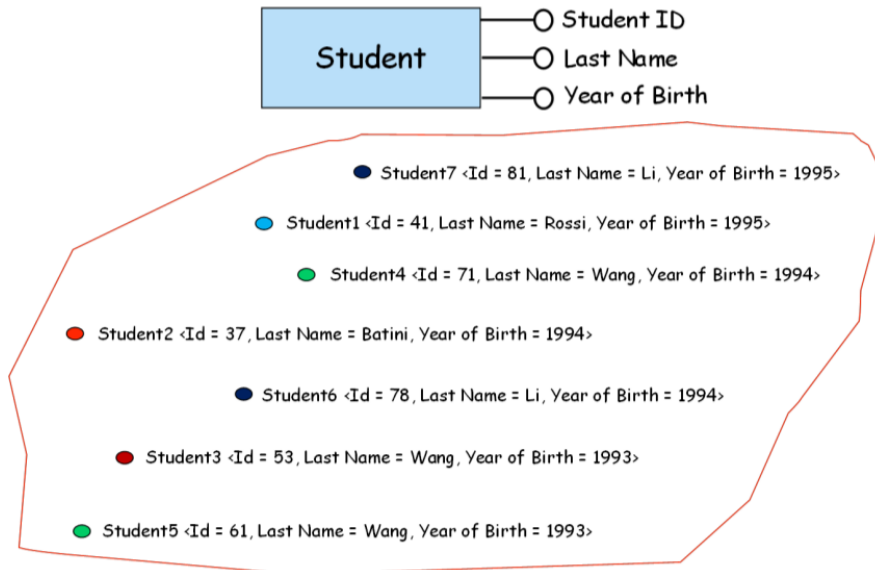
Corresponding compact notation

This is all for this lesson.

## Part 2 – Lesson 8 - Identifiers

### Internal identifier

Look at this entity and the related instance, that represents the students of a Chinese University, e.g. Harbin.



An entity and one instance of the entity

**Question 2.8** - Now reply to the following questions:

1. May two students (in the instance and in general) have the same Last Name?
2. May two students (in the instance and in general) have the same Year of Birth?
3. May two students (in the instance and in general) have the same Student Id?

## Discussion on Question 2.8

The answers are

1. yes, e.g. in China and in the instance there are a lot of Wangs,
2. yes, in a University lots of students are born in same year, and
3. No, a different Id is assigned to each Student.

We say that the attribute *Student Id* is an *identifier* of the entity Student, meaning that given a Student Id value, only one student may have such Id.

More formally.

**Definition** - An *Identifier* of an entity is one or more attributes of the Entity whose values uniquely identify each instance of the Entity.

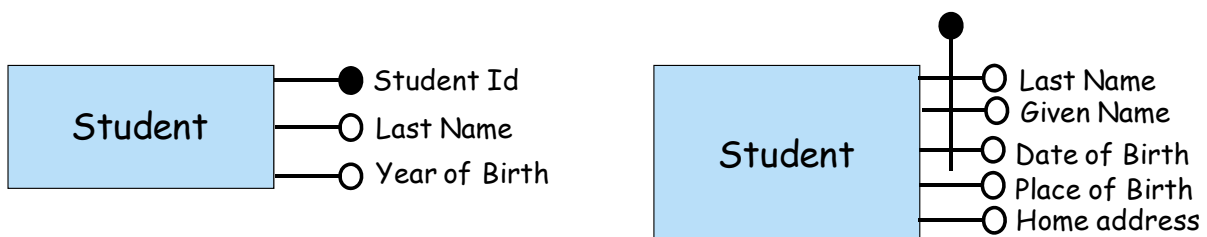
Identifiers are fundamental in our life to uniquely refer to a specific object of the perceived reality, and are fundamental in databases when you want in a query to find some property of a specific student, E.g. the *Last name*, in a table Students (Student Id, Last Name, Year of Birth).

Let us see a second example. Look at the following entity



In this case we do not have a Student Id, so we cannot identify students with one single attribute; we have to use instead several attributes, typically Last Name, Given Name, Date of Birth are enough.

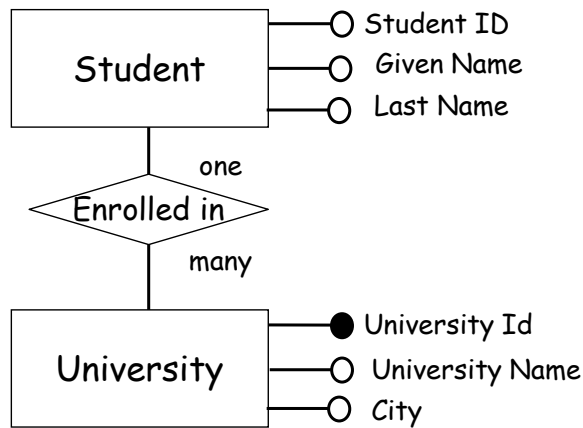
Identifiers composed by one or more attributes of an Entity are said *internal*, as they are composed of properties (namely, attributes) that are internal to the entity. They are represented in the two cases of a. one or b. more than one attributes with the following notation.



Internal identifiers made of one attribute and three attributes

External identifier

Now let us consider this new schema



A schema where the entity Student cannot be identified with an internal identifier

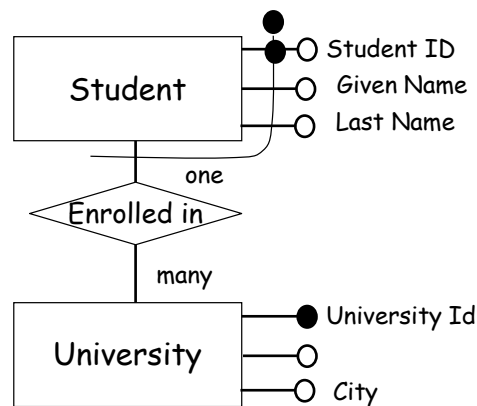
In this case we represent all the students of, say, Chinese universities. It may very well happen that in two different universities there are students with the same Id. We see that in this schema the *Student Id* is not enough to identify students.

**Question 2.9** - Which further information should we add to the Student Id to identify students? Do you have an idea?

### Answer to Exercise 2.9

I hope you said: the University. This is correct, we have to add the University so to distinguish e.g. the Student with Id = 37 of Harbin University, and the Student with Id = 37 of, say, Weihai University.

This type of identifier is called *external*, as, besides attributes, other properties of the entity are part of the identifier, that is one or more of the relationships in which the entity participates, in this case the relationship *Enrolled in*. We represent the identifier as in the following figure.



External identifier

**Exercise 2.7** – second part - Extend the Entity Relationship Schema of Exercise 2.8 with cardinalities of entities in relationships (extended notation and compact notation) and identifiers.

I recall you that the extended notation is (mincard,maxcard) where

- mincard can be 0 or 1
- maxcard can be 1 or n



**Discussion on Exercise 2.7 – second part**

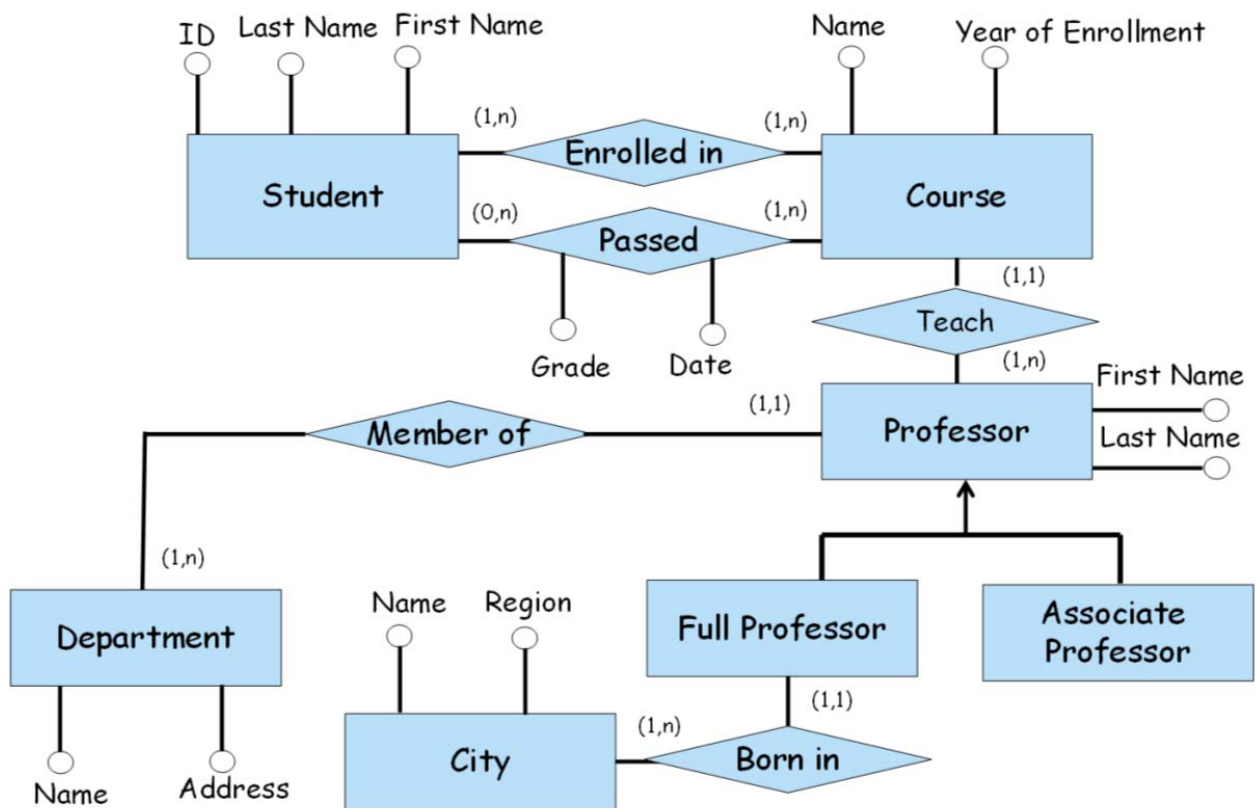
Extended notation for cardinalities

Some hint

Look at the relationship from the side for which you have to assign the extended cardinalities, and then ask yourselves, for instance for the Born in Relationship:

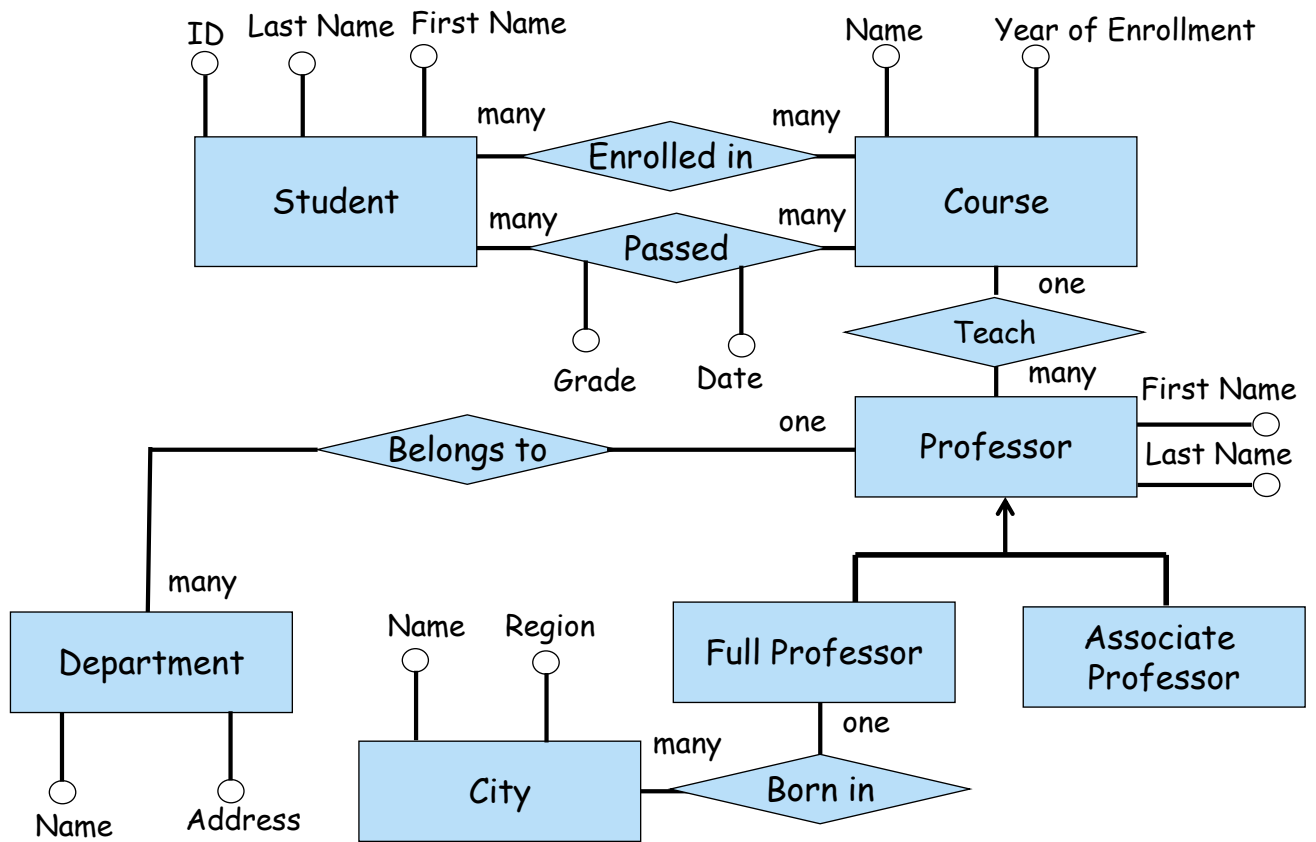
3. in how many Cities can be born at minimum a Professor? Obviously in this case → One!
4. in how many Cities can be born at maximum a Professor? Again, in this case → One!

Complete solution



Schema with compact notation for cardinalities

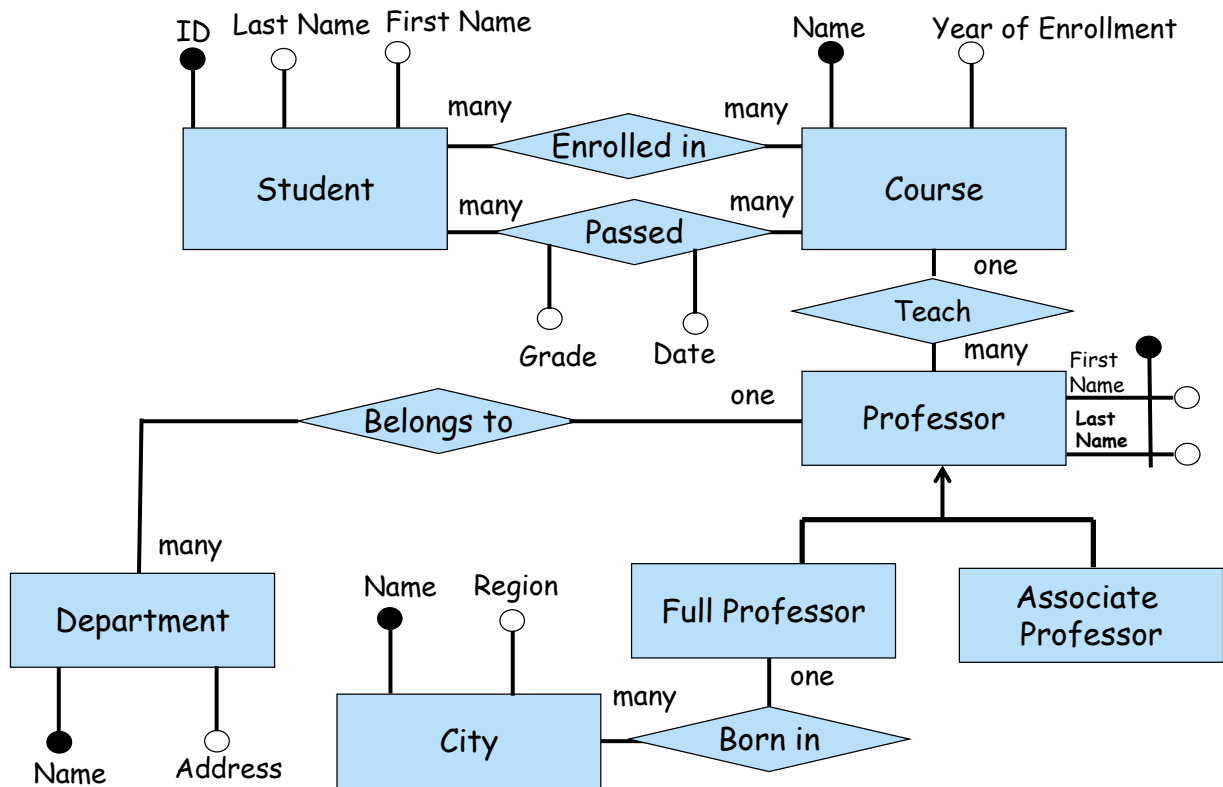
Answer for Compact notation



Schema with compact notation for cardinalities

### Identifiers

Here you have to remember that in general there are two types of identifiers, internal and external.



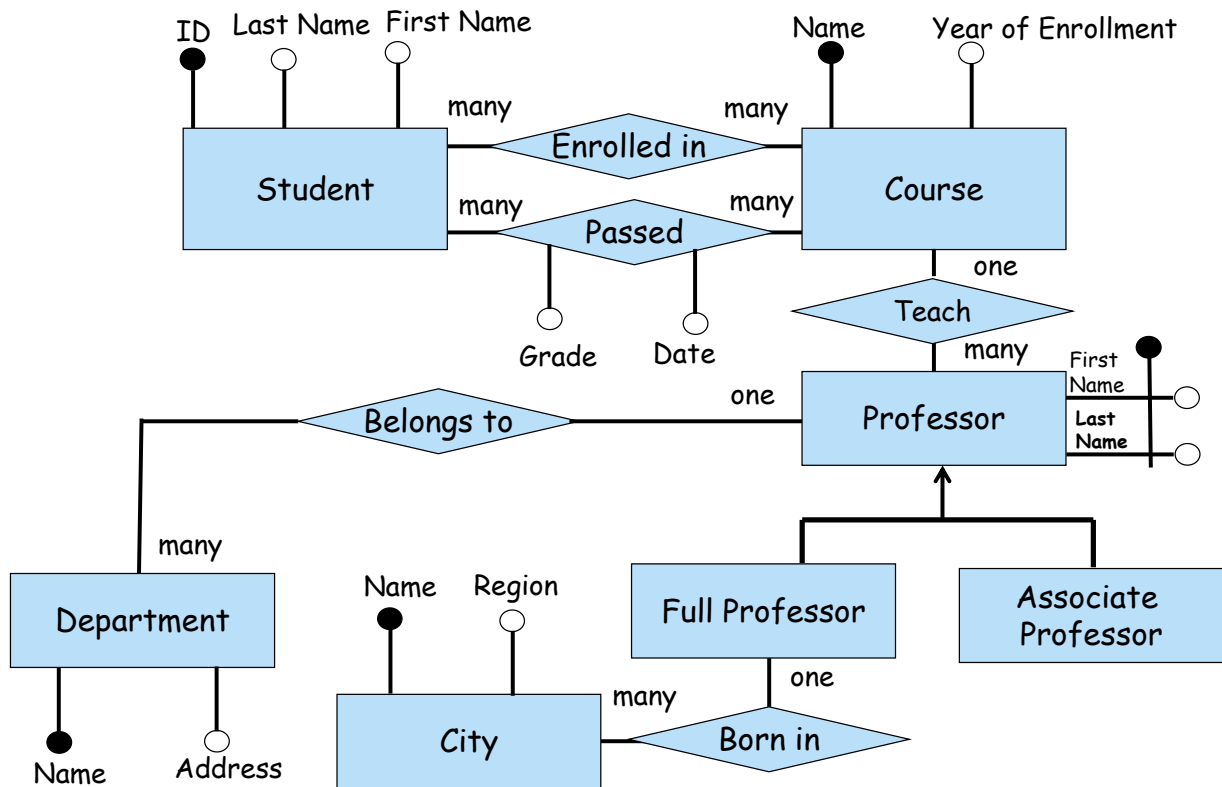
Schema with identifiers

In this case we have only internal identifiers.

This lesson ends here.

## Part 2 – Lesson 9 – Exercise

**Exercise 2.7** – third part - Assume we are in China. How should you modify the schema of Exercise 2.15 (reproduced below) in case we want to distinguish between Chinese Students and Foreign Students? For Chinese Students we want to represent the City of Birth with Name and Region, for Foreign Students we want to represent the Country and the Continent of Birth.

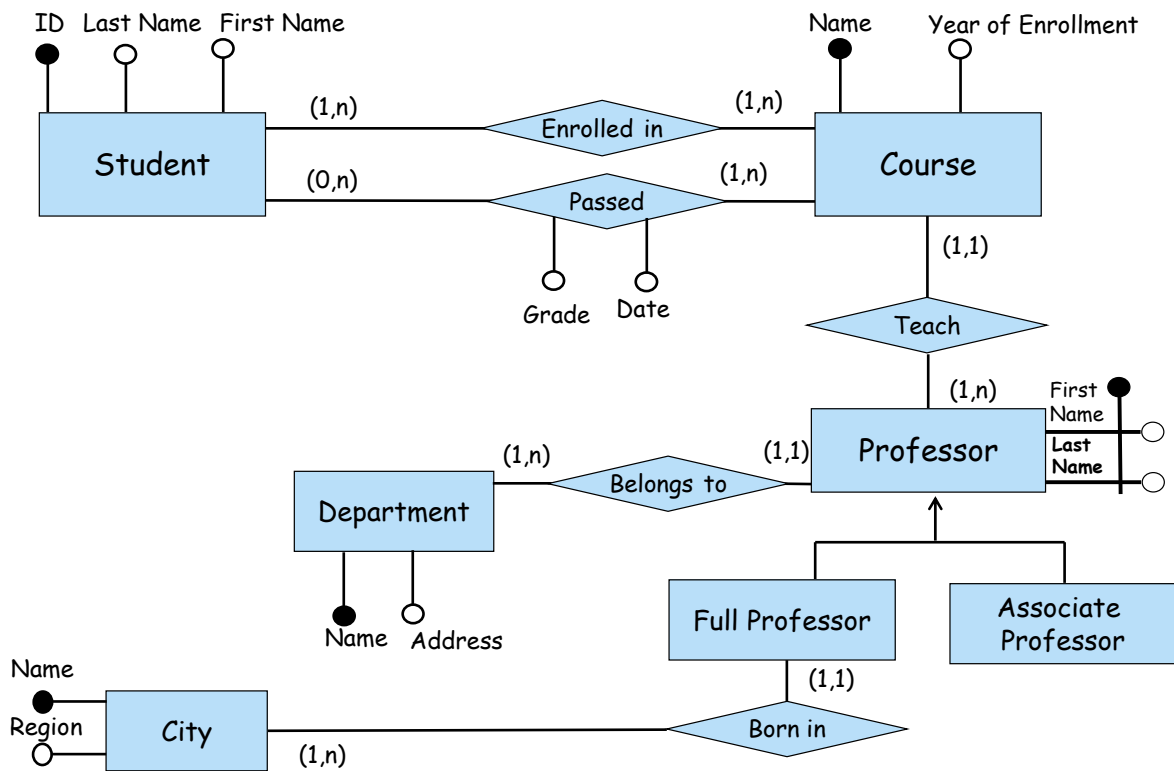


The schema at the end of Exercise 2.7 – second part

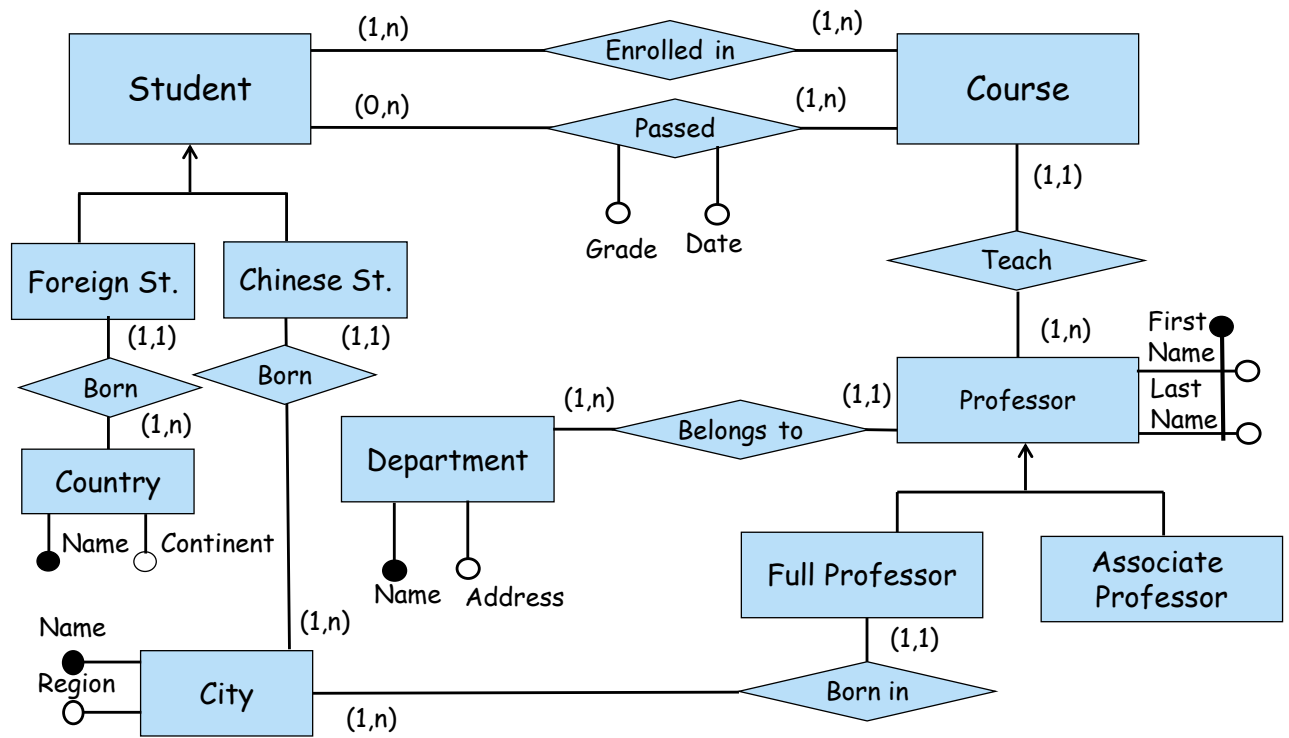
**Answer to Exercise 2.7 – third part**

New requirements make reference to the entity Student, and suggest to distinguish between two types of students. So, the construct we have to use is the generalization. Associated to the two new entities are different versions of the *born in* relationship. In one case we need a new entity Country, in the second case it is worthwhile to use an existing entity, namely City.

Let us modify just a little the layout of the diagram in such a way that we have space for the new schema to be designed.



The final schema is the following.



Final schema of Exercise 2.16

## Concepts introduced in the Second Part

### **Part 2 - Entity Relationship Model**

Diagrammatic Representation

Modeling Construct

Entity

Attribute of Entity

Relationship

Attribute of Relationship

Min/Max Cardinality of an Entity in a Relat.

Is-a Relationship between two Entities

Generalization Hierarchy among n Entities

Inheritance Property

Identifier of an Entity

Internal Identifier

External Identifier

## **Part 2 – Exercise assignment**

Solve exercises from 5.1 to 5.10 of Chapter 5 of Atzeni's book. Then compare your solutions with solutions provided in the course site.

Notice that in the Atzeni's book there is some different in modeling constructs and diagrammatic representations, in case of attributes and generalizations. Attributes may be composite, and generalizations are of different types, while we have defined one type of generalization. Please consult Atzeni's book if you want to understand better this point.





© Carlo Batini, 2015

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.