Course on Database Design

Carlo Batini

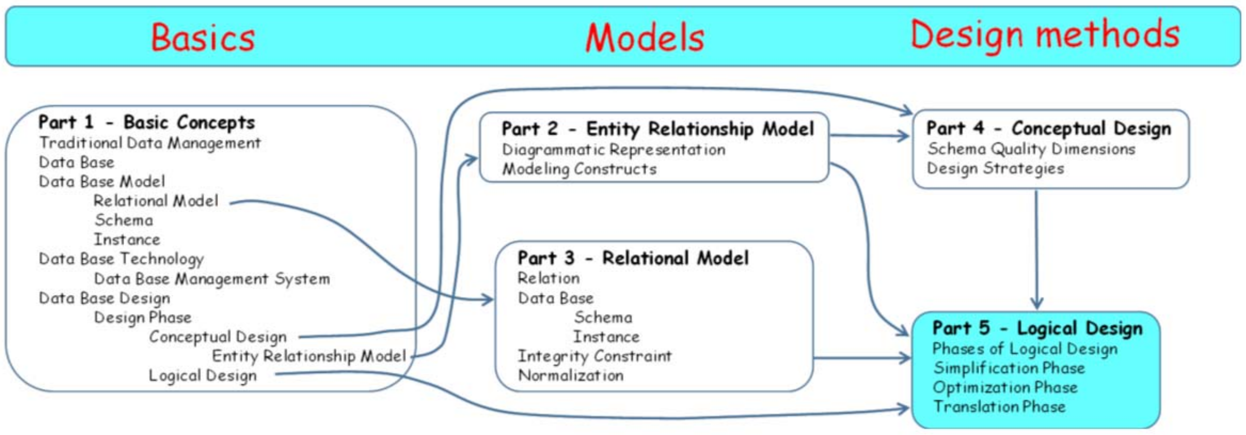
University of Milano Bicocca

Part 5 – Logical Design

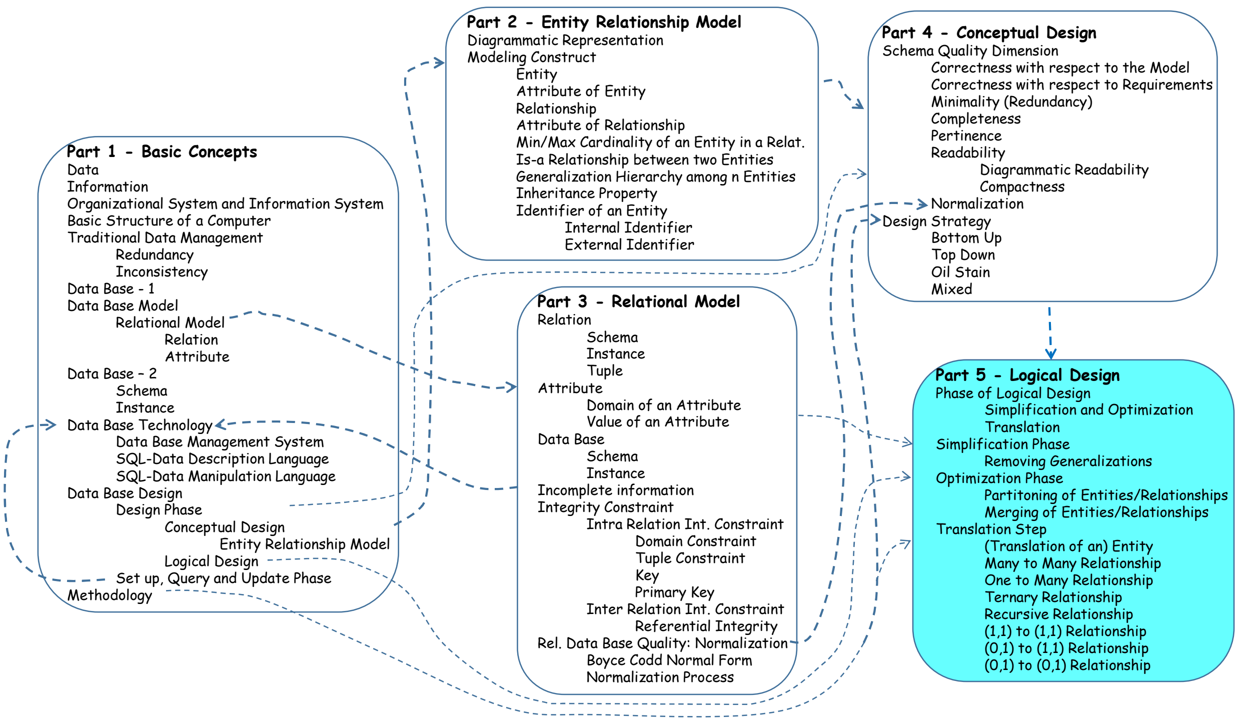
Part 5 – Lesson 1 – Introduction to Logical Design

This is the last part of the course, in which we apply all the topics learnt so far, with the goal of introducing a design methodology for the logical design of a database.

So far, we have investigated two models, the ER model adopted in the first phase of database design, conceptual design, and the relational model, adopted in logical design. The output of conceptual design is a conceptual schema, that in logical design has to be transformed into an equivalent representation in the relational model, the relational logical schema.



High level conceptual map



Low level conceptual map

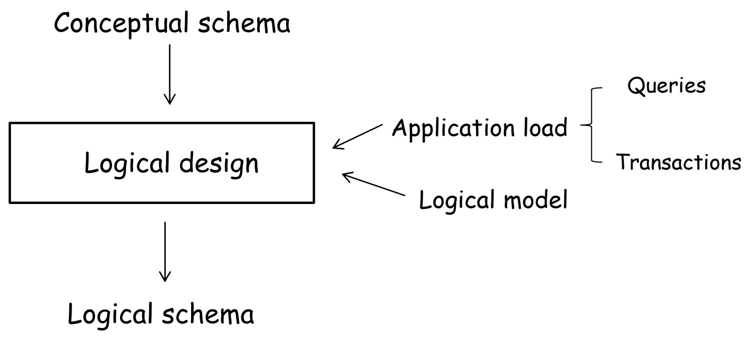
In order to translate the conceptual schema into the logical schema we have to define correspondences between constructs of the ER model and constructs of the logical model. Tables and attributes of the relational schema will be used as operands by queries and transactions of the application load; as a consequence, the process of translation has to be performed with the objective to optimize the use of resources by queries and transactions. Resources to be optimized are of two types,

a. space occupied by intermediate results in query execution (we do not consider transactions in the following, as usually queries are responsible of more relevant transfers) and

b. time needed to execute the query.

Other optimizations are characteristic of the *physical schema*, that defines the physical representation of data in secondary memory, such as e.g. the *sectors* in secondary memory where tables are stored, or the so called *block factor*, the minimum number of bits that are transferred each time from secondary memory to main memory, and viceversa. The optimization parameters of the physical schema are a high number, and in this course we will not deal at all with the many issues that arise in the optimal design of the physical schema. The topics is discussed in some detail in the Atzeni's book.

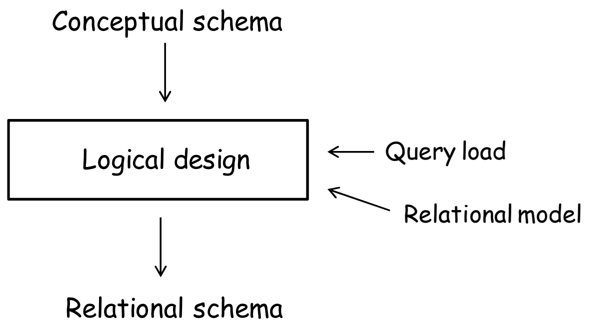
We now provide an introduction to main issues to be addressed in logical design. First of all, we provide an input-output view of logical design in general, see the following figure.



Logical design as a black box

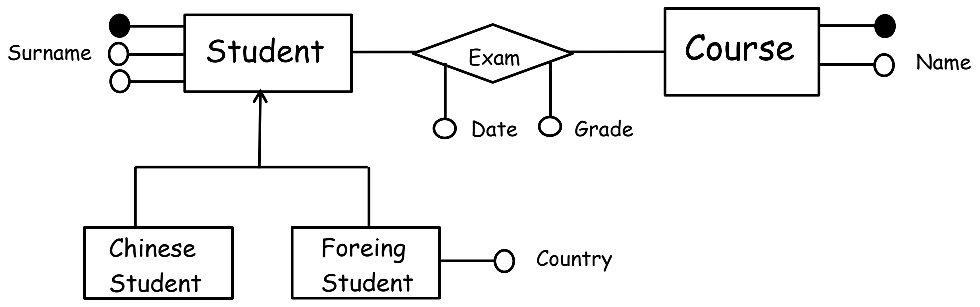
The figure tell us that logical design is an activity of transformation of the conceptual schema into the logical schema. The logical schema is expressed in terms of the logical model adopted. Furthermore, among the different equivalent schemas that we may produce, we have to choose the schema that optimizes the application load; the application load is made of the queries performed on the database (as I said, we omit transactions), ranked according to their frequencies.

We assume that the model for the logical schema is the relational model, so the input-output view is simplified as in the following figure.



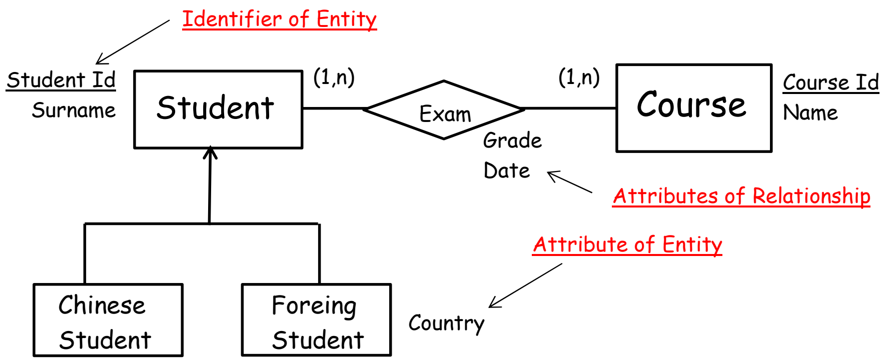
Logical design in the relational model

In this part of the course, we introduce a simplified graphical notation for ER diagrams. We define the notation using an example. In the following figure we see an ER diagram adopting the notation introduced in Part 2.



Diagrammatic notation for ER diagrams adopted so far

The corresponding diagram adopting the simplified notation is shown in the following figure.



Simplified notation adopted in this chapter

Entities, relationships, is-a hierarchies, generalizations and external identifiers adopt the same notation as in Part 2. The novelty is represented by attributes, that are listed close to the corresponding entity or relationship. When the internal identifier is an attribute, it is underlined.

The above schema is now adopted for the motivating example, that will allow us to introduce the relevant issues of logical design. The logical design process will be performed in the example within the following two assumptions.

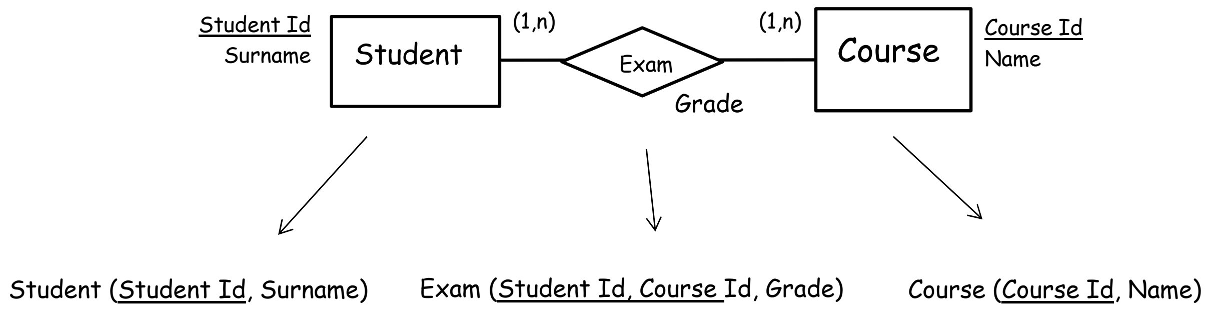
1. The relational model has *only one modeling construct*, the relation or table, composed of attributes. The many examples of correspondence we have seen between constructs of the ER model and the relation construct have shown that we will not have any problem in defining translation rules from entities and relationships to relations. The problem arises with Is-a hierarchies and generalizations, that do not have a direct counterpart in the relational model constructs. We have to manage this issue in logical design.
2. We assume that the application load is such that all queries visit together the entities Student and Course.

Let us now reply to the following questions: which are the typical steps of logical design?

First, we have to perform an activity of *model translation* from the conceptual model to the relational model. Such activity, remembering our experience in previous parts, seems relatively simple for entities and relationships.

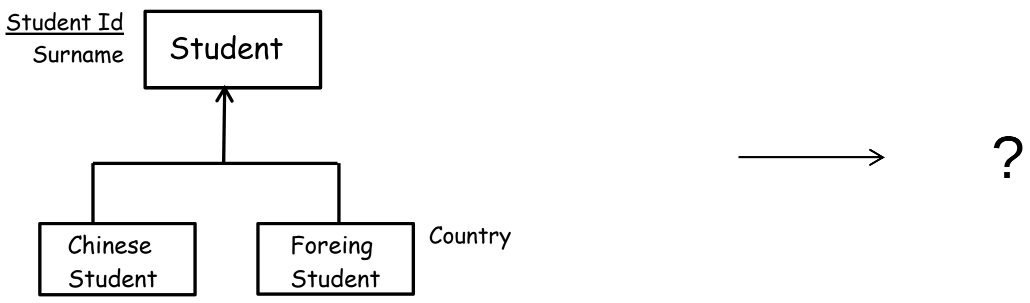
Question 5.1 - Try in our example to translate entities Student and Course and relationship Exam into relations in the relational model.

Discussion on Question 5.1 - I hope you have produced tables as in the following. Student is translated into a table having as key *Student Id*, Course into a table with key *Course Id*. Being *Exam* a relationship whose instances are pairs of instances of Student and Course, it is translated into a table with composed key *Student Id + Course Id*.



Translation of entities and relationships into Relational tables

What appears not immediate to perform is the translation of generalizations; see the following figure.



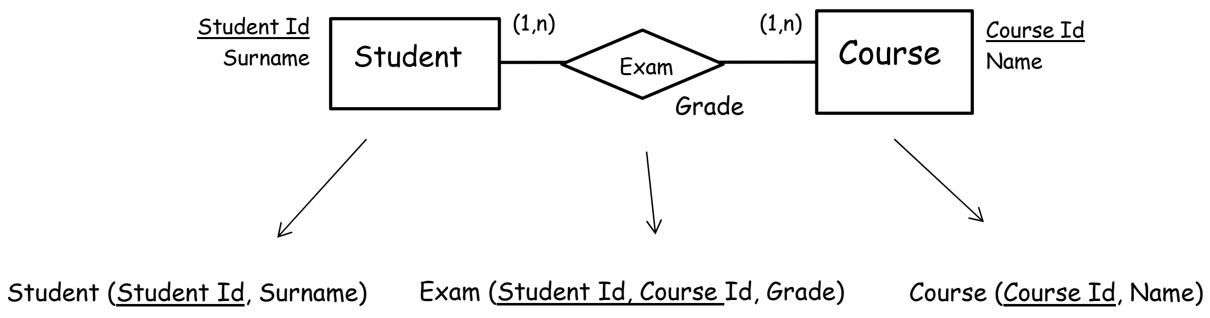
How to translate generalizations?

With reference to this case, and the related case of is-a hierarchies, we can manage these situations through a simplification activity, that transforms the generalization into a new ER schema that equivalently represents the generalization, but adopts only entity and relationship constructs. We see that besides translation, and before translation, we have to perform, a second step, namely *simplification*.

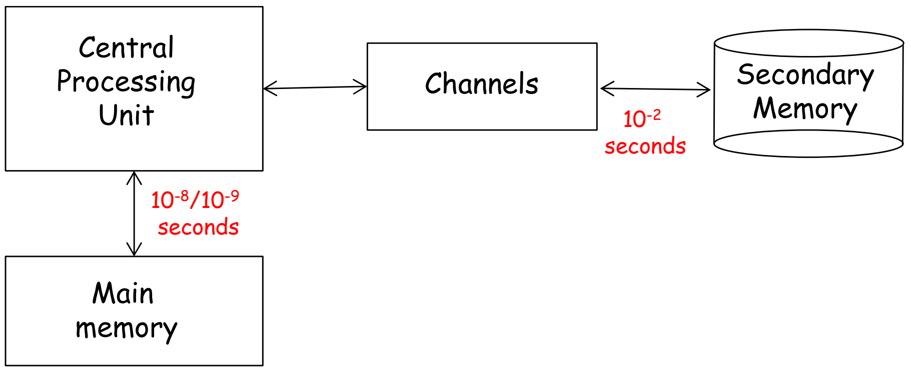
Look now at the second assumption we made previously.

2.the application load is such that all queries visit together the entities Student and Course.

What are the consequences of this assumption in the logical design activity? Consider the translation we performed previously, namely

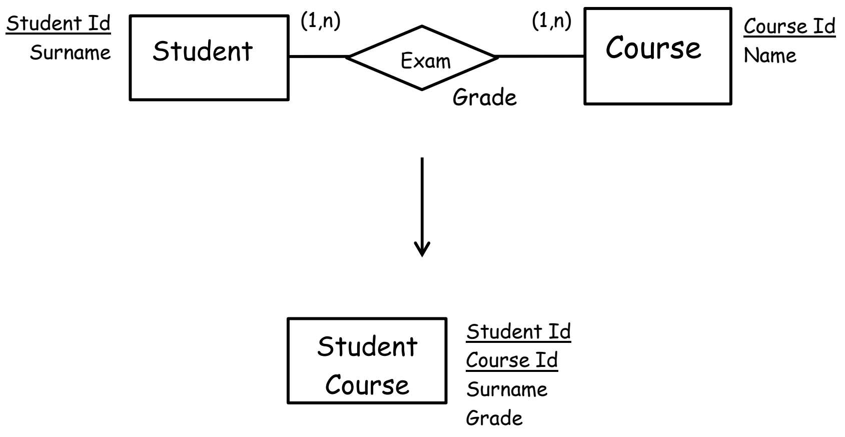


At query time, the execution of queries on the relational schemas will request that the three tables are linked together. However, remember the structure of a computer, which I reproduce in the following figure, in the part referring to central processing unit and memories.



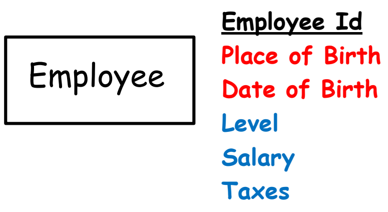
Computer structure: CPU and memories

To perform the linkage of tables (that, incidentally, in the query language SQL is performed with the *Join* operator), we have to transfer three tables from secondary memory to main memory. Intuitively, in this specific case in which all queries are performed on all the three tables, it is more efficient to transform the ER schema made of Student, Course and Exam into a unique entity, as in the following figure.



Transformation for achieving efficiency

Notice that we have denormalized the schema, to achieve more efficiency at query time. With this choice, a unique table will be created in the translation step. The above case is called *merging* in the following. As a second example, consider the following entity.



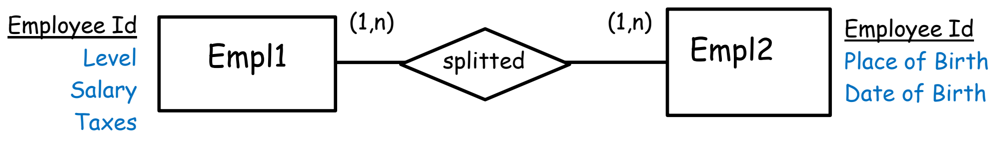
An entity with two groups of attributes accessed separately by queries

Assume that queries in the application load operate both on the identifier, and separately on the two groups of attributes, respectively:
a. Place of Birth + Date of Birth, and
b. Level + Salary + Taxes.

This assumption is reasonable, since the two groups of attributes refer to distinct properties of Employee, respectively

a. personal characteristics and
b. professional and economic characteristics.

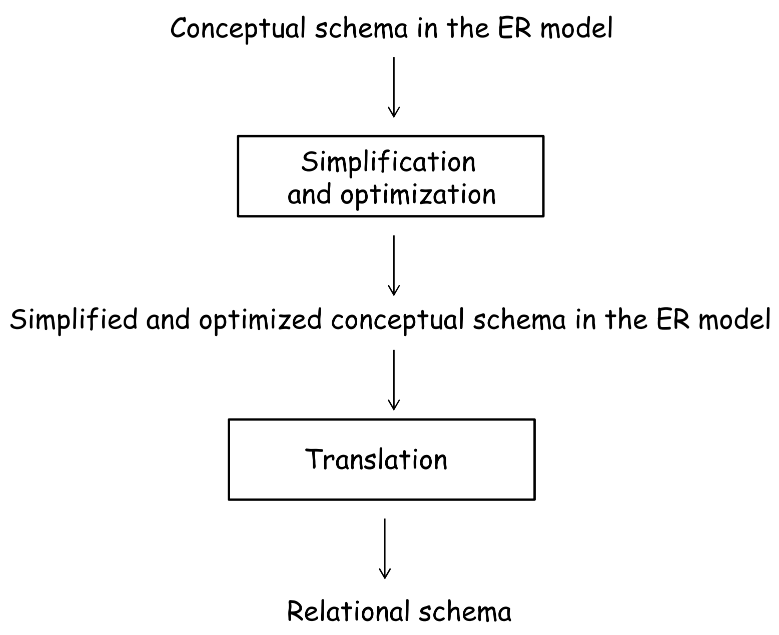
In this case, we can perform the opposite transformation w.r.t the previous one: we can decompose the entity *Employee* into two entities that equivalently represent employees but with different attributes, except the identifier that has to be reproduced for both entities. We can use the term *partitioning* for this type of optimization.



An example of partitioning

Therefore, we have identifies two types of *optimization activities*, respectively *merging* and *partitioning*. We will see soon that also the simplification activity comprises optimization decisions.

Considering that optimization, as simplification, is performed on the conceptual schema, we come to the conclusion that they are to be made before the translation in the relational model. In consequence of this, logical design has the following structure, represented in the figure.

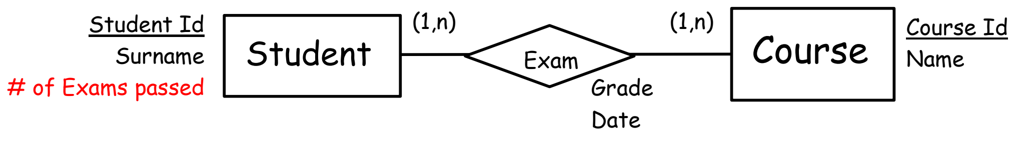

Conceptual schema in the ER model
Simplification and optimization
Simplified and optimized conceptual schema in the ER model
Translation
Relational schema

The three steps of logical design

Before concluding this introductory lesson we have to make two remarks, both referring to the Atzeni's book.

First, with reference to optimization activities, the book adopts an optimization methodology more complex than the methodology described in the following. For instance, it considers also the transaction load, which we have discarded; furthermore, it considers for entities and relationships their volume in terms of number of instances. See the book for more detail.

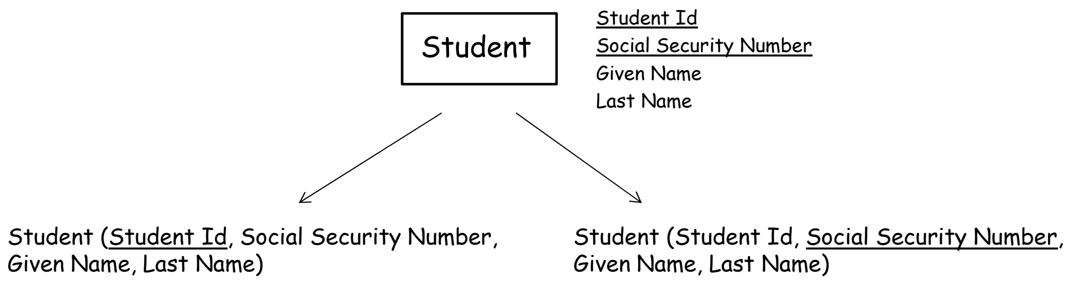
Secondly, the book mentions other two activities that we will not discuss in the following, and briefly mention now. The first activity is *redundancy analysis*. Look at the following schema.



An ER schema with redundancies

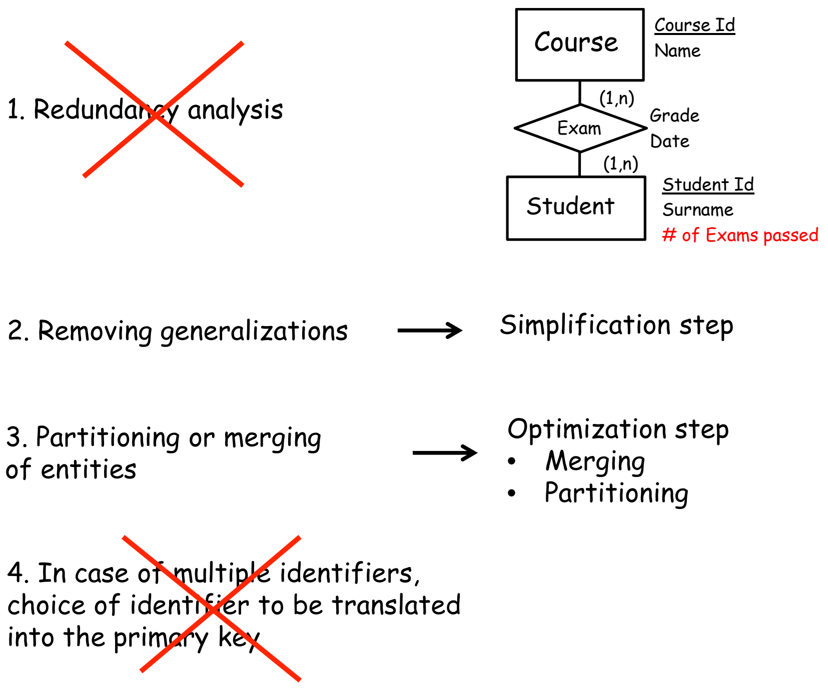
The attribute *# of Exams passed*, as we noticed in Part 3, when we discussed the *minimality* quality dimension, is redundant. It is up to logical design to decide whether to keep it in the relational schema, or else to delete it. Keeping it in the schema makes more efficient the execution of queries that involve the attribute, but forces to update the attribute when some transaction changes the instances of relationship Exam, e.g. inserting a new exam of a student.

The second activity is relevant when an entity has multiple identifiers. Such activity corresponds to the choice of the identifier to be translated as primary key of a relation. See the figure.


Student
Student Id
Social Security Number
Given Name
Last Name
Student (Student Id, Social Security Number, Given Name, Last Name)
Student (Student Id, Social Security Number, Given Name, Last Name)

An entity with two identifiers and the two possible choices as primary keys

We will not deal further with these two activities; the interested reader can study the Atzeni's book, Chapter 7. We show in the following figure our simplification and optimization methodology compared with the methodology described in the Atzeni's book.



Simplification and optimization in Atzeni's book and in this Part 5

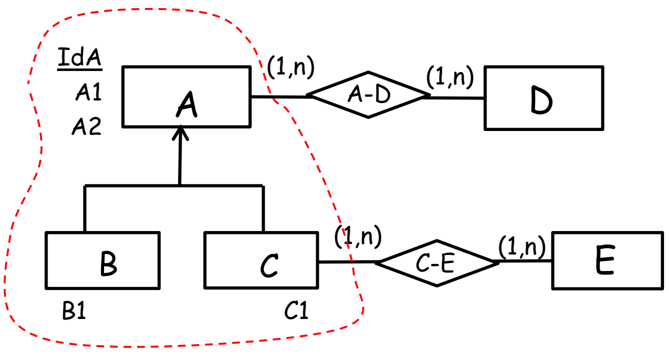
Now we have completed the lesson.

Part 5 – Lesson 2 – Simplification and Optimization Steps

As we said at the end of previous lesson, we discuss in the following:
a. with reference to simplification, the activity of *restructuring generalizations*, and
b. with reference to optimization, the activities of *partitioning and merging*.

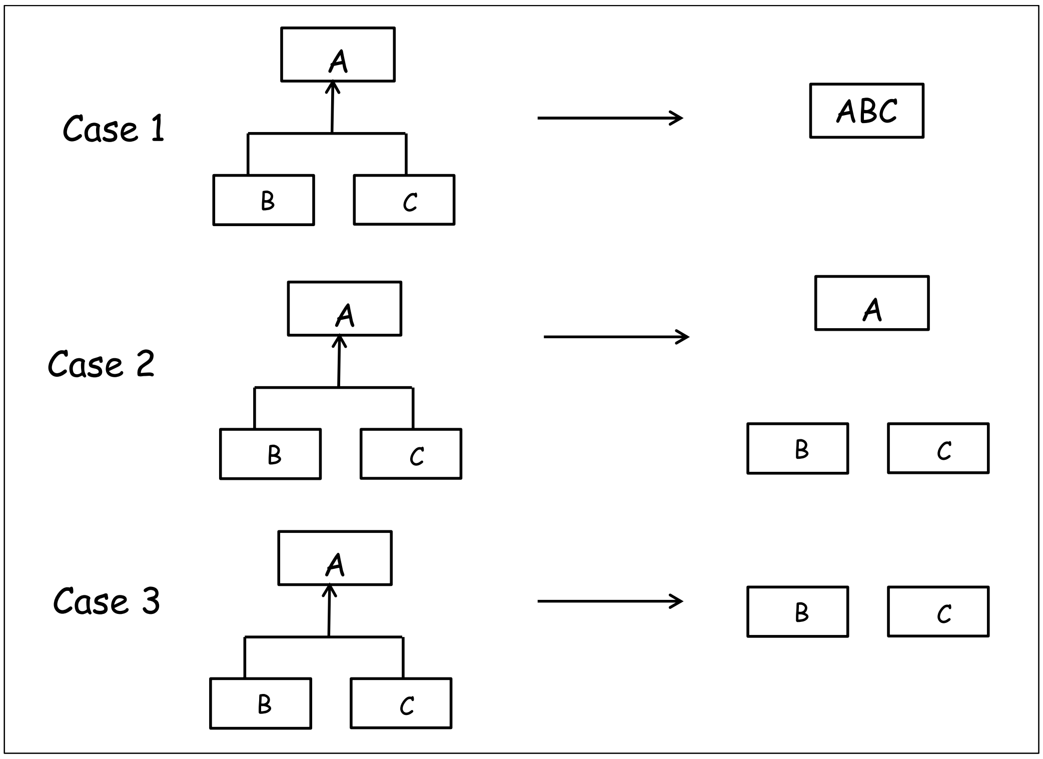
Simplification: restructuring generalizations

We will discuss this issue starting from the following example, in which we have identified with a closed dashed line the part of the schema to be restructured, and have connected with relationships A-D and D-E respectively entities A and C in the generalization.



The schema of the motivating example and the part on which we focus now

We have three most relevant cases, shown in the following figure.



Relevant cases of generalization restructurings

The choice among the three cases depends on the characteristics of the query load. In this sense, the simplification activity involves also optimization choices. Let us see the three cases.

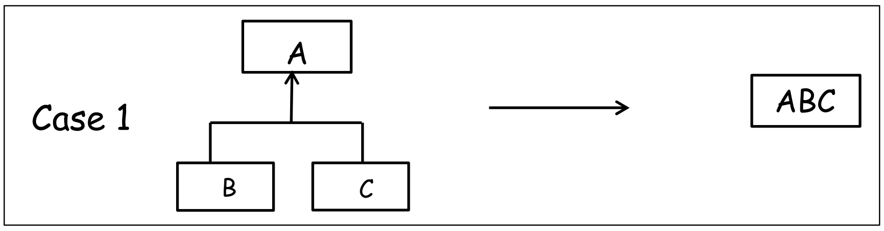
The first case is applied when the queries of the application load visit the three (or n, in the general case) entities all together.

The second case is applied when the queries of the application load visit separately the three entities.

The third case is applied when the queries of the application load visit separately the two child entities in the generalization, and do not visit the parent entity.

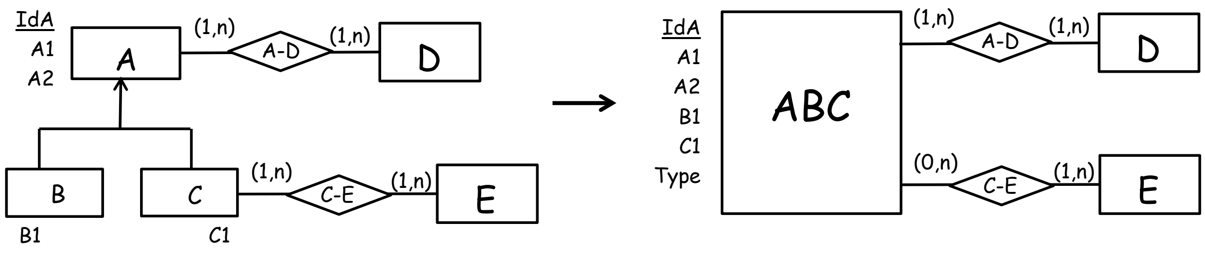
Let us see now how the transformation has to be performed in the three cases in order to keep unchanged the information content of the schema. We have to carefully associate to the new entity (or the new entities) the properties (attributes and relationships) associated to the entities in the generalization in the original schema.

Case 1



First case of restructuring

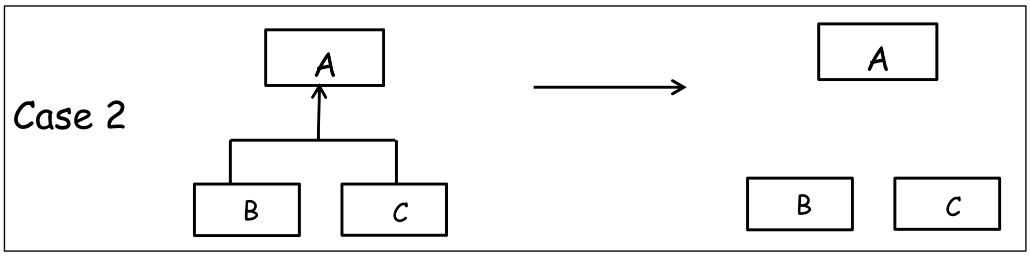
In our example, we have to perform the following transformation and assignment of properties.



Corresponding assignment of properties to new constructs in case 1

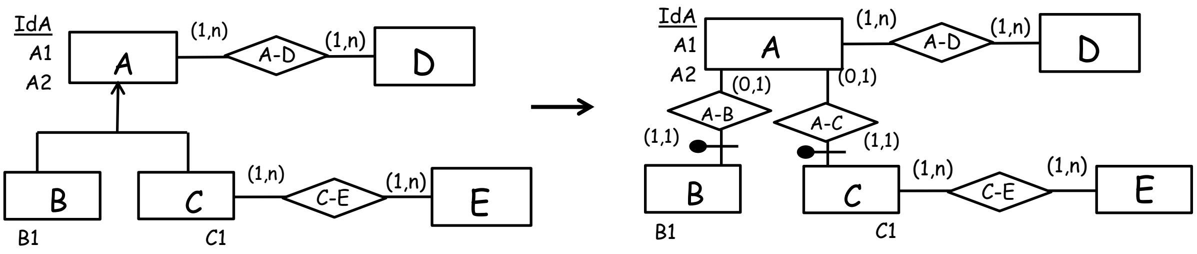
Notice that we have assigned to the unique entity ABC result of the transformation, all the attributes and relationships previously assigned to A, B, and C. Furthermore, we have added a new attribute Type to the entity ABC. The domain of Type is [B, C]; its role is to discriminate between instances of ABC corresponding to instances of B and to instances of C. Finally we have changed the minimum cardinality of ABC in relationship C-E, since the instances of C are a subset of the instances of A and we cannot be sure that all instances of A are connected with instances of E.

Case 2



Second case of restructuring

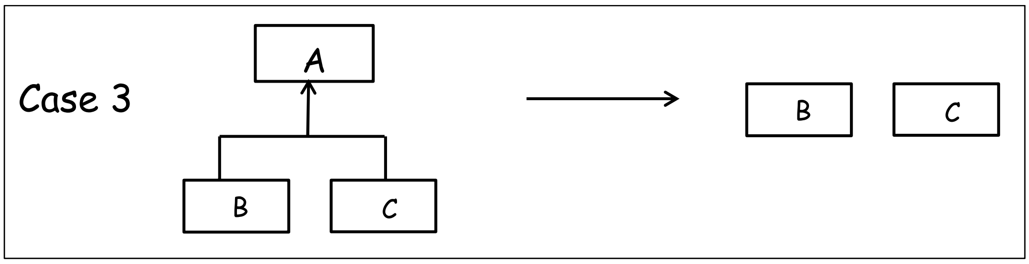
As to the second case of transformation, in our example we have to perform the following new assignement of properties.



Corresponding assignment of properties to new constructs in case 2

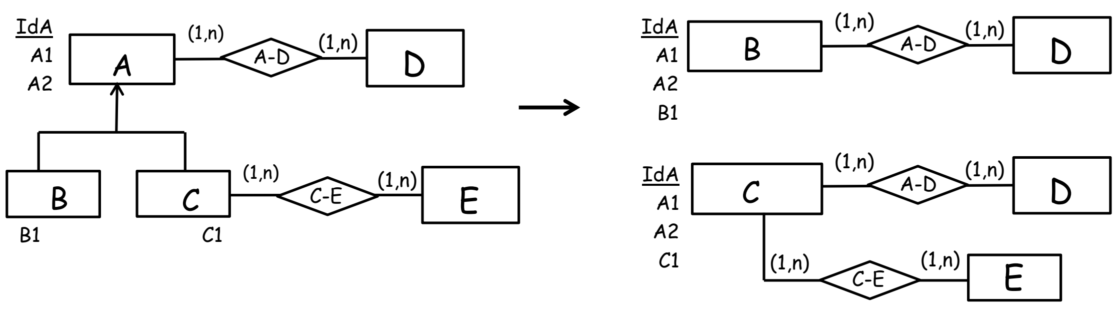
Here we connect entities B and C with A, to represent the relationships between the same entity instances in each pair of entities A/B and A/C. Furthermore, since the instances of B and C coincide with a subset of the instances of A, we add to B and C an external identifier. Furthermore, the two minimum cardinalities of A in relationships A-B and A-C are both 0, as some of the instances of A, as we said, are in common with B and others are in common with C.

Case 3



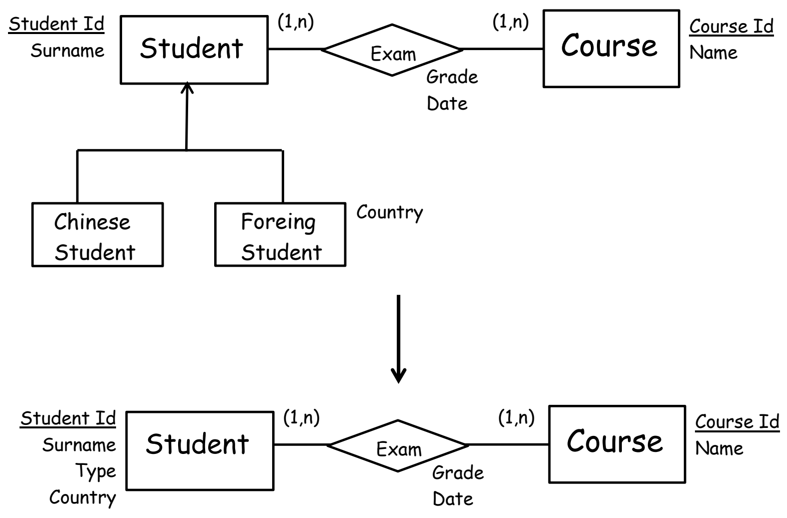
Third case of restructuring

In Case 3, see above, we assign the properties of A both to B and to C, and assign both to B and to C their previous properties.



Corresponding assignment of properties to new constructs in case 3

In the motivating example, due to the assumption that all queries visit entities *Student* and *Course* together, we may apply Case 1, which leads to the following transformation.

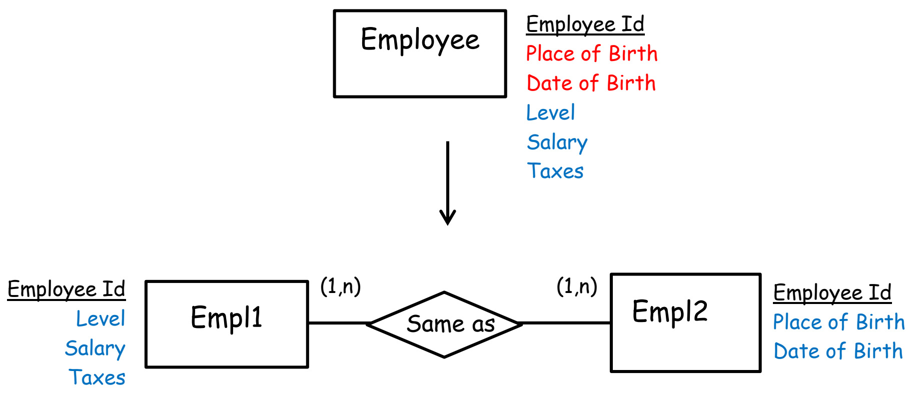


Restructuring generalizations in the motivating example

We have concluded the discussion of the simplification step.

<u>Optimization: partitioning</u>

We reproduce here the case of entity partitioning we discussed in the introductory lesson.

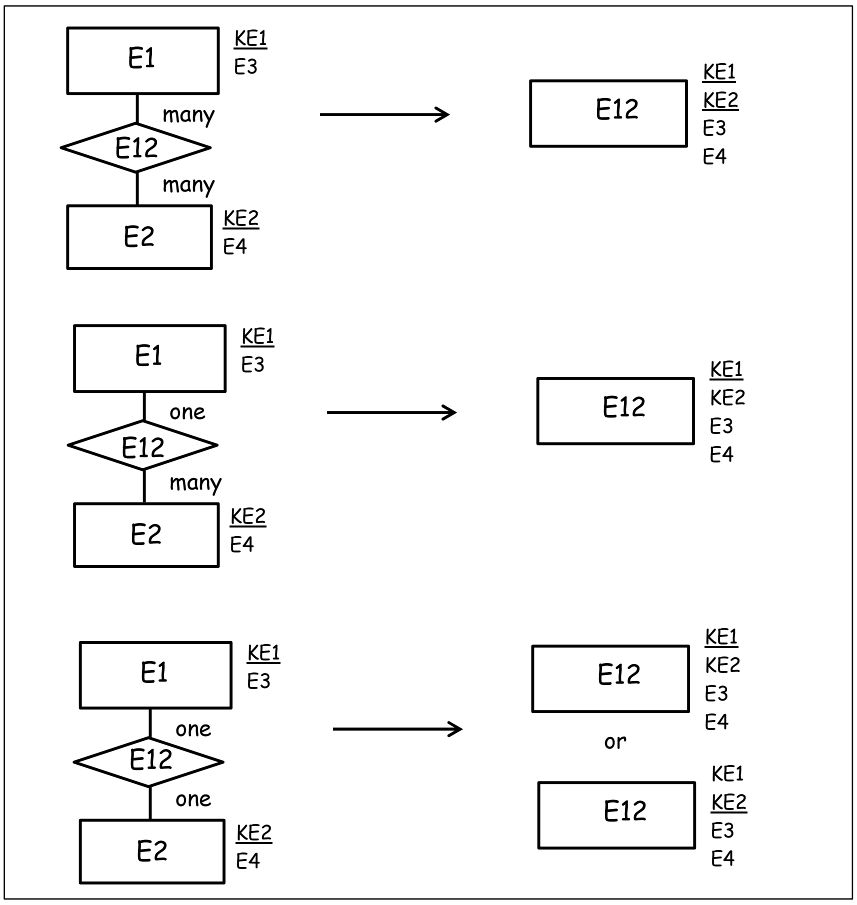


Entity partitioning

Besides partitioning entities, we may also partition relationships. We do not discuss further this case.

Optimization: merging

Merging of two entities (and the related relationship) is performed when in the query load queries visit together the two entities and the relationship. The identifier and attributes of the resulting entity depend on the type of the relationship, namely a. many to many, b. many to one and c. one to one. In the following figure, we see the three possible cases.



The three cases of entity (and related relationship) merging

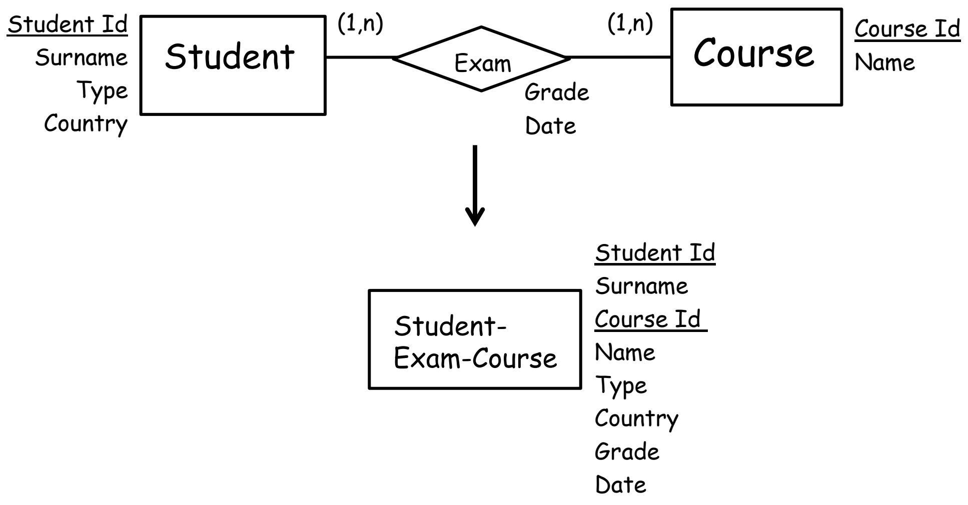
When the relationship is many to many, the identifier of entity E12 is the union of the identifiers of the entities to be merged. The reason of this rule is that in this case the instances of the entity correspond to the many to many instances of the relationship, and so, to be identified, we need to know the identifiers of the instances of both entities E1 and E2.

When the relationship is one to many, the identifier of entity E12 is the identifier of the entity on the *one* side, since the corresponding instance identifies also the unique instance of the entity related to it on the *many* side.

When the relationship is one to one, we can choose either the identifier of the first entity or the identifier of the second entity.

Notice that in the case of many to many relationships, merging leads to an un-normalized entity, that will generate a table non in BCNF. This is a typical design decision that is performed to make query execution more efficient. While we apply denormalization during the first phase of logical design, it can be also applied on the relational schema during the subsequent phase of physical design, which we do not discuss here.

In the motivating example, applying again the assumption that all queries visit entities Student and Course together, we may perform merging, obtaining as a result the following transformation.



Result of merging in the motivating example

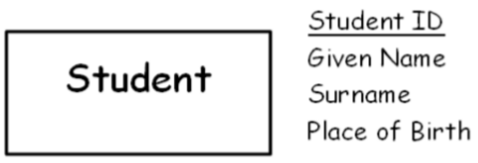
Part 5 – Lesson 3 – Translation Step – first part

In this lesson we start discussing the different cases of translation of ER constructs from the simplified ER schema to a relational schema. They are:

1. Entity with internal identifier
2. Many to many relationship
3. Many to many Ternary relationship
4. Many to many recursive relationship
5. One to many relationship
6. Entity with external identifier
7. One to one relationship; we will have to discuss three cases:
   a. (1,1) to (1,1)
   b. (0,1) to (1,1)
   c. (0,1) to (0,1).

In this lesson we discuss the first five cases. I need your collaboration in defining the different translations. You are aided by the examples given so far.

1. Entity with internal identifier



In the translation, we have to remember that an entity is a class of elementary instances sharing several properties of type attribute.

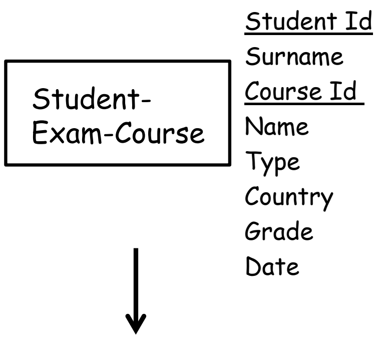
Question 5.2 – How do you translate this case?

Solution to Question 5.2

The translation of the above entity is the following relation schema.

Student (<u>Student Id</u>, Given Name, Surname, Place of Birth)

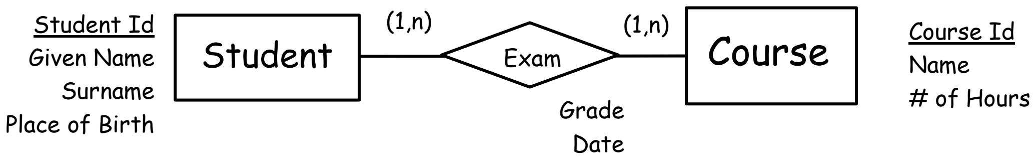
Since in our motivating example the ER schema is made of a unique entity, we may proceed to the translation of the simplified and optimized ER schema into the relational model.



Student-Exam-Course (<u>Student Id, Course Id</u>, Surname, Name, Type, Country, Grade, Date)

Applying the translation in the motivating example

2. <u>Many to many binary relationship</u>

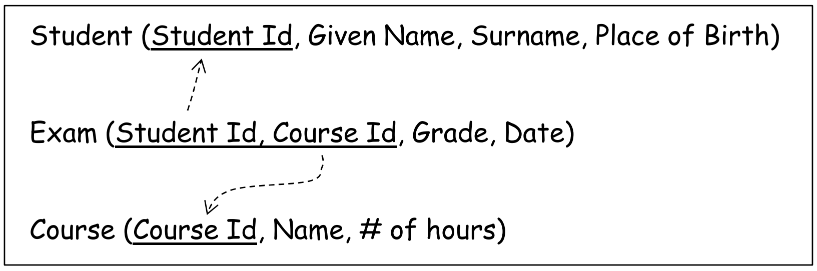


Example considered in the translation of many to many binary relationship

As in Question 5.1, we may represent the relationship Exam with a relation, whose key is the union of the identifiers of the two participating entities. This is reasonable, since the instances of Exam are pairs of instances of Student and Course. The table is completed with the possible attributes of the relationship. In our case we have for the whole above schema the following relational schema.

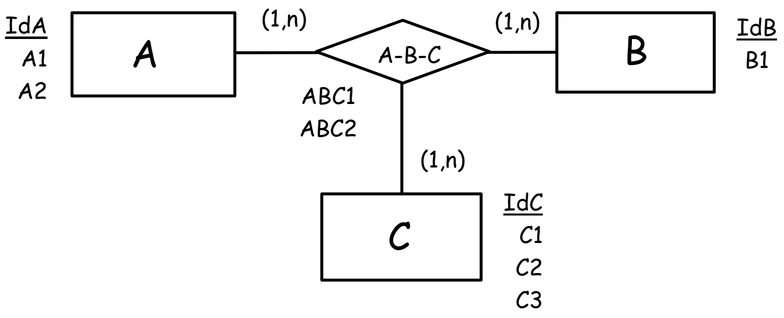
Student (<u>Student Id</u>, Given Name, Surname, Place of Birth) resulting from entity Student
Exam (<u>Student Id, Course Id</u>, Grade, Date) resulting from relationship Exam
Course (<u>Course Id</u>, Name, # of hours) resulting from entity Course.

Now, we should not forget adding referential integrity constraints, which relate the attributes participating in the key of the table *Exam*, with the keys of the two tables associated to the two entities.


Student (Student Id, Given Name, Surname, Place of Birth)
Exam (Student Id, Course Id, Grade, Date)
Course (Course Id, Name, # of hours)

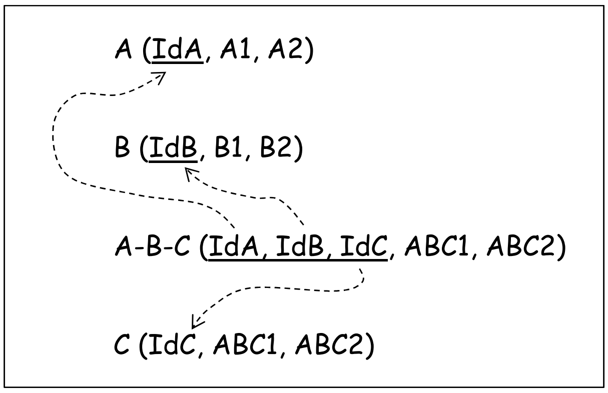
Do not forget referential integrity constraints…

3. Many to many ternary relationship



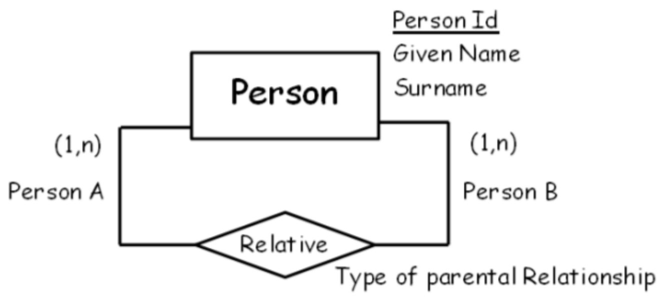
Translation of ternary relationship

This case is a simple extension of the previous one; the only change is that now the instances of the relationship are triples made of instances of three entities, so we have a key made of three attributes for the relation corresponding to A-B-C. Including also referential integrity constraints, we get the following schema.



Relational schema resulting from the translation of the ternary relationship and related entities

4. Many to many binary recursive relationship



Example of many to many recursive relationship

The many to many binary recursive relationship in the example above is defined on the entity Person, and expresses the parental relationships among a set of persons. For each pair of persons we want also to represent the type of parental relationships, e.g. child of, uncle of, etcetera.

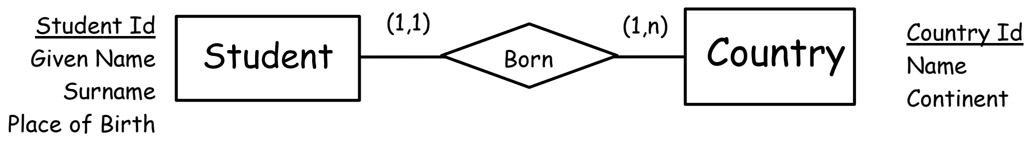
This case too is an extension of the many to many binary case considered above. In this case, the two entities involved in the relationship coincide. Therefore, the key is made of two occurrences of the same entity identifier; in the two occurrences, we have to adopt different names. Therefore, the resulting relational schema is the following.

Person (Person Id, Given Name, Surname)

Relative (Person IdA, Person IdB, Type of parental relationship)

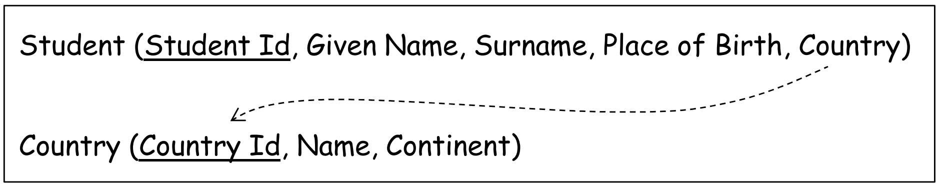
Translation of many to many recursive relationship

5. Translation of one to many relationship



Example of one to many relationship

A one to many relationship too is a particular case of a many to many relationship, so we could translate the two entities and the relationship with three tables. However, we have to observe that due to the one to many relationship, for each instance of Student we have a unique corresponding instance of Country in the Born relationship. Therefore, a functional dependency exists between the key of Student, namely Student Id, and the key of Country, Country Id. Therefore, we can save one table, including Course Id among the attributes of table Student. We obtain the following relational schema.


Student (Student Id, Given Name, Surname, Place of Birth, Country)
Country (Country Id, Name, Continent)

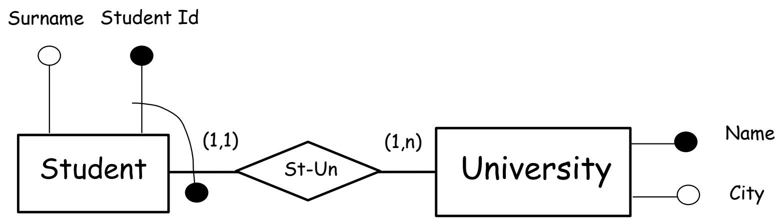
Translation of a one to many relationship

With this case, we have concluded the lesson.

Part 5 – Lesson 4 – Translation Step – second part

In this lesson, we consider the last cases of translation form ER constructs to the relational model.

6. Translation of Entity with external identifier



The example for translation of an entity with external identifier

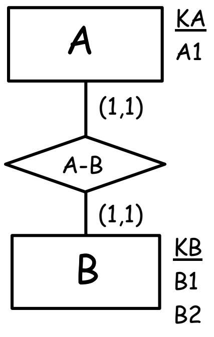
The external identifier means that the key of the table Student has to be extended with the identifier of the second Entity involved in the relationship. Therefore, the resulting pair of tables corresponding to the above schema is:

Student (Student Id, University, Surname)

University (Name, City)

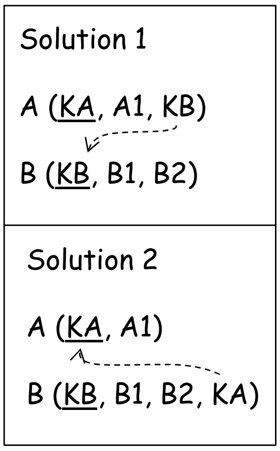
Resulting relational schema

7.1 Translation of one to one relationship: case of (1,1) to (1,1)



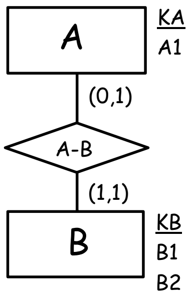
(1,1) to (1,1) relationship

A (1,1) to (1,1) relation is a particular case of one to many relationship, so two solutions are possible.

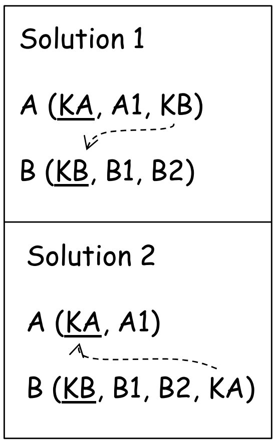


The two solutions are equivalent.

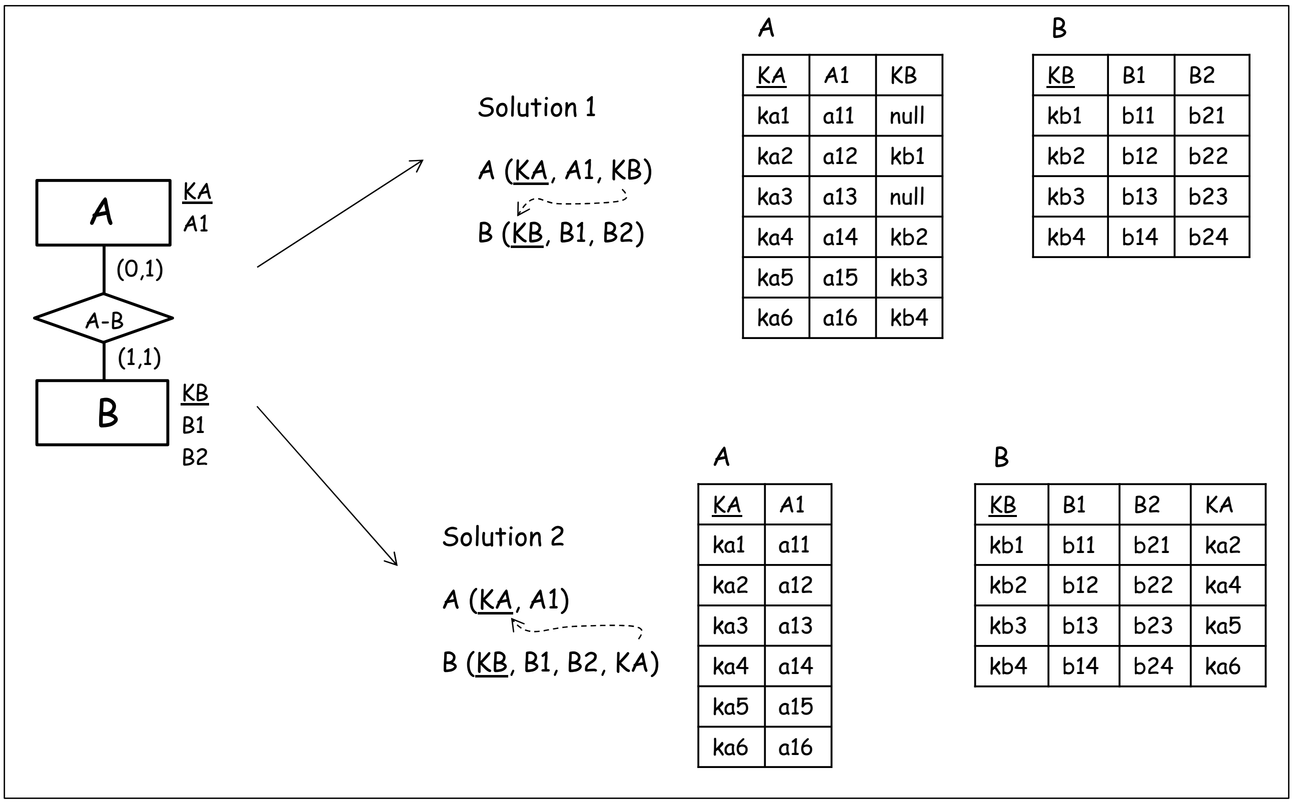
7.2 Translation of one to one relationship: case of (0,1) to (1,1)



Also in this case we have the two solutions as before:



Here a difference emerges in comparison to the previous case, due to (0,1) cardinalities of A. Let us see in the following figure an example of instances in the two solutions.



Solution 1

A (KA, A1, KB)

B (KB, B1, B2)

A

KA	A1	KB
ka1	a11	null
ka2	a12	kb1
ka3	a13	null
ka4	a14	kb2
ka5	a15	kb3
ka6	a16	kb4

B

KB	B1	B2
kb1	b11	b21
kb2	b12	b22
kb3	b13	b23
kb4	b14	b24

Solution 2

A (KA, A1)

B (KB, B1, B2, KA)

A

KA	A1
ka1	a11
ka2	a12
ka3	a13
ka4	a14
ka5	a15
ka6	a16

B

KB	B1	B2	KA
kb1	b11	b21	ka2
kb2	b12	b22	ka4
kb3	b13	b23	ka5
kb4	b14	b24	ka6

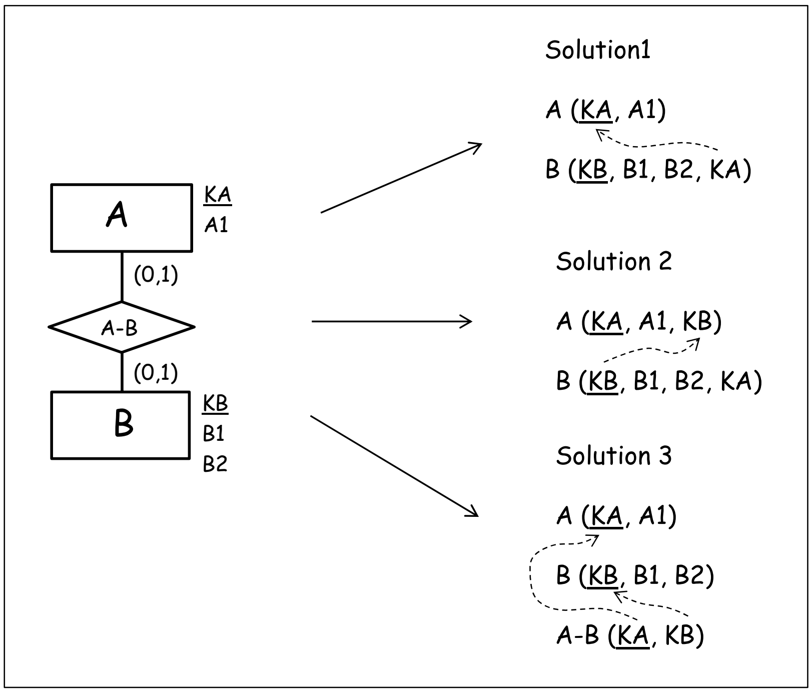
The two cases of translation in the case (0,1) to (1,1) and corresponding instances

We see that in Solution 1, due to the (0,1) cardinality of entity A, some instances of A are not connected with instances of B, and so in table A we have null values, that correspond to inefficient use of space.

In solution 2, due to the (1,1) cardinalities of entity B, we have no null values, since all instances of B are connected to instances of A, so KA is always different from null. In consequence of this, we prefer solution 2.

7.3 Translation of one to one relationship: case of (0,1) to (0,1)

In this case, besides the two solutions of previous cases, we have a third solution, see the figure.



The three solutions for the (0,1) and (0,1) case

The reason of introducing Solution 3 is that in this case for both entities A and B we have a minimum cardinality equal to 0. So, in case we know that only a limited number of instances of A and B are connected in the relationship A-B, then it is preferable to introduce a third table, corresponding to the A-B relationship, in which we represent only pairs of keys that refer to connected instances of A and B.

We have concluded the lesson.

Part 5 – Lesson 5 – Methodology for logical design

The above activities related to logical design need to be organized in a methodology. What is a methodology? It is a set of activities, ordered according to a plan, with an input and an output for each activity, that, when executed together, achieve an expected goal. In our case, the goal is to design a relational schema that is coherent with initial requirements; furthermore, among the different relational schemas that can be chosen, the resulting relational schema should be efficient with respect to the application load. The methodology for logical design is shown in the following box.

Methodology: Given an Entity Relationship schema S, to be translated into a Relational schema proceed as follows:

1. SIMPLIFICATION AND OPTIMIZATION – Using requirements referring to the query load, simplify and optimize the ER schema, producing a schema S'.
    a. Transform generalizations.
    b. Merge/Partition entities.

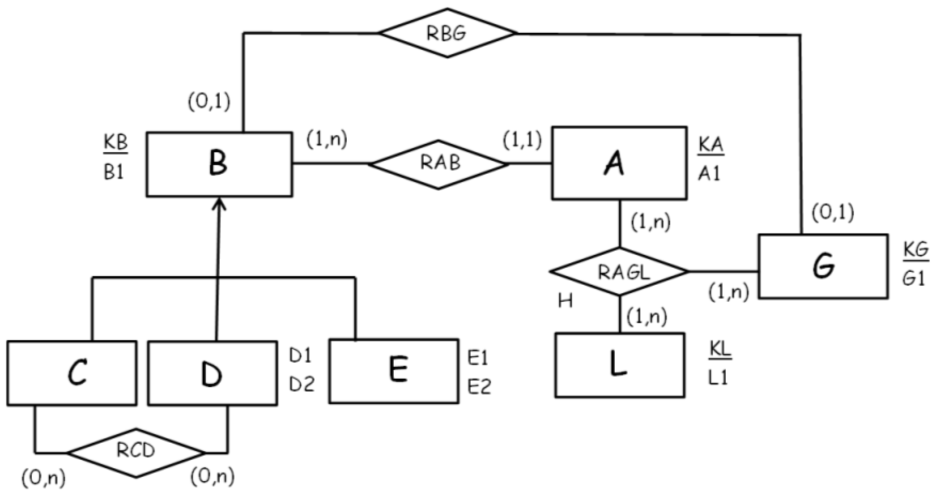
2. TRANSLATION - Starting from S'
    a. Focus first on *Entities*, and translate them into relations.
    b. Focus on *Entities with external identifier*, and extend the key of the corresponding Relation with the Identifier of the other entity in the relationship.
    c. Focus now on *binary, ternary, and recursive many to many relationships*; to represent them introduce a new relation.
    d. Focus now on *one to many Relationships*; to represent them, do not introduce a new Relation, but extend the relation on the "one" side.
    e. Finally, focus on *one to one Relationships*, and translate them according to the three cases:
        1. (1,1) to (1,1),
        2. (0,1) to (1,1),
        3. (0,1) to (0,1).

Notice that, with reference to the translation step, it is important that transformations involving entities (steps a. and b.) be performed first, to complete the generation of keys of relations associated to entities. The order among relationship transformations is not relevant: some of them create a new relation, others extend relations previously created, but they act independently from each other.

We now apply the methodology to three exercises.

Exercise 5.1

Initial schema



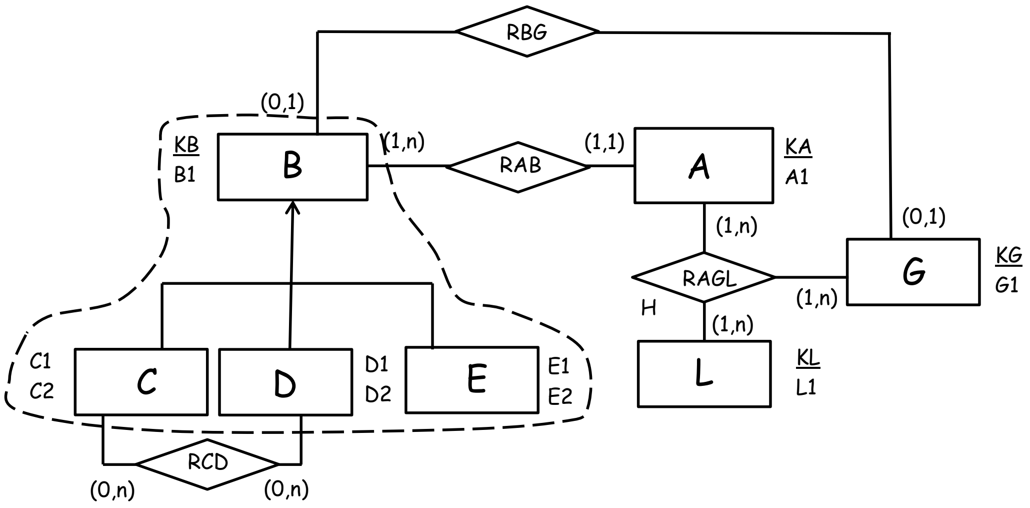
Requirements say that:

1. B, C, D, E are accessed by queries all together.
2. Instances of B and G are millions.
3. Instances of RBG are thousands.

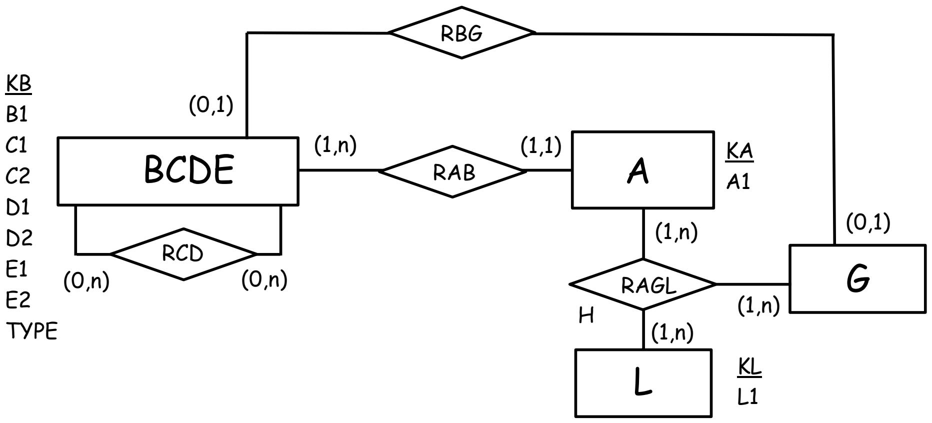
Input schema and requirements of Exercise 5.1.

Solution to Exercise 5.1 - Let us execute first the simplification and optimization step.

The simplification step involves the schema in the dashed closed line.



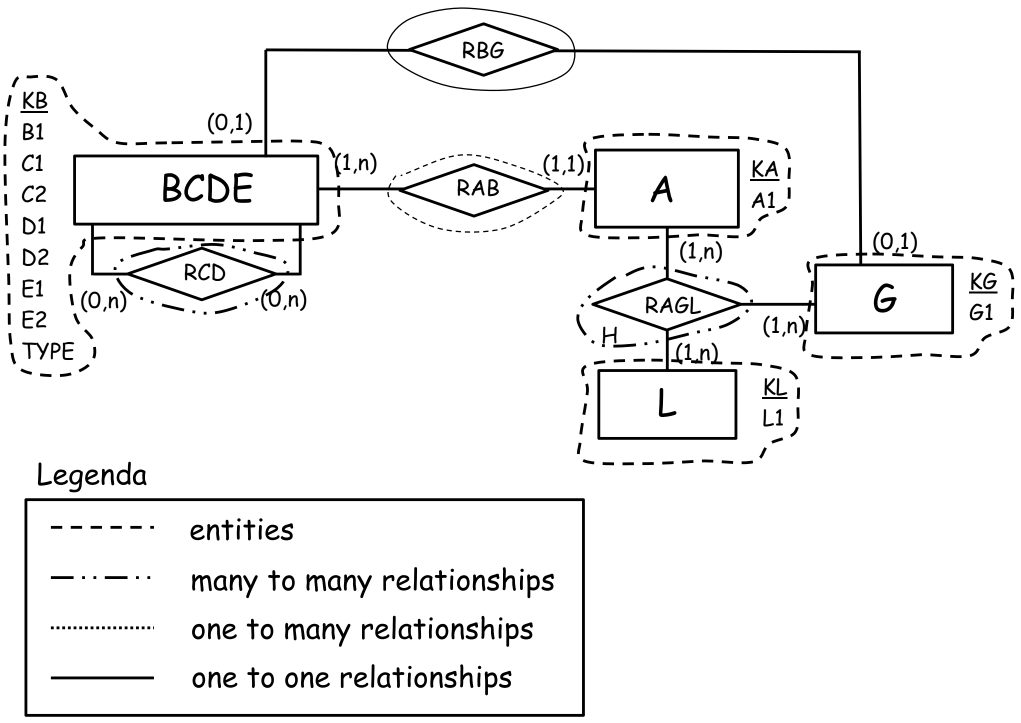
Requirement 1, that says that B, C, D, E are accessed by queries all together, suggests to collapse the four entities A,B,C,D into a unique entity. So we obtain the following schema.



Result of simplification in Exercise 5.1

We have no optimization transformations, since requirements do not refer explicitly to optimizations. So the final schema corresponds to the output of the simplification step.

We have now to apply the translation methodology. In the following figure, we show the different parts of the schema to be transformed.



At the end of the translation process, we obtain the following relational schema.

RBCDE (KB, B1, C1, C2, D1, D2, E1, E2, TYPE)

RA (KA, A1)

RG (KG, G1)

RL (KL, L1)

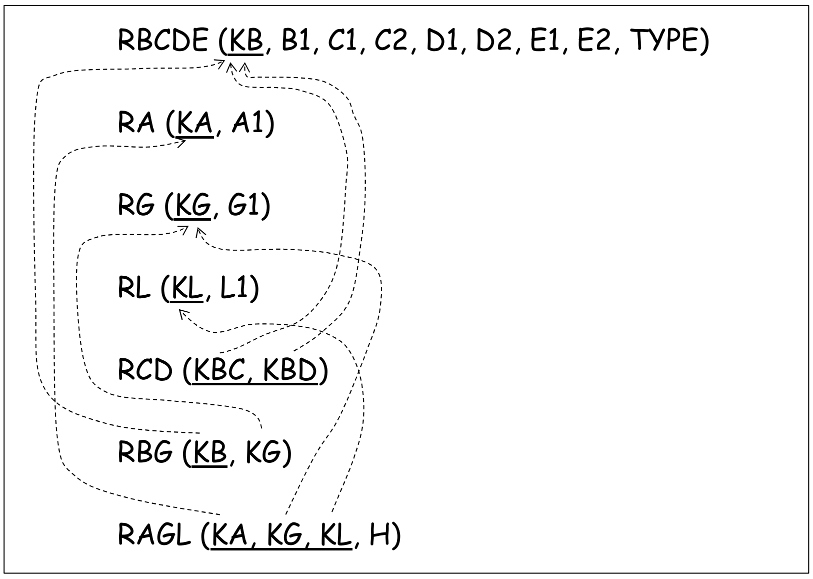
RCD (KBC, KBD)

RBG (KB, KG)

RAGL (KA, KG, KL, H)

Notice that relation RBG is created due to requirements 2 and 3, that lead to choose Solution 3 of the (0,1) to (0,1) case of translation.

We have now to add referential integrity constraints to the schema.



RBCDE (KB, B1, C1, C2, D1, D2, E1, E2, TYPE)

RA (KA, A1)

RG (KG, G1)

RL (KL, L1)

RCD (KBC, KBD)

RBG (KB, KG)

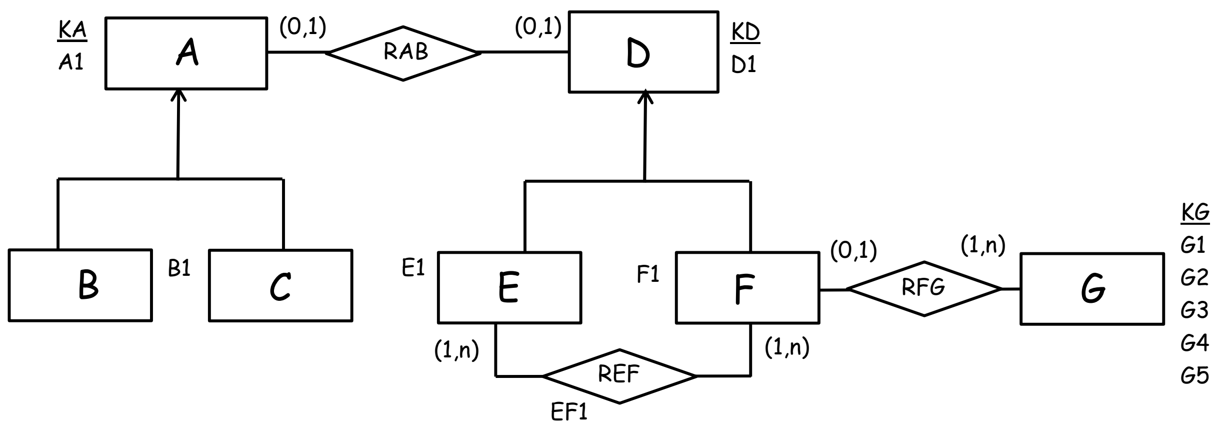
RAGL (KA, KG, KL, H)

This is the final schema for Exercise 5.1.

Part 5 – Lesson 6 – Exercise 5.2 and Exercise 5.3 on the University Database

Exercise 5.2

The input schema is shown below.



Requirements are:

1. Entities B and C are visited by different queries.
2. Entities D, E, and F are visited by the same queries.
3. Attributes KG, G1 and G2 and attributes KG, G3, G4, and G5 of entity G are visited by different queries.

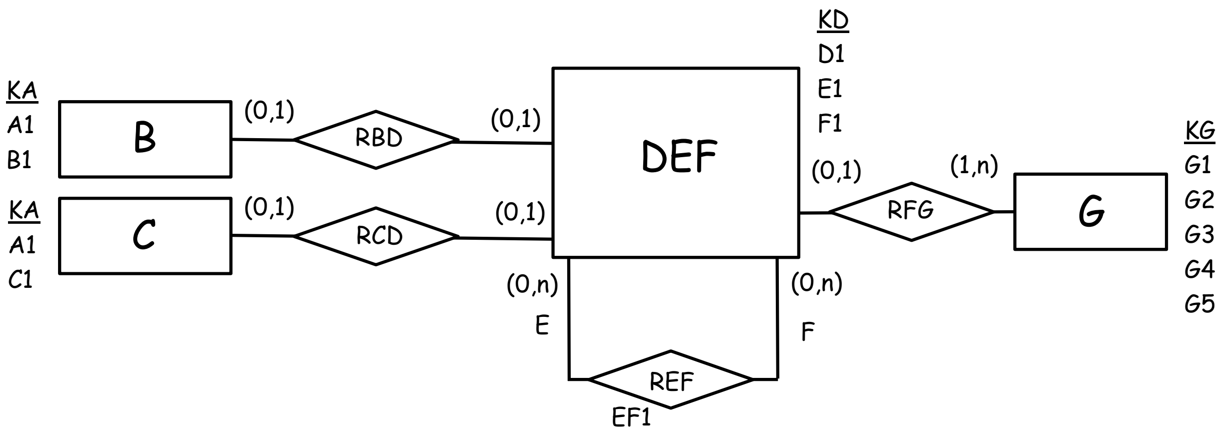
Solution to Exercise 5.2

We have to consider requirements 1 and 2 for the simplification step and requirement 3 for the optimization step. Leading to the following choices:

Simplification

Requirement 1 creates a schema with entity B and C separated.
Requirement 2 collapses D, E, and F into a unique entity.

The output of the simplification step is the schema:

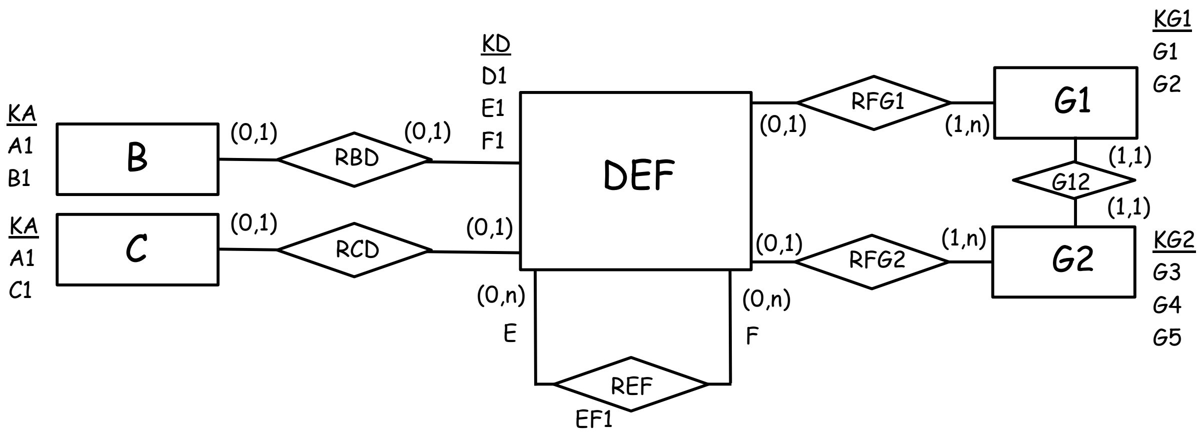


Output of simplification

Optimization

Requirement 3 leads to partition G into two entities.

The output of the optimization step is the schema:



Output of optimization

Translation

The output of the translation step is the following relational schema:

B (KA, A1, B1, KD)

C (KA, A1, B1, KD)

DEF (KD, D1, E1, F1, KG1, KG2)

G1 (KG1, G11, G21, KD, KG2)

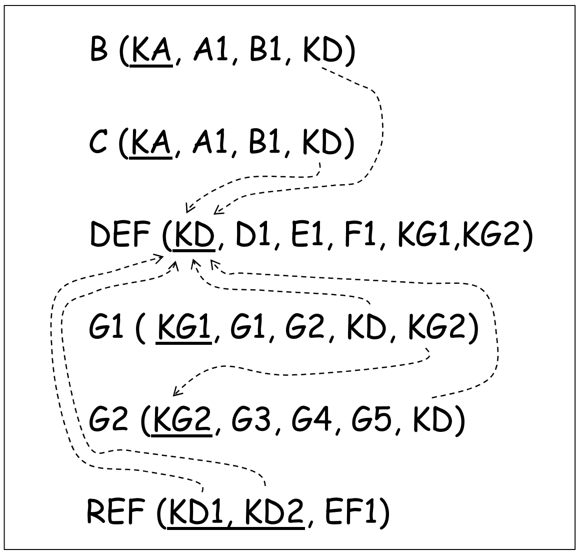
G2 (KG2, G3, G4, G5, KD)

REF (KD1, KD2, EF1)

Output of translation

Notice that we have added the attribute KD to relations G1 and G2 since the minimal cardinality of entity DEF is 0 (Solution 1 of the (0,1) to (1,1) case).

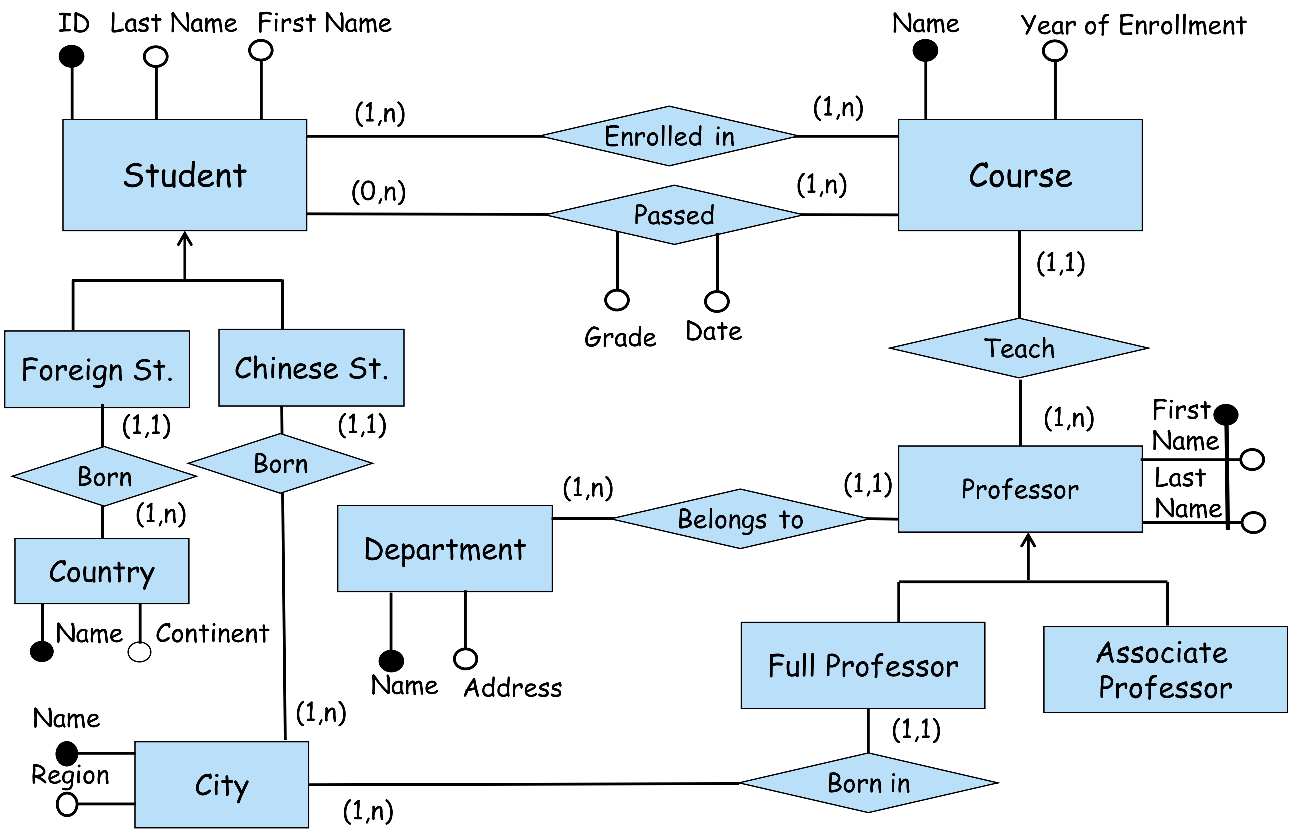
We now add referential integrity constraints leading to the final schema.



Final schema of Exercise 5.2

Exercise 5.3 - University database logical design

The final schema of conceptual design was the one in the following figure.



Here we make two assumptions.

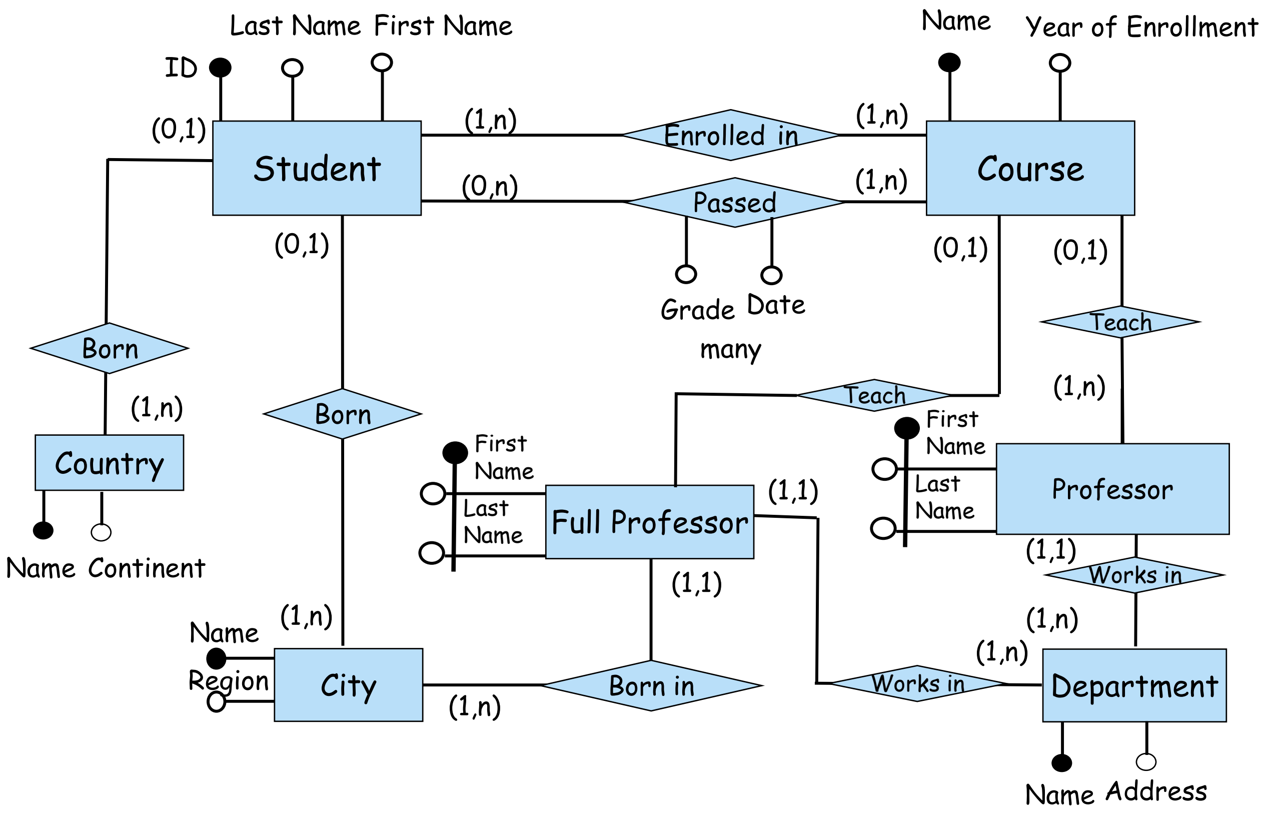
1. Queries on Students visit together Student, Foreign Student and Chinese Student.
2. Queries on Professors visit separately Full Professors and Associate Professors.

Solution to Exercise 5.3 – University database logical design

The two assumptions have to be considered in the simplification step.

Simplification and optimization

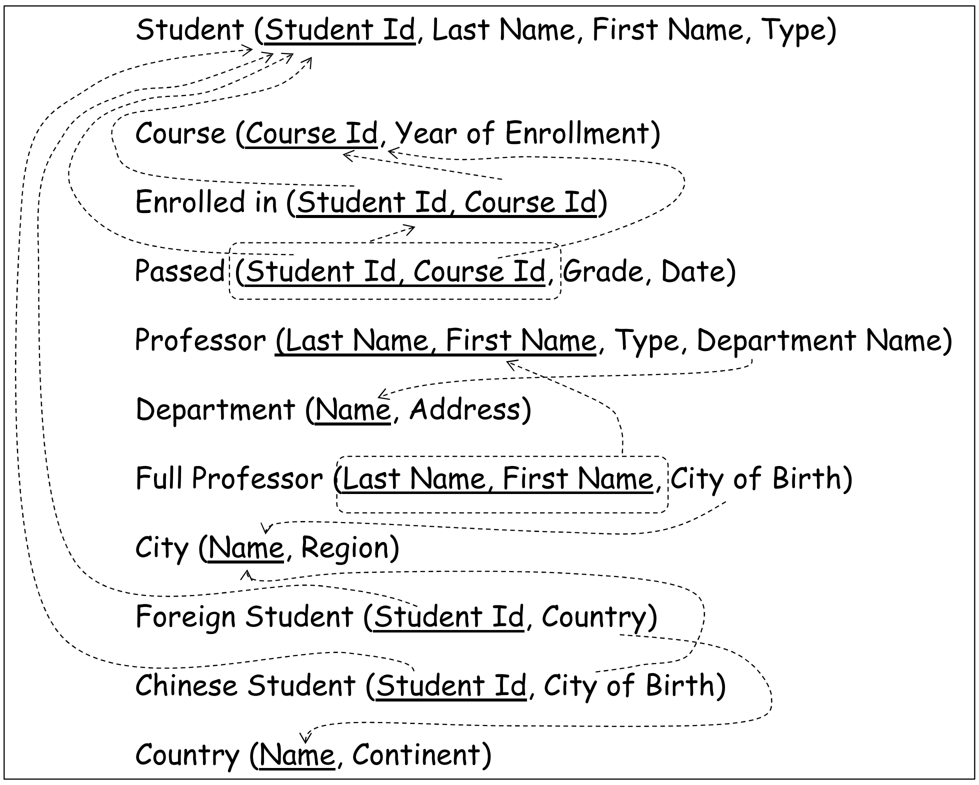
The output of the simplification step and of the optimization step is the following. Notice that now the minimum cardinality associated to the entity Student in the two *Born* relationships with Country and City are both 0. This is due to the fact that some students, namely foreign students, are to be linked with Countries, and other students, namely Chinese Students, are to be linked in Cities.



The result of the translation step is the schema:

Student (<u>Student Id</u>, Last Name, First Name, Type)
Course (<u>Course Id</u>, Year of Enrollment)
Enrolled in (<u>Student Id, Course Id</u>)
Passed (<u>Student Id, Course Id</u>, Grade, Date)
Professor (<u>Last Name, First Name</u>, Type, Department Name)
Department (<u>Name</u>, Address)
Full Professor (<u>Last Name, First Name</u>, City of Birth)
City (<u>Name</u>, Region)
Foreign Student (<u>Student Id</u>, Country)
Chinese Student (<u>Student Id</u>, City of Birth)
Country (<u>Name</u>, Continent)

The inclusion of referential integrity constraints produces the final schema.


Student (Student Id, Last Name, First Name, Type)
Course (Course Id, Year of Enrollment)
Enrolled in (Student Id, Course Id)
Passed (Student Id, Course Id, Grade, Date)
Professor (Last Name, First Name, Type, Department Name)
Department (Name, Address)
Full Professor (Last Name, First Name, City of Birth)
City (Name, Region)
Foreign Student (Student Id, Country)
Chinese Student (Student Id, City of Birth)
Country (Name, Continent)

Final schema of Exercise 5.3

With this last exercise, we end the course. I kept my promise to enable you to produce ER and relational schemas from requirements characterized by significant complexity.

I recall you that to achieve in an introductory course of databases a reasonable skill you have also to attend a course on database programming.

Best wishes for your present and future activity in data management!!!!

Carlo Batini

Concepts defined in Part 5

Part 5 - Logical Design

Phase of Logical Design
- Simplification and Optimization
- Translation

Simplification Phase
- Removing Generalizations

Optimization Phase
- Partitoning of Entities/Relationships
- Merging of Entities/Relationships

Translation Step
- (Translation of an) Entity
- Many to Many Relationship
- One to Many Relationship
- Ternary Relationship
- Recursive Relationship
- (1,1) to (1,1) Relationship
- (0,1) to (1,1) Relationship
- (0,1) to (0,1) Relationship

Part 5 – Exercise assignment

Remember that a more complex method is discussed in Atzeni’s book for this part. Once studied such method in Chapter 7 of the book, solve exercises from 7.1 to 7.4 of the book. Then compare your solutions with solutions provided in the course site.