Transforming *N*-ary Relationships to Database Schemas: An Old and Forgotten Problem

Author:

Rafael Camps
UPC/Barcelona(Spain)
RCAMPS@LSI.UPC.ES
FAX:93.896.77.00
TF: 93.645.01.11

Abstract:

The n-ary relationships, have been traditionally a source of confusion and still are. One important source of confusion is that the term "cardinality" in a relationship has several interpretations, two of them being very popular. But none of the two approaches, nor the two together, allow us to express all the possible cardinality patterns. The transformations from all the possible patterns to database schemas have never been fully discussed by the existing text-books and papers that deal with database design. Using the fourteen ternary patterns as example, we discuss these transformations particularly the transformations from the patterns ignored in the literature.

Index terms:

Relational database design, Functional dependencies, N-ary relationships, UML, SQL.

Transforming *N*-ary Relationships to Database Schemas: An Old and Forgotten Problem

Rafael Camps, Universitat Politècnica de Catalunya (Barcelona, Spain). rcamps@lsi.upc.es

1. Introduction

Although 25 years have past from the birth of the ER model (Chen paper [3]) and several hundreds of papers and books have been written about it, relationships of degree greater than two, the *n*-ary relationships, have been traditionally a source of confusion and still are. The majority of database designers are confused about *n*-ary relationships and do not use them or often use them wrongly. Mistakes in choosing the degree of relationships, wrong cardinalities, and errors in transforming ER models to relational schemas, are very common. The confusion should not surprise us after reading the literature. Some general examples: Teorey [19] says that ternary relationships are needed to express concepts that cannot be represented by several binary relationships, but other authors, as Dey-Storey-Barron [5] or Ullman-Widom [21], say that a higher-degree relationship may always be expressed as several binary relationships. The authors of UML [17] say that it is better to avoid *n*-ary associations because the definition of cardinalities is complicated.

One important source of confusion is that the term "cardinality" in a relationship has several interpretations or meanings. Although the cardinality is always defined as a way to express the desired structural restrictions in a relationship, there are several different approaches to the concept, two of them being very popular. The differences between these two approaches are not relevant for binary relationships, but they become important for a degree greater than two.

During the last decade several researchers have fully analyzed the *n*-ary relationships, trying to clarify their meaning and use in a systematic way. Very good examples of such a task can be Jones-Song [10] [11] and McAllister [15]. But they don't deal with the transformation of the relationships to a relational schema. Moreover, the text-books and papers that deal with database design are usually limited to a very general description of this transformation, and assume only one of the two popular approaches for the cardinality. But none of the two approaches, nor the two together, allow us to express all the possible cardinalities. So, these transformations have never been fully discussed by the existing literature.

The main purpose of this paper is to discuss the transformation of *n*-ary relationships to relational database schemas, specially the transformations from some patterns ignored in the literature. In discussing these transformations we disprove some very common beliefs about *n*-ary relationships. In sections 2 and 3 we present the two major approaches and introduce, using examples, the main concepts to be discussed. The transformation to relational model of ternary relationships using the two popular approaches is discussed in section 4. In sections 5 and 6, functional dependency patterns and embedded cardinalities are analyzed. The transformation for all the functional dependency patterns to normalized tables is presented in section 7. In sections 8 and 9 we discuss about minimum cardinality values and we do some comments on the traditional decomposition problem. And in the last section we summarize some conclusions.

2. Two popular kinds of cardinality

Let's see the example of fig.1. *Work* is a typical *one-to-many*, ***1:N*** , relationship that associates instances of the entity *Department* with instances of *Employee* entity. The symbol used in fig.1 to represent the relationship is the traditional diamond. The symbols that can be found in the literature to represent the classical *one* and *many* cardinality values are very varied, but we will use here a ***1*** and an ***N*** respectively.

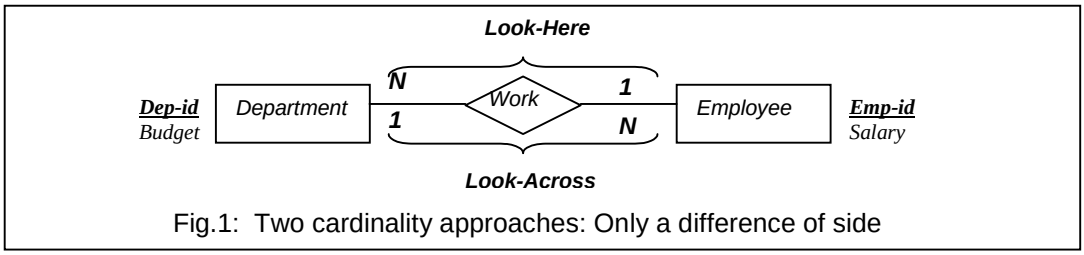


Fig.1: Two cardinality approaches: Only a difference of side

Anyway, here we are not worried about the symbols but about the semantics. The most popular approach to the concept of cardinality is the adopted by authors such as Chen [3], Teorey [20], Howe [9], Ullman-Widom [22], Jones-Song [11], Loizou-Levene [12], McAllister [15][16], and Mannila-Raiha [13], or by methods as UML [18]. We will name this approach as *Look-Across* (following Song-Evans-Park [19]) or *LA* for short. McAllister names it as *Chen style*. According to the *LA* approach, the meaning of the ***1*** and ***N*** symbols in fig.1 is: Each instance of *Employee* cannot be associated to more than ***one*** instance of *Department* via the relationship *Work,* but ***many*** instances of the entity *Employee* can be associated to each instance of the entity *Department*.

The other popular approach is the adopted by authors as Elmasri-Navathe [7], Batini-Ceri [1], Dey-Storey-Barron [5], and by methods as Merise [17] or Yourdon-Method [23]. We will label this approach as *Look-Here* (following Song-Evans-Park [19]) or *LH* for short. McAllister names it as *Merise style*. According to the *LH* approach, the meaning of the ***1*** and ***N*** symbols in fig.1 is: Each instance of *Employee* participates in no more than ***one*** instance of the relationship and each instance of *Department* can participate in ***many*** instances of the relationship *Work*. We could see the difference as a cardinality notation difference, simply a change of side of the cardinality symbols. In fact, the meaning of ***1:N*** relationships in terms of *functional dependencies* between keys, is exactly the same in both approaches:

Emp-id → *Dep-id*

When the cardinalities in the two sides are symbols ***N***, i.e. no functional dependencies exist, actually there is no differences between the two approaches. But the difference of meaning between them becomes very relevant when the degree is greater than two and at least one cardinality ***1*** exists, as we will see below.

If in an ER diagram, *R* is a relationship between *n* entities, $E_1, E_2, ..., E_n$ then each instance of *R* is an instance of the *n* degree product $E_1 \times E_2 \times ... E_i ... \times E_n$.

A cardinality symbol in the leg of E_k represents :

- in *LA* approach: The maximum number of instances of E_k that can be associated to each instance of the *n-1* degree product $E_1 \times E_2 \times ..\times E_{k-1} \times E_{k+1} \times ...\times E_n$. In other words, the maximum number of instances of E_k that may appear in association with a set of the other *n-1* entities.

- in *LH* approach: The maximum number of instances of *R* where E_k can participate. That is equivalent to say; the maximum number of instances of *n-1* degree product $E_1 \times E_2 \times ..\times E_{k-1} \times E_{k+1} \times ...\times E_n$ that can be associated to each instance of E_k .

Hence, if *R* is a ternary relationship between *A B* and *C*, and we are using *LA* cardinalities, and a ***1*** symbol appears in the leg of *A*, then each pair of instances *b,c* (*b* from *B* and *c* from *C*)

cannot be associated to more than one instance *a* (from *A*). But in the *LH* approach, each instance *a* cannot be associated to more than one pair of instances *b,c*.

At first glance it could look like the two approaches are complementary, so that using they together we could express all the structural constraints we need in *n*-ary relationships. But this is not true. In a ternary relationship, for example, there could be 12 different cardinalities (as we will see in section 5) but each one of the two popular approaches explicitly expresses only 3 of these cardinalities.

Note than in an *n*-ary relationship *R* each one of its instances involves *n* entity instances.

An *n*-ary relationship displaying *n* cardinalities, all of them according to the *LA* approach, will be qualified as *LA n-ary* relationship. And it will be qualified as *LH n-ary* if the *n* cardinalities are according to the *LH* approach.

Very often two values are used for each cardinality, the minimum and the maximum values. But the traditional cardinality notation it is only using the maximum value. We will assume that the minimum value is zero. We will discuss the case of minimum value = 1 in section 8.

3. An example: the *1:N:N* case

We will analyze all the ternary patterns and their transformation to relational database schemas. But we will start facilitating the comprehension of the importance of the differences between the two popular approaches, *LA and LH*, using an example. We will interpret the diagram of a ternary relationship with ***1:N:N*** cardinalities, fig.2, according to both approaches, and we will discuss their transformation to relational schemas. The entities are *Product* (identified by *Pro-id*), *State* (identified by *Sta-id*) and *Dealer* (identified by *Dea-id*). The relationship is named *Concession*. Each instance of the *Concession* relationship, is a concession of distribution to a dealer of a product into a state, and has a date of concession (*Con-date*).

3.1. *LA* approach

According to *LA, Look-Across,* approach, the semantics of fig. 2 is as follows: a dealer company can distribute several products and a product can be distributed by several dealers (each pair of one dealer and one state can be associated to many products and each pair of one product and one dealer can be associated to many states). But in each state a product cannot be distributed by more than one dealer.

In terms of functional dependencies, FDs, this semantics can be expressed as follows (the obvious internal dependencies in each entity, between their identification and their attributes, are not represented):

(Pro-id , Sta-id) → *Dea-id* because the ***1*** in the side of *Dea-id*
(Pro-id, Sta-id, Dea-id) → *Con-date* because *Con-date* is an attribute of the ternary relationship

These two dependencies can be reduced to: *(Pro-id , Sta-id)* → *(Dea-id, Con-date)*

A relational representation of this *LA* ternary relationship could consist of one table for each entity and one table for the *Concession* relationship (keys are underlined):

State (Sta-id, Sta-capital) *Product (Pro-id, Pro-type)* *Dealer (Dea-id, Dea-address)*

Concession (Pro-id, Sta-id, Dea-id , Con-date)

Each one of the attributes *Dea-id, Pro-id* and *Sta-id* in the table *Concession* should be declared as *not null foreign key*.

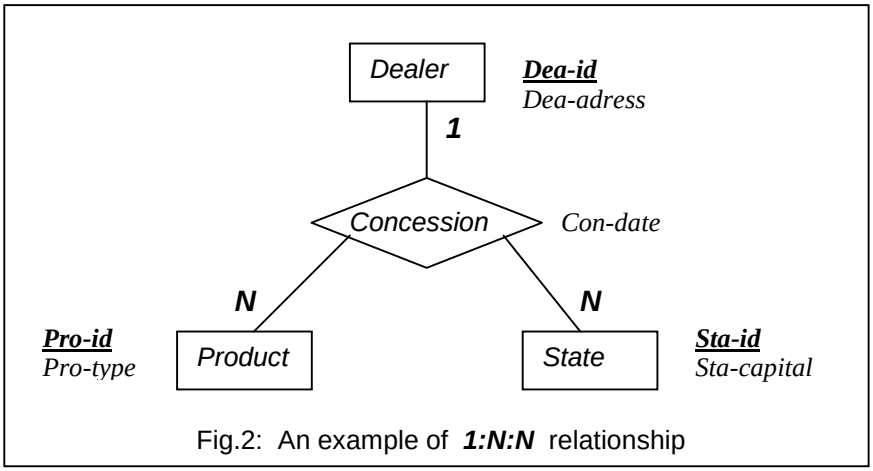


Fig.2: An example of ***1:N:N*** relationship

The possible lack of a dealer with a concession for a given pair product-state (because min=0) is expressed by the non-presence of a row in the *Concession* table for that pair of *Pro-id Sta-id.* Note that it is not possible to reduce the number of tables without losing their normalization (BCNF).

3.2. *LH* approach

Now let us interpret the fig.2 according to the *LH, Look-Here,* approach. Now, the ***1*** cardinality in *Dealer*, means that a dealer cannot participate in more than one instance of *Concession*, so it cannot be associated to more than one pair (*product,state*). In other terms, each dealer can distribute only one product and only in one state. The ***N*** in *Product* means that a product can be associated to many pairs (*state,dealer*). Expressing the model in terms of FDs we can write *Dea-id* → *(Sta-id , Pro-id)* because of the ***1*** in the side of *Dealer.* The attribute of the relationship depends also on *Dea-id*, so *Dea-id* → *Con-date.* So, each dealer has only one concession date*.* Also there are the obvious FDs from the key of an entity to its own attributes. The tables for the three entities will be the same as in the *LA* case, but for the *Concession* relationship could be (with same three *not null foreign keys* than before):

Concession (Dea-id, Pro-id, Sta-id, Con-date)

When a dealer has no concession, the fact is represented by a lack of a corresponding row in the *Concession* table.

If by error we input the diagram of fig.2, being an *LH* (or *LA*) ternary, to an automatic tool that interprets it as being *LA* (or *LH*) ternary, it would do completely erroneous assumptions, and the resulting relational schema would be wrong. But now we cannot solve the problem by simply reversing the position of cardinalities as we did in the binary case! It is very important to note that it is not possible to represent the semantics of this *LH* ternary (that corresponds to the pattern *#6* (fig.4) that will be discussed in sections 7.3 and 8.2) by an *LA* ternary. And the semantics of the *LA* ternary case (3.1) cannot be represented by an *LH* ternary.

3.3. Non-basic constructs

Note that the case of fig.2 interpreted as an *LA* ternary, cannot be modeled by binary relationships without adding textual constraints, except if we use some non-basic constructs. Remember, in the *LA* case a dealer can have the concession of several products in several states, but no more than one dealer can distribute a given product in a given state. We can express this *LA* ternary case semantics using binary relationships if we give them the ability of being related to a third entity; it is the special construct named *associative entity* (*intersection entity* or *gerund)*. In UML class diagrams it is represented by the *associative class*. In figure 3 the associative class *PS*, implicitly identified by the pair (*Pro-id, Sta-id*) can have attributes that are not from the ternary but from the binary relationship, as for example the price of the product in the state. The binary relationship *Concession* becomes the equivalent of the ternary*.* Other non-basic constructs as the *weak entity* (*ID-dependency*) [3] or the *composite entity* [7] could allow us to model the case using binary relationships, but that is not possible if we use only the ER basic constructs. To a study of decomposition of ternary relationships into two binary, see Jones&Song [11] and McAllister [16].

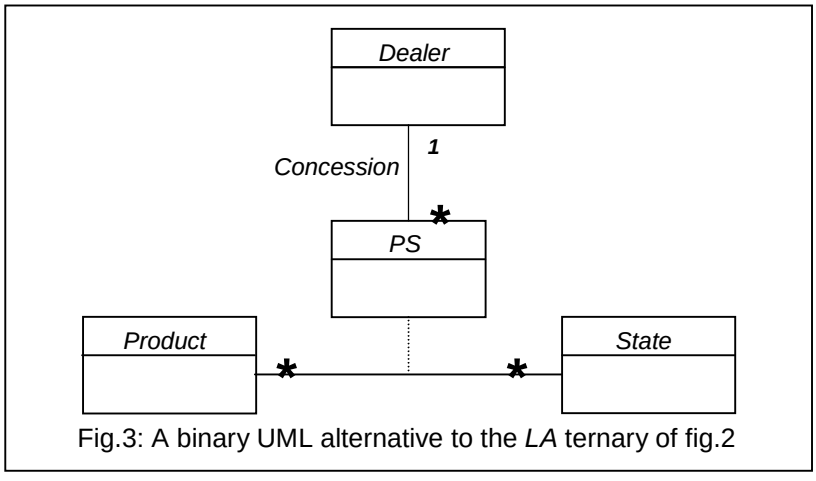


Fig.3: A binary UML alternative to the *LA* ternary of fig.2

4. Relational representation of *LA* and *LH* ternary relationships

In fig.4, we show translations of all the *LA* and *LH* ternary relationships *R* between entities *A, B* and *C*, to equivalent and normalized (at least in BCNF) tables *R.* We assume that the relationships are properly used, so all the FDs in the actual domain to be modeled, are exactly the same FDs implied by the diagram. The attributes of the relationship *R* are represented by the letter *r.* Keys are underlined. The FDs between the keys of *A, B* and *C*, are also represented. The ***N:N:N*** case has no FDs between keys, but obviously there still is the dependency of the attribute *r* on the ternary key: *(A-id,B-id,C-id)* → *r*.

In figure 4 we see the seven different FD patterns that are possible with the *LA* and*LH* approaches: four of them are using *LA* and other four are using *LH,* but one pattern is common to both approaches. As we will see in section 7 there still are seven more possible FD patterns, and three of them are special cases that cannot be represented using none of the two popular approaches. In the literature only four patterns are considered, the *LA* or the *LH* patterns depending on the approach adopted by the author.

In each of the ER diagrams of fig.4, except in the last one, it exists at least a ***1*** symbol. Then the relational representations for each of the two approaches are very different because the meaning of the ***1*** differs. In the last diagram, when the three symbols are ***N***, the two approaches have the same relational representation because in this case there is no FD between the keys of the entities *A, B*, *C*.

Each *LA* ternary relationship, if properly applied, cannot be represented by less than four normalized (BCNF) tables. The table *R* will have as much two-attribute keys as ***1***s exist in the relationship. If no ***1***s exist, the ***N:N:N*** case, then the key will be a three-attribute key. Each *R* table for the *LH* approach will have as much one-attribute keys as ***1***s exist. The ***N:N:N*** case is exactly like in the *LA* approach. More details can be found in section 7.

Remember that we are assuming that the minimum cardinallity value is 0. The relational representation when the minimum cardinality value is ***1***, is discussed in section 8.

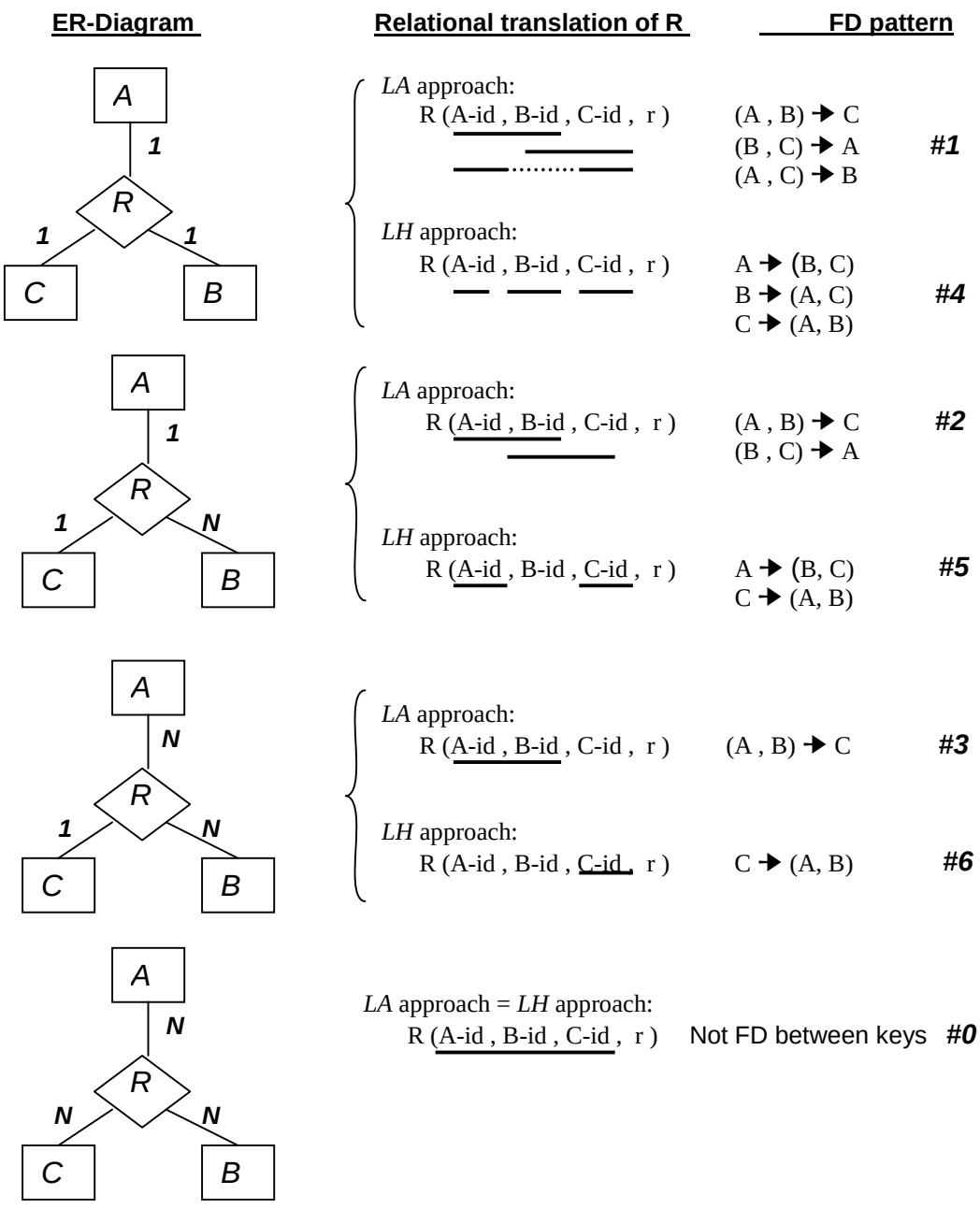


Fig.4 FD patterns and relational representation of *LA* and *LH* ternary relationships

5. Cardinality and three types of FD

As we have just seen, an *n*-ary relationship with *all-**N*** cardinalities, is translated to a table with an *n* attribute key. But in all the other cases, the keys have less than *n* attributes. The keys of *LA* relationships are always composed by *n-1* attributes, and the keys resulting from *LH* relationships are all single attribute. In terms of FDs, this can be expressed as follows for ternary relationships: The FDs represented by the value ***1*** in an *LA* cardinality, are of the form *(X,Y) → Z*. The FDs represented by the value ***1*** in an *LH* cardinality, are of the form *X → (Y,Z).*

Between each one of the three pairs of entities of a ternary relationship, we can figure an embedded binary relationship. A ***1*** in a cardinality of such embedded relationship is a dependency of the form $X \rightarrow Y$. These embedded binary dependencies are often labeled as *Semantically Constraining Binary, SCB* [10]. We will label them depending on their form, as:

LA	FD	if	it is	$(X,Y) \rightarrow Z$
LH	"	"	"	"	$X \rightarrow (Y,Z)$
Binary	"	"	"	"	$X \rightarrow Y$

In order to describe all the structural constraints between three related entities, so to fully describe a ternary relationship, we should define twelve different cardinalities:

- Three ternary cardinalities, where a ***1*** means an *LA* FD, to describe the maximum number of *Z*s associated to each pair *XY*
- Three ternary cardinalities, where a ***1*** means an *LH* FD, to describe the maximum number of pairs *YZ* associated to each *X*
- Six binary cardinalities (two for each pair of entities) where a ***1*** means a binary FD, to describe the maximum number of *Y*s associated to each *X*.

Figure 5 is like a framework for FD patterns and it shows graphically the 12 possible cardinalities (it is equivalent to the "cardinality tables" from McAllister [16]). The external lines, the triangle sides, are to describe the binary dependencies and can be viewed as if they were binary relationships (*SCB* relationships [10]) being an integral part of the ternary relationship. In order to maintain an uniform notation, the binary FDs will be always noted with a ***1*** in the side of the dependent entity.

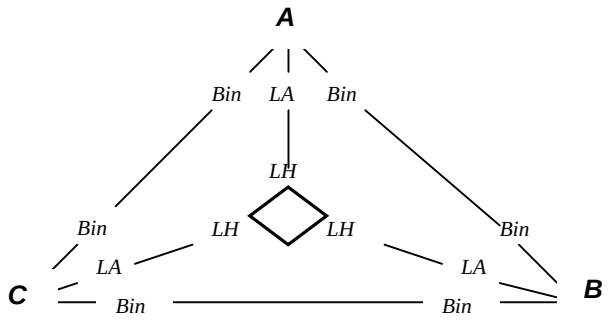
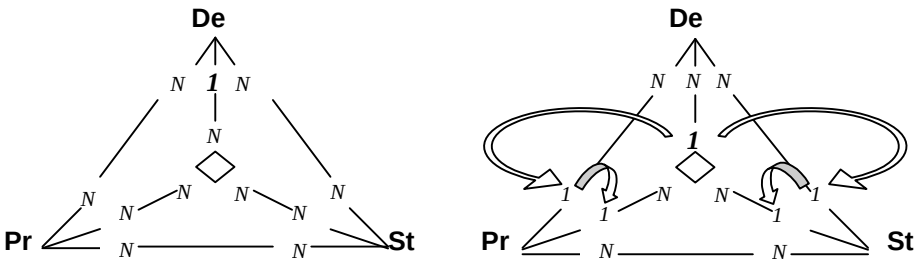


Fig.5: Twelve cardinalities

In figure 6 we represent the frameworks of the FD pattern for the *Concession* example of fig.2, for each one of the approaches *LA* and *LH*. Note that in the graphical FD pattern of fig.6b, we express the binary dependencies with a ***1*** in the side of the *Product* and *State* entities, although we are representing an *LH* case.



a) FD pattern (#*3*) for *LA* ternary b) FD pattern (#*6*) for *LH* ternary

Fig.6: The example of ***1:N:N*** relationship (fig2) revisited

In figure 7 we represent the framework for the FD patterns for all the *LA* and *LH* ternary relationships (patterns *#0* to *#6*) that appear in fig.4.

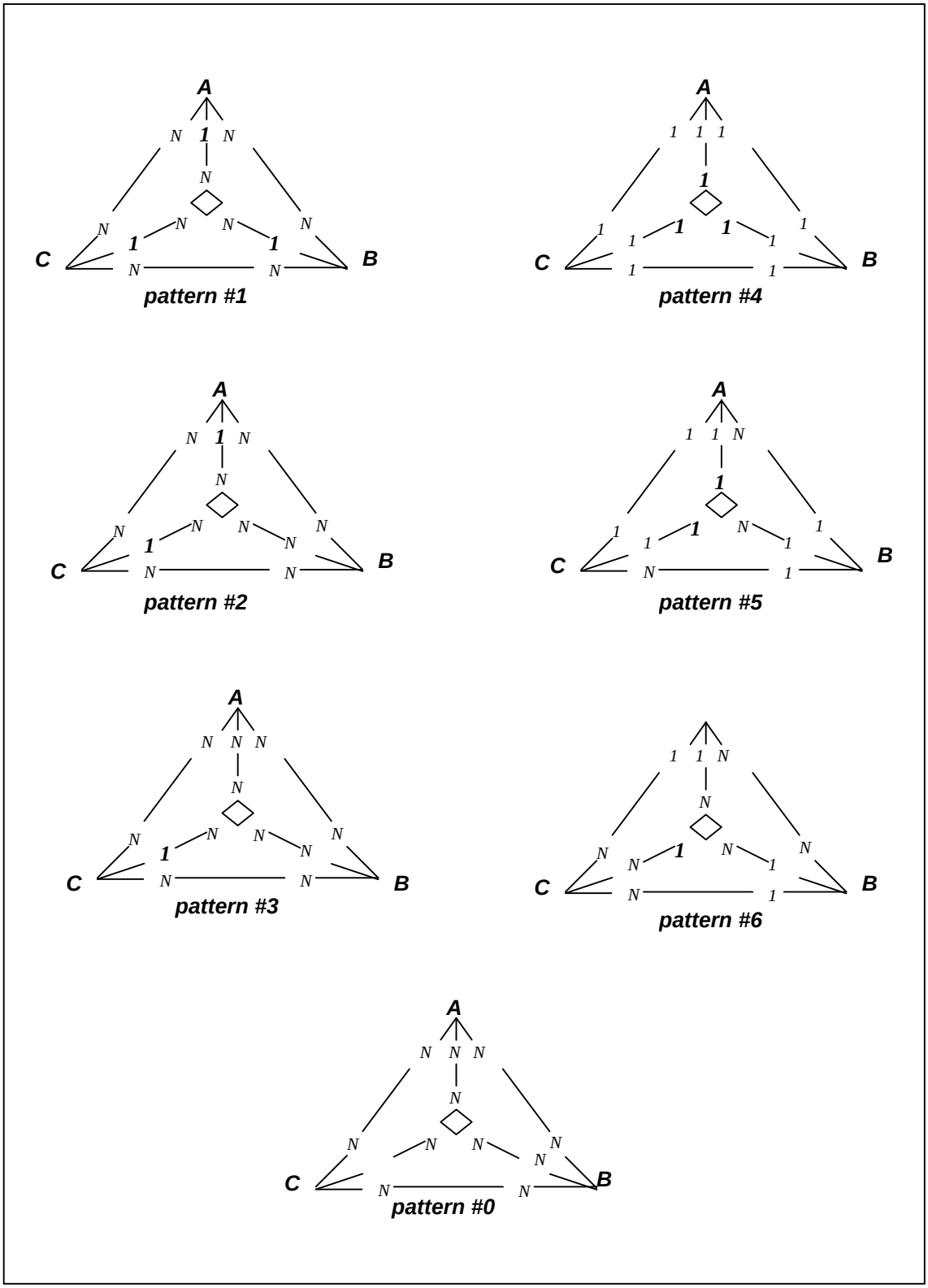


Fig.7: Frameworks for the FD patterns *#0* to *#6*

6. Implied and imposed cardinalities

In practice, during the preliminary steps of a database design, the relationships are often not used strictly. But, when the diagram is to be translated to a relational schema, perhaps using an automatic tool, it should be complete, with true *n*-ary relationships used properly, expressing the full set of FDs from the domain being modeled. But, as we will see in this section, the *LH* or *LA* approaches are not powerful enough to represent the full set of FDs of some cases.

6.1. Implied cardinalities

Each ***1*** in an *LH* ternary is an *LH* dependency and it is equivalent to two binary dependencies because

$$X \rightarrow (Y,Z) \;\Leftrightarrow\; X \rightarrow Y \text{ and } X \rightarrow Z$$

So, each ***1*** in an *LH* ternary, implies two ***1***s in the binary cardinalities. In the example of fig.6b, the ***1*** in the *LH* cardinality near the *Dealer,* produces two implied ***1***s because the dependency *Dealer* → (*Product,State*) implies two binary dependencies. The arrows in fig.6b are showing the FD implications:

Dealer → *Product* and *Dealer* → *State*.

But for an *LA* ternary (without imposed binary relationships) no binary FD is implied, so the implied binary cardinalities are always ***N:N***. For example, in fig.6a, the *LA* dependency (*Product,State*) → *Dealer* does not produce any binary dependency between pairs of entities.

Each ***1*** corresponding to a binary embedded cardinality, implies a ***1*** in an *LA* dependency because

$$X \rightarrow Y \;\Rightarrow\; (X,Z) \rightarrow Y$$

For example, in fig.6b the ***1*** in the binary dependency *Dealer* → *State,* produces an implied ***1*** in the *LA* cardinality near *State* because (*Dealer,Product*) → *State* . Symmetrically, the ***1*** in the binary dependency *Dealer* → *Product* produces an implicit ***1*** for (*Dealer,Sate*) → *Product.* So, each ***1*** in an *LH* cardinality produces four derived ***1***s: two binary and two *LA*.

A third type of implication is due to the transitivity property (see pattern *#9* in section 7.4):

$$X \rightarrow Y \text{ and } Y \rightarrow Z \;\Rightarrow\; X \rightarrow Z$$

A consequence of the FD implications is that not all the FD patterns obtainable combining the ***1*** and ***N*** values of the 12 cardinalities are valid. For a detailed analysis of implications between FDs (Armstrong rules) see McAllister [15]. The FD patterns in fig.4, do not show the implied, derived, dependencies.

Note that the set of bold ***1***s in each framework of figures 7, 10 and 12, correspond to a minimum set of FDs from wich all the non-bold ***1***s are derived.

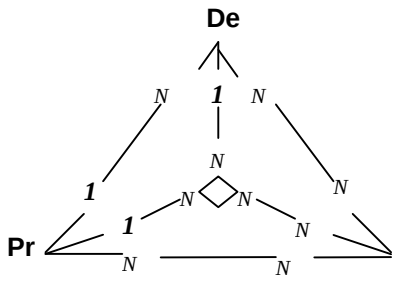
6.2. Imposed cardinalities

Remember that the semantics of fig.6a (or fig.2) is mainly a restriction; "A product in a state cannot have more than one dealer". It is an *LA* constraint: (*Product,State*) → *Dealer*

Now let us assume that we add the following constraint, "Each dealer cannot distribute more than one product": *Dealer* → *Product* . This new imposed restriction is an binary FD and it is not an independent binary relationship, but an integral part of the ternary, because each concession, each instance of this ternary, must relate one product, one state and one dealer. See fig.8. This imposed FD, produces a derived *LA* dependency: (*Dealer,State*) → *Product.*

Note that this pattern cannot be represented by any of the two popular approaches, *LA* and *LH*, because each one is expressing only three of the twelve possible cardinalities. They don't allow us to explicitly express the binary cardinalities for a ternary relationship. We will analyze this FD pattern in 7.5 (pattern #*12* in fig 12).

Elmasri and Navathe [7] propose the use of an ER notation representing *LA* and *LH* dependencies. So, in practice, what they are suggesting for *n*-ary relationships is the simultaneous use of both approaches. But this would not allow us to explicitly express binary cardinalities and therefore some FD patterns would be impossible to be expressed.



LA FD: (*Product,State*) ➔ *Dealer*
Binary FD: *Dealer* ➔ *Product*

The *1* in the *LA* cardinality, near the entity ***De***, represents the implied *LA* FD: (*Dealer,State*) ➔ *Product*

Fig.8: An imposed binary FD, to an *LA* ternary

See Dullea [6] and Jones-Song [11] for an analysis of the permitted impositions.

7. Valid FD patterns and their transformation to relational tables

As we have seen, it is possible to define 12 cardinalities for a ternary relationship. For $n = 5$ there are 180 cardinalities. For a relationship of degree *n*, it is possible to define $3^n-2^{n+1}+1$ different cardinalities (see McAllister [15]).

Each one of the cardinalities can have two values, ***1*** or ***N***. For a ternary relationship, combining all the cardinality values we obtain $2^{12} = 4096$ possibilities. But not all the combinations are valid. Moreover some combinations are equivalent because the symmetry of the ternary relationship. After eliminating all the impossible and all the redundant patterns, we obtain 14 unique FD patterns (see McAllister [15]) or closed set of FDs. In fact each different pattern corresponds to a *closure operation* (see Demetrovics [4]).

We will classify these fourteen FD patterns in five groups:

No-FD-pattern : ***#0***	***SCB-patterns*** **:** ***#7 #8 #9*** and ***#10***
LA-patterns **:** ***#1 #2*** and ***#3***	***Special-patterns*** **:** ***#11 #12*** and ***#13***
LH-patterns **:** ***#4 #5*** and ***#6***	

The first three groups are the patterns in fig.4 and fig.7 (patterns *#0* to *#6*). Their translation to relational tables has been discussed in section 4.

In order to facilitate the task of the reader that is also consulting the papers from McAllister [16] Jones-Song [11] and Thalheim [21] it follows an equivalence numbering of the cases:

our pattern #	0	1	2	3	4	5	6	7	8	9	10	11	12	13
McAllister [16]	--	--	--	--	10	6	3	1	2	5	7	4	8	9
Thalheim [21]	--	--	--	8	6	--	3	1	9	2	4	7	--	5
Jones-Song[11]	12	--	--	--	--	2,3,4	5	10	7	9	8	11	6	1

Fig.9: Equivalencies

7.1. *No-FD-pattern*

The pattern with no FDs (all the twelve cardinalities are ***N***) is the pattern *#0* (fig.4 and section 4). It is the only pattern that can be represented by any of the two approaches, *LA* or *LH.*

7.2. *LA-patterns*

The three patterns *#1 #2* and *#3* (fig.7 and section 4), together with pattern *#0*, correspond to the four classical *LA* ternaries that can be represented by the notations of Chen, Teorey, Howe, Ullman-Widom, Jones-Song, Loizou-Levene, McAllister, Mannila-Raiha, UML, IEFDX1, etc. The three patterns of this group are characterized by having at least one *LA* FD, but neither binary FD*s* nor *LH* FDs. As we will see later, all the other ten patterns (*#4* to *#13*) can be viewed as one *LA* ternary relationship with imposed binary dependencies, *SCB*s. Observe that in pattern *#1* all the implied binary cardinalities are ***N*** although all the ternary cardinalities are ***1***.

Note that none of these three *LA n*-ary relationships can be represented by *LH* relationships.

7.3. *LH-patterns*

The three patterns *#4 #5* and *#6* of fig.7, along with pattern *#0*, correspond to the four classical *LH* ternaries that can be represented with the notations of Elmasri-Navathe, Batini-Ceri, Dey-Storey-Barron, Merise, Yourdon-Method, etc. The three patterns of this group are characterized by having: at least one *LH* FD, and all their binary FDs are implied by the *LH* FDs (each ***1*** in an *LH* ternary implies two binary FDs) and all the *LA* are implied by the implied binary FDs. We show now the *LH* FDs and the implied FDs.

pattern	*LH* FDs	implied FDs	view as *LA* ternary
#4	$A \rightarrow (B,C)$	$\Rightarrow (A,C) \rightarrow B$	***1:1:1*** *LA* + two imposed ***1:1***
	$B \rightarrow (A,C)$	$\Rightarrow (A,B) \rightarrow C$	
	$C \rightarrow (A,B)$	$\Rightarrow (C,B) \rightarrow A$	
#5	$A \rightarrow (B,C)$	$\Rightarrow (A,C) \rightarrow B$	***1:1:1*** *LA* + two imposed ***1:N*** ($A \rightarrow B$ $C \rightarrow B$)
		$\Rightarrow (A,B) \rightarrow C$	
	$C \rightarrow (A,B)$	$\Rightarrow (C,B) \rightarrow A$	or + one imposed ***1:1*** ($C \leftrightarrow A$)
#6	$C \rightarrow (A,B)$	$\Rightarrow (C,A) \rightarrow B$	***1:1:N*** *LA* + one imposed ***1:N*** ($C \rightarrow A$)
		$\Rightarrow (C,B) \rightarrow A$	

For example, in the pattern *#6* (is the case of fig.2 considered as *LH*) the *LH* dependency is $C \rightarrow (A,B)$ that is equivalent to $C \rightarrow A$ and $C \rightarrow B$. From $C \rightarrow A$ we can derive the *LA* dependency $(C,B) \rightarrow A$, and from $C \rightarrow B$ we can derive the *LA* dependency $(C,A) \rightarrow B$.

In the column "view as *LA* ternary" we show that all the three patterns of this group can be seen as produced by one *LA* ternary relationship with some imposed binary FDs. Note that this *LA* view has some redundancy, for example the pattern *#6* can be expressed using a single *LH* dependency, $C \rightarrow (A,B)$ but the *LA* view oblige us to explicitly express three ***1***s.

The transformation to a relational schema of the ternary relationships produced by these three patterns, especially the pattern *#6*, is discussed in sections 3.2 and 8.2.

7.4. *SCB-patterns*

The seven FD patterns of the remaining groups (***SCB****-patterns* and ***Special****-patterns*) are characterized by having some imposed binary FDs. The *SCB-patterns* group, patterns *#7* to *#10*, is characterized by having all its *LA* and *LH* FDs being implied by the binary FDs (the bold ***1***s in figure 10). The four ***SCB*** patterns, as the four patterns we have just seen in the previous section, can be viewed as produced by one *LA* ternary relationship plus some imposed binary constraints:

pattern		binary FDs	implied FDs	view as *LA* ternary.
#7	*A → B*	*⇒ (A,C) → B*		***N:1:N*** *LA* + one imposed ***1:N***
#8	*A → B*	*⇒ (A,C) → B*		***1:1:N*** *LA* + one imposed ***1:1***
	B → A	*⇒ (B,C) → A*		
#9	*A → B*	*⇒ (A,C) → B*		***N:1:1*** *LA* + two imposed ***1:N***
	B → C	*⇒ (A,B) → C*		
		⇒ A → C ⇒ A → (B,C)		
#10	*A → B*			***1:1:N*** *LA* + one imposed ***1:N***
	C → B	*⇒ (A,C) → B*		+ one imposed ***1:1***
	B → A	*⇒ (B,C) → A*		
		⇒ C → A ⇒ C → (A,B)		

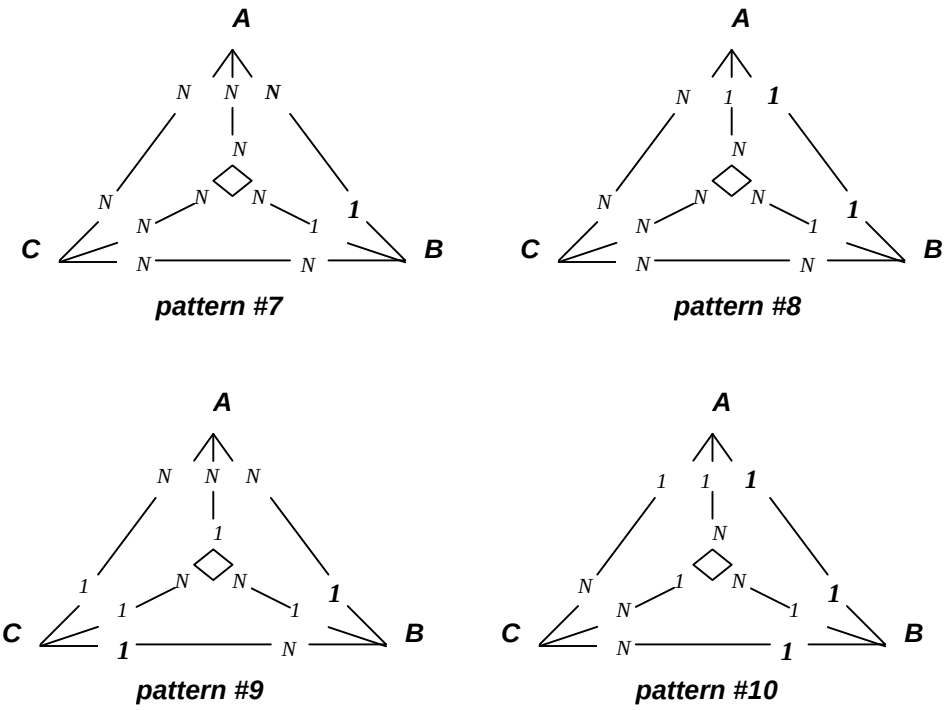


Fig.10: ***SCB-patterns***

Note that because an *LH* FD is equivalent to two binary FDs, the three ***LH*** patterns (*#4 #5 #6*) in section 7.3, could be seen as members of the ***SCB*** group.

We will discuss now relational representations for the four ***SCB*** patterns (fig.10). We will use one table for each entity:

A (A-id, a) B (B-id, b) C (C-id, c)

and we can add two tables in order to represent the relationship *R* with the attributes *r* , and the imposed binary FDs. For example, the FDs *A → B* and *B → A* can be represented by a table S :

S (A-id , B-id) with A-id and B-id defined as two alternative keys, not accepting nulls and as two foreign keys referencing tables A and B

The attributes *r* of a ternary relationship *R*, are functionally dependent on *(A-id,B-id,C-id).* But because the binary FDs of the patterns, the dependency of *r* can be simplified to:

pattern #*7* *(A-id,C-id)* → *r*
pattern #*8* *(A-id,C-id)* → *r* *or* *(B-id,C-id)* → *r*
pattern #*9* *(A-id)* → *r*
pattern #*10* *(C-id)* → *r*

For example, the table R will be R (A-id, C-id, r).

As a result, the schemas for the four ***SCB*** patterns can have five tables: the three tables A, B and C, and two additional tables, R and S, as in figure 11. Because all the instances of the tables are to be an integral part of the ternary relationship, we must add some *inclusion* restrictions [12] in each schema.

Pattern	**Additional Tables**	**Additional restrictions**
#7	S (A-id , B-id) R (A-id , C-id , r)	R•A-id ⊆ S•A-id (1) S•A-id ⊆ R•A-id (2)
#8	S (A-id , B-id) R (B-id , C-id , r)	
#9	S (B-id , C-id) R (A-id , B-id , r)	R•B-id ⊆ S•B-id (3) S•B-id ⊆ R•B-id (4)
#10	S (A-id , B-id) R (C-id , B-id , r)	

Fig.11: Additional tables and restrictions

For example, for pattern #*7* the two following conditions must be true:

- If an instance of A is related to, at least one instance of C, it should be related to one instance of B. This is expressed by the restriction (1) in figure 11.
- If an instance of A is related to an instance of B, it should be related to, at least one instance of C. This is expressed by the restriction (2) in figure 11.

Obviously, the attributes R•C-id, S•B-id and S•A-id should be declared as foreign keys of tables C, B and A respectively.

The restrictions (1) and (3) are key-based inclusions [12] so they can be expressed as foreign key restrictions. But restrictions (2) and (4) are non-key based inclusions, and can be declared in SQL, using *IN* conditions. For example, restriction (2) can be declared in the definition of table S, as follows:

CHECK (A-id *IN* (*SELECT* A-id *FROM* R))

Note that each pair of inclusions (1,2 and 3,4) in figure 11 is a referential loop, so the transactions that maintain the tables R and S, must do the controls at *commit time* (*DEFERRED,* in SQL terms).

It is possible to represent these ***SCB*** patterns, using less than five tables. For example the pattern #*7* can be transformed to a schema with only four tables, collapsing S with A:

A (A-id , B-id , a) B (B-id , b) C (C-id , c) A (A-id , C-id ,r)

Now A•B-id must accept nulls. Remember that we are assuming that the minimum values for the cardinalities are equal to zero. But to enforce A and R as being both an integral part of the ternary, the two restrictions (1) and (2) are now to be expressed in a slightly different way. We can declare the following assertion:

CHECK (((*SELECT* A-id *FROM* A *WHERE* B-id *IS NOT NULL*) *IN*
(*SELECT* A-id *FROM* R))
AND
((*SELECT* A-id *FROM* R) *IN*
(*SELECT* B-id *FROM* A *WHERE* B-id *IS NOT NULL*)))

7.5. Three special patterns

In this section we will discuss the last three remaining ternary patterns, the *Special-patterns*. These three patterns have specific and interesting characteristics:

pattern	*LA* and binary FDs	implied FDs	view as *LA* ternary
#11	$A \rightarrow B$ $C \rightarrow B$	$\Rightarrow$ $(A,C) \rightarrow B$	***N:1:N*** *LA + two imposed* ***1:N***
#12	$(B,C) \rightarrow A$ $A \rightarrow B$	$\Rightarrow$ $(A,C) \rightarrow B$	***1:1:N*** *LA + one imposed* ***1:N***
#13	$(B,C) \rightarrow A$ $A \rightarrow B$ $A \rightarrow C$	$\Rightarrow$ $(A,C) \rightarrow B$ $\Rightarrow$ $(A,B) \rightarrow C$ $\Rightarrow$ $A \rightarrow (B,C)$	***1:1:1*** *LA + one imposed* ***1:N*** $A \rightarrow B$

We present graphically in figure 12, the three patterns:

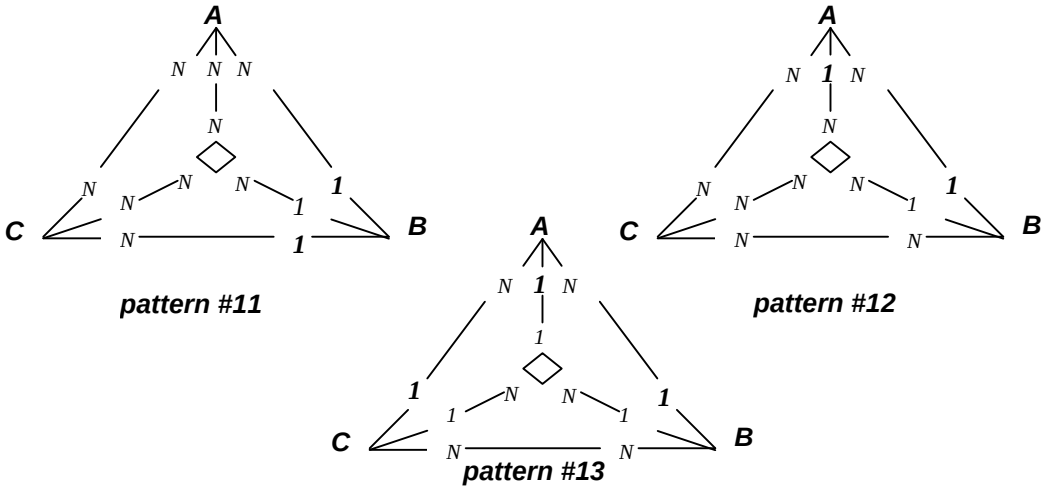


Fig.12: Three special cases

As we see, it is possible to represent these three cases using an *LA* ternary relationship if we augment it with one or two dependencies. But, we must add a restriction to express the inclusion of the binaries into the ternary. There is no way, in any of the two popular approaches, *LA* and *LH*, to properly represent these special cases in an ER diagram without a textual description of this additional restriction.

Observe that all the fourteen ternary patterns can be obtained using an *LA* ternary relationship plus some imposed binary FDs (in patterns *#5* to *#13*). But that is not possible using an *LH* ternary.

We will discuss now relational representations of the three special patterns. Remember that the ternary relationship, *R*, has its own attributes, *r*, and therefore they are functionally dependent on, at most, the three keys of entities *A, B* and *C*.

Pattern #11:

If we represent directly the full set of FDs for this pattern, the following database schema is obtained:

A (A-id, B-id, a) B (B-id, b) C (C-id, B-id, c) R (A-id, C-id, B-id , r)

We must declare five single-attribute foreign keys; R•A-id, R•B-id, R•C-id, C•B-id and A•B-id. The attributes C•B-id and A•B-id must accept nulls. And R•B-id must be declared as not null.

But the table R is not in 3NF (nor in 2NF) because of the binary dependencies *A-id→ B-id* and *C-id→ B-id* (see fig.12). These two FDs, represented in tables A and C respectively, already imply the *LA* dependency expressed in table R, *(A-id,C-id) → B-id*. So, we can take out the attribute B-id from table R and express the DB schema simply by:

A (A-id, B-id, a) B (B-id, b) C (C-id, B-id, c) R (A-id, C-id, r)

Note that this pattern can be considered as an ***SCB*** pattern because it can be seen as simply formed by the two binary FDs, the *LA* FD being implied by them.

Now the DB schem is normalized in 3NF (and also in BCNF) without loss of data and preserving all the FDs. But this DB schema (as the former non-normalized one) is not fully representing the semantics of our ternary relationship. The problem is that the two binary dependencies should be defined as forming part of the ternary, but in our schema they are independent of R. So we must add restrictions to enforce that R is not independent of the two binary FDs implied by the tables A and C. We could enforce:

a) $\pi_{\text{A-id,C-id}}$ R $\subseteq$ $\pi_{\text{A-id,C-id}}$ A*C

b) A•A-id $\subseteq$ R•A-id iff A.B-id is not null

C•C-id $\subseteq$ R•C-id iff C.B-id is not null

The constrain *a)* is more complex than an inclusion dependency. It involves a natural join operation that could be expressed in SQL92 using the following restriction in table R:

CHECK ((A-id, C-id) *MATCH* (*SELECT* A-id, C-id *FROM* (A *NATURAL JOIN* C)))

The constraints *b)* could be expressed as an assertion:

```
CHECK ( NOT EXISTS
        ( (SELECT A-id FROM A WHERE B-id IS NOT NULL)
                EXCEPT (SELECT A-id FROM R) )
 AND  NOT EXISTS
        ( (SELECT C-id FROM C WHERE B-id IS NOT NULL)
                EXCEPT (SELECT C-id FROM R) ) )
```

It is a very common belief among designers and educators, that given a set of FDs over a table resulting in a non-3NF situation, it is always possible to obtain a fully equivalent set of 3NF tables without adding other restrictions than the ones that can be expressed with candidate keys and inclusion dependencies (key or non-key based). But that is not actually true in this pattern *#11.* In order to obtain a representation in BCNF we need more powerful restrictions.

Other alternatives and a detailed analysis of this case *#11* can be found in [2].

Pattern #12:

According to the bold ***1***s in pattern *#12* (figure 12) the relationship of this case can be represented by:

R (A-id, C-id, B-id, r) plus the FD *A-id → B-id*

Note that this pattern is the pattern of the example of figure 8. The table R is not in BCNF because the FD *A-id → B-id* , but it is in 3NF. It is the example used in the literature to illustrate the case of a 3NF table that cannot be decomposed in BCNF without losing FDs. We can rearrange the DB schema to obtain BCNF tables, as follows:

R (A-id, C-id, r) A (A-id, B-id, a) B (B-id, b) C (C-id, c)

and declaring the three obvious single-attribute foreign keys. The attribute A•B-id must accept nulls. But then we lose the dependency *(B-id,C-id) → A-id.* We should enforce this lost FD, with an additional constraint that we can write in SQL-92, as the following assertion:

CREATE ASSERTION pattern#12
(*CHECK* (*UNIQUE* (*SELECT* B-id, C-id *FROM* (A *NATURAL JOIN* R))))

Observe that for pattern #*12*, as for pattern #*11*, it is not possible to obtain a fully equivalent set of tables in BCNF using only keys and inclusion dependencies (key or non-key based).

Pattern #13:

The pattren #*13* (see fig.12) can be reduced to: *(B,C) → A* plus *A → (B,C).* Therefore, it can be represented by the simultaneous utilization of *LA* and *LH* approaches, both with the values pattern ***1:N:N*** . So, the table R for the relationship, can be represented by:

R (A-id, B-id, C-id, r)

The two keys (one is single-attribute and the other is two-attribute) must be declared as not accepting nulls. The DB schema must be completed with the tables A, B and C, and the corresponding three single-attribute foreign keys, but no additional restrictions are needed.

This case is the only one among the fourteen possible cases, that produces a table R with a mix of multiple-attribute and single-attribute keys. Note that the table R is at least in BCNF (even in 4NF, because it has a single-attribute key).

8. Min-Max values and participation

8.1. Binary relationships

From the very beginning [3] the *participation* restriction was introduced to express for each entity participating in the relationship, if its participation was *mandatory* or was *optional*. The traditional cardinality, as defined in section 2, only considers its maximum value, but not the minimum. We assumed that the minimum value is zero. To express the participation, especial symbols are used. Several authors and methods, from both *LA* and *LH* approaches, explicitly express minimum and maximum values, ***min*** and ***max***, for each cardinality. The min value can be ***0*** or ***1*** (in this paper we do not consider cardinality values other than ***0, 1*** or ***N***). The meaning of max values, ***1*** or ***N***, is describable in terms of FD. But the meaning of min values, is conceptually different. The min value in binary relationships is equivalent to the participation restriction, but not in *LA* *n*-ary (*n*>2) relationships. A detailed analysis of the semantics of participation and min values can be found in Ferg [8].

In figure 13 we show a sample of different notations to express a case of *Project* and *Employee* (fig1) when projects can have no employees, but every employee works for *at least one* project and he/she can work for many projects. The *Project* entity is optionally participating in the *Work* relationship, but the entity *Employee* participation is mandatory in this relationship. In the case of binary relationships if the min value is *zero*, then the corresponding entity (which one, depends on the approach: *LA* or *LH*) is *optional*, but if the min value is *one*, then the entity is *mandatory*. So, in binary relationships the min value allow us to express, for both approaches, the *participation* restriction.

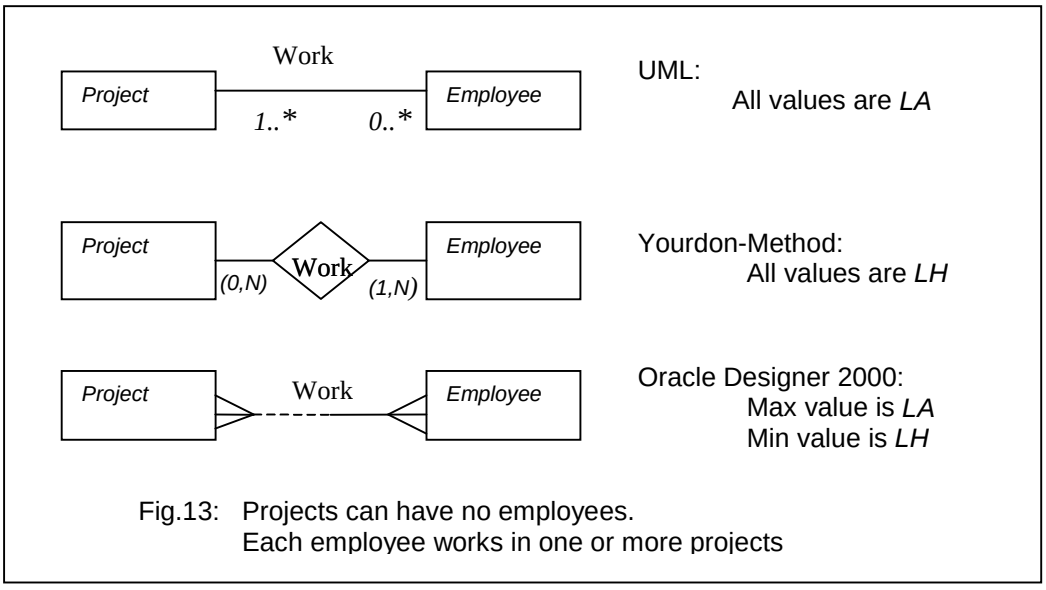


Fig.13: Projects can have no employees.
Each employee works in one or more projects

From now we will use the notation ***(min,max)***. So, a single symbol ***1*** in previous sections is a ***(0,1)*** and a single ***N*** is a ***(0,N)***. Note that in some notations, as for example in UML [18], the implicit minimum value for a single ***1*** is ***(1,1)***.

8.2. *N*-ary relationships

When we consider the cardinality of *n*-ary relationships, the meaning of the min value will be very different according to which one of the two approaches is adopted. Let us interpret the cardinalities of the example of fig.2 as being *LH* cardinalities (as in pattern #6) and assume that the cardinality near the *Dealer* is ***(1,1)***. The meaning of these values is that every dealer has one and only one concession. So every dealer must be associated to a single pair product-state. Every dealer must participate in the *Concession* relationship, but note that a pair product-state can be without concession to any dealer. This meaning is exactly the meaning of the classical participation restriction.

If we consider the cardinalities in fig.2 as being *LA* (it is a pattern *#3)* then the ***(1,1)*** cardinality near the *Dealer* entity will mean that every possible pair product-state must be associated to one and only one dealer. So, a product in a state must be distributed by one and only one dealer, but note that a dealer can have no concession at all. So, this restriction has not the same meaning of the participation restriction.

Let us now assume that the cardinality near the *Dealer* is *LH* and ***(0,1)***. The meaning of the zero is now that a dealer does not need to have a concession. The zero is expressing the optionality of *Dealer* entity. But if the cardinality is *LA,* the meaning of the zero is that a product in a state does not need to have a dealer. So, in *LA* approach, the min value constraint is no more the participation constraint for the entity. In *n*-ary relationships (*n*>2), the concept of participation of an entity in a relationship can not be expressed by the min value (zero or one) if it is an *LA* cardinality.

***LH* cardinalities, min= *1*:**

If in a ternary relationship R between entities A,B and C, it exists an *LH* ***(1,1)*** cardinality in the leg of entity A, this entity is mandatory. So, each instance of A has an associated instance of R, and vice versa. In the relational representation, table R will have the same key than table A, A-id, and we can enforce the restriction min = 1 using a referential loop or, easier, collapsing table A with table R:

In the example of figure 2, with the pattern ***1:N:N***, assuming an LH approach, the *Concession* table schema was

Concession (Dea-id, Pro-id, Sta-id, Con-date)

Obviously, the *Pro-id* and *Sta-id* attributes are to be defined as not null foreign keys. The key of the table *Concession* is *Dea-id*, the same key than *Dealer* table. But now every dealer must have associated a pair product-state, so the attributes *Dea-id* of the two tables can be defined as foreign keys referencing each other. Then the two tables can be collapsed to only one (without losing the normalization). We could eliminate the *Concession* table by moving *Pro-id, Sta-id* and *Con-date* to *Dealer* table as follows:

Dealer (Dea-id, Pro-id, Sta-id, Dea-address, Con-date)

In fact the functional dependency *Dea-id* → *(Sta-id , Pro-id)* can be decomposed into two more simple: *Dea-id* → *Sta-id* and *Dea-id* → *Pro-id,* and the diagram can be transformed to a pair of binary relationships.

All the *LH* ternaries could be represented by less than four tables if all the ***1*** cardinalities were ***(1,1)*** instead of ***(0,1)***. In the *LH* ***1:1:1*** case, the four tables could be collapsed to a single one, with three single-attribute candidate keys.

Let us now consider the *LH* ***(1,N)*** cardinality in the leg of entity A. Each instance of A has to be associated to at least one instance of R at least. That is a non-key based inclusion:

$A_{\bullet}\text{A-id} \subseteq R_{\bullet}\text{A-id}$

In the example of figure 2 assuming the *LH* approach (it is a pattern *#6*), and the cardinality near *Product* is a ***(1,N)*** cardinality. That means that each Product has to be associated to at least a concession. So:

Product.Pro-id $\subseteq$ *Concession.Pro-id*

that can be expressed in SQL as the following restriction in the definition of *Product* table.

CHECK (Pro-id *IN* (*SELECT* Pro-id *FROM* Concession))

***LA* cardinalities, min= *1*:**

Now we will analyze the translation of *LA* cardinalities having a min=1. The meaning of this restriction is that each possible combination of B instances with C instances, must have associated at least one instance of A. In practice, this is an extremely unusual case. To enforce this semantic we can utilize an inclusion involving a Cartesian product:

$$B_{\bullet}\text{B-id} \times C_{\bullet}\text{C-id} \subseteq \pi_{\text{B-id,C-id}} R$$

In the ***1:N:N*** example of fig.2, if the cardinality for *Dealer* is ***(1,1)*** but *LA* (pattern *#3*) then the *Concession* table will still be as in section 3.1

Concession (Pro-id, Sta-id, Dea-id , Con-date)

but now an additional restriction is needed to enforce the min=1 restriction. Every possible pair of (*Pro-id, Sta-id*) must have one concession. So, we can express it, in SQL-92, as :

CHECK ((*SELECT* Pro-id, Sta-id *FROM* Product *CROSS* State) *MATCH*
(*SELECT* Pro-id, Sta-id *FROM* Concession))

Another way to enforce this restriction is controlling that the number of concessions is equal to the result of multiplying the number of products by the number of states:

CHECK ((*SELECT COUNT(*) FROM* Concession) =

(*SELECT COUNT(*) FROM* Product) * (*SELECT COUNT(*) FROM* State))

Note that now it is not possible to reduce the number of tables without losing their normalization (BCNF). Although it is possible to draw an *LA* ER diagram for the *LH* case using two binary relationships, it is not possible to model the *LA* case using an *LH* ER diagram.

Remember that the min=1 in *LA* approach for relationships of degree greater than two, is no more the *mandatory participation* restriction, but a rather bizarre restriction. If we need to express the mandatory participation restriction, it will suffice an inclusion restriction as we explained for the *LH* min=1 case.

9. Decomposition

In this paper we have transformed from *n*-ary relationship to relations, through the analisys of the FDs. It could help a previous decomposition of the *n*-ary relationship to a set of relationships of lower degree. The decomposition to binary relationships is a classical problem and is tightly related with our problem. Although decomposition is not the subject of our paper, we will do some comments about it here.

Teorey [20] says that we should define a ternary relationship only when the concept cannot be represented by several binary relationships. But, some authors like Dey-Storey-Barron [5] say that a higher-degree relationship may always be expressed as several binary relationships. Apparently it exists a great confusion about the subject. Some researchers have published detailed analysis about decomposition from ternary to binary relationships: McAllister [16] says that ten patterns (*#4* to *#13*) are decomposable. And more recently, Jones-Song [11] observes that patterns *#10* to *#13* cannot be considered truly decomposable.

In fact all the authors are right, but each one use different meanings for essential words as *relationship* or *cardinality*: Teorey is only considering the three cardinalities that we labeled as *LA*. Dey-Storey-Barron use several non-basic constructs as *aggregate entities*, *association entities* and *weak entities* (as we saw in section 3 the use of ad-hoc non-basic constructs allow us to decompose *n*-ary relationships) and moreover they only consider the three cardinalities we labeled as *LH.* McAllister uses 12 cardinalities but he uses a non-basic construct, an ad-hoc graphical notation to add FDs to an ER diagram, Song-Jones also use 12 cardinalities but, as

all other authors, are not analyzing the influence of the minimum value of these cardinalities on decomposability, etc.

In this paper we have seen that if we do not limit the *participation* restriction to only the *optional* value, and we accept cardinalities with minimum values ***0*** or ***1***, none of the fourteen patterns can be always transformed to a database schema with tables having only single-attribute or two-attribute keys, without the need of adding other inclusion restrictions than the foreign keys. Then, if we accept that a ternary can be said decomposable only if the resultant ER diagram is limited to the use of basic constructs and not additional restrictions are to be added, none of the fourteen ternary patterns is always decomposable.

10. Final comments

To finish we summarize some of the conclusions, mainly the conclusions that do not coincide with very common beliefs.

It is a very common belief that transformation from ternary relationships to a database schema is a simple, well known and documented subject. One of the purposes of this paper has been to show that this is false.

LA style (Chen style) is more powerful than *LH* style (Merise style). All the possible FD patterns for ternary relationships can be obtained using the *LA* style: one *LA* ternary relationship plus some additional imposed binary FDs that can be seen as imposed *LA* binary relationships. But this would be not true for *LH* style. See section 7.

According to Loizou [12] "the central idea in relational databases design is that all the integrity constraints in the database should be describable in terms of keys and foreign keys". But, they are not. Many cases of *n*-ary patterns are not translatable to relational schemas using keys and key based inclusions (foreign keys) alone: non-key based inclusions are to be declared. But, moreover, several *n*-aries (two in the ternary case) cannot be translated even with such inclusions; more complex restrictions are needed. See 7.4, 7.5 and 8.2.

It is a very common belief that given a non-3NF table and a set of FDs, it is always possible to obtain a fully equivalent set of 3NF tables without adding other restrictions than the ones that can be expressed with keys and foreign keys. But that is not true. It is also a very common belief, that a ternary ER can always be represented by a BCNF set of tables, without using other restrictions than the ones that can be expressed with keys and inclusion dependencies (key or non-key based). But that is wrong as can be seen in patterns *#11* and *#12* in section 7.5.

Lastly, against a very common belief, none of the fourteen ternary patterns is always decomposable to binary relationships. See section 9.

References

[1] C.Batini, S.Ceri and S.B.Navathe, *Database Design: An Entity-Relationship Approach*, Prentice-Hall, (1991).

[2] R.Camps, "From Ternary Relationship to Relational Tables: A Case Against Common Beliefs", *ACM/SIGMOD-Record* (June 2002).

[3] P.P.S.Chen, "The entity relationship model: toward a unified view of data", *ACM Transactions on Database Systems*, **1** (1), 9-36 (1976).

[4] Demetrovics et al, "Minimum representations of closure operations", *Discrete Appl. Math*. **11** (1985).

[5] D.Dey, V.C.Storey and T.M.Barron, "Improving Database Design through the Analysis of Relationships", *ACM Transactions on Database Systems*, **24** (4), 453-486 (1999).

[6] J.Dullea and I.Y.Song, "An Analysis of Structural Validity of Ternary Relationships in Entity Relationship Modeling", *Proceed. 7th Intl. Conf. on Information on Knowledge Management,* 331-339 (1998).

[7] R.Elmasri and S.B.Navathe, *Fundamentals of Database Systems,* 3th ed., Addison-Wesley (1999).

[8] S.Ferg, "Cardinality Concepts in Entity-Relationship Modeling", *Proceed. 10th Intl. Conf. on Entity-Relationship Approach,* pp.1-30, T.J.Teorey (editor) (1991).

[9] D.R.Howe, *Data Analysis for Data Base Design*, Chapman&Hall (1990).

[10] T.Jones and I.Y.Song, "Analysis of binary/ternary cardinality combinations in entity-relationship modeling", *Data & Knowledge Engineering*, **19** (1), 39-64 (1996).

[11] T.Jones and I.Y.Song, "Binary Equivalents of Ternary Relationships in Entity-Relationship Modeling: a Logical Decomposition Approach", *Journal of Database Management,* April-June 2000, pp 12-19 (2000).

[12] G.Loizou and M.Levene, *A Guided Tour of Relational Databases and Beyond*, Springer-Verlag (1999).

[13] H.Mannila and K.J.Raiha, *The Design of Relational Databases,* Addison-Wesley, (1992).

[14] J.Martin, *Information Engineering:Planning & An*alysis. Prentice-Hall, (1990)

[15] A.J.McAllister, "Complete rules for *n*-ary relationship cardinality constraints"*, Data & Knowledge Engineering*, **27** , 255-288 (1998).

[16] A.J.McAllister and D.Sharpe, "An Approach for Decomposing *N*-ary Data Relationships" *Software Practice & Experience*, 28(2) , 125-154 (1998).

[17] A. Rochfeld and H. Tardieu, "MERISE: An information system design and development methodology", *Information & Management,* **6**, 143-159 (1983).

[18] J.Rumbaugh, I.Jacobson and G.Booch*, The Unified Modeling Language Reference Manual*, Addison-Wesley (1999).

[19] I.Y.Song, M. Evans and E.K. Park, "A comparative Analysis of Entity-Relationship Diagrams"*, Journal of Computer and Software Engineering*, **3** (4) , 427-459 (1995).

[20] T.J.Teorey, *Database Modeling & Design*, 3th ed., Morgan Kaufmann (1998).

[21] B.Thalheim, *Entity-Relationship Modelling*, Springer Verlag (1998).

[22] J.D.Ullman and J.Widom, *A First Course in Database Systems*, Prentice-Hall (1997).

[23] Yourdon,Inc., *Yourdon Method*, Prentice-Hall (1993).