From Ternary Relationship to Relational Tables: A Case Against Common Beliefs

Rafael Camps Paré
Universitat Politècnica de Catalunya (Barcelona) RCAMPS@LSI.UPC.ES

ABSTRACT

The transformation from n-ary relationships to a relational database schema has never been really fully analyzed. This paper presents one of the several ternary cases ignored by the ER-to-RM literature. The case shows that the following common belief is wrong: 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 candidate keys and elementary inclusion dependencies.

1. INTRODUCTION

The transformation from an *n*-ary relationship of the ER model to a relational database schema is considered by the literature as an elementary and well known problem. But according to my knowledge this subject is anywhere fully and systematically described.

It exists two popular approaches for expressing the cardinalities of the *n*-ary relationships: Chen approach [4] and Merise approach [10]. Chen approach is used by Teorey [12], Ullman-Widom [13], UML [11], IDEF1X [2], etc. Merise approach [10] is used by Elmasri-Navathe [5], Batini-Ceri-Navathe [1], Yourdon-Method [14], etc.

In the literature about database design, when the translation from ER to relational tables is described, is usually limited to a very general description, and only one of the two popular approaches is assumed for the cardinality. But none of the two approaches, nor the two together, allow us to express all the possible cardinalities (see McAllister [9]). So, transformations to relational tables from some ternary patterns are ignored in the literature. A detailed analysis of the transformation of all the patterns can be found in [3].

As an example we will present here a ternary case (section 2 and 3) with interesting characteristics (section 4) that challenge some very common beliefs (section 5).

2. AN ELEMENTARY EXAMPLE

We assume that if *R* is a relationship between the entities *A B* and *C*, then the symbol ***1*** in the leg of *A* means that each pair of instances, *b* from *B* and *c* from *C* cannot be associated to more than one instance *a* from *A*. Remember: each one of the instances of a ternary relationship involves actually three entity instances.

We will study the ternary relationship with ***1:N:N*** cardinalities in Figure 1. The entities are *Product* (identified by *Pro-id*) *State* (identified by *Sta-id*) and *Dealer* (identified by *Dea-id*). The semantics of the case is as follows: a dealer company can distribute several of our products and a product can be distributed by several dealers, but in each state a product cannot be distributed by more than one dealer. Each concession of distribution to a dealer of a product into a state, each instance of the ternary relationship, has a date of concession (*Con-date*). The meaning of the cardinalities is; each pair of one dealer and one state, can be associated to many products. Each pair of one state and one product cannot be associated to more than one dealer. Each pair product-dealer can be associated to more than one state.

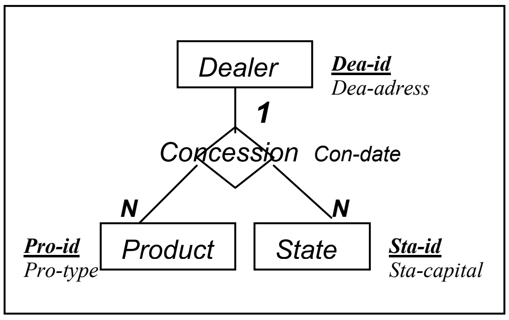


Figure 1: Example of *1:N:N* relationship

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 relationship

These two dependencies can be reduced to one:

(Pro-id , Sta-id) → *(Dea-id, Con-date)*

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

State (<u>Sta-id</u>, 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 *foreign key* and *not null.*

The possible lack of a dealer for a given pair product-state, 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 loosing their normalization (BCNF).

3. ADDING MORE FDs TO THE EXAMPLE

Now we will add to our elementary example two new functional dependencies:

Pro-id ➔ *Dea-id* and
Sta-id ➔ *Dea-id*

The two imposed structural constraints are, according to the terminology of Jones-Song [6], two *Constraining Binary Relationships*. The semantics of the new ternary pattern is now more restricted: *a)* each product is distributed only by one dealer, and *b)* in each state where it exists concessions, only one dealer exists.

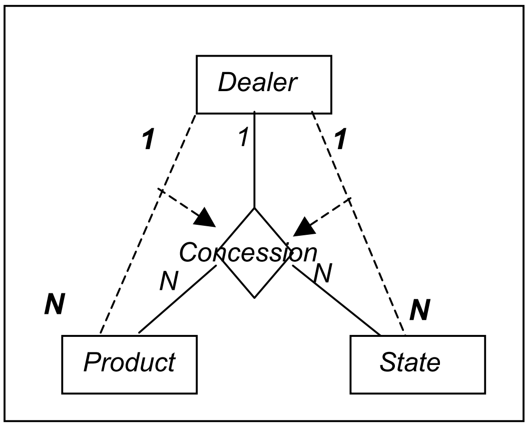


Figure 2: Two imposed FDs

Note that a product can still be distributed in several states and in a state is still possible to distribute several products.

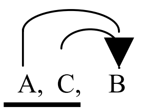
In Figure 2 we try to show graphically the new ternary. The dotted lines and arrows try to express the two imposed binary constraints as subsets of the ternary. In fact the *subset* term is not strictly correct because a constraint is not a subset of a relationship. Moreover, if we see the constraints as they were a set of instances of binary relationships, then a set of 2-tuples can not be a subset of a set of 3-tuples. In UML it exist the *{subset}* stereotype to express that an association is a subset of another association. Perhaps this UML stereotype can be used with the same meaning than the arrows in Figure 2, but the lack of a rigorous definition of UML in their reference manual [11] do not allow us to be sure.

There is no way to represent the pattern in Figure 2 (and some other patterns) using any of the most popular ER models, without adding text (or special symbols) to express that the binary restrictions are part of the ternary relationship.

Abstracting our case to a relationship R between the three entities A, B and C, the set of functional dependencies between the keys is:

(A-id,C-id) ➔ *B-id*
A-id ➔ *B-id*
C-id ➔ *B-id*

So the case is apparently simple and can be represented as:



This FD ternary pattern is exactly the pattern number 4 in the analysis of McAllister [9] or the pattern number 11 in the analysis of Jones-Song [7].

In our example, a valid set of instances for the new relationship *Concession* can be:

Pro-id	*Sta-id*	*Dea-id*	*Con-date*
Cocacola	Idaho	Smith&Sons	1996
Cocacola	Texas	Smith&Sons	1994
Fanta	Idaho	Smith&Sons	1994
Sprite	Kansas	FreeDrink	1998

Now it should be impossible to additionally insert the following instances, because the binary imposed FDs:

Cocacola	Kansas	Smith&Sons	2000
Sprite	Idaho	Smith&Sons	2000

As a consequence of the additional constraints, Cocacola cannot have a dealer in Kansas because Cocacola has already a dealer -Smith&Sons (in Idaho and Texas)- and Kansas already has a different dealer, FreeDrink (for Sprite). Similar considerations can be made for the pair Sprite/Idaho.

It could seem that with only two binary FDs we can characterize this case, because applying the *augmentation rule* (one of the Armstrong's rules [12]) that states

if *X* ➔ *Y* then *(X,Z)* ➔ *Y*

we derive the FD *(A-id,C-id)* ➔ *B-id* from both of our two binary FDs. But remember that the ternary relationship, R, has its own attributes, *r*, and therefore they are functionally dependent on the pair of keys of A and C. So, the full set of FDs is:

Key FDs:	Non-key FDs:
(A-id,C-id) $\rightarrow$ *B-id*	*(A-id,C-id)* $\rightarrow$ *r*
A-id $\rightarrow$ *B-id*	*A-id* $\rightarrow$ *a*
C-id $\rightarrow$ *B-id*	*C-id* $\rightarrow$ *c*
	B-id $\rightarrow$ *b*

4. FROM TERNARY TO RELATIONAL

If we transform our ternary relationship to relations, directly representing the full set of FDs of the pattern, we obtain the following database schema:

A ($\underline{\text{A-id}}$, B-id, a) B ($\underline{\text{B-id}}$, b)

C ($\underline{\text{C-id}}$, B-id, c) R ($\underline{\text{A-id, C-id}}$, B-id , r)

We are assuming that participation of entities A, B and C in the relationship R, is not mandatory. The attributes B-id of tables A and C are to be declared as accepting *nulls*. The attribute B-id of table R must be declared as *non-null*. The B-id attributes of tables A, C and R, are to be declared foreign keys of table B, and A-id and C-id of table R as foreign keys of tables A and C.

This schema has intertable redundancy (because B-id) and the table R is not in 3NF (nor in 2NF) because of the following two dependencies that have non-key determinants:

A-id $\rightarrow$ *B-id* *C-id* $\rightarrow$ *B-id.*

Moreover, the above relational schema is not fully representing the semantics of our ternary relationship. The two binary dependencies *A-id* $\rightarrow$ *B-id* and *C-id* $\rightarrow$ *B-id* in tables A and C, should be defined as part of the ternary, and in our schema they are not (they are independent of R). We could add the following *inclusion dependencies* [5]:

I) $\pi_{\text{A-id,B-id}}\, R \subseteq \pi_{\text{A-id,B-id}}\, A$

II) $\pi_{\text{A-id,B -id}}\, R \supseteq \pi_{\text{A-id,B-id}}\, A$

III) $\pi_{\text{C-id,B-id}}\, R \subseteq \pi_{\text{C-id,B-id}}\, C$

IV) $\pi_{\text{C-id,B -id}}\, R \supseteq \pi_{\text{C-id,B-id}}\, C$

The inclusion dependencies *I* and *III* can be easily expressed in SQL92, adding keys and foreign keys as follows:

- Table A: *UNIQUE (A-id, B-id)*
- Table C: *UNIQUE (C-id, B-id)*
- Table R:

FOREIGN KEY (A-id, B-id) REFERENCES A (A-id, B-id)
and
FOREIGN KEY (C-id, B-id) REFERENCES C (C-id, B-id)

Note that the *UNIQUE* declarations are partly redundant with the primary keys, but they make easy the control of the inclusion restrictions.

The inclusion dependencies *II* and *IV* are a little more complex to express in SQL:

- In table A *CHECK* ((A-id, B-id) *MATCH*
  (*SELECT* A-id, B-id *FROM* R))

- In table C *CHECK* ((C-id, B-id) *MATCH*
  (*SELECT* C-id, B-id *FROM* R))

With all these restrictions, the schema is now fully expressing the semantics of our case and avoids the possible update problems owning to the redundancy of B-id and to the lack of normalization of R.

Let us now start again with the original schema and assume that we desire to normalize table R. In order to normalize R we can take out its attribute B-id, because the additional FDs represented in tables A and C, already imply the dependency *(A-id,C-id)* $\rightarrow$ *B-id.* So, the schema of R can be reduced to:

R ($\underline{\text{A-id, C-id}}$, r)

Now the data base schema formed by the four tables A, B, C and R, is normalized in 3NF (and also in BCNF) without lost of data and preserving all the FDs. Obviously, we must define the attribute B-id of tables A and C, as foreign keys referencing table B, and accepting *nulls*.

But as before, this data base schema is not representing the inclusion of the binary restrictions in the ternary relationship. 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) $\pi_{\text{A-id}}\, A \subseteq \pi_{\text{A-id}}\, R$ iff A.B-id is not null

$\pi_{\text{C-id}}\, C \subseteq \pi_{\text{C-id}}\, R$ iff C.B-id is not null

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

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

The two 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)  )  )
```

A third possible transformation for our case is to take out the foreign key B-id from table A (or, symmetrically, table C). Now the schema will be:

A ($\underline{\text{A-id}}$, a) B ($\underline{\text{B-id}}$, b)

C ($\underline{\text{C-id}}$, B-id, c) R ($\underline{\text{A-id, C-id}}$, r)

We must declare A-id and C-id of table R as foreign keys of the corresponding tables, and B-id of the C table as foreign key accepting *nulls*. But now the FD *A-id* → *B-id* is not enforced. We can enforce it by:

```
CHECK   (UNIQUE   (SELECT A-id  FROM
      (SELECT DISTINCT A-id, B-id  FROM
                        (R NATURAL JOIN C ))))
```

Because R is ternary, the instances of C associated to instances of A must be exactly the same ones that are associated to instances of B. That can be enforced by an SQL check for equality (remember that C-id from table R is already defined as foreign key of C):

```
CHECK   ((SELECT COUNT(*) FROM  C
                     WHERE B-id  IS NOT NULL)
 =   (SELECT COUNT(DISTINCT C-id)  FROM R
                    WHERE  C-id NOT IN
    (SELECT  C-id  FROM  C WHERE B-id IS NULL)))
```

A fourth, and last, alternative transformation for our ternary case, consist of having B-id only in table R. Now we have no intertable redundancy but table R is not in 3NF. A-id, B-id and C-id in table R must be defined as foreign keys and not null. But we must enforce *A-id* → *B-id* and *C-id* → *B-id* in table R. So, we can write:

```
CHECK  ((UNIQUE   (SELECT A-id  FROM
           (SELECT DISTINCT A-id, B-id  FROM R)))
  AND     (UNIQUE   (SELECT A-id  FROM
           (SELECT DISTINCT A-id, B-id  FROM R))))
```

As we have seen , the non-3NF table
R ($\underline{\text{A-id, C-id}}$, B-id , r)
with the additional FDs *A-id* → *B-id* and *C-id* → *B-id,* cannot be decomposed in a fully equivalent set of 3NF tables if we use only candidate keys and inclusion dependencies.

5. AGAINST A COMMON BELIEF

The ER-to-RM literature ignores some interesting FD patterns. The case we have presented here is one of these cases. It looks very simple; apparently a harmless FD pattern. But it has an interesting behavior that have make us remember of Loizou[8] when he says that "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" although we know it's not always possible. It is a very common belief 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 candidate keys and inclusion dependencies (key or non-key based). But as we have seen in our case, it is not actually true. These restrictions are not powerful enough. If a correct database state is to be guaranteed more complex restrictions have to be used.

6. REFERENCES

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

[2] T.Bruce, *Designing Quality Databases with IDEF1X Information Models*, Dorset House Publishing, (1992).

[3] R.Camps, "Transforming N-ary Relationships to Database Schemas: An Old and Forgotten Problem", Research Repot LSI-5-02R of Universitat Politècnica de Catalunya (Spain).

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

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

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

[7] 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, 12-19 (2000).

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

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

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

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

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

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

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