An Analysis of a Methodology that Transforms the Entity-Relationship Model into a Conceptual Model for a Graph Database

Fernán Villa(✉), Francisco Moreno, and Jaime Guzmán

Departamento de Ciencias de la Computación y de la Decisión,
Universidad Nacional de Colombia, Medellín, Colombia
{favillao, fjmoreno, jaguzman}@unal.edu.co

Abstract. The graph databases (GDB) have gained a lot of importance in the last years; this is due to the necessity to store and manage very large volumes of data whose natural structure is a graph. However, nowadays there do not exist conceptual models widely accepted to represent a GDB. This fact implies that the analysts are guided considering their experience and best practices. There have been proposed different conceptual models for GDB; in this paper, we analyze a methodology that generates a conceptual model for a GDB from the entity-relationship (E-R) model. We explore several limitations of this methodology and offer some ideas for solving them.

Keywords: Graph databases · Entity-relationship model · Conceptual models
Model transformation

1 Introduction

The basic element of a graph database (GDB) [1] is a graph. A graph is composed of nodes and edges, which show and set up the relationships between the nodes, e.g., the friendship between two users, the distance between two cities. A GDB is appropriate for managing network applications such as social networks, biological networks [2], transport networks, genealogical networks, and citation networks, among others.

In this paper, we analyze a methodology [3] that generates a conceptual model for a GDB from the entity-relationship (E-R) model. We present several limitations of this methodology and offer some ideas for solving them. The remainder of the paper is organized as follows: in Sect. 2, we present the property graph model. In Sect. 3, we explore the Model-Driven Design of Graph Databases methodology. In Sect. 4, we study the methodology limitations and offer some ideas to solve them. Finally, we present the conclusions and future work.

M. H. Miraz et al. (Eds.): iCETiC 2018, LNICST 200, pp. 70–83, 2018.
https://doi.org/10.1007/978-3-319-95450-9_6

2 Property Graph Model

Today Neo4j [4] is a popular GDBMS (graph database management system). It was launched in 2007 and have been used by organizations such as NASA, Walmart, eBay, among others. This GDBMS uses a property graph model (PGM) to represent the domain of an application, it is a set of nodes related by directed edges. Nodes and relationships have properties (attributes). This model is used as well by other GDBMS such as TinkerPop [5] and Titan [6] and it is the base of GraphX (an API for managing graphs in Spark), among others.

The main elements for modeling are:

- **Nodes:** they are the basic model elements, represent the objects of interest for the application (entities in the real world), i.e., the objects that the analysts are interested to store information.
- **Relationship:** they represent the connections between two nodes. They must have a direction (a source node and a target node) and a type, which describes the nature of the relationship between the nodes, e.g., of friendship, possession, contract, among others. A relationship could have the same source and target node (recursive relationships). Between a couple of nodes, there could be several relationships (multigraph).
- **Properties:** They represent the attributes of nodes and relationships. In a node or in a relationship, a property is associated with a value (a property with its value is called key-value pair [7]). A node or relationship can have zero, one, or many properties.
- **Labels:** The labels allow the analyst to classify the nodes according to its role in the application. A node can have zero, one, or many labels, each with its corresponding name. In Fig. 1, we show three nodes with their labels and two relationships with their properties.

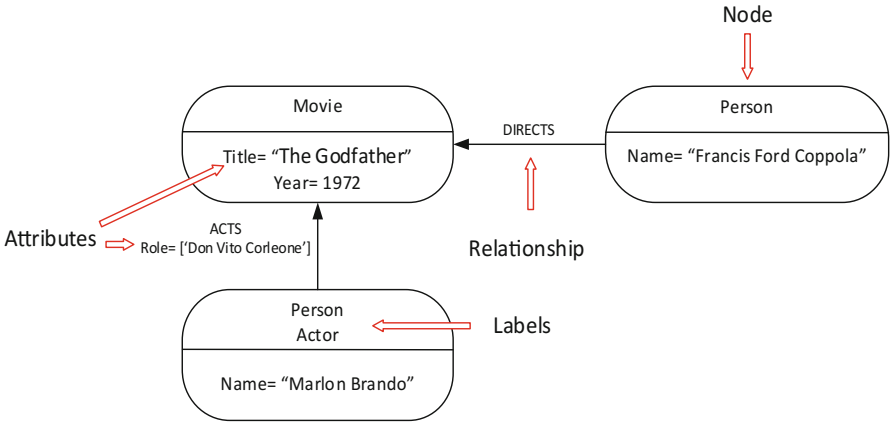

Fig. 1. Example of a PGM.

In addition to the PGM, several GDBMS are based in a hypergraph model. A hypergraph is a group of nodes and edges, but unlike a property graph it allows that a relationship connects more than two nodes or relationships. Thus, a hypergraph is a generalization of a property graph. Hypergraph DB [8] and Trinity [9] support hypergraphs.

3 Model-Driven Design of Graph Databases Methodology

In [3] it is proposed a methodology for modeling a GDB from the E-R model. Considering the relationships between the entities in the E-R model, it is obtained a model similar to the PGM.

To explain this methodology, we consider the E-R model example from Fig. 2. In this notation a rectangle represents an entity, a rhombus a relationship, a black circle a unique identifier, a white circle an attribute, and parentheses represent the cardinality (N stands for many).

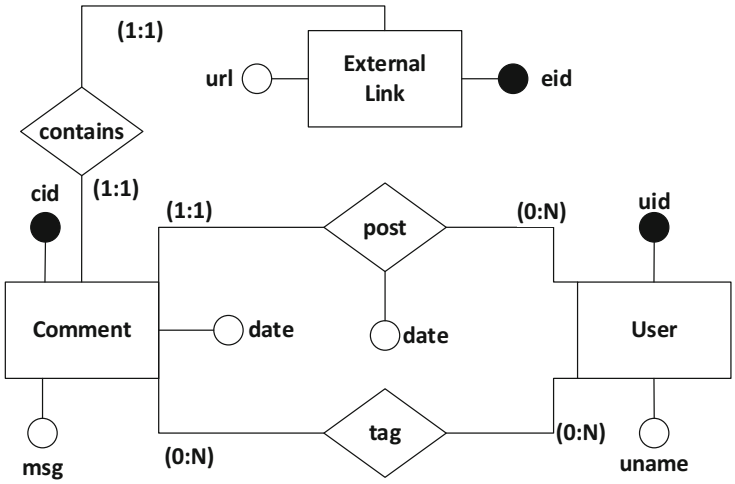

Fig. 2. E-R model.

The methodology has three steps:

3.1 Step 1. Apply Transformation Rules

The E-R model is transformed into an OE-R diagram (Oriented Entity-Relationship Diagram), i.e., a directed graph with labels and weights. The rules for transforming an E-R model into an OE-R diagram are:

(a) A one-to-one relationship is transformed into a bidirectional relationship. It is assigned a weight equal to zero, see Fig. 3a.

(b) A one-to-many relationship is transformed into a relationship that goes from the entity with cardinality one to the other entity. It is assigned a weight equal to one, see Fig. 3b.
(c) A many-to-many relationship is transformed into a bidirectional relationship. It is assigned a weight equal to two, see Fig. 3c.

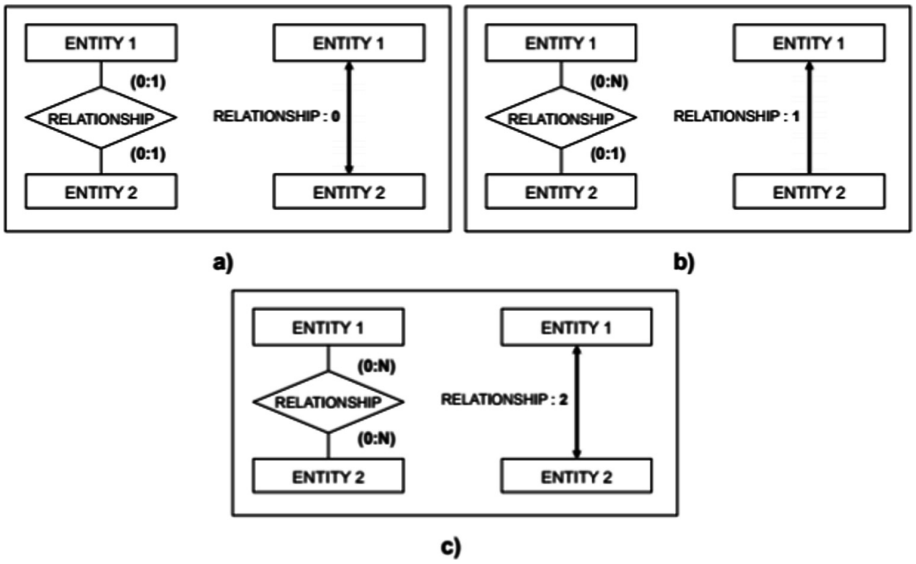

Fig. 3. Rules for transforming relationships: (a) One-to-one relationships, (b) One-to-many relationships, and (c) Many-to-many relationships. Source [3].

After applying these rules to the E-R model of Fig. 2, we obtain the O E-R diagram of Fig. 4

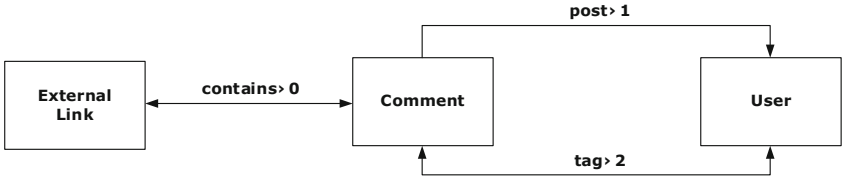

Fig. 4. O E-R diagram for the model of Fig. 2. Source [3].

3.2 Step 2. Merge Entities

This step is intended to merge entities whose instances use to appear together in the queries. To do this, the O E-R diagram is partitioned into groups of entities. To partition it, the authors define the functions W^+ and W^- for an entity n as follows:

$$W^+(n) = \sum_{e \in out(n)} weight(e). \quad (1)$$

$$W^-(n) = \sum_{e \in in(n)} weight(e). \quad (2)$$

Where $out(n)$ is the set of the outgoing relationships of n and $in(n)$ is the set of the incoming relationships of n. Thus, W^+ and W^- calculate; respectively, the weights of the relationships that go out and come in of an entity n.

For instance, for the *Comment* entity of the O E-R diagram of Fig. 4 we obtain W^+ (*Comment*) = 1 + 2 = 3 and W^- (*Comment*) = 0 + 2 = 2.

The partitions are formed in accordance to the following rules:

- Rule 1: An entity that is isolated, i.e., without relationships, forms a group by itself.
- Rule 2: If for an entity n is met that $W^-(n) > 1$ and $W^+(n) > 1$, then n will form a group (however, n could merge with some other entity, see Rule 3).
- Rule 3: If for an entity n is met that $W^-(n) \leq 1$ and $W^+(n) \leq 1$, then n is merged with other entity m, as long as between m and n there exist a relationship.

From the rules we conclude that:

(a) The merge of entities is done only when a node meets rule 3.
(b) The entities that participate in a many-to-many relationship do not merge, because for each of these entities W^- and W^+ will be greater or equal than 2 (because the weight of this type of relationship is 2) and; therefore, it does not meet Rule 3.
(c) The rules merge the entities that participate in a one-to-one relationship. However, the merge of the entities that participate in a one-to-one requires a more detailed analysis. For a discussion, see [10].
(d) With regard to the one-to-many relationships, these are not necessarily merged; indeed, as we saw in Sect. 4, there are cases in which none of these rules are met and *the methodology does not explain what must be done in such cases*.

After applying ese rules to the O E-R diagram of Fig. 4 we obtain the partition of Fig. 5.

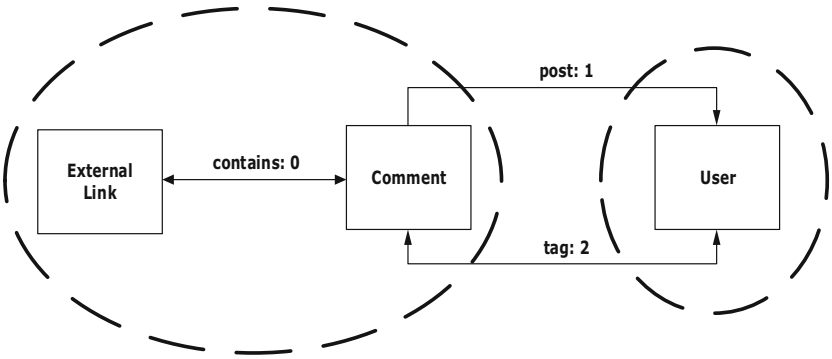

Fig. 5. Partitioned O E-R diagram of Fig. 4. Source [3].

3.3 Step 3. Conceptual Model

Finally, a conceptual model is defined for the GDB. This model is considered as a template for the creation of the instances of the GDB. The template includes all the attributes in this way: entityName.attributeName. See Fig. 6.

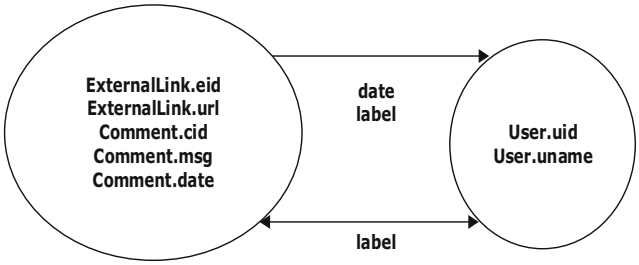

Fig. 6. Resultant template for the model of Fig. 5. Source [3].

Note that in the template appears the word label, which is an attribute that represents the name of the relationship. From the template of Fig. 6 it is possible to generate instances, as it is shown in Fig. 7.

4 Methodology Limitations

This methodology is a first step for the conceptual modelling of a GDB. The methodology generates a model with an abstraction level greater than other proposals, such as the PGM which models the GDB using instances. However, the methodology has some problems and disadvantages.

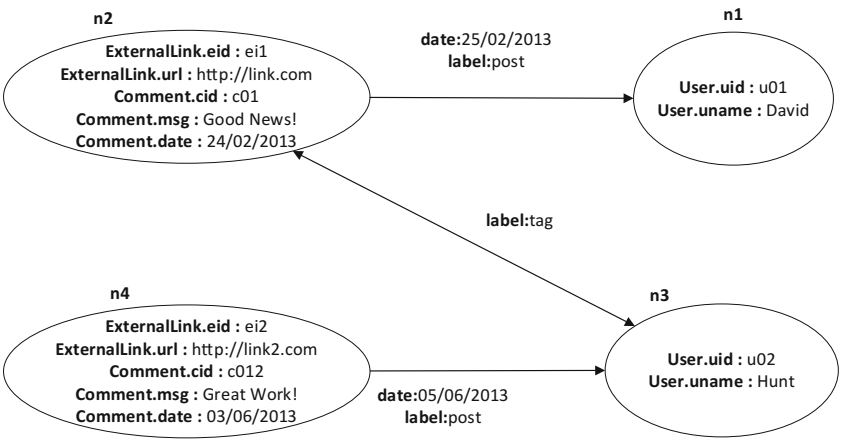

Fig. 7. Instances from the template of Fig. 6. Source [3].

4.1 Insufficient Rules

The proposed rules do not consider all the cases. In particular, it is not indicated how to proceed in these two cases:

- Case 1: if for an entity n is met that $W^-(n) > 1$ and $W^+(n) < 1$.
- Case 2: if for an entity n is met that $W^-(n) \leq 1$ and $W^+(n) > 1$.

For example, consider the E-R model of Fig. 8, where Entity1 = Employee, Entity2 = Company, Entity3 = Vehicle, Rel1 = Works for, Rel2 = Works for, Rel3 = Paints, and Rel4 = Drives. We show its corresponding O E-R diagram in Fig. 9.

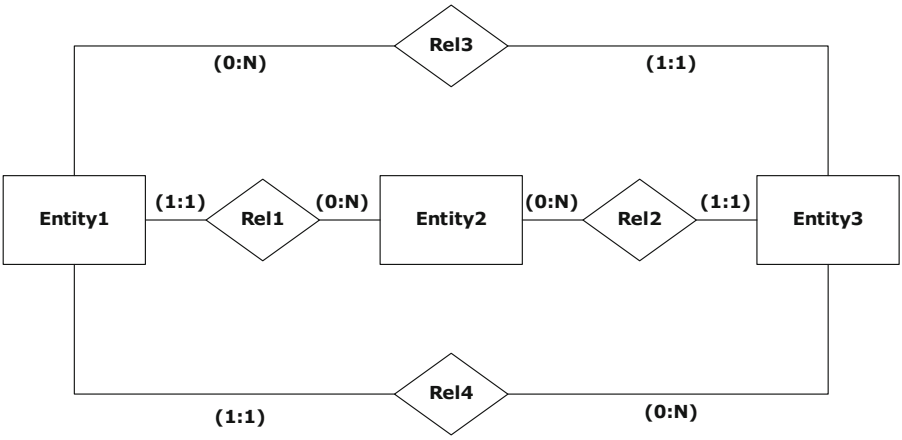

Fig. 8. E-R Model to exemplify insufficient rules.

In Table 1 we show the calculations of W^+ and W^- for each entity.

Because none of the entities meets the rules of the methodology, it is not possible to obtain a GDB template. A possible solution could be: given that all the relationships are of one-to-many type, then we define a node with all the attributes of the participating entities (this seems to be the intention of the methodology as suggested by the previous examples). Another alternative is to define three nodes as follows: (1) to merge Company, Employee, and Vehicle, (2) to merge Employee and Vehicle (relationship Paints), and (3) to merge Employee and Vehicle (relationship Drives). However, the appropriate solution will depend largely on factors not considered by the methodology (e.g., analysis of the most frequent queries in the database), see also Sect. 4.3.

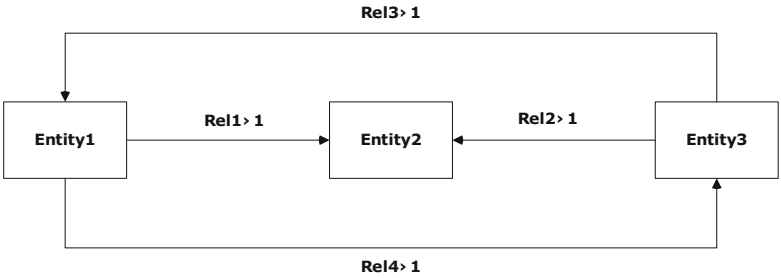

Fig. 9. O E-R diagram for the E-R model of Fig. 8.

Table 1. Calculation of W^+ and W^-.

Entity	W^-	W^+	Rule
Entity 1	1	1 + 1 = 2	None (Case 2)
Entity 2	1 + 1 = 2	0	None (Case 1)
Entity 3	1	1 + 1 = 2	None (Case 2)

4.2 Loss of Semantic Relationships

As entities merge, there is not information on relationships present among them in the original E-R model. This can lead to confusions or inconsistencies in the resultant template. Consider the E-R model of Fig. 10. We show its corresponding O E-R diagram in Fig. 11.

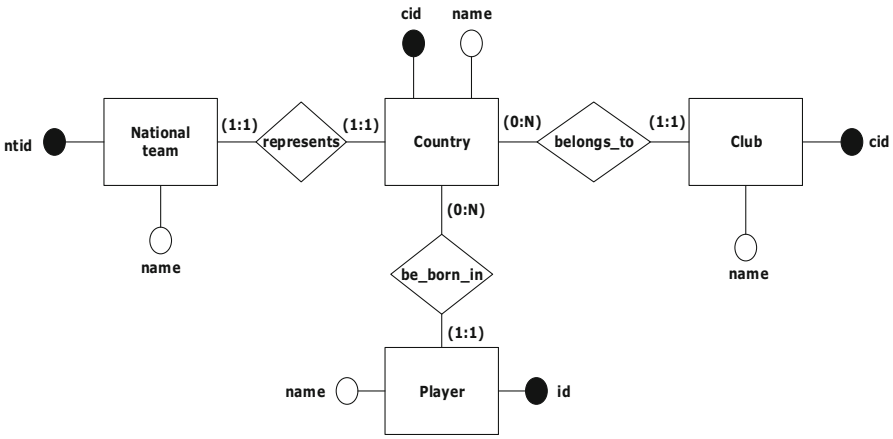

Fig. 10. E-R model with relationship between Country and Player.

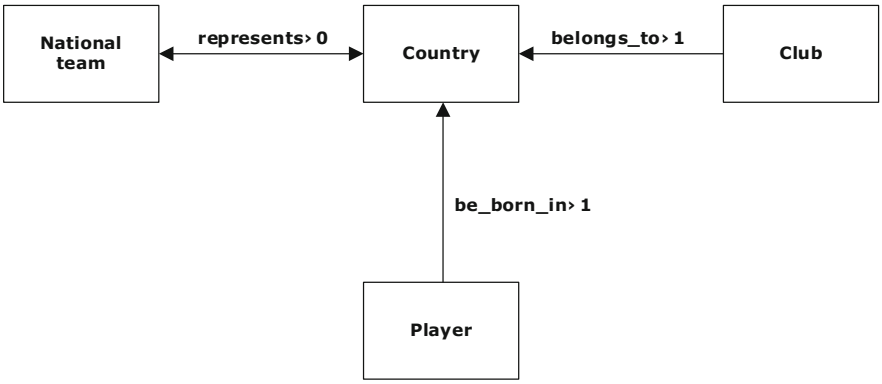

Fig. 11. O E-R diagram for the model of Fig. 10.

Table 2. Calculation of W^- and W^+.

Entity	W^-	W^+	Rule
National team	0	0	3
Country	1 + 1 = 2	0	None (Case 1)
Club	0	1	3
Player	0	1	3

In Table 2, we show the calculations of W^- and W^+ for each entity.

Although the Country entity does not meet any of the rules of the methodology, the other entities meet rule 3; therefore, they must be merged with an entity that they have at least a relationship. We show the resultant template in Fig. 12.

Fig. 12. Resultant template for the model of Fig. 11.

In generating instances with the resultant template some problems arise. For example, if we want to store the data of a player and his country of origin, it is not clear which values should be put in the attributes of the Club entity. The methodology does not indicate how to proceed in these cases.

Note how in the original E-R model, a player is related to a country and not with a club, the resultant template gives the impression that a player is also related with a club, which changes the semantics of the model. For example, in Fig. 13 we show an instance of the template of Fig. 12. The instance, gives the impression that a player belongs to the Boca Juniors club, something that does not correspond with the semantics of the original E-R model.

Fig. 13. Instance of template of Fig. 12.

In addition, if we apply the methodology to the models of Figs. 14 and 15, we obtain *the same template* to the model of Fig. 10.

That is, the methodology generates *the same template for three E-R models with different semantics*. In the resultant template it is not possible to determinate if the relationship is between a player and a club, or if it is between a player and a team, or if it is between a player and a country. The problem is that in the template there is not information about the relationships that existed between the entities in the original E-R model.

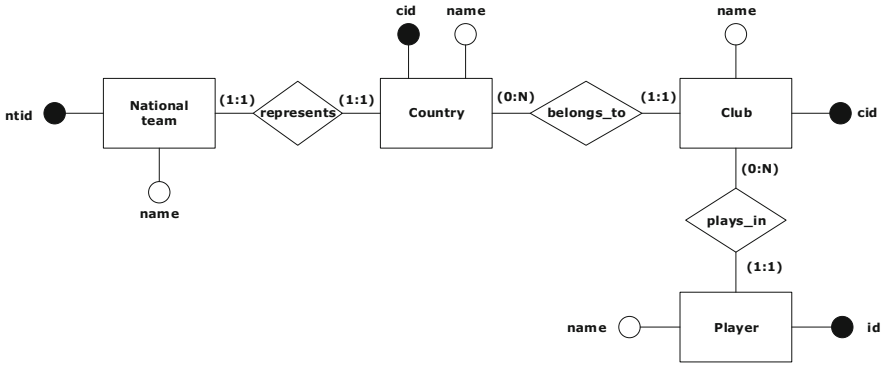

Fig. 14. E-R model with relationship between Club and Player.

Another example where there is a loss of semantic relationships is when between two entities, there is more than one relationship. For instance, consider the E-R model of Fig. 16, where we show a pair of entities with two relationships. After applying the methodology, we obtain the template shown in Fig. 17(a).

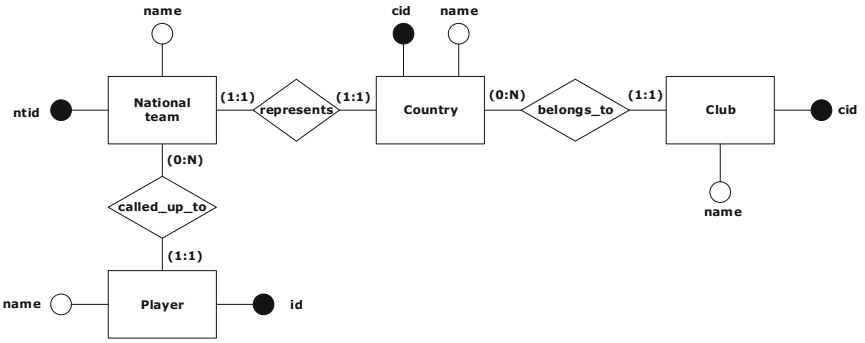

Fig. 15. E-R Model with relationship between National team and Player.

If we use the template of Fig. 16(a) for creating an instance, it is not possible to find if the relationship that exists between the person and the company is of type "Represents" or "Works for", because the template does not include information about the relationship that there was between the two entities. Yet, as in the E-R model a company is related to two kinds of persons (employees and representatives), we change the template as it is shown in Fig. 17(b). Note that in the new template, we include the name of the relationship in the attributes of the entity Person, to distinguish if the person is an employee or a representative. In addition, the relationship ¨works for¨ has an attribute "start_date" but after merging the entities this relationship disappears (it should be included in the resultant template). *These aspects are not considered in the methodology*.

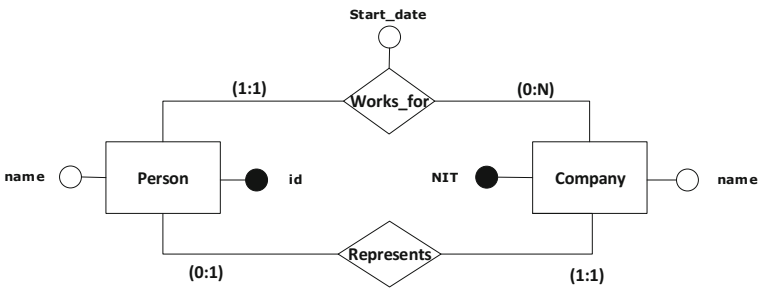

Fig. 16. E-R model with two relationships between two entities.

4.3 Other Limitations

(a) There is a lacking specification for the optionality of the attributes. The methodology does not offer tools that indicate which attributes are mandatory or optional, neither which are unique identifiers. For example, in the template of Fig. 17(a) it is not indicated that "Company.id" and "Person.id" correspond to the unique identifiers of their corresponding entities in the original model.

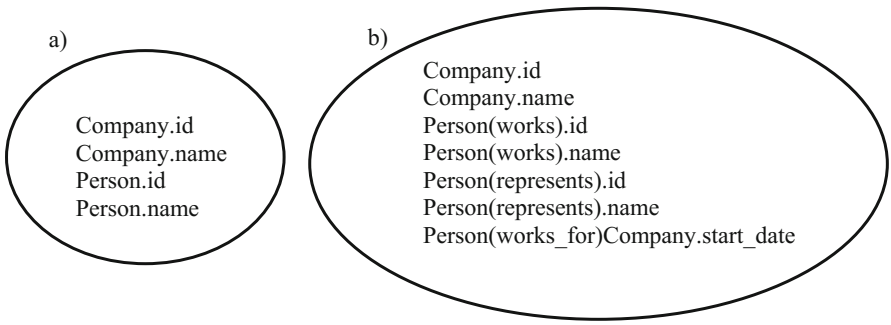

Fig. 17. Template for the model of Fig. 16: (a) generated by the methodology and (b) template proposal.

(b) There are no rules for generalization and exclusive relationships. In the methodology are not considered generalization (inheritance) nor exclusive relationships.
(c) Lack of analysis about the expediency of using a GDB [11]. The methodology is based on a database conceptual model, the E-R model and *mechanically* generates a template (model) for a GDB. However, it is not analyzed if a GDB is appropriate for an application. Although the decision for transforming an E-R model into a conceptual model for a GDB is, in a great part, a responsibility of the analyst team, the methodology could be enriched with elements (e.g., with the most

Table 3. Hints for improving the methodology.

Corresponding limitation	Solution or hint
4.1	Extend or change the current rules to consider all the cases
4.2	Include information about the relationships that disappear after merging entities, e.g., for each group of entities in the resultant template, we could specify in an annex (metadata) the relationships that existed between the corresponding entities in the original model. The template must also include the attributes of such relationship, e.g., the attributes could be named like this: relationshipName_attributeName (see example in Fig. 17)
4.3a	It must be included symbols to represent the mandatory attributes and those corresponding to unique identifiers. For example, it could be used a notation like the proposed in [13]
4.3b	Propose elements to represent in the resultant template generalizations and exclusive relationships. For example, it could be used a notation similar to the proposed in [13, 14]
4.3c	It must be considered the most frequent queries, the data volumes, and the database schema to decide if it is convenient to use a GDBMS
4.3d	It must be considered the most frequent queries to decide the convenience to merge some entities

frequent queries, see next item) which help to determine how convenient is to do the transformation or *transform only a part of the original model*.

(d) There is a lacking analysis on convenience to merge determined entities [12]. This aspect is related with the previous one. For instance, suppose the methodology merge the entities *A*, *B*, and *C* and that in the application the most frequent queries only require data from *A* and *B* but not from *C*. Considering this aspect, it is not convenient to merge *C* with *A* and *B*.

4.4 Some Recommendations

In Table 3 we show some hints to be developed in future works, which could help to improve the methodology.

5 Conclusions

In this paper, we analyzed the methodology "Model-Driven Design of Graph Databases" that transform an E-R model into a conceptual model for a GDB (Template Graph). This methodology presents several disadvantages and problems. We analyzed these aspects and offer some ideas for solving them. Perhaps the main problem is the omission of the relationships in the resulting template; this can lead to semantic confusions (with regard to the original model, i.e., the E-R model) when the template is generated, as we showed in Sect. 4.2.

In future works, in addition to those which can be derived from the identified problems in Table 3, it could be developed a similar methodology to model other non-relational types of databases [15]. Finally, as one of the referees suggested, concepts such as weak entity, strong entity, associative entity should be considered in the transformation from the E-R model to the GDB model. Constraints are also missing in the methodology.

Acknowledgments. This paper has been supported by the research group "SintelWeb, Sistemas Inteligentes Web".

References

1. Patil, S., Vaswani, G., Bhatia, A.: Graph databases - an overview. Int. J. Comput. Sci. Inf. Technol. **5**(1), 657–660 (2014)
2. Angles, R., Gutiérrez, C.: Survey of graph database models. ACM Comput. Surv. **40**(1), 1–39 (2008)
3. De Virgilio, R., Maccioni, A., Torlone, R.: Model-driven design of graph databases. In: Yu, E., Dobbie, G., Jarke, M., Purao, S. (eds.) ER 2014. LNCS, vol. 8824, pp. 172–185. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12206-9_14
4. Shimpi, D., Chaudhari, S.: An overview of graph databases. In: International Conference in Recent Trends in Information Technology and Computer Science (ICRTITCS 2012), Mumbai, India (2012)

5. The Apache Software Foundation: Apache TinkerPop. http://tinkerpop.apache.org/docs/current/tutorials/getting-started/. Last accessed 05 May 2018
6. DataStax: Titan Documentation. http://s3.thinkaurelius.com/docs/titan/1.0.0/getting-started.html. Last accessed 05 May 2018
7. Neo Technology: What is a Graph Database and the Property Graph? https://neo4j.com/developer/graph-database/#property-graph. Last accessed 05 May 2018
8. Iordanov, B.: HyperGraphDB: a generalized graph database. In: Shen, H.T., et al. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 25–36. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16720-1_3
9. Shao, B., Wang, H., Li, Y.: Trinity: a distributed graph engine on a memory cloud. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 505–516 (2013)
10. Bernik, J.P.: A translation of the one-to-one relationship for introductory relational database courses. Newsletter ACM SIGCSE Bull. **35**(4), 66–67 (2003)
11. Batra, S., Tyagi, C.: Comparative analysis of relational and graph databases. Int. J. Soft Comput. Eng. **2**(2), 509–512 (2012)
12. Pinto, Y.: A framework for systematic database denormalization. Global J. Comput. Sci. Technol. **9**(4), 44–52 (2009)
13. Barker, R.: CASE Method: Entity Relationship Modelling (Computer Aided Systems Engineering). Addison-Wesley Professional, Massachusetts (1990)
14. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design: An Entity-Relationship Approach. Addison-Wesley Professional, Massachusetts (1991)
15. Vera, H., Boayentura, W., Holanda, M., Guimaraes, V., Hondo, F.: Data modeling for NoSQL document-oriented databases. In: Proceedings of the 2nd Annual International Symposium on Information Management and Big Data - SIMBig, vol. 1478, pp. 129–135 (2015)