Applying a Fuzzy Approach to Relaxing Cardinality Constraints

Harith T. Al-Jumaily, Dolores Cuadra, and Paloma Martínez

Dept. of Computer Sciences, University of Carlos III in Madrid
{haljumai,dcuadra,pmf}@inf.uc3m.es

Abstract. In database applications the verification of cardinality constraints is a serious and complex problem that appears when the modifications operations are performed in a large cascade. Many efforts have been devoted to solve this problem, but some solutions lead to other problems such as the complex execution model or an impact on the database performance. In this paper a method to reducing and simplifying the complex verification of cardinality constraints by relaxing these constraints using fuzzy concepts is proposed.

1 Introduction

In database design methodology such as [1] and [2], there are processes devoted to transform conceptual into logical schemata. In such processes, semantic losses are produced because logical constructs are not coincident with conceptual constructs. The cardinality constraint is one of these constraints that can be established in a conceptual schema. It has dynamic aspects that are transformed to the relational model as certain conditions to verification the insertion, deletion, and update operations which modify the database. This type of constraint is usually represented by minimum and maximum bounds [3]; the minimum cardinality constraint (also called participation constraint [4]) represents the minimum number that an entity instance must participate in a relationship. There are two types of participation, partial and total. The maximum cardinality constraint represents the maximum number that an entity instance participates in the relationship [5]. A correct transformation of these constraints into the relational model is necessary in order to preserve the semantics that reflects the Universe of Discourse. Therefore, the relational model supports mechanisms such as primary key, foreign key, etc. to express these constraints, but these mechanisms are not sufficient to reflect all the required semantics, therefore it can be enhanced by using triggers and stored procedures [6]. The main goal is to keep the semantics inside the database and not in the applications accessing it.

In many database applications the verification of cardinality constraints is a seriously and complex problem that appears when the modification operations are performed in a large cascade. In such applications, may be it is difficult to address all objects that could be affected by such operations as well as the verification of the cardinality constraints of these objects. Therefore, many research efforts have performed different and complex methods to solve this problem. In some solutions,

the applications developers must intervene to verify the cardinality constraints in more than one object at the same time. So, if the intervention is correct, other problems such as the complex execution model or the impact on the database performance may be produced.

We believe that there is a relationship among the database semantics, the complex implementations to verify this semantics, and the database performance. These three elements could be represented as three lines of a triangle (Figure 1), the vertexes (abc) of this triangle can be moved only in two directions, vertical and horizontal. When the vertex (a) is moved up the base of the triangle (bc) will be reduced, this means that, when all the semantic constraints are verified, the final result will produce an impact on the performance of this database.

Therefore, in this work we will propose a method to reduce the complex verification of cardinality constraints. The method can be applied to any conceptual model but in this work we will use the Extended Entity/Relationship (EER) as reference. The rest of this work is organised as follows. In section (2) related work is shown. Section (3) is devoted to explain the partial and total participations as well as fuzzy participations. In section (4) our method to relaxing the cardinality constraints will be presented. Finally, some conclusions are exposed.

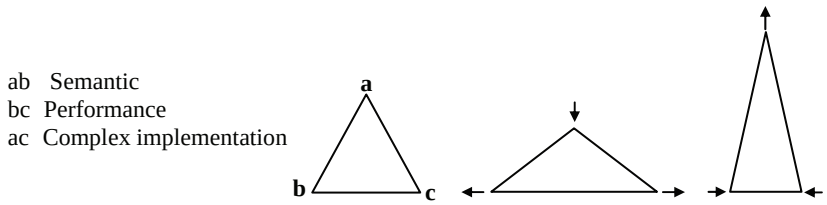

Fig. 1. Relationship Triangle

2 Related Work

An approach to verifying cardinality constraints is explained in [7]. It provides triggers to verify the cardinality constraints for each relationship object in the relational model. These triggers are activated when the modification operations are actually performed. When a constraint violation is produced the adopted action is rolled back and the transaction is aborted. The experiments have shown that the performance evaluations of triggers and stored procedure are equally well [8] [9]. The work presented in [10] stablished that a modifying operation upon an entity affects only the relationships that are connected to it while a modifying operation upon a relationship affects only the entities participating in it. Three actions are adopted when a constraint violation is produced, (a) the operation may be rejected, (b) triggered another operation to satisfy the constraint, or (c) updating the adjacent element by nullifying integrity type. While the contribution in [11] was in the automatic creation of structured methods on top of a given EER schema, these

methods are to prove cardinality faithful and terminating. For example, when an entity is inserted, all associated binary relationships are studied to see whether their cardinality constraints are not violated. So, asking the user for all relationships related to the inserted entity is required and this process is repeated until constraints are satisfied. In order to terminate propagation a null entity could be inserted.

Our work addresses the issue of insert operations because an inserted instance in the database needs to satisfy the cardinality constraints with all associated elements, and to insert more than one instance at a time could be required. Therefore, in this paper a method to divide the relational schema of a database into several subschemata is proposed. This solution will reduce the number of objects in each subschema and consequently to reduce the number of objects that must be verified when an insert operation is performed. The final objective is to simplify the complex implementation of verifying cardinality constraints. We believe that an insertion operation could be easier controlled if few associated elements are verified whatever approach is used. We will do this by taking advantage of the definition of the optional and fuzzy participations [12]. So, we consider that the optional and the fuzzy participation roles are end points of verifying the cardinality constraints in a relational schema. The fuzzy participation roles are used in many works such as [13] where several fuzzy conditions on each instance have been defined and a trigger that checks the value of a quantifier is used to verify these conditions. If this value is less than the minimum percentage that constraints must satisfy in a database, then the DBMS must produce an error message informing about the non-fulfilment of this constraint.

3 Total/Partial and Fuzzy Participations

Let $R = (r_1E_1,\ldots, r_nE_n, A_{1,\ldots,} A_s)$ be a n-ary relationship with s attributes, where each r_i is the role that plays an entity E_k in the relationship R (Figure 2).

We define R^t as a set of instances in R. An element r^t in R^t is a vector of n components, where each r_i position in the n-vector represents a participation role, and it contains an instance of the entity identifier which participates with that role. Thus, a set of the instances R^t of R is a subset of the Cartesian product of the identifier instances of the entities that participate in R and the attribute domains that belong to it.

$$R^t \subseteq r_1E_1 \text{ x}\ldots\text{x } r_n E_j \text{ x } \text{dom}(A_1)\text{x}\ldots\text{x } \text{dom}(A_s) \quad (1)$$

Therefore:

$|R^t|$ = Number of instances that belong to R.

$|E_1|$ = Number of instances that belong to E_1.

According to the definition of cardinalities in [3], we define the cardinality constraints of an entity E_i as;

$$Card(E_i, R^t) = (n,m), \text{ IFF } n \leq |\{ a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n \in E_1,\ldots,E_{i-1}, E_{i+1}, \ldots E_n \,/\, (a_1,\ldots, a_{i-1},a_i ,a_{i+1},\ldots,a_n) \in R^t \}| \leq m \quad (2)$$

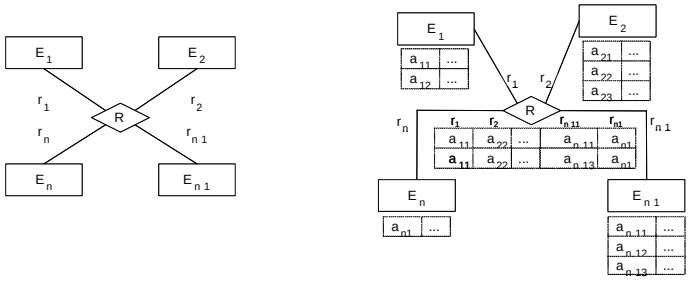

Fig. 2. Representation of a n ary relationship R and the set of instances associated to it

A set of instances that fulfils the minimum and maximum cardinality constraints (n, m), is defined as;

$$C_i = \{ a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n \in E_1,\ldots,E_{i-1}\ldots, E_{i+1}, \ldots E_n \,/\, n \leq |\{ (a_1,\ldots, a_{i-1},a_i ,a_{i+1},\ldots,a_n) \in R^t \}| \leq m\} \quad (3)$$

Therefore:

$|C_i|$ = Number of instances that fulfil the cardinality constraints of E_i.

$|C'_i|$ = Number of instances that not fulfil the cardinality constraints of E_i.

We propose to use a relative quantifier Q_{ri} that shows the percentage of instances that not fulfil the cardinality constraint in the role r_i of the entity E_k in a relationship R, as shown:

$$Q_{ri} = |C'_i| * |R^t|^{-1} \quad (4)$$

The quantifier Q_R shows the percentage of semantic loss in a n-ary relationship R and be calculated as shown:

$$Q_R = \sum_{i\ 1}^{n} (|C'_i| * |R^t|^{-1}) \quad (5)$$

We can easily calculate this quantifier in a real database applying directly the previous equation (4) on each element associated with a relationship. This will give us an idea about the percentage of the instances that do not fulfil the cardinality constraints in this role, and help us to decide improving the verification of the cardinality constraints. It is possible to predefine a maximum limit to semantic loss,

but when this value increases significantly, the improvement mechanisms could be enabled to recover the semantics. These mechanisms could be enabled or disabled depending on the actual cardinality semantics. For example, in certain relationships if the predefined limit is (Q_R<=0.002) then the improvement mechanisms should be activated when (Q_R>0.002).

In the total participation it is necessary to ensure that all instances must fulfill the cardinality constraints of the role r_i, that is, ($|C'_i|$=0), whenever an insertion operation is performed, the semantic loss percentage in this role must be Q_{ri}=0.

$$Q_{ri} = |C'_i| * |R^t|^{-1} = 0 * |R^t|^{-1} = 0$$

With the partial participation we could have instances that do not fulfil the cardinality constraints of the role r_i. Therefore, the cardinality constraint of r_i could be consistent when the value ($|C'_i|$>0), whenever an insertion operation is performed, the semantic loss percentage of this relationship could be Q_{ri}>0.

$$Q_{ri} = |C'_i| * |R^t|^{-1} > 0$$

We define the fuzzy participation similarly to the partial participation i.e., let R be a relationship and E_i, E_j are the entities involved in this relationship. If the role r_i corresponds to total participation in R and there is an instance in E_i which does not satisfy the constraints then we define this case as fuzzy participation role. Therefore, the cardinality constraint of r_i could be consistent when the value ($|C'_i|$>0), whenever an insertion operation is performed, the semantic loss percentage of this relationship could be Q_{ri}=[0,1].

$$Q_{ri} = |C'_i| * |R^t|^{-1} \in [0, 1]$$

4 Relaxing Cardinality Constraints

Taking into account the designer's point of view, there are two types of elements that can be distinguished in a EER schema; first order elements are the most important in the model and they always need mechanisms to verify its constraints, especially for these objects, it would be very useful applying the fuzzy participation; and second order elements, less important in the schema and which do not requiere to verify its constraints at the same time when a modification is done, so it would be useful applying a polling system and periodically verification.

Our method could be used on an actual database, where the developer or the administrator of a database can periodically gathers statistics about each element in it, these statistics are used to know exactly where the semantic losses are produced. A quantifier Q_{ri} is calculated for each role in the schema, if the value of any quantifier is greater than the predefined limit, a verification tool such as triggers system or others could be activated. It is not a good way to increases the number of ends points because this produces a high loss of cardinality semantics.

A partial view of a schema for a university is used to illustrate the approach of verifying cardinality constraints, figure (3). The two relationships in the ER schema are translated into relations containing as foreign keys the primary keys of the

associated entities. The foreign key options (***On Delete Cascade*** and ***On Update Cascade***) are available to both relations.

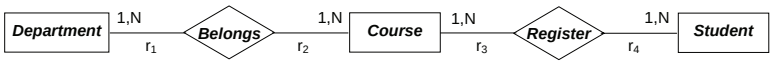

Fig. 3. ER schema for the case study

The cardinality constraints are verified by using a trigger-based approach [5] when deleting or updating the PKs, FKs are produced. But, the insertion operations need more effort to verification. For example, when inserting a tuple into ***Course***, it is very necessary to satisfy the roles $\mathbf{r_2}$, $\mathbf{r_3}$ because each course belongs to at least one department and one or several students have to be registered in each course. But, what happen if we need to relate the inserted course to a new student or a new department?. So, we must satisfy also the roles $\mathbf{r_1}$, $\mathbf{r_4}$. In such model, controlling the cardinality constraints needs complex predefined systems, and this may be produce other problems in the execution model of these systems or an impact on the database performance.

So, we suggest here that achieving more precise verification is very important whenever a high percentage of semantic loss is produced. In a real database, we can calculate this percentage in each element. These percentages give us an image of the semantic loss in the schema, as well as the elements that need to be verified. So, we need to gather statistics at time ***t*** about the instances participating in each entity and each role. In our example, at ***t*** there are 28 departments, 1244 courses, 16538 students, 3318 tuples in ***Belongs***, and 114564 tuples in ***Register***. From these, 4 courses in $\mathbf{r_2}$, 6 courses in $\mathbf{r_3}$, and 13 students in $\mathbf{r_4}$ are not participating in any relationships. The semantic loss percentage in each relationship objects $\boldsymbol{Q_{Belongs}}$ and $\boldsymbol{Q_{Register}}$ are calculated from equation 5, as shown:

$$Q_{\text{Belongs}} = (|C'_{r1}| + |C'_{r2}|) * |R^{t}_{.\text{Belongs}}|^{-1}$$

$$= (0* 3318^{-1} + 4* 3318^{-1})*100 \cong 0.12\%$$

$$Q_{\text{Register}} = (|C'_{r3}| + |C'_{r4}|) * |R^{t}_{.\text{Register}}|^{-1}$$

$$= (6*114564^{-1} + 13*114564^{-1}) * 100 \cong 0.017\%$$

The quantifier $\boldsymbol{Q_{Belongs}}$ shows the percentage of the instances that do not satisfy the cardinality constraints in ***Belongs***. While $\boldsymbol{Q_{Belongs}}$ shows the percentage of the instances that do not satisfy the cardinality constraints in ***Register***. Here, the database designer would decide if the semantic loss percentages are high or not. According to his decision, the cardinality constraints could be verified. Nevertheless, we think that as priority, we must do the best in those elements which have high semantic loss such as ***Belongs***.

But yet, in large database schemata the verification needs to be more simply. Moreover, we can consider only those roles that have high quantifier. The semantic loss percentage in each participations roles $\boldsymbol{r_1}$, $\boldsymbol{r_2}$, $\boldsymbol{r_3}$ and $\boldsymbol{r_4}$ are calculated from equation 4, as is shown below:

$$Q_{r1} = |C'_{r1}| * |R^t._{Belongs}|^{-1} = 0 * 3318^{-1} *100 = 0$$

$$Q_{r2} = |C'_{r2}| * |R^t._{Belongs}|^{-1} = 4 * 3318^{-1} *100 = 0{,}12\%$$

$$Q_{r3} = |C'_{r3}| * |R^t._{Register}|^{-1} = 6 * 114564^{-1} *100 = 0{,}005\%$$

$$Q_{r4} = |C'_{r4}| * |R^t._{Register}|^{-1} = 13 * 114564^{-1} *100 = 0{,}011\%$$

When the previous results are compared, we find that ***r₂*** has the highest percentage of semantic loss. Although the participation type in all roles is total, the developer should give the priority to ***r₂*** when he wants to verify the constraints. In such a way, we propose a method to select only those roles that need to be verified, saving much effort and time, reducing the errors, and getting better results when we evaluate the database performance.

Let the relation ***Department*** with 28 tuples and let each department associated to the same number of ***Course***, then this number ($n_{i.r1}$) is calculated by dividing the total number of the tuples in ***Belongs*** by the total number of departments in ***Department***, and let us do the same to find ($n_{i.r2}$, $n_{i.r3}$, $n_{i.r4}$) as is shown below:

$n_{i.r1}$ $= |R^t._{Belongs}| * |E_{DEPRTS}|^{-1} = 3318 * 28^{-1}$
$\cong 118$
(each department is related to 118 Courses).

$n_{i.r2}$ $= |R^t._{Belongs}| * |E_{COURSES}|^{-1} = 3318 * 1244^{-1}$
$\cong 3$
(each course is related to 3 departments).

$n_{i.r3}$ $= |R^t._{Register}| * |E_{COURSES}|^{-1} = 114564 * 1244^{-1}$
$\cong 92$
(each course is related to 92 students).

$n_{i.r4}$ $= |R^t._{Register}| * |E_{STUDENTS}|^{-1} = 114564 * 16538^{-1}$
$\cong 7$
(each student is related to 7 Courses).

This example shows clearly that the major semantic losses are produced in ***r₂***, ***r₄***, therefore, more efforts are required to verify it than to verify ***r₁***, ***r₃***. But, in the case of deletions or updates the verification of ***r₁***, ***r₃*** should be more important than ***r₂***, ***r₄***, because if an old department is deleted without semantic verification many relationships (≅118) may be deleted from ***Belongs*** because of the referential integrity actions. This deletion may be lead to damage in cardinalities semantics more than if an old student and his relationships (≅7) are deleted.

These results can be extended to database design phase, i.e., the designer of the database who has the better knowledge about the Universe of Discourse, could consider the following aspects during the database design:

- It is possible to leave without verification the roles that have total participation if the associated elements to these roles have a fixed number of instances or the insertion operations that are produced in these objects are very limited. But, in this case we must carefully verify the remaining operations such as deleting and updating. For example, ***Department*** has a fixed number of departments and consequently the insertion operations on it are very limited.

- In a binary relationship if the entities have approximately the same number of instances, then we can verify the entity, which gets more modifications (insert, delete, and update) at ***t***. For example, if the numbers of instances in $\boldsymbol{E_1}$, $\boldsymbol{E_2}$, $\boldsymbol{R^t}$ are $|E_1|$, $|E_2|$, $|R^t|$ respectively, and ($|E_1| \cong |E_2|$). Let we suppose that 30% and 5% of the instances are modified in $\boldsymbol{E_1}$, $\boldsymbol{E_2}$ respectively. The total instances in R^t that are modified by the referential integrity rules are ($e_{i.r1} = 0.3 * (|R^t| * |E_1|^{-1})$) and ($e_{i.r2} = 0.05 * (|R^t| * |E_2|^{-1})$). Because of ($E_1| \cong |E_2|$), we find that ($e_{i.r1} > e_{i.r2}$). So we can say that E_1 has more modifications at ***t*** and consequently its cardinality constraints need more verification.

5 Conclusions

The cardinality constraint is one of the most important constraints that can be established in a conceptual schema but the verification of these constraint is very difficult, especially in the case of insertions due to logical model constructs are not coincident with the conceptual model ones. Therefore, our work is addressed to the issue of the insertion operations. Some research prototypes have performed different and complex methods to solve this problem. Some solutions would lead to other problems such as the complex execution model or the impact on the database performance.

We propose a method to simplify the verification of cardinality constraints although three aspects have to be considered; (1) a minimum threshold of temporarily semantic loss in cardinality constraints could be allowed, it is measured periodically as a relative quantifier, and calculated by dividing the total number of the instances which do not fulfil the constraints by the total number of the instances in the associated relationship; (2) the designer should trust in his own design, because he must decide this threshold depending on the importance of each element; (3) the database must be periodically submitted to a polling system to recover the losses of cardinality constraints.

References

1. Elmasri, R. Navathe, S.: Fundamentals of Database Systems, Third Edition, Addison Wesley, 2000.
2. Toby J. Teorey: Database Modeling & Design, third edition, Morgan Kaufmann Series in data management systems, 1999.
3. Teorey, T., Yang, D., Fry, J. A Logical Design Methodology for Relation Databases Using the Extended Entity Relationship Model. Computer Surveys, Vol. 18. No. 2. 1986.
4. II Yeol Song, Mary Evans, E.K. Park, A Comparative Analysis of Entity Relationship Diagrams, Journal of Computer and Software Engineering, 3(4), 427 459, 1995.
5. Chen, P.: The Entity Relationship Model Toward a Unified View of Data, ACM Transactions on Database Systems, Vol. 1, N. 1. 1976.

6. D. Cuadra, C. Nieto, E. Castro, P. Martínez M. Velasco: Preserving relationship cardinality constraints in relational schemata, Database Integrity: Challenges and Solutions, Ed: Idea Group Publishing, 2002.
7. H. Al Jumaily, D. Cuadra, P. Martínez. PANDORA CASE TOOL: Generating triggers for cardinality constraints verification in RDBMS. IADIS International Conference, Portugal, 2003.
8. H Al Jumaily, D. Cuadra, P. Martínez. Incorporando Técnicas Activas para la Conservación de Semántica en la Transformación de Esquemas. VIII Jornadas de Ingeniería del Software y Bases de Datos 12 14 Noviembre 2003, Alicante.
9. Norman W. Paton, Active Rules in Database Systems, Springer Verlag, New York, 1998.
10. Lazarevic, B., and Misic. Extending the entity relationship model to capture dynamic behaviour. European Journal Information Systems 1 (2) pp. 95 106. 1991.
11. Balaban, M., and Shoval, P. MEER An EER model enhanced with structure methods. Information Systems 27, pp 245 275, 2002.
12. Guoqing Chen, Fuzzy Logic in Data Modeling; Semantic, Constraints, and Database Design. Kluwer Academic Publishers, London 1998.
13. Galindo J., Urrutia A., Carrasco R., Piattini M.: Fuzzy Constraints using the Enhanced Entity Relationship Model. XXI International Conference of the Chilean Computer Science Society, (Chile). 2001.