# TOTAL RELATIONSHIP INTEGRITY CONSTRAINTS

*Hussien Hassan Aly*

Computers and Automatic Control Department, Faculty of Engineering,
Alexandria University, Alexandria, Egypt.

**ABSTRACT**

We identify some integrity constraints that are not supported by current relational database management systems. These constraints span more than one table and can enhance the current RDBMS by providing some sort of encapsulation. We propose a complete set of operations together with a suggestion for an extension of the DDL of SQL so that such constraints can be declaratively included in the schema definition of database systems.

***Keywords***: Database system, Integrity constraints, Total relations, Object-relational systems, SQL

## INTRODUCTION

Currently, relational data models and the supporting database management systems (RDBMS) are the most widely used systems. However, one of the main points against RDBMS is that they do not provide encapsulation for the represented entities in the data model [1]. Encapsulation is a major issue in the object oriented approaches [2] that guarantees the isolation of different object classes from changes occurred to other object classes and treats object classes as an abstract data type[3]. Currently, many attempts are made to marriage the relational technology and the object oriented technology [4,5]. Extending the RDBMS to support encapsulation will reduce the gap between the two technologies.

Also, the normalization process - usually applied to the relational data model - may decompose a relation schema into smaller schemata in order to eliminate some update anomalies that may arise due to functional dependence between the attributes[6]. This decomposition breaks the encapsulation even more since now each of the smaller relations can be accessed and maintained separately. So, to get some kind of encapsulation, the RDBMS must provide automatic mechanisms that maintain separate but related relations. The only declarative mechanism provided by the current RDBMS to deal with related relations is by referential constraints[7]. Other methods such as trigger proceedures exist but are not declarative in nature and require some programming [8].

In fact as we will demonstrate in the next section, while the normalization process eliminates some redundancies and update anomalies, it may also breaks some of the integrity constraints that may be required in the system. Also, many of the natural constraints require that the designer of a database write some code for the RDBMS to ensure the integrity of the system with respect to these constraints which usually span more than one table. It will be much better if we can enhance RDBMS to automatically support such constraints in the same declarative manner as they support referential and other integrity constraints.

In this paper, we identify a kind of constraints that span more than one table and are not considered by current RDBMS. We call these constraints *total relationship integrity constraint (TRIC)*. These are conditions set by the designer on a many-many relationship between two entities in the E-R diagram representing the data model of the system. We then suggest complete set of operations that are needed

to automatically maintain the database with respect to this kind of constraints. These operations include insertions and/or deletions from more than one table. The idea is that to make the system treats these operations as atomic operations exactly as it treats the operations for referential integrity enforcement. The user now does not write any code to enforce these constraints but only declare them in the database schema. In order to declare such constraints to be recognized by a RDBMS, we suggest an extension of the SQL data definition language. In particular, the CREATE TABLE statement is augmented by some clause called "TOTAL" clause in order to register such a constraint into the schema.

The rest of the paper is organized as follows: The next section presents a motivating example and identify the kind of integrity constraints that we need to be supported by the RDBMS. The section that follows, the formal treatment of the identified constraint is given. Also, the mathematical notations and terminology that we use in the formal treatment are briefly summarized. In the last section, we suggest the formal specification of the operations and the SQL-like syntax for RDBMS to support the total integrity constraints. Finally, at the end of the paper the conclusion is given.

## A MOTIVATING EXAMPLE

The relational model supports various kinds of integrity constraints such as entity-integrity (candidate and primary keys), referential constraints (foreign keys), and domain constraints (domain values rules). See for example Reference 7 for an extensive treatment of such constraints. Of these constraints, Only the foreign key constraints involve more than one relation. In other words, the entity integrity constraints assume that each tuple in the relation represents a complete instance of an object. This is not always true when we consider normalized relations as discussed in the introduction.

Also, there are some constraints that may exist on relationships between entities and are not supported by the existing relational model. To demonstrate some of these constraints, we consider a simple example of the relationship between a STUDENT entity and a COURSE entity. The E-R diagram in Figure 1 illustrate this relationship as many-many relationship. In other words, a student may be enrolled in more than one course, and a course may have more than one student. The double line between the STUDENT entity and the diamond representing the ENROLL relationship indicates that we have the rule that " every student must be enrolled in some course". It is important to note that this is not a total functional mapping, instead it is a total relation.

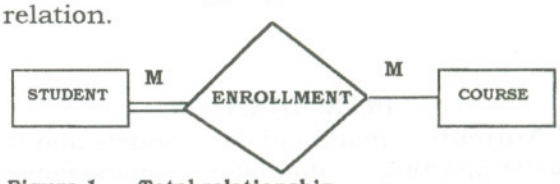


*Figure 1* **Total relationship**

Now transforming this simple E-R diagram into relational schema will result in three tables. A STUDENT table, a COURSE table, and an ENROLL table corresponding to the many-many relationship. The question that we address in this paper is how to reflect - declaratively in the relational schema - the rule that every student must participate in the ENROLL relationship? Notice that normalization principles forces us to map the ENROLL relationship to a separate table. Otherwise, the tables will not be even in the 2NF. Also this kind of constraints can't be implemented by any of the referential, entity, or the general CHECK rules that are supported by some current RDBMS. To maintain such constraints, insertions and/or deletions from two different tables may be necessary.

Since these kind of constraints actually dictates that all the instances in the domain (or range) of the relationship must participate by at least one tuple, we will call these constraints *total relationship constraints (TRIC)*. In the next section, we briefly review some terminology and notations that we will use in the rest of the paper.

## NOTATIONS AND TERMINOLOGY

Since we are going to specify some operations to be supported by the RDBMS, we use one of the languages that are designed for formal specifications of software systems. The language we chose is the Z language since its mathematical foundations and concise notations make it very close to the relational algebra. However, for our purpose, we will try to use the minimum notations. In particular, we use the notations for relations and functions and the related operators. Also, we will ignore some syntactical issues that are not in concern here. We refer the reader to Reference 9 for an introduction to the Z language and the notations that will be used here.

Z language is a strongly typed language, and in Z, relations and functions are complex types that built up from the power set and product constructors. Relations and functions usually are supported in the basic Z library. So, in order to define a relation type R between two types X and Y, we write R: $X \leftrightarrow Y$. Functions are special case of relations in which an element in the domain is allowed to relate with only a single element in the range. A function type F from X to Y is written as F: $X \rightarrow Y$.

A surjective function is denoted by $\twoheadrightarrow$ while an injective function is denoted by $\rightarrowtail$. If the function is partial, i.e. it is defined on a subset of the domain, then the notation is decorated with a small bar. For example, a partial surjective function is denoted by $\mid\!\!\twoheadrightarrow$ , and a bijective function is denoted by $\rightarrowtail\!\!\!\!\twoheadrightarrow$ .

Some operators that are defined on relations and function include dom(R) to get the domain of R, ran(R) to get the range of R. Also, domain restriction ($X \lhd R$) and range restrictions ($R \rhd Y$) to get the corresponding tuples in R that have specified domain and range elements respectively.

*Insert*

| S: $PN_1$ |
|---|
| s?: $N_1$ |
| s? $\notin$ S |
| S' = S $\cup$ {s?} |

***Figure 2.*** Insert operation schema

An operation "OP" schema in Z has two parts: A declaration part and a predicate part. An example operation is shown in Figure 2 which corresponding to insert an integer into a set S. A variable decorated by "'" is the value after the operation. An input/output variable is decorated by ?/! respectively. In the predicate part, all predicates in different lines are assumed to be conjunctive.

## TOTAL RELATIONSHIP INTEGRITY CONSTRAINTS

A total relationship integrity constraint (TRIC) is a constraint that can be specified on the relationship between entity sets. For simplicity, we will deal only with binary relationships.

As illustrated in the motivating example given earlier the TRIC requires that the domain of ENROLL relationship must be equal to the set of STUDENTS, and this relationship is represented in the relational model by a separate table different from the two tables representing the participating two entity sets. In other words, we could not encapsulate this relationship with any of the participating entities.

Formally, a total relation 'R' between two sets 'X' and 'Y' is written as:

$$R: X \leftrightarrow\!\!\!\!\rightarrow Y \mid \forall x \in X \bullet \exists y \in Y \bullet (x,y) \in R$$

To maintain the validity of this constraint, we have to make sure that dom(R) =X is true all the time. The system should compensate for actions such as updating X and updating R.

Now, in order for the RDBMS enforces automatically this constraint, we have to include two things:

1 The specification of operational procedures that the DBMS should perform for every action that may affect the integrity of the DB w.r.t. the TRIC.
2. The declaration of this constraint in the schema.

In the following two subsection we address these two problems in details.

**Specification of the Operational Procedure**

Given a total relationship constraint R: X$\leftrightarrow$ Y as defined above. The DB integrity w.r.t. this TRIC may be affected in the following cases: Insertion into X , deletion from X , deletion from Y and deletion from R.

It is worth noting that the case of insertion into Y does not affect the DB integrity, and the case of insertion into R is totally covered with the referential integrity constraint capabilities of the DBMS.

In the following, we give the formal specification together with a brief discussion for each of the above operations that should be performed in order to guarantee the integrity of the DB with respect to the TRIC. Also, we assume tables X and Y be of types $X$ and $Y$ respectively.

***1-Insertion into X :***

Inserting a new member "x" into X should be accompanied by inserting at least one new member into R. In other words, the transaction for insertion into X should also contain the appropriate set y$\subseteq$Y of values to indicate the relationship R with the new inserted "x". So, formally this operation can be written as a Z schema as shown in Figure 3.

*insert into* X

| |
|---|
| R: $X \leftrightarrow Y$<br>x?:X<br>y?: Y |
| y?$\subseteq$ Y<br>x?$\notin$ X<br>X' = X $\cup${x?}<br>R' = R $\cup$({x?}$\times$ y?)<br>Y' = Y |

***Figure 3*** **Schema for insert into X**

***2-Deletion from X :***

This operation should be accompanied with a cascade deletion of every tuple in R that represent a relationship with the candidate of deletion. The schema for this operation is shown in Figure 4.

*delete from* X

| |
|---|
| R: $X \leftrightarrow Y$<br>x?:X |
| x?$\in$ X<br>X' = X - {x?}<br>R' = R - ({x?} <\| R)<br>Y' = Y |

***Figure 4*** **Schema for delete from X**

***3-Deletion from Y :***

Deletion from Y may affect the TRIC if the deleted element will leave some element x$\in$X without any R relationship. If this is the case, then either all such x's should be deleted also, or a substitution is made to keep the validity of the TRIC. In general, Assuming that entity X and entity Y are independent, then deletion from one of them should not imply the deletion from the other one and the operation should not be done. The Z specification is shown in Figure 5.

*delete from* Y

| |
|---|
| R: $X \leftrightarrow Y$<br>y?:Y |
| y?$\in$ Y<br>Y' = Y - {y?}<br>R' = R - (R \|> {y?})<br>X' = X<br>dom(R') = dom(R) |

***Figure 5*** **Schema for delete from Y**

***4-Deletion from R :***

As in the previous case, the deletion of tuples from R should not leave some element x$\in$X without any R relationship with some y $\in$Y. Figure 6 contains the specification schema for this operation.

*delete from* R

| R: $X \leftrightarrow\!\!\!\rightarrow Y$<br>r?:R |
|---|
| r? $\in$ R<br>R' = R - {r?}<br>Y' = Y<br>X' = X<br>dom(R') = dom(R) |

***Figure 6*** **Schema for delete from R**

**Declaration of TRIC**

To declare a TRIC for a DBMS, we suggest extending the syntax of the SQL data definition statements. By adding a new TOTAL clause to the CREATE TABLE statement for the table corresponding to the relationship with TRIC. The general syntax of this new clause is

[TOTAL *constraint_name* ON d*omain_ table_ name* TO *range_table_name*
INSERT [RESTRICT | DEFAULT = *value* | *select_statement*]]

There are some conditions on the CREATE TABLE statement which include a TRIC:

- The *domain_table_name* and *range_table_name* must be referenced as FORIEGN in the same CREATE TABLE statement that contains this clause,
- The *domain_table_name* foreign key must have ON DELETE CASCADE.
- The *value* must have the same structure as the key for *range_table_name.*
- The *select_statement* must return a single value with the same structure as the key for *range_table_name.*

## CONCLUSION

In this paper, we identify the total relationship integrity constraint. We propose an extension to the current RDBMS to support this kind of constraints declaratively. We give specification of all the operations required to maintain this constraint and suggest extension to SQL to declare such constraints.

## REFERENCES


1. Arun K. Thakore, Stanley Y.W. Su, Herman X. Lam; "Algorithms for Asynchronous Parallel Processing of Object-Oriented Databases"; IEEE Transactions on Knowledge and Data Engineering, Vol. 7, pp. 487-504, No. 3, (1995).
2. P.M.D. Gray, K.G. Kulkarni, N.W. Paton; "Object-Oriented Databases - A semantic Data Model Approach"; Prentice-Hall, (1992).
3. J.V. Gutlay; "Abstract Data Types and the Development of Data Structures"; Communications of ACM , Vol. 20, pp. 396-404, (1977).
4. M. Carey, D. Dewitt, J. Naughton, M. Asgarian, P. Brown, J. Gehrke, D. Shah; "The BUCKY Object-Relational Benchmark"; ACM SIGMOD, pp. 135-146, (1997).
5. Jorng-Tzong Horug, Gwo-Dang Chen, Cheng-Yan Kao,Baw-Jhune Liu; "An extension of a relational query language to capture more information from objects with many-many relationships"; IEEE International Conference on Systems, Man, and Cybernetics , Vol. 2, pp. 1497-1502, (1994).
6. A.V. Aho, C.Beeri, J.D. Ullman, "The theory of Joins in Relational Databases", ACM TODS, Vol. 4, No. 3, pp. 297-314, (1979).
7. J.C. Date "An Introduction to Database Systems", 6th edition, Addison-Wesly, (1995).
8. C. Machgeels, "A Procedural Language for Expressing Integrity Constraints in a Relational Database." In G. Nijssen (ed.): Modeling in Data Base Management Systems, North-Holland (1976).
9. Deri Sheppard, "An Introduction to Formal Specification with Z and VDM" McGraw-Hill, (1995).


Received September 29, 1997
Accepted March 23, 1998

# القيود التكاملية ذات العلاقات الكلية

حسين حسن على

قسم الالات الحاسبة والتحكم الالى - جامعة الاسكندرية

## ملخص البحث

فى هذا البحث، تم تحديد نوع جديد من القيود التكاملية على نظم قواعد البيانات. تربط هذه القيود بين أكثر من جدول وتحتاج تغليفا للعمليات اللازمة لتطبيقها من قبل نظم إدارة قواعد البيانات. وقد اقترحنا مجموعة تامة من العمليات لتمكين نظم إدارة قواعد البيانات من تنفيذ هذه القيود. كذلك اقترحنا امتدادا لفقرات تعريف البيانات الخاصة بلغة **SQL** لتمكين المستخدم من إعلان هذه القيود فى المخطط التعريفى لنظم قواعد البيانات.