Dynamic Referential Integrity Model for Relational Databases

Ing. Is-Boset Jiménez Alvarado[1], Mtro. Juan Ramos Ramos[2], Dr José Juan Hernández Mora[3], Ing. Carolina Tlapalamatl López[4]

[1, 2, 3, 4] Tecnológico Nacional de México / Instituto Tecnológico de Apizaco, Carretera Apizaco Tzompantepec, esquina con Av. Instituto Tecnológico S/N, Conurbado Apizaco - Tzompantepec, Tlaxcala, Mex. C.P. 90300

Abstract: ***A Relational Database Model (BDR) is a database organization and management model consisting of storing data in tables made up of rows and columns. The relationship and integrity of the data between the different entities can be achieved by implementing the concept of Referential Integrity, which consists of the fact that the foreign key of a table must always refer to a valid row of the table to which reference is made. A BDR model will always have a number of at least two tables that are interrelated with each other through a primary key and a foreign key. A problem that often occurs in the design and implementation of a BDR is the necessary incorporation of new tables that are essential due to changes in user requirements, in turn implying that these new data entities implement referential integrity to ensure the values of specific attributes of other tables, for example; a predefined catalog of information. In this case, the modification to the Database (BD) and to the applications of interface with the end user would necessarily require those persons who fulfill the functions of design and administration of the databases, including the modification in the code of the user interfaces and the system backend. As a result of a research process, a Dynamic Referential Integrity Model is proposed, consisting of a solution that allows the inclusion of new data entities (catalogs) from which the concept of referential integrity can be applied dynamically, at the Database level, for new attributes.***

Keywords: Referential Integrity, Relational Database, Primary Key, Foreign Key

1. Introduction

Whenever a Relational Database (BDR) is designed, it is possible to establish referential integrity towards other tables, through primary keys and foreign keys in order to guarantee the relationship between data tables. A foreign key allows to guarantee that the value that appears in a relation for a given set of attributes, appears in the same way in par to a determined set of attributes in another relation. Those internal and external keys can be specified as part of instructions in a structured query language (SQL), creating the table as follows: "create table" using the "primary key" and "foreign key" clauses. Foreign key declarations are specified using the SQL Data Definition Language (LDD) which is provided by the database management system which, depending on the database engine, will allow users to carry out the definition tasks. of structures, which will store the data without mentioning the procedures and functions that are required to manage the data. Once the referential integrity is implemented, it is graphically reflected in the entity-relationship diagram of the database in such a way that when adding, deleting and updating a record, the actions are performed in the tables that are related to each other. In other words, referential integrity in a database is the constant correctness and compatibility of data [1].

2. Background

Between 1960 and 1970, computers had limited storage and processing capabilities. Although the applications were not as demanding as the current ones, it took a lot of work to achieve the expected results.
Currently, all people interact with data that, in some way, is stored in a physical medium and associated with a computer system that registers them and allows their access.

A **database** is a set of data structured and defined through a specific process, which seeks to avoid redundancy, and which will be stored on some mass storage medium [1].

Database management system: A Database Management System (DBMS) or DGBA (Data Base Management System) is a set of non-visible programs that administer and manage the information contained in a database. Database managers make it possible to manage all access to the database as they are intended to serve as an interface between the database, the user and the applications [2].

In a simplified way, a database management system can be seen as a layer of software that controls all access to the database.

The DBMS can implement instructions given by different users with different effects in a database. These instructions are grouped into: DDL (Data Definition Language).

2.1 DDL

It is the set of orders that define the structure of a database [2].

Relational data model: The relational data model belongs to the group of record-oriented data models. Today it is the model of greatest use and dissemination in different types of organizations.

2.2 Basic concepts in the relational model

The fundamental concept in the relational model is that the data is represented in a single way, at the level of abstraction that is visible to the user, that is, specifically as a tabular structure, made up of rows and columns or as a table with values [1] (See figure 1).

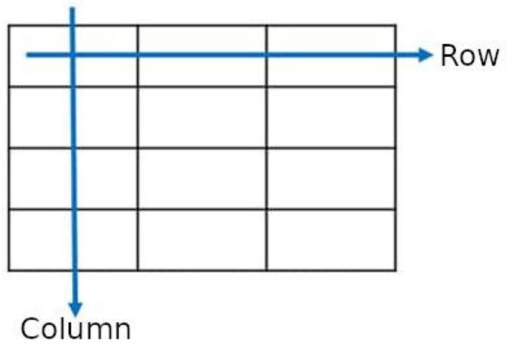


Figure 1: Tabular structure

This structure is formally called a relationship, although informally it is known as a table.

Thus, relationship is nothing more than a two-dimensional, or double-entry, representation consisting of rows or tuples and columns or attributes. It is the primitive and fundamental concept of the model [2].

Within the database design, the relationships represent the entities that were molded. The entities have attributes that distinguish them and each one of them is linked to a particular domain.

2.3 Primary Key

These keys play a role of great importance in the design of the database, since, if they are not correctly determined, the stored data will be unequivocally identified and this causes the repetition of tuples and the reduction of the data, which will generate more problems serious according to the progress in the implementation of a system.

The primary key is the identifier that the designer defines for each entity that will be stored in a relational database. It meets characteristics such as uniqueness and minimality [2].

In short, the primary key is the form or mechanism of identification of an instance within an entity. Expressed differently, it is how a tuple is identified within a relationship.

2.4 Foreign key

The foreign key is also of great importance in a relational system, it is an attribute or union of attributes, which allows the combination of data from the different relations of the system [2].

2.5 Normalization

One of the most important points to take into account in the design of a relational database is its normalization, implementing at least three ways:

- First normal form: Delete the repeated groups from the individual tables. Create a separate table for each related data set and Identify each related data set with a primary key.
- Second normal way: Create separate tables for sets of values that apply to multiple records and relate these tables to a foreign key.
- Third normal way: Delete the fields that do not depend on the primary key.

The three forms of normalization are fundamental to the process of organizing data in a database, where the rules for creating tables and defining the relationships between them must be taken into account. These rules are designed to protect data, in addition to providing a more flexible design to eliminate redundancies and inconsistent and unnecessary dependencies between tables [1].

2.6 Triggers

A trigger is a command that the system automatically executes as a side effect of modifying the database. To design a trigger mechanism, two requirements must be met [1]:

- Specify the conditions under which the trigger will be executed. This breaks down into an event that causes the trigger check and a condition that must be met to run the trigger.
- Specify the actions to be performed when the trigger is executed.

2.7 Relational algebra

The objective of teaching Relational Algebra is to facilitate learning to write SQL statements, given by manipulating data from relations with operations.

The result of applying a Relational Algebra operation is another relationship. This allows, then, the combination of operations and their nesting, since the entry of an operation could be the result generated by another previous operation [2].

From the point of view of relational algebra, let r1 (R1) and r2 (R2) be relations with the primary keys C1 and C2, respectively (where R1 and R2 denote the set of attributes of r1 and r2, respectively), it is said that a subset R1 of R2 is a foreign key that refers to C1 of the relation r1 if it is required that, for each tuple t2 of r2, there must be a tuple t1 of r1 such that t1 [C1] = t2 [R1]. Requirements of this type are called referential integrity constraints, or subset dependencies.

By default, in SQL foreign keys refer to the attributes of the primary key of the referenced table. SQL also supports a version of the *"references"* clause where you can explicitly specify a list of attributes for the referenced relationship. The specified list of attributes, however, must be declared as a candidate key of the relation to which it refers.

In this way, the connections between the various tables of the relational databases are handled, as shown in Figure 2.

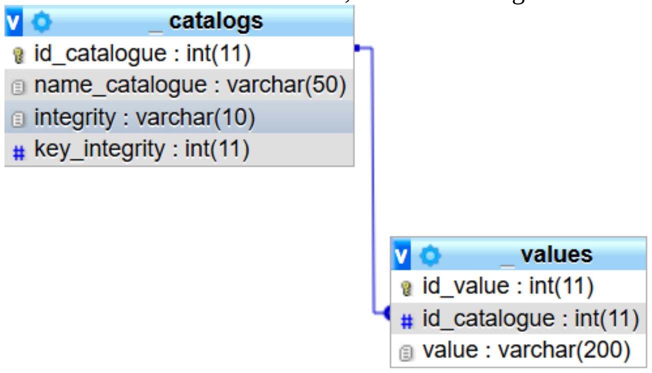


Figure 2: Connections between the tables of the relational databases

3. Solution Development

3.1 Problems of referential integrity for large databases

The success in the design of a BDR depends largely on the understanding of the requirements of the users and the complexity of the relationships between the different tables that make up the database. However, when various information tables are required to guarantee the integrity of the data, the database grows and the structure and amount of operations between the different entities is complicated, not to mention that in many cases where the number of Tables are larger and in turn they are always linked by primary keys and foreign keys to each other, the design of the entity-relationship model diagram becomes increasingly difficult to understand and modify, due to the large number of tables generated [1].

It is worth mentioning that an additional problem in the design of a BDR occurs when the database requires the inclusion of new information entities (tables), attending to new user needs, necessarily depending on a referential integrity to ensure the values in attributes specific, which in turn depend, for example, on a predefined catalog of information. In this case, the modification to the database and the interface applications with the end user necessarily requires those people who carried out the design and fulfill the administration function, as well as the design and coding of the user interfaces. , generating additional work to the versions with which it operates in a production environment.

3.2 Specific solution proposal

To counter this problem that arises from the implementation of the concept of referential integrity in a database, a Dynamic Referential Integrity Model (MIRD) is proposed, which consists of a solution that allows the inclusion of new data entities (catalogs) from of which the concept of referential integrity can be applied dynamically, at the database level, for new attributes, with the following characteristics:

1) ***A "catalogs" table*** that will allow storing the various attributes of the information.
2) ***An attribute "key_integrity"*** in the table *"catalogs"* that will allow establishing the relationship of each defined attribute towards its possible integrity values.
3) ***A "Referential_Integrity" table*** that will store the possible integrity values that the attributes can take in the *"catalogs"* table.
4) ***A compound key is suggested*** in the *"referential_integrity"* table, made up of the *"integrity_key"* attribute and an additional *"consecutive"* attribute to guarantee unique records in that table.
5) ***A "Values" data table*** that will store the useful values to an organization, depending on the possible attributes established in the *"catalogs"* table and valid by referential integrity through the values of the "*referential_integraty*" table.

3.3 Diagram of the dynamic referential integrity model

According to the characteristics mentioned in the previous point, the diagram for a dynamic referential integrity model can be seen graphically, as shown in Figure 3:

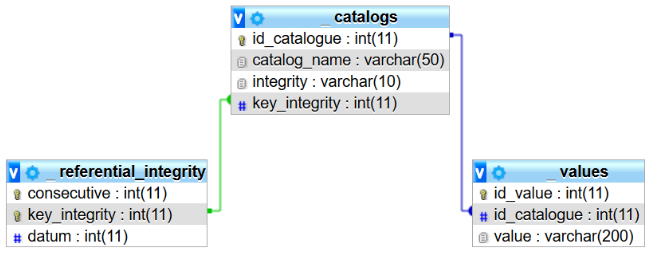
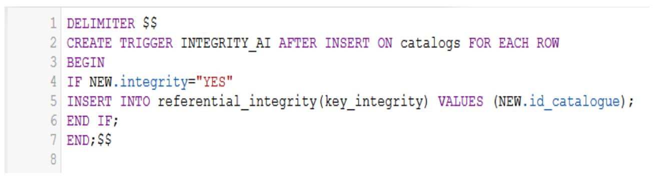


Figure 3: Relational scheme for the dynamic Referential Integrity Model.

3.4 Trigger to generate the value of "key_integrity"

To generate the value of *"integrity_key"* automatically in the *"catalogs"* table, a trigger will be used, which will have the function of being activated every time a new record is generated in the table, only when the *"integrity"* field Have the parameter (YES) as its value, once this is confirmed by the trigger called *" INTEGRITY_AI "*it will perform a new action that will insert a new record equal to the *" referential_integrity"* table, this is where the execution of a second trigger begins called *"KEYI_AI"* which will be activated when verifying that a new record in the *"referential_integrity"* table has been created. This second trigger will only have the task of inserting the value of *"integrity_key"* in the linked field *"integrity_key"* of the *"catalogs"* table , *see Figure 5*[3].

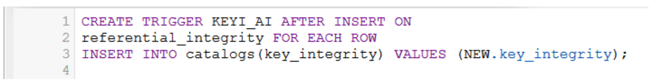
The unique values and keys that will be generated for the *"integrity_key"* field within the *"referential_integrity"* table will be a unique identifier that will highlight the difference between the unique keys of each catalog. For this purpose, see Figure 4:



```
DELIMITER $$
CREATE TRIGGER INTEGRITY_AI AFTER INSERT ON catalogs FOR EACH ROW
BEGIN
IF NEW.integrity="YES"
INSERT INTO referential_integrity(key_integrity) VALUES (NEW.id_catalogue);
END IF;
END;$$

```

Figure 4: Function of the first trigger.



```
CREATE TRIGGER KEYI_AI AFTER INSERT ON
referential_integrity FOR EACH ROW
INSERT INTO catalogs(key_integrity) VALUES (NEW.key_integrity);
```

Figure 5: Function of the second trigger.

4. Application Case

In order to verify the operation of the proposed Dynamic Referential Integrity Model, its use is implemented in the development of a web application for an information system of statistical variables in the matter of justice delivery.

In order to adapt the user interface to capture the information in the corresponding catalogs, you must consider the behavior of the triggers already defined in the BDR, therefore, the capture interface must be made up of two different forms, as shown in Figures 6 and 7.

In the first form, the two fields that are necessary for the registration of each new catalog will be saved, starting with the name of the catalog that refers to the field "catalog_name", followed by the field that contains the question "Does this catalog have referential integrity? ” (See Figure 6), which must be declared through the "integrity" field in the same table shown in figure 7:

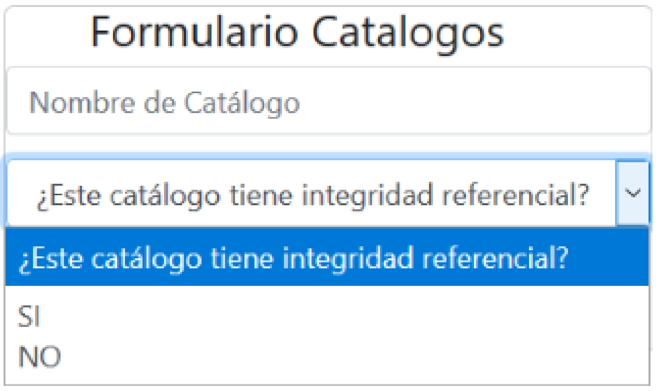

Formulario Catalogos
Nombre de Catálogo
¿Este catálogo tiene integridad referencial?
¿Este catálogo tiene integridad referencial?
SI
NO

Figure 6: Form for registering each new catalog

Once the catalogs that do have referential integrity have been defined, they will be shown through a *"select"* in the interface, which will obtain its content through a defined query, depending on whether you are working with a programming language and / or a specific framework.

After selecting the catalog to be fed, the catalog value is filled, which corresponds to the second form (see figure 7), which is saved in the "value" field of the table called "values", likewise , a function will be defined in the code to insert the value of the id of the selected catalog:

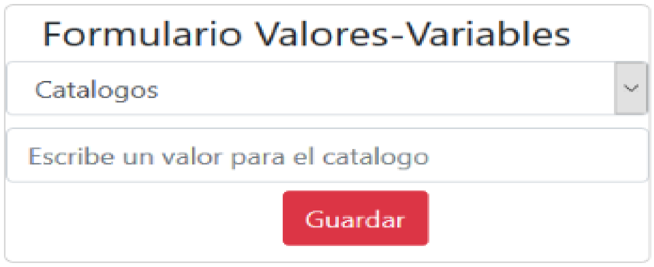

Formulario Valores-Variables
Catalogos
Escribe un valor para el catalogo
Guardar

Figure 7: Form for filling in the catalog value

5. Conclusions

Once the database model with dynamic referential integrity was designed and implemented, it was concluded that the model meets the specifications and meets all the needs of database systems with large volumes of information that interact between if through a set of relationships between the tables, thus adding "*Dynamic Referential Integrity"*, so that the advantages shown by the implementation of this model are of great help when working on systems that have multiple catalog management, facilitating and speeding up each of the development phases of the systems and mainly their databases, making them simpler, digestible and lightweight, especially when it is necessary to add more information catalogs to the DB.

The proposed Dynamic Referential Integrity Model allows the incorporation of new data entities with reference to other existing tables, with the simple intervention of the end user, without requiring the Database Administrator or the developers of the applications that serve as user interface and system operation at the backend level.

References

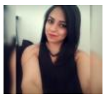
[1] Silberschatz, Korth & Sudarshan (2014). Fundamentos de base de datos.
[2] Reinosa, Jose E., Maldonado Alejandro, Muñoz Roberto, (2012). Bases de Datos, Argentina.
[3] Ding Li, Zeng Fanjin, Chen Xiaoji, and Liu Jingzhong (2013). Research on SQL Server Trigger to Implement Referential Integrity.

Author Profile

Is-boset Jiménez Alvarado has a degree in Engineering in Information and Communication Technologies from the Instituto Tecnológico de Apizaco de Tlaxcala, from 2016. He is currently studying the masters in Computer Systems in Software Engineering from the TecNM / Instituto Tecnológico de Apizaco.

Juan Ramos Ramos has a degree in Computer Science from the Instituto Tecnológico de Apizaco, from 1993. He is also a Master in Computer Science and Telecommunications from the Instituto de Estudios Universitarios, A.C.; he works as a full-time professor at the TecNM / Instituto Tecnológico de Apizaco in the area of Systems and Computing, teaching at the undergraduate and postgraduate level, in the areas of Programming and Software Engineering.

José Juan Hernández Mora has a Doctor of Teaching Excellence degree from the University of Los Angeles, 2019. He also has a degree in Computer engineer from the Universidad Autónoma de Tlaxcala, from 1994. Master in Computer science at the National Center for Research and Technological Development of the TecNM, 2003. Research professor at the Tecnológico de Apizaco del TecNM. Teacher of the Master of computer systems of the Instituto Tecnológico de Apizaco.


Carolina Tlapalamatl López has a degree in Engineering in Information and Communication Technologies from the Technological University of Tlaxcala, since 2018. She is currently studying the master's degree in Computer Systems in Software Engineering from the Technological Institute of Apizaco