European Journal of Research Development and Sustainability (EJRDS)
Available Online at: https://www.scholarzest.com
Vol. 3 No. 11, November 2022
ISSN: 2660-5570

INTEGRATION AND VALIDITY OF DATA IN INFORMATION SYSTEMS

Ahmad Hamid Shakir
Students Affairs and Registration Department, College of Nursing , Al-Muthanna University, AlMuthanna, Iraq.
Amajic-1984@mu.edu.iq

Article history:

Received: 1st September 2022
Accepted: 1st October 2022
Published: 4th November 2022

Abstract:

In this article, we will discuss a set of applied solutions related to creating a set of data constraints, which allow organizations to organize their work in terms of using information stored in relational databases as well as maintaining them through a database management system. The theoretical basis for these constraints is the embedding mechanisms that are used in this article in order to use null values, and null values are used for their practical importance. In order to obtain correct results for this problem, the term non-periodic databases was introduced, and in this article it was studied in detail, by providing many explanations that prove their scientific importance and effectiveness. Here, the automatic construction of different reference constraints is introduced, as well as an algorithm to remove unwanted referential integrity constraints

Keywords: Database, Database management system, Database integrity

1. INTRODUCTION

In all institutions, we need effective and reliable ways to save, install, update and follow up data on a continuous basis and with high accuracy. Databases are the most widely used systems for storing and storing customer data, stocks, or any information about the company. The integration of databases is a necessary need where the saved data can be combined with other applications or files and used by staff cadres or heads of institutions for different purposes, an example of this is the work of reports, data analysis and other necessary needs.

The process of collecting data from multiple sources such as databases, cloud, repositories, files, etc. to create a clean, unified and comprehensive version of the enterprise which is known as database integration. The benefit of the database integration process is to make the data accessible to all employees in the organization as well as customers without the need to repeat it.

To understand the process of database integration, we will use the following example: A certain company saves and stores its accounting data in Oracle databases, and at the same time, this company stores and saves customer data in Salesforce databases. By using these database integrations, all employees can access the data collected for both systems in one dedicated location such as a unified repository or a unified database. Other organizations and some companies use database integration in websites to standardize data.

1.1. Domain restriction

When defining the structure of database relations, restrictions on the allowed values in columns can be set. The definition of an attribute type on a relation specifies a base constraint that is controlled by the DBMS. The specified column cannot contain a value that conflicts with the selected type, such as a character string in a column whose type is Date. In addition, restrictions can be added to the attribute values in the relationship column, for example, the date must be set in a certain interval. An attempt to enter a value for this attribute that is outside the specified range will be blocked by the DBMS.

1.2. Entity Integrity

Each relation must have a primary key that has a unique non-null value in each row of the relation. The basis for determining the primary key is the set of functional dependencies formed by the database designer. The PRIMARY KEY property is assigned to the set of relation attributes defined in this way. Within a relationship, this property can only be specified once. Another way to ensure that the values of a subset of attributes for alternative keys of a relation are unique is to assign the UNIQUE property to that subset.

1.3. Referential Integrity

If, in some respect, attribute values in a row can only take on values that have matching attribute values in another respect, then this constraint is called referential integrity. To implement it, the DBMS has a foreign key apparatus. This defines the main relation and the subordinate relation: there cannot be rows in the subordinate relation that do not have a corresponding row in the main relation. The database management system will not allow an operation to add a row to a child relation if there is no corresponding tuple in the main relation, and the operation of deleting a

row in the main relation will be rejected if there is a corresponding related row in the child relation (dangling references are prohibited).

It should be noted that integrity constraints can interact with each other and, moreover, contradict each other[1][2]. Domain constraints, in turn, can significantly increase referential integrity, up to blocking entry of records into a subrelation. This paper assumes that referential constraints are not affected by other constraints. This does not deprive the study of relevance, since typed inclusion dependencies are used, which are of great practical importance, but do not interact with functional dependencies. Domain constraints can only narrow the range of acceptable values, but this does not contradict the results of this article. The constraint interaction problem can be explored further as an extension of the presented research results.

Currently, various options for automating the construction of the first two types of integrity constraints are known, and referential integrity remains without due attention. However, when determining referential integrity, automation tools can be used.

With the correct (classical) design of the DB schema [3][4], in most cases, foreign keys are set on the attributes of the primary key of the main relation. This fact will be used to automate the construction of links on the database schema.

Often, objects duplicate the structure of the enterprise's workflow, Which leads to the violation of data freedom and independence, as well as the destruction of the structure of the database, with the need for its modernization. Then, relationships between objects are immediately created, also based on the intuition of the designer (ER-diagram - entity-relationship). A typical mistake with this approach is the "load" of links with the semantics of the application, i.e., database objects are "hidden" in the links. requirements for the qualification of the designer. Examples of tools for such systems include ERWin, BPWin, etc.

When converting an ER diagram into a database schema, the mechanism of automatic link generation is used. In this case, the transformation rules are heuristic with many exceptions and unrealizable situations, the reason for which is the ambiguous interpretation of entities and relationships. Therefore, it makes no sense to talk about the correctness of the results of this process.

The purpose of this work is to study the properties of referential integrity constraints and implement software that allows you to automate the construction of a correct and non-redundant set of links on a database schema that implements referential integrity constraints. Based on the development of a new mathematical apparatus and on the basis of the well-known mechanism for using links in the database schema, algorithms for the automatic formation of referential integrity constraints have been developed. The correctness and nonredundancy of the results of the constructions are proved.

2. Problem statement

Referential integrity constraints on data are one of the main types of database constraints that are supported by most existing DBMS. To do this, the DBMS uses the links established on the database schema. Path $U = \{A_1, A_2, \ldots, A_n\}$ — set of attributes defined in the database; [Ri] is the attributes by which the relationship(Ri) is defined., $[R_i] \subseteq U$; $R = (R_1, R_2, \ldots, R_k)$ — DB; $S = \{[R_1], [R_2], \ldots, [R_k]\}$ — database schema, $1 \le i \le k$. The values of the attributes by which the relations are connected are borrowed from the main relations (dictionaries) into the subordinate relations. The relationship (1:M or 1:1) is established between them, and it is a relationship from the official to the employee, and this is considered essential in the formation of links [5]:

Definition 1: Let $[R_i]$ and $[R_j]$ be relationship schemes (not necessarily distinct),V ⊆ Ri and W ⊆ Ri, determinants V = determinants W , then the relation $R_i[V] \subseteq R_j[W]$ is called an inclusion dependence.

In this definition $|V|$ is the cardinality of the set V ; $R_i[V] = \pi_V(R_i)$ is the projection of the relation R_i over the attributes of V .

We must bear in mind that there are practical aspects used in the embedding processes. We assume that ($W = V$) is one of the necessary conditions for making a connection, and these embedding operations are called (printing) [6][7]. In applied problems, one often has to deal with uncertain data, when some of the characteristics of objects, except for the identifier (primary key), are unknown. In this case, the condition $\pi_V(R_i) \subseteq \pi_W(R_j)$ in Definition 1 may not be satisfied, i.e., there may be tuples with undefined values for R_i. These tuples correspond to tuples in R_j , but with specific values. In database technology, this correspondence is called extended association. Further definition 1 will be changed in accordance with this addition. Consider an example where a 1 : 1 relationship should be established between relationships.

Example: Let $A1$ - "Personnel number of the employee", $A2$ - "Name of the employee", $A3$ - "Date of dismissal of the employee". There are functional dependencies: $A1 \rightarrow A2$ and $A1 \rightarrow A3$. The areas of definition of these dependencies are different: the first dependency is defined for the entire set of employees, and the second one is defined only for the laid-off employees. This should serve as the basis for constructing the decomposition: $R_i[A1,A2]$ and $R_j[A1,A3]$. 1 : 1 relationship is established between R_i and R_j, where R_i is the main relationship, and R_j is the subordinate one. Note that such a decomposition solves the problem of knowingly null values: in the combined relation $R[A1,A2,A3]$, the value of the attribute $A3$ will have a null value for all non-dismissed employees. The formal definition of the domain of inclusion dependence is a time-consuming task that requires a separate study, in this article we will limit ourselves to this example to understand the term "domain of definition of functional dependence".

Let's introduce the notation: PK(R_i), or simply PK(i), It acts like a primary key which is related to R_i; $L(i, j, X)$ is a 1:1 or 1 : M relationship from R_i to R_j , established by the set of attributes X, where R_i is the main relationship, and R_j is the subordinate relationship; $L_1(i, j, X)$ is a 1 : 1 relationship from R_i to R_j ; $L_M(i, j, X)$ is a 1 : M connection from R_i to R_j . Note that for Ri, there can be several alternative primary keys and many relationships in which R_i is master or slave.
Next, we will consider the definition of relationships between database relations, which we will use as extended referential integrity constraints that take into account the presence of null values.
Definition 2: A relationship $L_1(i, j, X)$ is allowed between relations R_i and R_j if X = PK(R_i) = PK(R_j) and for any realizations of R_i and R_j $\pi_X(R_j) \subseteq \pi_X(R_i)$.
Nulls are not allowed in this definition because the attributes of X are in both respects components of the primary key.
Definition 3: A relation $L_M(i, j, X)$ is admissible between relations R_i and R_j if PK(R_i) ≠ PK(R_j) and PK(R_i) ⊆ [R_j].
In definition 3, the relation R_j may contain null values for attributes that do not belong to the primary key. Note that definitions 2 and 3 correspond to typed containment dependencies that are supported by the DBMS through the creation of foreign keys. The integrity constraint given by the connection $L_M(i, j, X)$ does not imply the fulfillment of the condition πx (Rj) Subset πx (Ri), where X = Ri ∩ Rj, since attributes of X that do not belong to PK(R_j) can take undefined values, while in R_i they have specific meanings. The meaning of the restriction is that the undefined value of any attribute in R_j can only be replaced by the defined value that is in R_i. The search for links corresponding to definitions 2 and 3 is quite simply algorithmized, which makes it possible to detect most of the referential integrity constraints in automatic mode[8].
The links below are used in database schemas:

1. At least one is mandatory.
2. only one.
3. There is no or optional one.
4. There is no or set of options.

Since the links in the database schema are paired, link 1 is a special case of a 1: M link on the side of the second relation, link 2 is a special case of a 1:1 link on the side of both relations, link 3 coincides with a 1:1 link on the side of the second relation, bond 4 coincides with bond 1: M on the side of the second relation. Other relationship options that have relationship type 3 on the side of the first relationship contradict the definition of an inclusion dependency and are not a constraint on referential integrity of the data. Such connections will be further ignored. A special case is the 1:M relation, which allows tuples with null common attribute values in the second relation when they are not key. Since the undefined value is not equal to the defined value, this relationship option contradicts the inclusion dependency, but remains an integrity constraint. Let us leave such a connection under consideration, calling it an extended connection. Thus, the task is to develop algorithms for generating referential integrity constraints in accordance with definitions 2 and 3.
Let us formulate the acyclicity condition for the set of relations by analogy with the acyclicity condition for inclusion dependencies [9], but taking into account extended relationships.
Definition 4: A set of relations R will be called acyclic if there is no ordered subset of relations
$\{ R_{m(1)}, R_{m(2)}, \ldots, R_{m(s)} \} \subseteq R$
such that there are connections
$T(n(1), n(2), X_1)$, $T(n(2), n(3), X_2)$, ... , $T(n(s), n(1), Xs)$ ……. (1)
$s > 1$ and X_1 Subset X_2 Subset · · · Subset Xs
Otherwise the set of relations R will be called cyclic. Sequence (1) may contain extended links.

Theorem: A connection is redundant if there are connections.
$T(i, n(1), X_0)$, $T(n(1), n(2), X_1)$, ... , $T(n(p), j, Xp)$ …… (2)
X Subset PK(i) Subset X_s Subset $R_n(s)$, s = 2, 3, . . . , p ……. (3)
where m is an array of relationship numbers.
Proof : Note that the conditions PK(i) ⊆ Rj and PK(i) ⊆ $Rm(1)$ are a consequence of the presence of the links $T(i, j, X)$ and $T(i, n(1), X_0)$, respectively.

1. Let conditions (2) and (3) be satisfied. Since the set of attributes PK(i) is present in each relation of the chain
   $R_i, R_{n(1)}, R_{n(2)}, \ldots, R_{n(p)}, R_j$ …….. (4)

then by definition it participates in all bonds of the sequence (2); Consequently,
$\pi_{PK(i)}(R_i)$ Subset $\pi_{PK(i)}(R_{m(p)})$ Subset · · · · · · Subset $\pi_{PK(i)}(R_{m(p)})$ Subset $\pi_{PK(i)}(R_j)$.
This sequence of inclusions ensures that R_j does not contain a tuple with the value PK(i) not contained in other chain relations (4). This serves as a stronger constraint than the constraint $L(i, j, X)$ defines and, therefore, it is redundant in accordance with Definition 3. Let the constraint $L(m(p), j, X_p)$ in the sequence (2) be extended, and the relation $L(i, j, X)$ is not extended. The other relationships in sequence (2) cannot be extended since the common attributes in the second relationship must be key. No tuple with empty values of all attributes PK(i) will appear in relation to R_j, since this constraint is set inside R_j . Therefore, the connection $L(i, j, X)$ can be removed[10].
Suppose that in chain (4) there is a relation $R_{m(s)}$ that does not completely contain the set of attributes PK(i). Then, when passing from $R_{m(s-1)}$ to $R_{m(s)}$, the restrictions on the attributes PK(i)−[$R_{m(s)}$] due to the sequence of links (2) are lifted.
Let PK(Ri - the key of the relation Ri) corresponding to the relation $T(i, j, X)$, Then:

for each $L(i, j, X)$ in L
$l = 1$
$m(l) = i$
iterations = true
do while iterations
for each $L(v, w, X_l)$ in L
where $L(i, j, X) \neq L(v, w, X_l)$
substitution = false
if $v \in m[1, \ldots, l]$ and $PK(R_l) \subseteq R_w$ then
if $w = j$ then
$L = L - L(i, j, X)$
exit do
else
if $w \in m[1, \ldots, l]$ then
$l = l + 1$
$m(l) = w$
substitution = true
endif
endif
endif
endfor
if not *substitution* then *iterations* = false
end do
endfor

CONCLUSION

The paper considers the simplest types of paired data integrity constraints, borrowed from object-oriented data models. However, in practice, there are structural restrictions on data that connect not only pairs, but also more database components in one restriction, for example, for the fifth normal form, when three or more relations obtained after decomposition must be connected. To solve this problem, it will be necessary to develop a formal theory of inclusion dependencies (the theoretical basis of referential integrity) for the case of a set of relations: building a non-redundant set of dependencies, acyclicity, developing algorithms for automatically generating programs (triggers) that serve constraints.

REFERENCES

1. Gasca R.M., P,erez-, Alvarez J.M. Compliance validation and diagnosis of business data constraints in business processes // Inform. Syst., 2015.
2. Agrawal & Gehani, ODE (Object Database and Environment): The language and the Data Model, Sigmod Record 2017.
3. Fishman et al, IRIS: An object oriented database management system, ACM Trans Office Information Syst., Vol 5 No 1, Jan 2011.
4. S. Zdonik and D. Maier, Fundamentals of Object-Oriented Databases, in Readings in Objectoriented Database Systems, Morgan Kaufman, San Mateo, Ca., 2013.
5. Pastor, J.A.; Olivé. A. "Supporting Transaction Design in Conceptual Modeling of Information Systems", 7th Int. Conf. on Advanced Information Systems (CAiSE'95), Finland, June 2014.
6. Teniente, E.; Urpí, T. "A Common Framework for Classifying and Specifying Deductive Database Updating Problems", 11th Int. Conf. on Data Engineering (ICDE), Taipei (Taiwan), 2012.
7. Bancilhon, F. "Supporting View Updates in Relational Data Bases" In Data Base Architecture (Bracci and Nijssen Eds.), North Holland, Amsterdam, 2015.
8. Sadri,F.; Kowalski,R. "A theorem-proving approach to database integrity". In Minker,J. (Ed.) "Foundations of deductive databases and logic programming", Morgan Kaufmann Pub., 2010.
9. Bancilhon,F.; Ramakrishnan,R. "An amateur's introduction to recursive query processing strategies". Proc. ACM SIGMOD Int. Conf. on Management of data. Washington DC., May,2010.
10. Kowalski,R.; Sadri,F.; Soper,P, "Integrity checking in deductivedatabases". Proc. of the 13th. VLDB Conference, Brighton 2014.