

12. Hisham Adnan "An Approach to Solve Incomplete Database Problems", 1999 Msc Thesis to University of Technology


20. IEEE Computer Society "Data Engineering", December 2000 Vol. 23 No. 4


constraints as well as of data entry applications.

2. Application Specific Business-Rule proposed has the properties of practicality, simplicity, usability, minimalism, consistency, universality, and precise results.

3. Functional dependency proposed rules can be used efficiently for Null existing problems solving, practically, a designer must put all these rules determining the suitable attributes which can be treated by this approach, it gives a precise answers and results.

4. A metadata approach will be more efficient and practical when the vendors of DBMS provide the solution within the software, because it is a database function, the DBMS can handle any number of relations putting the users aware of any load, DBMS would provide facilities for generating metadata-relations and managing all the operations done previously by the user, transparently update the data among related relations after updating null values by any user at any site of the distributed database. The DBMS optimizer will be care of the performance. Metadata approach is recommended for commercially implementation.

5. Tuple-marks proposed approach is a practical useful solution for treating null values, marking them, and preventing side effects of 3VL.

6. A combined treatment for null values in distributed database is one of the powerful solutions to provide a unified treatment of different types of nulls at difficult databases.

References


(5) Fernando Lozano "Introduction to Relational Database Design" 1999.


This approach is based on two rules of solution:

First Rule (Rule-1): The extracting of unique marker $va(n)$ is the intersection of $va(n1)$ and $va(n2)$.

The table below shows the results of different inputs.

<table>
<thead>
<tr>
<th>$va(n1)$</th>
<th>$va(n2)$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>va-mark</td>
<td>va-mark</td>
<td>va-mark (or -)</td>
</tr>
<tr>
<td>ce-null</td>
<td>ce-null</td>
<td>ce-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>ma-null</td>
<td>ce-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>li-null</td>
<td>li-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>pe-null</td>
<td>-</td>
</tr>
<tr>
<td>ce-null</td>
<td>lm-null</td>
<td>li-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>va-mark</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>li-null</td>
<td>li null</td>
</tr>
<tr>
<td>ma-null</td>
<td>pe-null</td>
<td>pe-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>lm-null</td>
<td>lm-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>va-mark</td>
<td>ma-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>pe-null</td>
<td>pe-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>li null</td>
<td>-</td>
</tr>
<tr>
<td>pe-null</td>
<td>lm-null</td>
<td>pe-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>va-mark</td>
<td>-</td>
</tr>
<tr>
<td>li null</td>
<td>li-null</td>
<td>li-null (or -)</td>
</tr>
<tr>
<td>li null</td>
<td>va-mark</td>
<td>va-mark (or -)</td>
</tr>
<tr>
<td>lm-null</td>
<td>va-mark</td>
<td>va-mark (or -)</td>
</tr>
<tr>
<td>lm-null</td>
<td>li-null</td>
<td>lm-null (or -)</td>
</tr>
<tr>
<td>lm-null</td>
<td>li null</td>
<td>lm-null (or pe-null)</td>
</tr>
</tbody>
</table>

Composition Rule-1

Second Rule (Rule-2):

This rule takes the composition to be the union of $va(n1)$ and $va(n2)$.

The possible values as $va(n1) \cup va(n2)$.

All the rest results will be extracted similarly as shown in next table.

<table>
<thead>
<tr>
<th>$va(n1)$</th>
<th>$va(n2)$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ce-null</td>
<td>ce-null</td>
<td>ce-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>ma-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>li-null</td>
<td>li-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>pe-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ce-null</td>
<td>lm-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>ma-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>li-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>pe-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>lm-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>ma-null</td>
<td>va-mark</td>
<td>ma-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>pe-null</td>
<td>ma-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>li-null</td>
<td>lm-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>lm-null</td>
<td>lm-null</td>
</tr>
<tr>
<td>pe-null</td>
<td>va-mark</td>
<td>ma-null</td>
</tr>
<tr>
<td>li-null</td>
<td>li-null</td>
<td>li-null</td>
</tr>
<tr>
<td>li-null</td>
<td>li-null</td>
<td>li-null</td>
</tr>
<tr>
<td>li-null</td>
<td>li-null</td>
<td>li-null</td>
</tr>
<tr>
<td>li-null</td>
<td>va-mark</td>
<td>li-null</td>
</tr>
<tr>
<td>lm-null</td>
<td>li-null</td>
<td>lm-null</td>
</tr>
<tr>
<td>lm-null</td>
<td>lm-null</td>
<td>lm-null</td>
</tr>
</tbody>
</table>

Composition Rule-2

5.1 Conclusions

It is a common belief that null values are one of the important features of a database application. Whether single data source of distributed data sources, it must be prevented.

1. Preventing Null values is obviously an important step to reduce problems. This requires an appropriate design of the database scheme and integrity
4. A Combined Treatment Approach

There are many different types of null values, a practical work recommended on providing a combined treatment deals with different types of nulls, in this approach we try to do that. [4], [6], [26], [23], [20], [13].

A category of the reasons of missing data proposed which are:
1. Certainly nulls
2. Maybe null
3. Place empty null
4. Limited maybe nulls

The null attributes may be filled in with markers of any one of the following types:

<table>
<thead>
<tr>
<th>Marker</th>
<th>Representing for</th>
</tr>
</thead>
<tbody>
<tr>
<td>va-mark</td>
<td>Any element from the domain</td>
</tr>
<tr>
<td>ce-null</td>
<td>Certain null</td>
</tr>
<tr>
<td>ma-null</td>
<td>Maybe null</td>
</tr>
<tr>
<td>pe-null</td>
<td>Place empty null=Vs where Vs is subset from the domain of the attribute A</td>
</tr>
<tr>
<td>li-null</td>
<td>Limited null=Vs where Vs is subset from the domain of the attribute A</td>
</tr>
<tr>
<td>im-null</td>
<td>Limited maybe null</td>
</tr>
</tbody>
</table>

Let:

A be a attribute
D be a domain of a given attribute
T be a tuple
Va be a function which takes a marker and returns a set of possible values.

The table below shows relations between markers and the function Va.

<table>
<thead>
<tr>
<th>X</th>
<th>Va (X) Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>va-mark</td>
<td>{v}</td>
</tr>
<tr>
<td>ce-null</td>
<td>∅</td>
</tr>
<tr>
<td>ma-null</td>
<td>Vs U {∅}</td>
</tr>
<tr>
<td>pe-null</td>
<td>{v}</td>
</tr>
<tr>
<td>li-null</td>
<td>Vs</td>
</tr>
<tr>
<td>im-null</td>
<td>Vs U {∅}</td>
</tr>
</tbody>
</table>

The following properties are true:

- If t.A is a va with va(t.A) = {v}, then (t.A = v)
- If t.A is a ce-null, then (t.A ∈ Dom (A))
- If t.A is a li-null with va(t.A) = V, then (t.A ∈ V)
- If t.A is a ma-null, then (t.A ∈ Dom (A) ∩ (t.A = 1))
- If t.A is a pe-null, then (t.A = 1)
- If t.A is a im-null, with va(t.A) = Vs U {∅}, then (t.A ∈ ∅) ∩ (t.A = 1)

Implementation

For Combined Treatment Approach implementation, we proposed two solutions or rules for null values treatment, especially for distributed database systems. These approaches are based on null markers composition, which exists for a certain tuple in more than one site.

Assume that:

- t1, t2 attribute values at site1, site2 for attribute A
- n1, n2 markers of attributes
- va(n1) value of attribute A according to t1
- va(n2) value of attribute A according to t2
3. Metadata Approaches

Rule04: A new relation will be created automatically by the application programs control when such null values occurrences during data insertion and manipulation. The new relations attributes will be (pkvalue, relation, column, status), where pkvalue is the primary key value and relation is the source relation name. There will be data in the metadatanrelation associated with the source relation. That will be the duty of the application program or precisely triggers design for that purpose to maintain and update the metadata-relation transparently after any knowledge of the null value changed, when the unknown values become known the row updated will be triggered to delete associated row in the metadata-relation. All operations will be controlled using this approach.

41, 51, 19, 26, 23, 20, 13, 25

PKvalue: The primary key value of the tuple contains null value.
Relation: The relation to which the null value tuple belongs.
Column: The attribute name (domain) which the null value includes.
Status: The status of the null value which is shown on table 3.22.

For charges relation we assume that Charge_P may take Null values.

Rule04 Algorithm Steps:

2. Use charges relation
   If chargedprice not Null: goto 9.
3. Open metadata relation.
4. Pkvalue=Chargecode
   Relation= 'charges'
   Column = 'chargedprice'
   Status = 'Unknown' or 'Mapp' or 'NSup': goto 6.
5. If a query request for charge code exist then: goto 8 else goto 6.
6. Use metadata relation and search for pkvalue=chargecode if exist then: goto 8 else: goto 10.
7. View table without chargedprice
8. View table with chargedprice
9. End
2. Functional Dependency Approach

Rule03: For treatment of room’s status Null values, RULE03 is provided. (5), (19), (26), (23), (20), (18)

Rule03 Algorithm Steps:
1. Open para.mdb
2. Use rooms relation
3. If rooms.status not=Null where=roomno
4. Use bookings
5. If wroom exist: goto 7 else: goto 6
6. Display status= "Unknown" then: goto 11
7. If checkin<>current date and checkout=blank then: goto 7.a else: goto 7.b

7.a display status= "Occupied" then:
goto 10
7.b : goto 8
8. Use maint
9. If wroom exist then: goto 9.a else: goto 9.b
9.a display status= "Available" then:
goto 10
9.b : goto 10
10. If from Date not empty and empty
    Today then: goto 10.a else goto 10.b
10.a display status= "In Maintenance" then: goto 11
10.b display status= "Available" then:
goto 11
11. End

Rule03: Functional Dependency Flowchart for Null Value Treatment
Rule02: After implementing RULE01 if there is no such exact price value extracted from a similar hotel “Dijlah”, there is a need to extract a near value depending on roomtype, checkin, checkout attributes.

Rule02 Algorithm Steps
1. Use paradise roomprice relation
2. Wfrom=roomprice. From_data, uto=roomprice. To_data
3. Use paradise bookings relation
4. Check for wfrom=checkin and uto=checkout
   a. if exist then: goto 5
   b. else: goto 9

5. Roomtype=Reservation ID then goto 6
6. Use Bill relation then: go to 7
7. Check for ReservationID=writeserv
   a. if exist then: goto 7
   b. else display “room price not found” then: go to 9
8. Appcost=cost/unto-wfrom then goto 12
9. Open SUN, mark through ODBC
10. Use roomprice relation
11. Find for wfrom-roomprice. from_data, uto-roomprice.to_data
12. Appcost=roomprice.price
13. Display a message “price is calculated assumable value”
14. End.

Rule02: A Business Rule Flowchart for Null Value Treatment
START RUL01

select a hotel code

Request room price

Is Null

No

Reconnect ODBC

Is connection Done

Yes

search for room type

If Exist

Yes

appeast = price
price = appeast
display room price

"The Rule is done perfectly"

END
في بحثنا هذا حاولنا تقديم حلول شاملة
لخدمة المستخدمين الشابيين، فقد تم تصميم نظام
قواعد معلومات متوزعة ليصبح الأرضية اللازمة
لتقديم وتطبيق الحلول المقدمة، أن النظام هو
((إدارة المقصور)) شركة مفترضة بمثابة أربعة
فنادق في العراق متوزعة في مدن وأماكن جغرافية
مختلفة، وتم تصميم النظام وبرمجته باستخدام لغة
( Access/Visual Basic)

للثلاثة فنادق أسا الفنادق الرابع فاستخدم
كما واستخدمت مجموعة (Visual FoxPro)
الدوار المرغوب لتأمين الربط بين مختلف
قواعد المعلومات. تم تقديم خمسة حلول مختلفة
للاعجال حسب النية اللاغية كمحاولة لتقديم
حلول عامة يمكن تطبيقها عمليا في بيئة قواعد
العلومات المتوزعة.

Hotel Database Tables and Relationship

Proposed Approaches for Null Values Treatment

For approaches are proposed for treatment of Null values in distributed
database applications, the summaries written below give practical use and
another known approach is implemented if for nulls treatment. (1),
(4), (13), (4), (5), (19), (26), (23), (20)


Rule01: For a RoomPrice relation
when price is Null for example we can
extract the exact value from the rule as
shown below: [5], (19), (26), (23), (20),
(12)

Rule01 Algorithm Steps
1. Open a hotel database
2. Use roomprice relation
3. Check for roomprice. price if exit then:
goto 10 else utype=roomprice. roomtype
4. Open db and db through ODBC
5. If connection failure: goto 1 else
6. Use roomprice relation then
7. Search for utype, if exist then: goto 8,
else, perform Rule02
8. Approve roomprice, price
9. Display a message "$RULE01 is done
perfectly" then: goto 10
10. End.
Null Values Treatment in Distributed Databases

By
Saran Akram Abd Al-Majeed
Information Technologies Center

Abstract

There has been a great deal of discussion about null values in relational databases. The relational model was defined in 1969, and Nulls was added in 1979. Unfortunately, there is not a generally agreeable solution for null values problem.

Null is a special marker which stands for a value undefined or unknown, which means that no entry has been made, a missing value mark is not a value and not of a data type and cannot be treated as a value by Database Management System (DBMS). As we know, distributed database users are more than a single database and data will be distributed among several data sources or sites, it must be precise data, the replication is allowed there, so complex problems will appear, then there will be need for perfect practical general approaches for treatment of Nulls.

A distributed database system is designed, that is “Hotel reservation control” system, based on different data sources at four sites, each site is represented as a Hotel, for more heterogeneity different application programming languages there are five practical approaches, designed with their rules and algorithms for Null values treatment, through the distributed database sites. [1], [2], [3], [4], [5], [9]