REMOVING RESTRICTIONS IN THE RELATIONAL DATA BASE MODEL:
AN APPLICATION OF PROBLEM SOLVING TECHNIQUES

Laurent Siklóssy
Dept. of Information Eng.
University of Illinois at Chicago
Chicago, IL 60680 (USA)

Jean-Louis Laurière
Institut de Programmation
Université Paris VI
4, Place Jusieu
75250 Paris Cedex 05 (FRANCE)

ABSTRACT
The principal restrictions previously placed on the relational data base model have been removed in the L2 model presented here.

We extend the model to include null (i.e. unknown and non-relevant) values (even in keys), repetitions of tuples, functional dependencies, a very rich set of constraints and information originating from several sources.

The programmed problem-solver ALICE is utilized to manipulate the constraints and to simplify relations: to answer a query, ALICE selects both the tuples which might answer the query upon appropriate substitutions for unknown values.

1. INTRODUCTION
We assume that the reader is familiar with the relational data base model [Codd, 1975; Date, 1981; Ullman, 1980]. In practice, some values may not be known in a data base; they will be called unknown values.

In other cases, to avoid a multiplicity of similar data bases, non-relevant values are introduced. For example, the value of the attribute "Name of Spouse" is non-relevant for a person who is not married. Unless we permit such non-relevant values, we would be obliged to divide a data base on persons into two similar data bases: one for married persons; a second one for unmarried persons. Unknown and non-relevant values are combined in the term null values.

Data bases with null values have been studied by a variety of authors [Codd, 1979; Grant, 1977; Imielinski et al, 1981; Lipski, 1979, 1981; Siklóssy, 1981; Vassiliou, 1979, 1980], among others. Generally these authors have placed a variety of restrictions on the use of null values. Some authors do not permit non-relevant values. Others do not permit null values in a key.

We shall show that it is possible, with the help of the problem-solving program ALICE [Laurière, 1976] to lift the restrictions previously advocated. In addition, our extended model will admit functional dependencies and a rich set of constraints, since ALICE can manipulate these dependencies and constraints.

If queried, ALICE can indeed return both the tuples which satisfy the query, and also those tuples which might satisfy the query upon appropriate specifications of unknown values.

We shall first discuss the restrictions pre-

viously imposed upon null values, and will argue that their justification is tenuous at best. Then, through a set of examples, we shall describe some extensions of data bases with null values that ALICE can process.

2. ARE RESTRICTIONS ON NULL VALUES NECESSARY?
Two principal restrictions on null values have been proposed: no null values are permitted in a key, and there should be no duplication of the same tuple. The first restriction is forcefully stated, for example by Codd (1979, p. 400):

Rule 1 (Entity Integrity): No primary key value of a base relation is allowed to be null or to have a null component.

Insisting upon a fully defined key for each tuple automatically prohibits the existence of two (or more) tuples having identical keys. Such tuples can be merged (or recognized as inconsistent).

The above restrictions appear motivated by access necessities: if a key (or part of a key) is unknown, how can one access tuples? In practice, though, such restrictions prohibit the representation of much information that occurs frequently, and leads to a model of restricted practical use. Indeed, let us consider a data base of individuals, with a variety of properties for that individual and let us assume that a person's name is the key to the individual (and her properties).

With the restrictions that we have just mentioned, it is not possible to represent the information provided by observer 1:
"I saw somebody (but I don't know her name) with only one arm, wearing green trousers", since the key (the person's name) is unknown, and hence null. Nor is there a way to represent the information provided by observer 2: "I saw the same person as observer 1, and I noticed that she had hazel eyes. But I don't know her name either".

If we assume that a data base is grown piece-meal, as a result of the contribution of a variety of observers (or informants), then we must remove the above restrictions. A tuple can be viewed as the result of an observation by an observer. Observations are then often incomplete. The question becomes: Can one still compute? That is, is it possible to answer queries from such unrestricted data bases? We shall now show, through a series of examples, that indeed we can still compute.

3. THE L² MODEL: AN EXTENDED RELATIONAL DATA BASE MODEL
We shall now describe our extended relational model, which will be referred to as the L^2 model for short. Although the model can be described formally and precisely, we shall attempt a more intuitive and explanatory description here.

A relation in the L^2 model consists of two-dimensional tables. The columns are labelled as attributes; the rows will also be referred to as tuples. We can think initially of each row as representing a set of partial observations by an informer on some individual (or entity). The system ALICE will try to recognize that several rows refer to the same individual.

The following values of attributes in a tuple are presently accepted in the L^2 model, and can be processed by ALICE:

1°/-Scalar values, such as "blue", "69", "potato".
2°/-Ranges of scalar values, such as blue or green or red, meaning the value is blue, green or red; or such as (22..35], meaning the value is between the values of 22 and 35, with ends included.
3°/-Non-relevant values, denoted NR. An attribute may have several non-relevant values. For example, the attribute "Name of Spouse" may have the value "NRl" to mean "never married", or NR2" meaning "Spouse is Dead". ALICE must be given the specific semantics of non-relevant values by means of formal definitions.
4°/-Unknown values. The attribute for a tuple has a value which is unknown in the observation represented by the tuple. Unknown values are represented by: ?. If two unknown values are the same, as for example in "I saw the same person as you did", then the unknown is indexed, for example as: ?5.
5°/-Functional dependencies, from one set of attributes to another set of attributes in a relation.
6°/-Keys, a special case of functional dependencies.
7°/-A variety of constraints.

ALICE was designed to manipulate constraints, and has proved very successful. Examples of constraints on the relation R, with attributes A and B, could be:

"For each value of A, there may be at most two values of B". (Application: a society where a person may have at most two spouses!)

"The values of A must be at least twice larger than the values of B".

Since ALICE has been described extensively elsewhere, we shall not describe it here. The examples will illustrate her capabilities. (The original ALICE required no modifications for the present applications).

4. RELATED WORK

Few works in the area of extensions of relational database systems (to null values or ranges, for example) include running programs. An exception is Lipski (1979). Lipski describes a theory and a program to manipulate data bases where values are within a range. We note that the L^2 model includes such ranges and, in addition, null values, functional dependencies and a rich set of constraints.

ALICE has been run on the two problems mentioned by Lipski, which we shall repeat here:

Problem 1: The Database is:

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>AGE</th>
<th>DEPT #</th>
<th>HIREDAYEAR</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,40)</td>
<td>(1,2,3,4,5)</td>
<td>70..80</td>
<td>(0,40)</td>
<td></td>
</tr>
<tr>
<td>x₁</td>
<td>60..70</td>
<td>(1,2,3,4,5)</td>
<td>70..75</td>
<td>(10000)</td>
</tr>
<tr>
<td>x₂</td>
<td>52..56</td>
<td>(2)</td>
<td>72..76</td>
<td>(0,20000)</td>
</tr>
<tr>
<td>x₃</td>
<td>30</td>
<td>(3)</td>
<td>70..71</td>
<td>(0,40)</td>
</tr>
<tr>
<td>x₄</td>
<td>(0,40)</td>
<td>(2,3)</td>
<td>70..74</td>
<td>(22000)</td>
</tr>
<tr>
<td>x₅</td>
<td>32</td>
<td>(4)</td>
<td>75</td>
<td>(0,40)</td>
</tr>
</tbody>
</table>

The query is:

[Dept # in (2,3)]∧
[(Salary < 10000)∧ HIREDAYEAR [22]∧
[(Age > 50)∧(Salary < 150000)].

ALICE determines (in 1.5 seconds on a 370/168 computer) that x₃ and x₄ satisfy the query; that x₂ might satisfy the query; and that x₁ and x₅ do not satisfy the query. These results agree with Lipski's.

Problem 2: The Database is:

<table>
<thead>
<tr>
<th>NAME</th>
<th>SALARY</th>
<th>STATUS</th>
<th># CHILDREN</th>
<th>DEPT #</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>(15000,18000)</td>
<td>[Married]</td>
<td>{3,4,5}</td>
<td>{2,3,4}</td>
</tr>
<tr>
<td>Brown</td>
<td>(18000)</td>
<td>[Married]</td>
<td>{2}</td>
<td>{2}</td>
</tr>
</tbody>
</table>

The query is:

[(Dept # = 3)∧(Name = Jones)∧
[(Salary < 15000)∧(# Children > 2)∧
[(Salary in (10000, 20000))∧(Status = Married)].

ALICE determines (in 1 second) that "?" might answer the query, while Brown does not. Again, these results agree with Lipski's. See the exact formulation of these problems in the ALICE language in the annex.

5. EXAMPLES OF MANIPULATIONS IN THE L^2 MODEL AND SOLUTIONS BY ALICE

We shall briefly describe a problem illustrating the representational capabilities of the L^2 model and the database reduction capabilities of ALICE*. When given a data base, ALICE first reduces the

* The example is somewhat artificial, so that it remains simple to describe.
database by eliminating redundant tuples, and finding values or ranges for unknowns.

The database is:

<table>
<thead>
<tr>
<th>Tuple</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(04)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>(04)</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>(06)</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The constraints are:

(1) Keys: A ≠ (B,C)
(II) F.D: C → D
(III) Not more than 2 C's for 1 B
(IV) At least two different tuples with C = cl
(V) 0 < A < 10;
(VI) A integer.
(VII) A ≥ 2D.

Notice null values in the key A (in tuples 4, 6, 8 and 10) and again in the B component of the key (B,C) (in tuples 1 to 5 and 8).

Indexed unknowns are used to indicate equality of unknowns, as in 6.A = 7.A, or explicitly to indicate inequality. The constraint on line 3: "?5 ≠ ?1" means that: 3.B is equal to ?5 but must be different from ?1.

Solution by ALICE (4 seconds):

By constraint (I): 3.A = 6. Tuples 1 and 2 give indeed that 1.A and 2.A form together the set (O,4). Since A is a key, if 3.A = 0 then tuple 3 must be either tuple 1 or tuple 2. But both cases are impossible due to: ?5 ≠ ?1 and c3 ≠ c4. .74 = ?10 = (01) by constraints (II) and (VII). .72 = ?7 by (II). .71 = 10 by (V) and (VII); ?6 < 3 by (VII).


The reduced database becomes:

<table>
<thead>
<tr>
<th>Tuple</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(04)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>(04)</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>(06)</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The following questions are answered by ALICE in a total of 3 seconds:

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2: Tuples with B=b1? A2: Surely: 7, 9; Maybe: none.</td>
<td></td>
</tr>
<tr>
<td>Q3: Tuples with B=b2? A3: Surely: none; Maybe: 1, 2, 3, 4, 5, 8.</td>
<td></td>
</tr>
<tr>
<td>Q5: Tuples with D=0? A5: Surely: 7; Maybe: 1, 2, 3, 5, 8.</td>
<td></td>
</tr>
<tr>
<td>Q6: Tuples with A=0? A6: Surely: none; Maybe: 1, 2.</td>
<td></td>
</tr>
</tbody>
</table>

(Actually A6 is more precise; Surely: either 1 or 2 but not both. Tuple 8 may not answer Query 6 by one of the last conclusions of the previous reduction process).

The reader can verify, by solving the problem, that some of the deductions, if not very difficult, are not trivial either.

NOTE: For each problem, ALICE receives the formal definitions of the keys, constraints and queries. She does not know the application a priori but interprets these formal data on the data base. An optimised "frozen" version of the system for this kind of application would probably reduce the running times significantly.

6. CONCLUSIONS

The L² data base model is a natural extension of the relational data base model. It allows null values in any field, including keys; the repetition of tuples, and therefore information from different and possibly coupled sources; and a great variety of constraints. Therefore, we can now model situations that were forbidden in previously proposed extensions.

The problem-solver ALICE reduces a given data base, finds contradictions in it, and answers queries. The performance of ALICE in other domains indicates that constraints that are significantly more powerful than those previously allowed, or even those shown in this article, can be successfully accepted in a relational database.

We are pursuing our study of the L² model, in particular in the areas of non-relevant nulls (which were not given in our examples here); secondary storage handling; and the incorporation of significantly more challenging constraints.

Acknowledgments

Some of this work was pursued while one of the authors (LS) was Visiting Professor at the University of Paris IX Dauphine. The support of Paris IX is gratefully acknowledged.

ANNEX

The statements in ALICE of two of the problems, together with the run of one, follow:

Problem 1: Give the constraints:

| range of age: ia = 0; | sa = 100. |
| range of department: id = 0; | sd = 5. |
| range of hireyear: ih = 70; | sh = 80. |
| range of salary: is = 0; | ss = 90. |
Given the sets: $X = [l, l]$
$DA = [ia, sa]$
$DD = [id, sd]$
$DH = [ih, sh]$
$DS = [is, ss]$

Find the functions:
$AGE$ from $X$ to $DA$
$DEP$ from $X$ to $DD$
$HYE$ from $X$ to $DH$
$SAL$ from $X$ to $DS$

With the constraint (question):
$\text{Dep}(X) > 2$ and $\text{Dep}(X) \leq 3$ and
$\text{Sal}(X) > 10$ or $\text{Hye}(X) \leq 12$ and
$\text{Age}(X) < 50$ or $\text{Sal}(X) > 15$

All these formulas are written, for the system ALICE, in polish notation. The full problem specifications follow:

The other problems are given in the same manner: objects and relations are formally defined; key or (FD) correspond to partial or total onto functions.

Problem 2 gives for instance:

Comments: the set $DN$ of the possible names is read as data: \{Brown, Jones, Smith\}, the set $DM$ is \{married\}. The form of the constraints is similar.

The run is:
call .II(m)

REFERENCES