Database Relations with Null Values

(Extended Abstract)

Carlo Zaniolo

Bell Laboratories
Holmdel, New Jersey 07733
U.S.A.

1. INTRODUCTION

The problem of providing a formal treatment for incomplete database information and null values represents one of the thorniest research issues in database theory. A particular source of difficulties has been the generalization of the relational model to include null values. Here, we introduce a novel approach which extends key features of the relational model:

1. its set-theoretic foundations — which are preserved through a lattice-based generalization of the relational algebra — , and
2. efficient query-evaluation algorithms based upon the well-known correspondence between the relational calculus and the relational algebra.

Moreover, this generalization completes the data type "relation" with respect to the relational algebra — much in the same way that rational numbers complete the number system with respect to the four arithmetic operations.

2. SOLUTION APPROACH

Previous approaches have treated various interpretations of null values, including the *unknown* interpretation [2,4,8] (a value exists but is not known), the *nonexistent* interpretation [7], and these two combined [10]. Other authors have considered more informative interpretations to model disjunctive information [6] and probability distributions [11].

A main problem with the "unknown" interpretation is the computational complexity that it requires to answer certain queries [5,8,13]. For this interpretation, however, Codd has proposed an approach based upon three-valued logic, which is computationally simple and yields a straightforward generalization of the relational algebra [2]. Unfortunately, this approach fails to produce the correct answer for certain queries — e.g., those involving tautologies [4,5]; moreover it does not preserve any of the set-theoretic properties of relations since expressions such as $R=R$ and $R \cup S \supseteq R$ evaluate to MAYBE, rather than TRUE, when the relations R and S contain nulls.

In contradistinction to the previous approaches we focus on an interpretation of nulls which is less informative than every other interpretation. This is the *no information* interpretation which is demonstrated by the following example.

Say that a database contains a relation EMP with columns (attributes), E#, NAME, SEX, and MGR# (the E# of the employee's manager). Say that the current content of the relation is:

EMP	(E#,	NAME,	SEX,	MGR#)
	1120	SMITH	M	2235
	4335	BROWN	F	2235
	8799	GREEN	M	1255

Table 2.1. The employee relation.

Say now that the database administrator (anticipating future needs of the enterprise) decides to change the schema to include a new column TEL#, to contain the home number of each employee, to be entered in the

database when this information become available. Therefore, the database administrator is faced with the problem of having to operate, at least for the immediate future, with an expanded schema, while no change in the information content of the database has occurred. The obvious solution is to view the database as follows:

EMP	(E# ,	NAME,	SEX,	MGR#,	TEL#)
	1120	SMITH	M	2235	—
	4335	BROWN	F	2235	—
	8799	GREEN	M	1255	—

Table 2.2. The employee relation after the addition of the new attribute TEL#.

Thus, the TEL# entries in the rows of our relations have been filled with the symbol "—", which, from now on, we use to denote a null value. Table 2.2 demonstrates a very plausible and useful usage of null values. Clearly, neither the "unknown" nor the "nonexistent" interpretation is applicable in this situation. Here the symbol "—" neither denotes that a telephone number of a given employee does not exist, nor that the telephone number exists but is not known. Here the null value simply denotes that *no information* whatsoever exists on the TEL# of an employee. Thus our null value can be regarded as a place holder for either a nonexistent or an unknown value.

On the contrary, if "—" were interpreted as either "unknown" or "nonexistent", then the relation of Table 2.2, would contain more information than that of Table 2.1, and this would contradict the assumption that no additional data was gathered and stored when the schema was modified. Under the "no information" interpretation of nulls, it is instead correct to say that the relations in these two tables are *information-wise equivalent*. The notion of information-wise equivalence is central in our approach and will be further discussed and formally defined later.

Our objective is to define an extension of the relational data model to include "no information" null values. This approach will allow us to extend the relational algebra and the set-theoretic properties of relations, and to use Codd's simple three-valued logic for query execution.

3. *EXTENDED RELATIONS*

From now on, we will refer to relations with null values simply as relations. To denote relations without nulls we explicitly say "fully defined relations" or "total relations".

A relation R, defined over a set of attributes $W = \{A_1,...,A_n\}$, is denoted R(W). Underlying each variable attribute $A_i \in W$, there is a domain denoted $DOM(A_i)$. We *extend* each domain to include the distinguished symbol ***ni*** which denotes the null value under the "no information" interpretation. For $A \in W$, an A-value is an assignment from the *extended* A-*domain*. Generalizing this notion, an X-value, where $X \subseteq W$, is an assignment of values to the attributes in X from their respective extended domains. A relation R(W) is a set of W-values. The elements of this set are called rows or tuples of R.

A relation can be represented as a table, where the rows represent the tuples of the relation and the columns correspond to the attributes of the relation. In our tables we represent ***ni*** by the dash "—" (see Table 2.2). Say that r is a W-value, e.g., some tuple of R(W). Also, let $A \in W$ and $X \subseteq W$. Then r[A] and r[X] respectively denote the A-value and the X-value of r. We will assume, without loss of generality, that all the attributes of our relations are contained in a finite universe of attributes, U. We use the first letters of the alphabet, such as A, B, and C, to denote single attributes in U, and the last letters, such as W, X, Y, and Z, to denote subsets of U.

The notion of *more informative tuple* [12] supplies the cornerstone of our approach.

Def. 3.1. An X-value r is said to be *more informative* than a Y-value t, when for each $B \in Y$, if t[B] is not ***ni*** then $B \in X$ and $r[B] = t[B]$.

Thus, r must match t in each nonnull value of t. We write $r \geq t$ to denote that r is more informative than t. Conversely, if $r \geq t$ we say that t is less informative than r and write $t \leq r$. If $r \geq t$ and $t \geq r$, then we say that r and t are (information-wise) equivalent and write $r \cong t$.[1] For instance, say that

$$r_1 = (5555, JONES, -, 2231)$$

$$r_2 = (5555, JONES, F, 2231)$$

denote values of {E#, NAME, SEX, MGR#}, and

1. The relationship "$\geq$" is reflexive and transitive. Thus, "$\cong$" is an equivalence relation since it is also symmetric.

$r_3 = (5555, JONES, F, 2231, -)$

$r_4 = (5555, JONES, F, 2231, 2639452)$

denote values of {E#, NAME, SEX, MGR#, TEL#}. Then, $r_1 \leq r_2$, $r_2 \cong r_3$ and $r_3 \leq r_4$. (Also, note that each tuple in Table 2.1 is equivalent to the corresponding enlarged tuple in Table 2.2.) Let $X \subseteq Y \subseteq U$. Given an X-value r, an equivalent Y-value $t \cong r$ can be constructed from r by filling the (Y-X)-values with nulls. Therefore, we will prescribe by convention that, if r is an X-value and the attribute A is not in X, then $r[A] \cong ni$. Therefore, any two tuples consisting only of null values are equivalent and any such tuple is equivalent to the tuple consisting of *ni*. These tuples will be called null tuples. A tuple without nulls will be called *total* and a tuple with a total X-value will be called *X-total*.

Two tuples r_1, r_2 will be called *joinable* when the following is true for each $A \in U$:

If $r_1[A] \neq r_2[A]$, then either $r_1[A] = ni$ or $r_2[A] = ni$.

A tuple t will be called a *join* of two tuples r_1 and r_2, denoted $t \cong r_1 \vee r_2$, when r_1 and r_2 are joinable and for each $A \in U$,

$$t[A] = \begin{cases} r_1[A] & \text{if } r_1[A] \geq r_2[A] \\ r_2[A] & \text{if } r_2[A] \geq r_1[A]. \end{cases}$$

The notion of being more informative can be extended to relations, which will be said to be *more informative than* or to *subsume* other relations.

Def. 3.2. A relation R_1 subsumes a relation R_2, written $R_1 \supseteq R_2$, when for each tuple $r_2 \in R_2$ there is a tuple $r_1 \in R_1$ with $r_1 \geq r_2$.

This "$\supseteq$" relationship is transitive and reflexive. We can now define the notion of information-wise equivalence as follows:

Def. 3.3. The relations R_1 and R_2 are information-wise equivalent, written $R_1 \cong R_2$, when $R_1 \supseteq R_2$ and $R_2 \supseteq R_1$.

The equivalence relation "$\cong$" (reflexive, symmetric, and transitive) partitions the universe of relations into disjoint subclasses. We can thus use the basic generalization mechanism used to extend natural numbers to real numbers, to introduce the notion of *extended relations* (written x-relations for short).

Def. 3.4. An x-relation is an equivalence class under $\cong$. The class of relations equivalent to R is denoted $\hat{R}$. R is called a *representation* of $\hat{R}$.

Thus $\hat{R}1 = \hat{R}2$ iff $R_1 \cong R_2$. We can now define the notion of set inclusion or set containment for x-relations.

Def. 3.5. $\hat{R}1$ contains $\hat{R}2$, written $\hat{R}1 \supseteq \hat{R}2$, when R_1 subsumes R_2.

Clearly if $R_1' \cong R_1$, $R_2' \cong R_2$ and $\hat{R}_1 \supseteq \hat{R}_2$ then $\hat{R}_1' \supseteq \hat{R}_2'$, as expected. Moreover,

Proposition 3.1. $\hat{R}_1 = \hat{R}_2$ iff $\hat{R}_1 \supseteq \hat{R}_2$ and $\hat{R}_2 \supseteq \hat{R}_1$.

We will also say that R_1 properly contains R_2, and write $\hat{R}_1 \supset \hat{R}_2$, when $\hat{R}_1 \supseteq \hat{R}_2$ but $\hat{R}_1 \neq \hat{R}_2$. The converse of $\supseteq$ and $\supset$ will be denoted by $\subseteq$ and $\subset$, as usual.

It is also convenient to generalize the notion of a tuple being an element, or a member, of an x-relation as follows:

Def. 3.6. A tuple t is said to x-belong to, or to be an x-element of, $\hat{R}$, written $t \tilde{\in} \hat{R}$, when, for some R' in $\hat{R}$, $t \in R'$.

The following proposition supplies a simpler characterization of $\tilde{\in}$.

Proposition 3.2. $t \tilde{\in} \hat{R}$ iff there exists a tuple $r \in R$ s.t. $r \geq t$.

Thus a tuple t belongs to an x-relation iff its representation contains a tuple which is more informative than t. Also we will say that a tuple t x-belongs to a relation R, and write $t \tilde{\in} R$, to denote that for some $r \in R$, $r \geq t$. Although "$\tilde{\in}$" is now used in two different contexts no confusion arises, since $t \tilde{\in} \hat{R}$ iff $t \tilde{\in} R$. We also write $t \not\tilde{\in} R$ or $t \not\tilde{\in} \hat{R}$ to denote that $\neg(t \tilde{\in} R)$ or $\neg(t \tilde{\in} \hat{R})$ holds.

Given a set of tuples $\{t_1, t_2, ..., t_n\}$, one can eliminate all tuples that are less informative than some other tuples, and enlarge the others to their equivalent U-values. The x-relation represented by the set of U-values so obtained will be denoted:

$$\{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_n\}.$$

We can now define union, x-intersection, and difference using this handy notation. (The reason for the term x-intersection will become clear later.) We have:

union: $$\hat{R}_1 \cup \hat{R}_2 = \{\hat{r} \mid r \tilde{\in} \hat{R}_1 \text{ or } r \tilde{\in} \hat{R}_2\} \quad (3.1)$$

x-intersection: $$\hat{R}_1 \cap \hat{R}_2 = \{\hat{r} \mid r \tilde{\in} \hat{R}_1 \text{ and } r \tilde{\in} \hat{R}_2\} \quad (3.2)$$

difference: $$\hat{R}_1 - \hat{R}_2 = \{\hat{r} \mid r \tilde{\in} \hat{R}_1 \text{ and } r \tilde{\notin} R_2\} . \quad (3.3)$$

All these operations have the substitution property with respect to equality, as expected.

The union and the x-intersection respectively define the least upper bound and the greatest lower bound with respect to the partial ordering $\tilde{\supseteq}$. In fact:

Proposition 3.3. If $\hat{R} \tilde{\supseteq} \hat{R}_1$ and $\hat{R} \tilde{\supseteq} \hat{R}_2$ then $\hat{R} \tilde{\supseteq} \hat{R}_1 \cup \hat{R}_2$.

Proposition 3.4. If $\hat{R} \tilde{\subseteq} \hat{R}_1$ and $\hat{R} \tilde{\subseteq} \hat{R}_2$ then $\hat{R} \tilde{\subseteq} \hat{R}_1 \cap \hat{R}_2$.

Thus we have a lattice of x-relations, with the well-known properties associated with it [1]. This lattice is also *distributive* [13].

Our lattice has a bottom element, denoted $\hat{\emptyset}$, which is characterized by the property that for every x-relation $\hat{R}$, $\hat{R} \cap \hat{\emptyset} = \hat{\emptyset}$. $\hat{\emptyset}$ can be represented by an empty relation. The top of our lattice, denoted $\hat{TOP}_U$, is characterized by the property that $\hat{R} \cup \hat{TOP}_U = \hat{TOP}_U$, for all $\hat{R}$. If $U = \{A_1,\ldots,A_p\}$ then $\hat{TOP}_U$ can be represented by

$$TOP_U = DOM(A_1) \times \cdots \times DOM(A_p) .$$

In general, an x-relation $\hat{R}$ does not have a complement (i.e., there is no relation $\hat{R}'$ for which $\hat{R} \cap \hat{R}' = \hat{\emptyset}$ and $\hat{R} \cup \hat{R}' = \hat{TOP}_U$). This can be seen from the following example:

$$U = \{A,B\}, \quad DOM(A) = \{a_1\}, \quad DOM(B) = \{b_1,b_2\} .$$

Thus the following two tuples are x-elements of $\hat{TOP}_U$:

$$r_1: (a_1,b_1)$$
$$r_2: (a_1,b_2) .$$

Now take a relation R x-containing r_1 but not r_2. Then an x-relation $\hat{R}'$, to yield $\hat{R} \cup \hat{R}' = \hat{TOP}_U$, must have r_2 as an x-element: $r_2 \tilde{\in} \hat{R}'$. But then the tuple (a1, —) x-belongs to both $\hat{R}$ and $\hat{R}'$. Therefore it also belongs to $\hat{R} \cap \hat{R}' \neq \hat{\emptyset}$.

Likewise, the intersection $(\hat{R}_1 - \hat{R}_2)_2$ may not be empty. However, the following two properties hold:

Proposition 3.5. For any two x-relations $\hat{R}_1$ and $\hat{R}_2$, $\hat{R}_1 \tilde{\supseteq} \hat{R}_2$: $(\hat{R}_1 - \hat{R}_2) \cup \hat{R}_2 = \hat{R}_1$.

Proposition 3.6. If $\hat{R} \cup \hat{R}_2 = \hat{R}_1$ then $\hat{R} \tilde{\supseteq} (\hat{R}_1 - \hat{R}_2)$.

Thus $(\hat{R}_1 - \hat{R}_2)$ is the smallest x-relation (in terms of $\subseteq$, of course) whose union with $\hat{R}_2$ will give $\hat{R}_1$. Therefore, x-relations define a distributive, pseudo-complemented lattice [1], where the pseudo-complement of $\hat{R}$, denoted $\hat{R}^*$, is defined (U being the universe of discourse) as

$$\hat{R}^* = \hat{TOP}_U - \hat{R} .$$

Thus, $\hat{R}^*$ is the smallest x-relation which when unioned with $\hat{R}$ gives $\hat{TOP}_U$. Pseudo-complemented, distributive lattices are also known as *Brouwerian* lattices after Brouwer and Heyting (1930), who characterized an important generalization of Boolean algebra having very similar properties [1].[2] Brouwerian lattices have many interesting properties [1,3]. In particular it is known that the pseudo-complements of such a lattice form a Boolean lattice. In our case the set $\{\hat{R}^*\}$ is simply the family of total x-relations with scope U, the universe. It is also known that every Brouwerian lattice (our x-relations) and the Boolean lattice of its pseudo-complements share the join — i.e., the union — and the (pseudo)-complement operation. However, the proof that the two may have two different meet operations was published in its full generality only in 1962 [3]. Now, x-relations supply a most interesting example of such a difference: The

2. More precisely x-relations form the dual of a Brouwerian lattice, where the pseudo-complement of an element a is usually defined as the largest element a* for which $a \wedge a^* =$ bottom.

meet for the complements (U-total x-relations) is the usual set intersection while the meet for x-relations is the x-intersection (3.2). Obviously these two are different, as illustrated by the simple case of the following two x-relations on the universe U = {A, B}:

$$\hat{R}_1 = \{\hat{}\ (a,b_1)\}, \quad \hat{R}_2 = \{\hat{}\ (a,b_2)\}\ .$$

Here the set intersection of R_1 and R_2 is empty while the x-intersection of $\hat{R}_1$ and $\hat{R}_2$ x-contains the tuple (a,—).

The notion of *minimal representation* is convenient for representing and handling x-relations.

Def. 3.7. A relation R constitutes a minimal representation for $\hat{R}$, when no proper subset of R is also a representation of $\hat{R}$.

A minimal representation can be constructed by starting with an arbitrary one and removing every tuple which is less informative than some other tuple. This process can be regarded as an extension of the one of removing duplicate tuples in tables representing conventional relations.

The minimal representation of an x-relation over a given attribute set is unique. As shown by examples in Tables 2.1 and 2.2, however, an x-relation can have two distinct minimal representations over two different sets of attributes. The minimal set of attributes on which an x-relation can be represented will be called its *scope*.

Two x-relations will be called *separate* if they have disjoint scopes.

Definitions (3.1 - 3.3), in their present form, are not conducive to efficient implementation. In fact, the definition of $\tilde{\epsilon}$ suggests a combinatorial explosion in which a plethora of less informative tuples are tested and possibly included in the result relation. This problem can be solved by deriving equivalent formulations which do not use $\tilde{\epsilon}$, as discussed in [13].

4. *QUERY EVALUATION and RELATIONAL ALGEBRA*

The solution here proposed is similar to Codd's TRUE-evaluation of queries. However, it uses a different interpretation of three-valued logic and a new treatment of sets.

Predicate calculus based languages contain simple relational expressions such as

t.A θ m.B

t.A θ k

where t and m are tuple variables, A and B are attributes, k is a (nonnull) constant, and θ is one of the comparison operators, $>, <, =, \geq, \leq, \neq$. If the A-value of t is null then these two relational expressions evaluate to ***ni***. Also if the m.B value is null then "t.A θ m.B" evaluates to ***ni***. Otherwise these expressions evaluate to TRUE or FALSE as usual. Boolean expressions combining relational expressions like the above are evaluated according to Table 4.1.

OR	T	F	*ni*
T	T	T	T
F	T	F	*ni*
ni	T	*ni*	*ni*

AND	T	F	*ni*
T	T	F	*ni*
F	F	F	F
ni	*ni*	F	*ni*

	NOT
T	F
F	T
ni	*ni*

Table 4.1 Three-Valued Logic Tables

The answer to a query is computed by selecting those tuples which evaluate to TRUE and discarding those which evaluate to FALSE or ***ni***.

The three-valued logic and method of query evaluation described above are equivalent to Codd's TRUE-evaluation strategy. It has been shown that this strategy does not produce the correct answer for the "unknown" interpretation, for queries which correspond to tautologies [4,5]. Fortunately the ***ni*** interpretation avoids this problem. Take for instance a query containing the following expression (say that e is a tuple variable ranging over the relation EMP):

$$(e.SEX = \text{"F"} \wedge e.TEL\# > 2634000) \vee e.TEL\# \leq 2634000.$$

Now, consider the second tuple in Table 2.2:

(4335, BROWN, F, 2235, —).

If the null value "—" is interpreted as the place holder of an existing although unknown TEL#, then it is clear that whatever number we substitute for "—" the above expression evaluates to TRUE. Thus this tuple contributes to the output. Under the ***ni*** interpretation, however, the null value fills in for both unknown and nonexistent values. Now, an object which does not exist has no property and does not satisfy any relational expression [7,10]. Therefore a TEL# which does not exist is neither greater than 2634000, nor smaller than

nor equal to it. Thus, the above condition evaluates to *ni* for tuples with a null TEL# which, therefore, do not contribute to the TRUE-evaluation of our query.

We can now define the operations of selection and join in conformity with the query interpretation discipline just discussed. The *selection operation* is defined as follows:

$$\hat{R}[A\theta B] \quad \text{and} \quad \hat{R}[A\theta k],$$

where A and B are two attributes in U from the same underlying domain, k is a constant from DOM(A) — not the *ni* symbol — and θ denotes a relational operator such as $=$, $>$, etc. The definitions of these two operations for x-relations are:

$$\hat{R}[A\theta B] = \{\hat{}\, r \mid r \in R \text{ is A-total and B-total and } r[A]\,\theta\, r[B]\}$$

$$\hat{R}[A\theta k] = \{\hat{}\, r \mid r \in R \text{ is A-total and } r[A]\,\theta\, k\}\ .$$

We will next define a version of the natural join operation, where the columns on which the join is performed are explicitly stated. The *join on X* of $\hat{R}_1$ and $\hat{R}_2$, denoted $\hat{R}_1 <\bullet X> \hat{R}_2$, is defined as follows:

$$\hat{R}_1 <\bullet X> \hat{R}_2 =$$

$$= \{\hat{}\, r \mid r \text{ is X-total and } \exists r_1 \in R_1, \exists r_2 \in R_2 \text{ s.t. } r = r_1 \vee r_2\}\ .$$

Thus the join on X is constructed by joining those tuples in R_1 and R_2 which have fully defined identical x-values. The *cartesian product* $\hat{R}_1 \times \hat{R}_2$ represents a special subcase of this join, obtained by letting $\hat{R}_1$ and $\hat{R}_2$ be separate and X be the empty set. As in the case of total relations, the various θ-joins can thus be defined as selections on the cartesian product:

$$\hat{R}_1[A\theta B]\hat{R}_2 = (\hat{R}_1 \times \hat{R}_2)[A\theta B]\ .$$

The projection of a relation $\hat{R}$ on a set of attributes X, denoted R[X], is defined as follows:

$$\hat{R}[X] = \{\hat{}\, r[X] \mid r \in R\}\ .$$

In the previous definitions, we have used the operator "ϵ" rather than "$\tilde{\epsilon}$". The inconsistency here is only apparent, since the replacement of "ϵ" by "$\tilde{\epsilon}$" in all the formulas above yields relations which are information-wise equivalent to the originals. The definitions of division and information-preserving joins, called outer joins in [2], are given in [13].

A complete relational algebra consists of the following operations [9]: union, difference, cartesian product, selection and projection. For traditional relations, these operations are defined only when the operands satisfy certain applicability conditions (e.g. to perform a union or a difference the relations must be union-compatible, and to perform a select the operands must contain all the attributes used in the selection expression). For x-relations, however, these operators are defined and applicable *independently of the attribute sets of the operands.* Thus x-relations have the closure property with respect to all operators of the complete relational algebra. Also, the proposed generalization is consistent since these operators reduce to their traditional counterparts for x-relations representable by total relations such that the traditional operators are applicable. Therefore, x-relations complete the data type "relation" with respect to the relational algebra, much in the same way that rational numbers complete the number system with respect to the four arithmetic operations.

Acknowledgments

The author would like to thank Bob Horgan, Ed Lien, and Eric Wolman for their comments and useful suggestions.

REFERENCES

[1] Birkoff, G., *Lattice Theory,* American Mathematical Society, Providence, R.I., 1967.

[2] Codd, E. F., "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. Database Systems,* Vol. 4, No. 4, 1979, pp. 397-434.

[3] Frink, O., "Pseudo-complements in Semilattices," *Duke Math. J.,* Vol. 29, 1962, pp. 505-514.

[4] Grant, J., "Null Values in a Relational Data Base," *Information Processing Letters,* Vol. 6, No. 5, 1977, pp. 156-157.

[5] Imielinski, T. and W. Lipski, "On Representing Incomplete Information in a Relational Database," *Proc. 7th Int. Conf. on Very Large Data Bases,* 1981, pp. 388-397.

[6] Lipski, W., "On Semantic Issues Connected with Incomplete Information Databases," *ACM Trans. Database Systems,* Vol. 4, No. 3, 1979, pp. 262-296.

[7] Lien, Y. E., "Multivalued Dependencies with Null Values in Relational Databases," *Proc. 5th Int. Conf. on Very Large Data Bases,* Rio de Janeiro, Brazil, 1979, pp. 155-168.

[8] Reiter, R. "Data Bases, a Logical Perspective", *Proc. Workshop on Data Abstraction Databases and Conceptual Modeling,* Pingree Park, Colorado, June 23-26, 1980, pp. 174-176.

[9] Ullman, J. D., *Principles of Database Systems,* Computer Science Press, Potomac, MD, 1980.

[10] Vassiliou, Y., "Null Values in Data Base Management: A Denotational Semantics Approach" *Proc. ACM SIGMOD Int. Conf. Management of Data,* Boston, Massachusetts, May 30 - June 1, 1979, pp. 162-169.

[11] Wong, E., "A Statistical Approach to Incomplete Information in Database Systems," Technical Report CCA-80-08, May 30, 1980, Computer Corporation of America (575 Technology Square, Cambridge, Massachusetts 02139).

[12] Zaniolo, C., "Relational Views in a Database System; Support for Queries," *Proc. IEEE Computer Applications and Software Conf.,* Chicago, Illinois, November 8-11, 1977, pp. 267-275.

[13] Zaniolo, C., "Database Relations with Null Values," 1981, submitted for publication.