# NAIVE TABLES

Gösta Grahne
Department of Computer Science
Concordia University, Montreal, Canada
grahne@cs.concordia.ca

**SYNONYMS**

Relations with marked nulls; Extended Relations

**DEFINITION**

The simplest way to incorporate unknown values into the relational model, is to allow *variables*, in addition to *constants*, as entries in the columns of relations. Such constructs are called *tables*, instead of *relations*. A table is an incomplete database, and *represents a set of complete databases*, each obtained by substituting all variables with constants. Different occurrences of the same variable (marked null) are substituted with the same constant. The substitution is thus a function from the variables and constants, to the constants, such that the function is identity on the constants. A table $T$ then represents the set of relations, denoted $rep(T)$, defined as $\{v(T) : v \text{ is a valuation } \}$. Then the *certain answer* (crossref) to a query $q$ on a table $T$, denoted $sure(q, T)$ is the set of tuples that occur in every answer obtained by applying the query to every database in $rep(T)$. In other words the certain answer to $q$ on $T$ is $sure(q, T) = \cap q(rep(T))$.

**MAIN TEXT**

To illustrate the above concepts, let tables $T_1$ and $T_2$ be as below, and let $q$ be the relational expression $\sigma_{A=a \vee A=c}(\pi_{AC}(R_1 \bowtie R_2))$. (The schema of $T_i$ is that of $R_i$, $i = 1, 2$.) Then applying $q$ to $T_1, T_2$, which is denoted $\sigma_{A=a \vee A=c}(\pi_{AC}(T_1 \bowtie T_2))$, yields table $q(T_1, T_2)$ below.

$T_1$

| A | B |
|---|---|
| $a$ | $X$ |
| $Y$ | $b$ |
| $c$ | $b$ |

$T_2$

| B | C |
|---|---|
| $X$ | $d$ |
| $b$ | $Z$ |

$q(T_1, T_2)$

| A | C |
|---|---|
| $a$ | $d$ |
| $c$ | $Z$ |

The variables/null-values are written in uppercase, to clearly distinguish them from the (lowercase) constants. Note however that $q(T_1, T_2)$ is not (necessarily) yet the certain answer. How was $q(T_1, T_2)$ derived from $q$ and $T_1, T_2$, and how is the certain answer $sure(q, (T_1, T_2))$ obtained from $q(T_1, T_2)$? The answer to the second question is very simple: just drop all tuples containing variables from $q(T_1, T_2)$. The remaining tuples form the certain answer. In the example the certain answer consists of tuple $(a, c)$ only. The answer to the first question is not much more complicated: evaluate $q$ on the tables, treating variables are "constants," pairwise distinct, and distinct from all "real" constants. This is also known as the Naive evaluation of $q$ on $T$ [2, 3]. In the example above, tuple $(a, X)$ joined with tuple $(X, c)$ since they have the same value, represented by $X$, in the join column. This is done even though the "actual" value of $X$ is not known, since in any valuation $v$ the two occurrences of $X$ are mapped to the same value. On the other hand, when performing the selection $\sigma_{A=a}$ the tuple $(Y, Z)$ is not picked, since there is a least one valuation $v$, for which both $v(Y) \neq a$ and $v(Y) \neq c$. A characterization of the correctness on the Naive evaluation is given below.

Before going to the characterization, note that is not always ideal to return the certain answer only. Namely, if the answer to $q$ is to be materialized as a view for further querying, essential information is lost if the tuples with variables are dropped. For a simple example, if evaluating $\pi_A$ on $sure(q, (T_1, T_2))$ gives tuple $(a)$ as sure answer, whereas evaluating $\pi_A(q(T_1, T_2))$, puts tuples $(a)$ and $(c)$ in the sure answer. As a consequence, query evaluation would not be compositional, unless $q(T_1, T_2)$ is stored as an "intermediate" answer. This "intermediate" answer is called the *exact answer* in [1], where the theory of query rewriting in information integration systems is extended to use the exact answer, instead of the certain one.

The correctness and completeness criteria for tables and query-evaluation is formalized using the notion of representation system [2]. Here an alternative, equivalent formulation given in [3] is used: Consider a class of tables $\mathsf{T}$, and a query language $\mathcal{Q}$. A triple $(\mathsf{T}, rep, \mathcal{Q})$ is said to be a *representation system* if for every table $T \in \mathsf{T}$, and for every applicable $q \in \mathcal{Q}$, there exist a function (here also named) $q$, such that

$$\cap\, rep(q(T)) = \cap q(rep(T)), \text{ and} \tag{1}$$

$$q' \circ q(T) = q'(q(T)), \tag{2}$$

for all applicable $q' \in \mathcal{Q}$.
Condition (1) says that the system can correctly compute the certain answer, and condition (2) states that the computation has to be uniformly recursive, following the structure of $q$. The important result is now that the class of Naive tables, and the class of all negation-free relational algebra expressions, together with Naive evaluation form such a representation system. And this result comes without any computational penalty.

## CROSS REFERENCE

Incomplete Information, Naive tables, Certain answer, Maybe answer.

## RECOMMENDED READING


[1] Gösta Grahne, Victoria Kiricenko: Towards an algebraic theory of information integration. *Inf. Comput.* 194(2): 79-100 (2004)
[2] Tomasz Imielinski, Witold Lipski Jr.: Incomplete Information in Relational Databases. *J. ACM* 31(4): 761-791 (1984)
[3] Witold Lipski Jr.: On Relational Algebra with Marked Nulls. *PODS* 1985: 201-203