The incomplete database

Karen L. Kwast
University of Amsterdam
Departments of Mathematics and Computer Science,
Plantage Muidergracht 24, 1018 TV, Amsterdam

Abstract

The introduction of *nulls* (unknown values) in the relational database calls for an extension of the theoretical foundation of the database model. Nulls are alien to classical logic, in which the relational database model is rooted. This has led to all sorts of counterintuitive *ad hoc* solutions, reasonable in one place, but awkward in others.

A sound model-theoretical foundation of nulls in a relational database based on modal logic is presented here. The modal interpretation of queries is easy to comprehend and intuitively correct. Partial interpretations, which are to be preferred from a computational point of view, are inadequate for arbitrary queries.
Formulas that have an identical partial and modal interpretation are called *safe.* Safety guarantees on the one hand that the partial answer is *meaningful* and on the other hand that the modal interpretation is *finitely computable.* The suitability of modal logic to model nulls is illustrated by a short discussion of the effect of nulls on database integrity.

1 Introduction

The relational database model has in recent years been extended to cope with missing data ([Codd, 1979]). A relation may contain incomplete tuples, because for some objects an attribute is inapplicable *(name-of-spouse)* or because it is unknown *(blood-group).* Absent data are unproblematic from a theoretical point of view (see [Lien, 1979]), and will not be discussed here. The mathematical properties of unknown data, however, are insufficiently known. As a consequence, the treatment of unknown values in relational databases tends to be illogical or unnatural (see [Date, 1989],[Codd, 1990]). This paper deals with the theoretical foundation of nulls in a relational database.

1.1 Unknown values

Let *nulls* $(\omega_1, \omega_2, \ldots)$ represent the unknown values in a relational database. Indices are used to distinguish between different unknown values and multiple occurrences of the same unknown value. These indices will be hidden from the user, but they are essential to preserve basic algebraic laws such as

$$R \subseteq \Pi_{XY}(R) \bowtie \Pi_{XZ}(R)$$

The main problem with nulls is that they lack identity criteria: there is no way to decide whether '$\omega = 5$' or rather '$\omega \neq 5$', except by revealing w's identity. This means that '$\omega = 5$' is neither true nor false, and thus contradicts the Law of the Excluded Middle. Unfortunately, the relational database model is firmly rooted in classical logic, so the introduction of nulls calls for major changes. The following sequence of equivalent terms, for instance, must be relinquished, since in the presence of nulls the first term is but a part of the last one:

$$\sigma_{A \text{ not null}}(R) \equiv \sigma_{A=5}(R) \cup \sigma_{A\neq 5}(R) \equiv$$
$$\sigma_{A=5 \vee A\neq 5}(R) \equiv \sigma_{\text{TRUE}}(R) \equiv R$$

The precise point at which this sequence breaks down will depend on the type of logic that is adopted.

1.2 Choice of logic

Since the relational database model is set in *classical logic,* attempts have been made to incorporate nulls into classical logic. The simplest solution is to use *defaults,* reserved values like xxx or 999, providing no system support of nulls at all ([Date, 1989]). The relational tuples have been extended with additional attributes to encode the level of uncertainty ([Biskup, 1981], [Ola, 1989]). This does not suffice to preserve classical tautologies, such as '$\omega = 5 \vee \omega \neq 5$', so equations have, been interpreted through *uniform null substitution* ([Grant, 1979]) or *normal form transformation* ([Lipski, 1977], [Vassiliou, 1979]). There is no formal justification for the restriction of these variants of *supervaluation* to equational expressions. Moreover, supervaluation leads to an interpretation that is not compositional, so:

$$\sigma_{A=5}(R) \cup \sigma_{A\neq 5}(R) \not\equiv \sigma_{A=5 \vee A\neq 5}(R)$$

In recognition of w = 5's being neither true nor false, there have been proposals that use *many-valued logic* ([Codd, 1979],[Codd, 1986], [Zaniola, 1984]) or *partial logic* ([Demolombe *et* a/., 1988]). The missing data create a gap in the truth value assignment, not only for equations but for relational literals as well. Unfortunately, this implies that classical tautologies are no longer valid, in that they have no truth value for tuples

with nulls. Since query optimization heavily depends on term rewriting, this will have all sorts of unexpected side-effects:

$$\sigma_{A=5 \vee A \neq 5}(R) \not\equiv \sigma_{\mathrm{TRUE}}(R) \equiv R$$

The third approach, the one I advocate for nulls in a relational database, is to change to *modal logic* ([Lipski, 1977],[Lipski, 1979], [Lipski, 1981], [imielinski and Lipski, 1984b], [Levesque, 1984]). Modal logic is a conservative extension of classical logic, so all classical laws are preserved, and the correct inequivalence emerges:

$$\sigma_{A \text{ not null}}(R) \not\equiv \sigma_{A=5}(R) \cup \sigma_{A \neq 5}(R)$$

A drawback of modal logic is its computational complexity. Nevertheless, it is the only approach that is consistently intuitive and easy to comprehend.

1.3 Choice of paradigm

Under the *database-as-theory* paradigm (see [Reiter, 1984], [Demolombe *et* a/., 1988]) nulls are Skolem constants and part of the language. This leads to 'extended relational theories' for which the Unique Names Axioms, which specify the identity criteria for constants and nulls, are no longer complete. As a consequence, the query evaluation algorithm is only sometimes complete ([Reiter, 1986], [Vardi, 1986], [Demolombe *et* a/., 1988]). This paper adheres to the *database-as-model* paradigm, but all results can be translated to the database-as-theory paradigm (see [Vardi, 1986]). The choice bet ween the two paradigms is irrelevant, as is illustrated by De molornbe & Farinas del Cerro in [1988], where they discuss the "extended Herbrand model'. One advantage of the database-as-model approach is that nulls need not be added to the query language, but may remain in the realm of the database implementation.

1.4 Outline of this paper

It is well-known that the meaning of an incomplete relation is the set of its completions ([Biskup, 1981], [Vardi, 1986]). The main innovation of the present proposal is to use a set of *revelations* to represent those completions, and to incorporate them explicitly into the interpretation of relational queries **(§ 2).** Query evaluation will be hard to compute inductively, and is therefore linked to -by way of example- a partial interpretation **(§ 3).** Working with a modal interpretation ensures that the limitations of using nulls can be established with greater case and precision **(§ 4).** The intuitive correctness of employing modal logic to represent unknown values is illustrated in a discussion of database integrity **(§ 5).**

2 A relational model for nulls

After all is said and done, a complete relational database DB is just a set of tables, an *instance* of a database scheme Σ, *acceptable* if it satisfies some set of constraints *X.* As such it is a model $< \mathcal{D}, \mathcal{I} >$, with D the domain of relevant objects and $\mathcal{I}$ the interpretation of relation names as relations over *V.*

Each constant c $\in$ CON will denote a unique object $\underline{c} \in \mathcal{D}$, and each object *d* $\in$D must have a name. These assumptions are reasonable in a database context; if in addition all objects have different names, as required by the Unique Names Axioms, then the identity = is interpreted as $\{< d, d > \mid d \in \mathcal{D}\} = \{< \underline{c}, \underline{c} > \mid \underline{c} \in CON\}$.

2.1 Extend the model, not the language

Suppose the database is provided with a *finite* set of nulls $\Omega = \{\omega_1, \ldots, \omega_k\}$. Then the model contains relations over the *extended domain* $\mathcal{D} \cup \Omega$, but the interpretation of = can not be $\{< d, d > \mid d \in \mathcal{D} \cup \Omega\}$, since that would make '$\omega = 5$' false instead of unknown. Especially when ω ranges over a finite domain, such as days in the week, we cannot pretend that w is different to them all.

There is no need to extend the query language with names for nulls. Queries that refer to nulls should be reformulated by means of existential quantification, addressing the null through its context, a partial tuple in an incomplete relation.

The identifying index is not relevant to external users. Indices are analogous to the surrogates used for tuple identification; they are there for the sake of the system, but the external user cannot address them. In other words, nulls are part of the incomplete relational *model,* not part of any relational query language.

2.2 Revelations

The ultimate meaning of the null is modeled by a set G of *revelations.* The term 'revelation' is chosen to indicate that the unknown identity of the nulls is *revealed* eventually, but we want to reason over the incomplete database without waiting for revelations (cf *symbolic constraint solving,* see [Denneheuvel *et* al., 1991]). Alternative terms would be 'null-assignment'' or even 'substitution', though the latter term sounds more innocent than is justified by its elusive denotation.

A revelation $\mathbf{g} : \Omega \rightarrow \mathcal{D}$ is a function that assigns a proper value to each null, thus resolving the nulls in the relations, and completing the database: For each $\mathbf{g} \in \mathbf{G}$ and $\underline{\mathbf{R}}$ in DB, $\mathbf{g}[\underline{\mathbf{R}}]$ is a complete relation over *V.*

The revelations G represent all possible *completions* of the database. Partial knowledge of the identity of the nulls can be encoded in G's structure. The database integrity constraints, for instance, may resolve some nulls, and rule others unequal (see § 5 below).

2.3 Truth definition

Queries to the database will be formulated in a Codd-complete query language, such as the relational algebra. Any query can be translated into first order predicate logic, however, so the truth definition is here recursive over logical connectives and quantifiers, extended -for the sake of the nulls- with the modal operators K for 'it is known that' and M for 'it is conceivable that'.

In the clauses below the assignment *h* is a standard assignment $h \in \mathbf{H}$: $\mathbf{VAR} \rightarrow \mathcal{D}$, to resolve free variables. Existential quantification is interpreted by means of x-cylindrification:

$$\mathcal{C}_x\mathcal{I}(\varphi) := \{< g, h > \mid h =_x h' < g, h' > \in \mathcal{I}(\varphi)\})$$

The symbol $=_x$ denotes the equality of two assignments on all variables, with the possible exception of *x.*

Definition 1 incomplete instance
A model $< \mathcal{D} \cup \Omega, \mathcal{I}, G >$ is an incomplete modal instance of a relational database scheme Σ iff every n-ary relation name R in Σ is interpreted by $\mathcal{I}(R) = \underline{R} \subseteq (\mathcal{D} \cup \Omega)^n$ and every formula φ by $\mathcal{I}(\varphi) \subseteq G \times H$, as follows:

$$
\begin{array}{lcl}
\mathcal{I}(Ra \ldots x) & := & \{< g, h > \mid < \underline{a}, \ldots, h(x) > \in g[\underline{R}]\} \\
\mathcal{I}(a = b) & := & \{< g, h > \mid \underline{a} = \underline{b}\} \\
\mathcal{I}(x = b) & := & G \times \{h \in H \mid h(x) = \underline{b}\} \\
\mathcal{I}(x = y) & := & G \times \{h \in H \mid h(x) = h(y)\} \\
\mathcal{I}(\neg\varphi) & := & G \times H \setminus \mathcal{I}(\varphi) \\
\mathcal{I}(\varphi \wedge \psi) & := & \mathcal{I}(\varphi) \cap \mathcal{I}(\psi) \\
\mathcal{I}(\exists x\, \varphi) & := & C_x \mathcal{I}(\varphi) \\
\mathcal{I}(\mathbf{K}\varphi) & := & G \times \{h \in H \mid G \times \{h\} \subseteq \mathcal{I}(\varphi)\}
\end{array}
$$

Definition 1 can be explained as follows. The interpretation of formulas is presented in an algebraic manner, to emphasize the connection with the relational algebra. Equivalently, it can be given as a modal truth definition, since the revelations G represent a set of possible worlds $W \equiv \{ w_g \mid g \in G \}$, and the incomplete instance represents a Kripke model, which is of course an appropriate model for unknown values.

Lemma 1 *$< \mathcal{D} \cup \Omega, \mathcal{I}, G > \equiv < \mathcal{D}, \mathcal{I}', W >$, in such a way that*

- $W \equiv \{ w_g \mid g \in G \}$
- $\mathcal{I}'(w_g)(R) = g[\underline{R}]$
- $< \mathcal{D}, \mathcal{I}', W >, w_g \models \varphi[h]$ *iff* $< g, h > \in \mathcal{I}(\varphi)$

The model $< \mathcal{D}, \mathcal{I}', W >$ is a connected **S5** Kripke model, which yields a very simple modal logic.

Nulls are modeled as parameters, and only through the revelations G are they existentially quantified. To describe the incomplete model, one can replace every ω by an existentially quantified variable, but '$\exists\omega\varphi$' makes no sense, since ω is not a variable in the query language.

Given lemma 1 it is easy to see that $< g, h > \in \mathcal{I}(\varphi)$ has the intuitive meaning that the restriction of h to FV(φ) denotes a tuple for which φ is true, *provided* that g is the correct revelation of unknown values.

Example 1 Let R(ABC) be interpreted as

$$\underline{R} = \{< a, b, c >, < d, \omega_{7639}, f >\}.$$

$\mathcal{I}(Rdxy) = \{< g, h > \mid h(x) = g(\omega_{7639})\ \&\ h(y) = f\}$
$\mathcal{I}(\exists x\ Rdxy) = \{< g, h > \mid h(y) = f\}$
If the unknown B value happens to be e, then '$< e, f >$' is a correct answer to the query 'xy such that $Rdxy$'; f is a solution to the query 'y such that $\exists x\ Rdxy$' independent of the unknown value of the tuple. •

The correct revelation g is not known, but by adding a modal operator **K** or **M** it becomes superfluous.

Example 2 The query 'Is it true that *Rdef*?' is not very satisfactory answered by the condition '$\omega \mapsto e$', that is 'yes, provided *Rdef*. The modalized versions do make sense: the query 'Is it known that *Rdef*?' yields 'No ...', and 'Is it possible that *Rdef*?' receives a final 'Yes !'. •

$\mathcal{I}(\mathbf{K}\varphi)$ does not depend on g, and the answer to an open query is

$$\mathcal{I}_*(\varphi) := \{h \in H \mid < g, h > \in \mathcal{I}(\mathbf{K}\varphi)\}$$

and if necessary the dual $\mathcal{I}^*(\varphi)$, the projection of $\mathcal{I}(\mathbf{M}\varphi)$ on H. Unfortunately, it is not possible to compute $\mathcal{I}_*$ by independent induction, that is, without first computing $\mathcal{I}$.

Example 3 Let R(ABC) be interpreted as before.
$\mathcal{I}_*(Rdxy) = \emptyset$
$\mathcal{I}_*(\exists x\ Rdxy) = \{h \in H \mid h(y) = f\}$
$\mathcal{I}_*(\neg Rdxy) = \{h \in H \mid h(y) \neq f\}$
No tuple xy is known to satisfy $Rdxy$. Still, it is known that $y \mapsto f$ satisfies $\exists x\ Rdxy$, and it is known that h satisfies $\neg Rdxy$ as long as $y \not\mapsto f$, regardless of $h(x)$. •

3 Safe queries

The only drawback of modal logic is its computational complexity. In comparison it is very easy to compute a partial interpretation, namely by a finite induction. Queries φ on which these interpretations agree will be called *safe*.

3.1 Partial logic

A partial interpretation of open queries will either implicitly or explicitly make use of *extended assignments* H^+: VAR $\rightarrow \mathcal{D} \cup \Omega$. The positive answers $\mathcal{J}_+(\varphi)$ are computed by straightforward induction, with an accompanying set of negative answers $\mathcal{J}_-(\varphi)$. This guarantees that all the *de Morgan duality laws* are validated, so each formula can be transformed into prenex normal form.

Definition 2 partial instance
A model $< \mathcal{D} \cup \Omega, \mathcal{J} >$ is a partial instance of a relational database scheme Σ iff every n-ary relation name R in Σ is interpreted by $\mathcal{J}(R) = \underline{R} \subseteq (\mathcal{D} \cup \Omega)^n$ and every formula φ is interpreted by a pair $\mathcal{J}_+(\varphi), \mathcal{J}_-(\varphi) \subseteq H^+$, as follows:

$$
\begin{array}{lcl}
\mathcal{J}_+(R(a \ldots x)) & := & \{h \in H^+ \mid < \underline{a} \ldots h(x) > \in \underline{R}\} \\
\mathcal{J}_+(x = y) & := & \{h \in H^+ \mid h(x) = h(y)\} \\
\mathcal{J}_+(x = a) & := & \{h \in H^+ \mid h(x) = \underline{a}\} \\
\mathcal{J}_+(a = b) & := & \{h \in H^+ \mid \underline{a} = \underline{b}\} \\
\mathcal{J}_+(\neg\varphi) & := & \mathcal{J}_-(\varphi) \\
\mathcal{J}_+(\varphi \vee \psi) & := & \mathcal{J}_+(\varphi) \cup \mathcal{J}_+(\psi) \\
\mathcal{J}_+(\varphi \wedge \psi) & := & \mathcal{J}_+(\varphi) \cap \mathcal{J}_+(\psi) \\
\mathcal{J}_+(\exists x\, \varphi) & := & \mathcal{C}_x^+(\mathcal{J}_+(\varphi)) \\
\mathcal{J}_+(\forall x\, \varphi) & := & \mathcal{D}_x^+(\mathcal{J}_+(\varphi)) \\
 & & \\
\mathcal{J}_-(R(a \ldots x)) & := & \{h \in H^+ \mid < \underline{b} \ldots \underline{d} > \in \underline{R} \Rightarrow \\
 & & \quad < \underline{b} \ldots \underline{d} > \neq < \underline{a} \ldots h(x) >\} \\
\mathcal{J}_-(x = y) & := & \{h \in H^+ \mid h(x) \neq h(y)\} \\
\mathcal{J}_-(x = a) & := & \{h \in H^+ \mid h(x) \neq \underline{a}\} \\
\mathcal{J}_-(a = b) & := & \{h \in H^+ \mid \underline{a} \neq \underline{b}\} \\
\mathcal{J}_-(\neg\varphi) & := & \mathcal{J}_+(\varphi) \\
\mathcal{J}_-(\varphi \vee \psi) & := & \mathcal{J}_-(\varphi) \cap \mathcal{J}_-(\psi) \\
\mathcal{J}_-(\varphi \wedge \psi) & := & \mathcal{J}_-(\varphi) \cup \mathcal{J}_-(\psi) \\
\mathcal{J}_-(\exists x\, \varphi) & := & \mathcal{D}_x^+(\mathcal{J}_-(\varphi)) \\
\mathcal{J}_-(\forall x\, \varphi) & := & \mathcal{C}_x^+(\mathcal{J}_-(\varphi))
\end{array}
$$

The cylindrifications $\mathcal{C}_x^+$ and $\mathcal{D}_x^+$ are over the extended domain $\mathcal{D} \cup \Omega$. Inequality is defined over $\mathcal{D}$ alone, so if $h(x) \neq h(y)$, then $h(x), h(y) \in \mathcal{D}$. Two tuples are inequal, $< d_1 \ldots d_n > \neq < e_1 \ldots e_n >$, if there is at least one place at which they disagree: for some i: $d_i \neq e_i$.

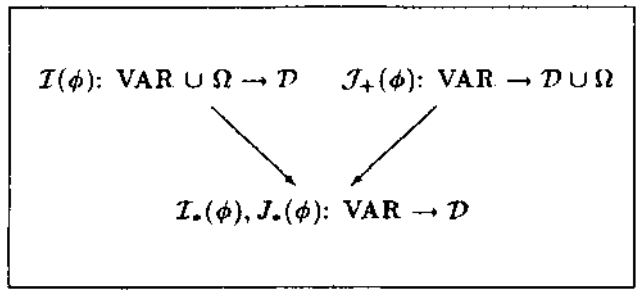

Figure 1: Restricted interpretations

3.2 Comparison on H

To compare the modal and the partial interpretations we need to restrict the latter to H, see figure 1. This can be done without loss of expressihility, since any reference to nulls can be avoided by additional quantification.

$$\mathcal{J}_*(\varphi) := \mathcal{J}_+(\varphi) \cap H \qquad \mathcal{J}^*(\varphi) := H \setminus \mathcal{J}_-(\varphi).$$

Both the modal and the partial interpretation cannot be computed on H directly. The intersection of $\mathcal{J}_+(\varphi)$ with H interferes with quantification, but otherwise $\mathcal{J}_*$ can itself be computed by induction over the structure of the query. For modal logic, answers are lost in case of disjunction and existential quantification.

Theorem 2 *Comparison of the 2 interpretations*
1. answers known to be true

$\mathcal{J}_*(\varphi) \subseteq \mathcal{J}^*(\varphi)$	$\mathcal{I}_*(\varphi) \subseteq \mathcal{I}^*(\varphi)$
$\mathcal{J}_*(\neg\varphi) = \mathbf{H} \setminus \mathcal{J}^*(\varphi)$	$\mathcal{I}_*(\neg\varphi) = \mathbf{H} \setminus \mathcal{I}^*(\varphi)$
$\mathcal{J}_*(\varphi \vee \psi) = \mathcal{J}_*(\varphi) \cup \mathcal{J}_*(\psi)$	$\mathcal{I}_*(\varphi \vee \psi) \supseteq \mathcal{I}_*(\varphi) \cup \mathcal{I}_*(\psi)$
$\mathcal{J}_*(\varphi \wedge \psi) = \mathcal{J}_*(\varphi) \cap \mathcal{J}_*(\psi)$	$\mathcal{I}_*(\varphi \wedge \psi) = \mathcal{I}_*(\varphi) \cap \mathcal{I}_*(\psi)$
$\mathcal{J}_*(\exists x\ \varphi) \supseteq \mathcal{C}_x(\mathcal{J}_*(\varphi))$	$\mathcal{I}_*(\exists x\ \varphi) \supseteq \mathcal{C}_x(\mathcal{I}_*(\varphi))$
$\mathcal{J}_*(\forall x\ \varphi) \subseteq \mathcal{D}_x(\mathcal{J}_*(\varphi))$	$\mathcal{I}_*(\forall x\ \varphi) = \mathcal{D}_x(\mathcal{I}_*(\varphi))$

2. answers that may be true

$\mathcal{J}^*(\neg\varphi) = \mathbf{H} \setminus \mathcal{J}_*(\varphi)$	$\mathcal{I}^*(\neg\varphi) = \mathbf{H} \setminus \mathcal{I}_*(\varphi)$
$\mathcal{J}^*(\varphi \vee \psi) = \mathcal{J}^*(\varphi) \cup \mathcal{J}^*(\psi)$	$\mathcal{I}^*(\varphi \vee \psi) = \mathcal{I}^*(\varphi) \cup \mathcal{I}^*(\psi)$
$\mathcal{J}^*(\varphi \wedge \psi) = \mathcal{J}^*(\varphi) \cap \mathcal{J}^*(\psi)$	$\mathcal{I}^*(\varphi \wedge \psi) \subseteq \mathcal{I}^*(\varphi) \cap \mathcal{I}^*(\psi)$
$\mathcal{J}^*(\exists x\ \varphi) \supseteq \mathcal{C}_x(\mathcal{J}^*(\varphi))$	$\mathcal{I}^*(\exists x\ \varphi) = \mathcal{C}_x(\mathcal{I}^*(\varphi))$
$\mathcal{J}^*(\forall x\ \varphi) \subseteq \mathcal{D}_x(\mathcal{J}^*(\varphi))$	$\mathcal{I}^*(\forall x\ \varphi) \subseteq \mathcal{D}_x(\mathcal{I}^*(\varphi))$

Theorem 2 can be found in several places in the literature ([Lipski, 1977], [Reiter, 1986], [Demolombe *et al.*, 1988]), though the precise details will depend on the details of the particular model. Consider for instance the set of answers (see [Reiter, 1986]) deducible from the database:

$$\| \varphi \| := \{ h \in H^+ \mid DB \vdash \varphi[h] \}.$$

The evaluation algorithm proposed is $\mathcal{J}_+$; completeness is guaranteed if $\mathcal{I}_*$-inequalities do not arise.

All classes of unproblematic queries suggested in the literature are *safe* under the present definition.

Definition 3 *A formula φ is* **safe**
iff for every partial $\mathcal{J}$ and modal $\mathcal{I}$: $\mathcal{J}_*(\varphi) = \mathcal{I}_*(\varphi)$, provided that $\mathcal{J}(R) = \mathcal{I}(R)$ for every relation name R.

3.3 Unsafe queries

Before searching for classes of safe queries, first an example of an unsafe query, which is incorrectly answered under the partial interpretation $\mathcal{J}_*$.

Example 4 Any comparison expression F is safe, but F cannot be safely combined with a relational atom. If $\underline{R} = \{< a, b, \omega >\}$:
$\mathcal{J}_*(\exists y\ R(axy)) = \mathcal{I}_*(\exists y\ R(axy)) = \{x \mapsto b\}$
$\mathcal{I}_*(\exists y:\ R(axy) \wedge (y = 5 \vee y \neq 5)) = \{x \mapsto b\}$
$\mathcal{J}_*(\exists y:\ R(axy) \wedge (y = 5 \vee y \neq 5)) = \emptyset$ •

The partial instance as defined here does not make use of the set of revelations G, so any query that depends indirectly on the structure of this G cannot be correctly evaluated. To avoid this unfairness it will be assumed that G is the full set $\mathcal{D}^\Omega$. The next example shows that, unless some sort of fairness condition is laid on G, even a relational atom may be unsafe.

Example 5 Let $\underline{R} = \{< a, \omega_1, c >, < a, b, \omega_2 >\}$.
Suppose that for all $g \in G$ either $g(\omega_1) = b$ or $g(\omega_2) = c$.
Then $\mathcal{J}_*(Rxbc) = \emptyset \neq \{x \mapsto a\} = \mathcal{I}_*(Rxbc)$. •

3.4 Classes of safe queries

Positive selections in combination with union, join and projection can be evaluated over incomplete relations (see [imielinski and Lipski, 1984b]). These relational terms correspond roughly with positive queries as defined by Reiter (see below), and with formulas without negation. The translation of positive selections is of the format $\varphi \wedge E$, with E an equational expression.

Definition 4 positive
A relational atom is positive.
If φ and ψ are positive, then $\exists \mathbf{x}\ \varphi$, $\forall \mathbf{x}\ \varphi$, $\varphi \wedge E$, $\varphi \wedge \psi$, and $\varphi \vee \psi$, are all positive.

Reiter allows restricted quantification only, to avoid unreasonable queries. He defines positive queries accordingly ([Reiter, 1986]):

Definition 5 Reiter's positive
An atomic formula is positive.
If φ and ψ are positive and T is a finite type, then $\varphi \wedge \psi$, $\varphi \vee \psi$, $\exists x \in T:\ \varphi$, and $\forall x \in T:\ \varphi$, are all positive.

One would like to see that all literals are safe. Since conjunction and universal quantification are unproblematic, then so are conjunctive queries:

Definition 6 Reiter's conjunctive
A formula φ is conjunctive iff φ is of the format $\forall \mathbf{x} \in \mathbf{T}:\ \psi_1 \wedge \ldots \wedge \psi_n$, where each conjunct ψ_i is a literal, an atom or negated atom.

Demolombe & Farinas del Cerro ([1988]) define a larger set of conjunctive queries.

Definition 7 *A formula φ is* conjunctive
iff φ is a conjunction of $[\neg][\exists\ \mathbf{x}]\ \psi$, with ψ a disjunction of atoms and $\mathbf{x}$ free in one of the disjuncts.

In order to prove that a negated atom such as $\neg\ Rxab$ is safe, we need to sharpen the basic clause $\mathcal{J}_-$ a little, as can be seen from the following example.

Example 6 Let $\underline{R} = \{< k, \omega_{245}, \omega_{245} >\}$.
$\mathcal{I}_*(\neg Rxab) = H \neq \{h \mid h(x) \neq k\} = \mathcal{J}_*(\neg Rxab)$ •

To counter this example it suffices to redefine the notion of tuple inequality:

Definition 8 uniform tuple inequality
$< d_1 \ldots d_n > \neq < e_1 \ldots e_n > :=$
$\forall g \in \mathcal{D}^\Omega :< g(d_1) \ldots g(d_n) > \neq < g(e_1) \ldots g(e_n) >$

This is a generalization of strong tuple inequality, since $d_i \neq e_i$ implies that $d_i, e_i \in \mathcal{D}$, so for all revelations $g(d_i) \neq g(e_i)$.

The change is significant. Apparently we need a little modal information even for a simple literal. It is not much, just the set of all possible revelations $\mathcal{D}^{\Omega}$, but it is a clear indication that modal logic will become indispensable when queries get more complex.

Once the partial interpretation is adapted in this manner, we can prove that all given sets of queries are safe:

Theorem 3 *If (p is a positive or conjunctive query, then* $\mathcal{J}_*(\varphi) \quad \mathcal{I}_*(\varphi)$

The set of semantically safe queries can easily be extended using modal equivalences, leading to larger syntactic sets of queries (see [Kwast, 1999]).

Definition 9 OK, XDE
OK $\rightarrow$ positive | OK $\wedge$ OK | $\forall$xOK ¦ $\neg$XDE
XDE $\rightarrow$ rel. atom | XDE $\vee$ XDE | $\exists$xXDE | $\neg$OK

Theorem 4 *Let φ be* **OK**, *then* $\mathcal{I}_*(\varphi) = \mathcal{J}_*(\varphi)$.

4 Interpretation versus computation

There are two ways to appreciate safety. On tin* one hand, safe queries are queries for which the modal interpretation gives the intuitive *meaning* of the partial interpretation. This feeling is strengthened by the fact-that the partial interpretation of an unsafe query equals the modal interpretation of a modalized version of the query, as in the following example.

Example 7 Let $\underline{R} = \{< a, \omega_{111} >, < b, \omega_{111} >\}$.
$\mathcal{J}_*(\exists x\, \neg Rax) = \mathcal{I}_*(\exists x\, \mathbf{K}\neg Rax) \neq \mathcal{I}_*(\exists x\, \neg Rax)$.
$\mathcal{J}_*(\exists x : (Rax \wedge x = 5) \vee (Rbx \wedge x \neq 5))$
$= \mathcal{I}_*(\mathbf{K}(\exists x\, Rax \wedge x = 5) \vee \mathbf{K}(\exists x\, Rbx \wedge x \neq 5))$
$\neq \mathcal{I}_*(\exists x\, (Rax \wedge x = 5) \vee (Rbx \wedge x \neq 5))$. •

As is suggested by the second query, there is, unfortunately, no *uniform* translation from the one logic to the other. Every partial answer can be expressed by a modalized formula, of course, but it cannot be predicted by which particular formula. It must be concluded that there remains a class of queries for which the partial interpretation is inappropriate. Some answer is computed, but there is no intuitive idea about what it might be.

4.1 Finite domains

Obviously, the definition of a partial model can be changed, for instance by the addition of G into the interpretation of $\mathcal{J}_+$ and/or $\mathcal{J}_-$. To be sure, in [Demolombe *et* a/., 1988] the partial induction has access to G to determine the basic clauses.

In this manner the partial interpretation is set on a modal base. This does not suffice to make it modal logic, of course. For one thing, partial logic cannot cope with finite domains:

Example 8 Let $\mathcal{D} = \{a, b, c\}$ and $\underline{R} = \{< \omega, c >\}$.
$\mathcal{J}_*(Rac \vee Rbc \vee Rcc) = \emptyset \neq H = \mathcal{J}_*(\exists x Rxc)$
$\mathcal{I}_*(Rac \vee Rbc \vee Rcc) = \mathcal{I}_*(\exists x Rxc) = H$

In a database context many nulls will refer to a finite domain, such as *day-in-the~week* or *blood-group.* A partial interpretation, even when set on a modal base, is not adequate for queries in which finite domains are involved.

4.2 Reasonable queries

If there is no uniform translation from partial logic into modal logic, we may turn in the opposite direction and use partial logic to *compute* the modal interpretation. Obviously, this only works for safe and 'reasonable' queries, queries which can be finitely computed if the database is complete. If there is no other simple way to compute the modal interpretation, then that is a good reason to forbid the query. It might be cheaper to try and find out what the missing data should have been by using the telephone or a nationwide poll.

4.3 Supervaluation

A good example of a computation method for a distinct class of queries is *normal form transformation* used to evaluate projection-selection queries. Imielinski &: Lipski prove in ([1984b]) that $< @\text{-}tables, Rep, \{\Pi\ \sigma\} >$ is a *representation system.* This means that projection-selection queries can be correctly represented by functions on @-tables, relations with unindexed nulls. In our terminology: PS-queries are safe, on the assumption that all nulls occur uniquely in the tables, to guarantee that two nulls need never be identified. As was shown before (see example 4), PS-queries are not safe with respect to the partial interpretation, but they are safe with respect to supervaluation of the normalized form.

5 Nulls and integrity

The effect of nulls on integrity has not drawn as much attention as the problems of query evaluation. The traditional definition of a database satisfying its constraints is not adequate (see [Reiter, 1988]). Codd's suggestion ([1979], [1986]) that nulls should just be ignored, suspending judgement to the time of their revelation, is no solution either. Not every incomplete tuple may be added to the database.

The model for nulls presented here is well suited to describe database integrity.

Definition 10 *An incomplete database is* acceptable *iff there exists an acceptable revelation, that is, one that yields a database that satisfies all its constraints.*

This simple definition can be compared to the definitions proposed in literature (see [Reiter, 1988]): It is not enough if the atomic facts plus constraints are satisfiable; they must be satisfied by a completion deriving from a revelation. Still, the atomic facts could never imply the constraints. So definition 10 is a suitable adaptation of the Closed World Assumption.

5.1 What should hold

The set of constraints will divide the set of revelations in two: acceptable and unacceptable revelations. Relative to the former set we define a modal operator X with the intended meaning *what should hold,* that is, what holds in all acceptable completions.

Definition 11 $DB \models \mathbf{X}\varphi$ *iff* $\exists g \in G$: g *acceptable, and* $\forall g \in G$: g *acceptable* $\Rightarrow$ $DB, g \models \varphi$.

This definition, being based on Kripke semantics, captures the interaction of integrity constraints with unknown values in an intuitive manner (see [Kwast, 1999]).

Example 9 Suppose $R(ABCD) : A \rightarrow B, C \rightarrow D$ and let $\underline{R} = \{< a, b, \omega_1, d >, < a, \omega_2, p, q >\}$.
$DB \models \mathbf{X}Rabpq \quad (: \omega_2 = b)$
$DB \models \mathbf{X}\neg Rabpd \quad (: \omega_1 \neq p)$
$DB \models \mathbf{X}\forall x : Rabxd \supset x \neq p$
The insertion of the tuple $< e, f, \omega_1, r >$ into $\underline{R}$ will make the database unacceptable. •

5.2 Queries versus constraints

To distinguish formulas that denote constraints, such as $R : X \rightarrow Y$, from relational facts and queries, the former will contain a leading X operator:

$$\mathbf{X}(\forall x, y, y', z, z' \; Rxyz \wedge Rxy'z' \supset y = y')$$

Queries are practically never formulated with the help of X, or any analogous operator. Evaluated over total relations X is vacuous, of course, but in the presence of nulls the interpretation of a query φ which does not involve X will contain answers that contradict the relational dependencies, and it will miss others that should have been there. Unfortunately, it cannot be hoped that integrity considerations are actually employed in answering queries, on account of the ensuing complexity.

6 Conclusion

The overall conclusion of this paper is that any sound model-theoretic foundation of nulls in a relational database must be based on modal logic. That modal logic is appropriate is obvious, but it is necessary as well, as there is no other way to assess the intuitive correctness of an evaluation method for wide classes of queries.

References

[Biskup, 198l] J. Diskup. *A formal approach to null values in database relations.* In: H.Gallaire et al. (eds.) Advances in database theory, Vol. 1. 1981

[Codd, 1979] E.F. Codd. *Extending the relational database model to capture more meaning.* In: ACM Transactions on Database Systems, Vol. 4. 1979

[Codd, 1986] E.F. Codd. *Missing information (applicable and inapplicable) in relational databases.* In: ACM SIGMOD, Vol. 15. 1986

[Codd, 1990] E.F. Codd. *The Relational Model for Database Management: version 2.* Addison-Wesley, 1990

[Date, 1989] C.J. Date. *Null Values in Database Management.* In: Relational Database: Selected Writings. Addison-Wesley, 1986

[Demolombe *et al.*, 1988] R. Demolombe & L. Farinas del Cerro. *An algebraic evaluation method for deduction in incomplete databases.* In: Journal of Logic Programming, Vol. 5. 1988

[Denneheuvel *et al,* 199l] S. van Denneheuvel & K.L. Kwast. *Weak equivalence for constraint sets.* In: Proceedings of IJCAI'91, Morgan Kaufmann, 1991

[Grant, 1979] J. Grant. *Null values in the relational database.* In: Information Processing Letters, Vol. 6, 1979

[Imielinski and Lipski, 1984a] T. Imielinski &: W. Lipski. *The relational model of data and cylindric algebras.* In: Journal of Computer and System Sciences, Vol. 28. 1984

[Imielinski and Lipski, 1984b] T. Imielinski &: W. Lipski. *Incomplete information in relational databases.* In: Journal of the ACM, Vol. 31. 1984

[Kwast, 1999] K.L. Kwast. *Nulls in a relational database.* Forthcoming.

[Levesque, 1984] H. Levesque. *Foundations of a functional approach to knowledge representation.* In: Artificial Intelligence, Vol. 23. 1984

[Lien, 1979] Y. Edmund Lien. *Multivalued dependencies with null values in relational databases.* In: Proceedings of IEEE, 1979

[Lipski, 1977] W. Lipski. *On the logic of incomplete information.* In: T. Gruska (ed.) Proc. of the 6th symp. on Math. Found, of Comp. Science, LNCS, Springer Verlag. 1977

[Lipski, 1979] W. Lipski. *On semantic issues connected with incomplete information databases.* In: ACM Transactions on Database Systems, Vol. 4. 1979

[Lipski, 1981] W. Lipski. *On databases with incomplete information.* In: Journal of the ACM, Vol. 28. 1981

[Ola, 1989] A. Ola & G. Ozsoyoglu. *A family of incomplete database models.* In: P. Apers & G. Wiederhold (ed.) Proceedings of the 15 VLDB. 1989

[Reiter, 1984] R. Reiter. *Towards a logical reconstruction of relational database theory.* In: M. Brodie et. al. (eds.) On Conceptual Modeling. Springer Verlag, 1984

[Reiter, 1986] R. Reiter. *A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values.* In: Journal of the ACM, Vol. 33. 1986

[Reiter, 1988] R. Reiter. *On Integrity Constraints.* In: M. Vardi (ed.) Proc. of the 2nd conf. on Theoretical Aspects of Reasoning about Knowledge, 1988

[Vardi, 1986] M. Vardi. *Querying Logical Databases.* In: Journal of Computer &. System Sciences, Vol. 33. 1986

[Vassiliou, 1979] Y. Vassiliou. *Null values in data base management / a denotational approach.* In: ACM SIGMOD 1979

[Zaniola, 1984] C. Zaniola. *Database relations with null values.* In: Journal of Computer and System Sciences, Vol. 28. 1984