# A Family of Incomplete Relational Database Models†

*Adegbemiga Ola‡ and Gultekin Ozsoyoglu*

Department of Computer Engineering and Science, Case Western Reserve University

## ABSTRACT

In this paper, we utilize intervals for unknown values in incomplete relational databases. We use tables to represent unknown relations. First, we define three partial tuple types in a table to specify incompleteness relationships among tuples of the same table. For tuples of different tables, we distinguish between the cases where incompleteness are introduced at the *relation level, tuple level* or *attribute-value level.* And, based on these relationships among tuples in different tables, we present a family of incomplete relational database models.

For each of the models, the query evaluation is *sound* (i.e., no incorrect results are derivable). None of the models is *complete* (i.e., all valid conclusions are derivable). We briefly compare two of the models in the family with other approaches.

Considering each table tuple as a set of d-dimensional cubes, each model in the family of models presented in this paper can be considered as a geometric database model. We are presently implementing a version of one of the models. We briefly summarize the geometric operations and the primitive update semantics being utilized in the implementation.

## 1. Introduction

Null values and incomplete information in databases have received much attention in recent years, for instance see [Codd 79, Vass 79, Lips 79, Gran 79, Gran 80, Bisk 81, Bisk 83, ImiL 84, Zani 84, AbKG 87]. Partial information in databases in the form of possible values is allowed in [Lips 79] and [Gran 80]. Imilienski and Lipski [ImiL 84] give conditions that ensure *soundness* and *completeness* of query evaluation in a certain sense, as opposed to the correctness of individual operators.

In this paper, we introduce a family of incomplete information models for the relational model. The main characteristic of all the models in this family is that the partial knowledge about an unknown value is specified as possible values in a set of intervals rather than an arbitrary set. For example, the unknown tuple t=(5, 9) is represented by the tuple, say, ([1,10], 9), in which the first component is the interval [1,10] containing the unknown value 5 in tuple t. Figure 1 illustrates tuples $p_1$=([3,5], [8,10], [3,5]) and $p_2$=([10,12], {[13,15],[20,22]}, [8,10]) of table U with scheme U($A_1$, $A_2$, $A_3$). The points or tuples in the set of cubes (also called d-rectangles) are called *candidate tuples* for the unknown tuple t, and exactly one of the points is the unknown tuple. This approach allows database operations to be transformed into operations in Computational Geometry, leading to efficient operator evaluations [OlaO 88a].

Following [ImiL 84] we use the terminology that a table in the incomplete database environment *represents* a relation some tuples of which are unknown. Tables contain *partial tuples* (i.e., tuples with incomplete components) as well as total tuples (i.e., tuples whose components are all known).

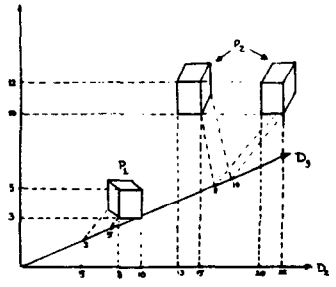

**Figure 1.** Geometric View of Partial Tuples

| Table Tuple Type | Semantics of the Represented Relation tuple |
|---|---|
| type 0 | Total tuple |
| type 1 | Exists and Is-uniquely-represented |
| type 2 | Exists and May-be-represented-already |
| type 3 | May-not-exist |

**Figure 2.** Semantics of Tuple Types

We now define new tuple types. A partial tuple consists of two or more tuples, exactly one of which is the unknown tuple. Let a table *U represent* a relation r, then the following tuple types are allowed in U.

(a) a *type 0 total* tuple is the usual relation tuple without unknown values.

(b) a *type 1 partial* tuple in *U* represents an unknown tuple t which *exists*, and no other tuple in U represents t.

(c) a *type 2* partial tuple p in *U* represents an unknown tuple t of r which is known to *exist*. However, another tuple p' of U, physically distinct than p, may also represent t. That is, p may represent a tuple in r which is already represented by some other tuple in U.

(d) a *type 3* partial tuple p in U represents an unknown tuple in r that may exist (i.e., a *maybe* tuple); the set of possible tuples for p includes the *null tuple* (a special tuple Ø denoting



† This research is supported by the National Science Foundation under Grants DCR-8605554 and IRI-8811057.

# The present address of A. Ola is Department of Computer Science, University of Dayton, Dayton, OH 45469.

the nonexistent tuple).
Figure 2 summarizes the different tuple types. We give an example.

**Example 1:** Let us consider relation r in figure 3, consisting of four tuples $t_1$, $t_2$, $t_3$ and $t_4$. If the last digits of the second components in $t_1$, $t_2$ and $t_4$ are unknown then r is represented by table $U$. The partial tuples in U are type 1 because they represent tuples which are known to be distinct. Table $V$ is the projection on attribute $A_2$ of $U$. The tuples $p_1'$ and $p_2'$ are type 1 in the projection because the unknown tuples represented by $p_1$ and $p_2$ of $U$ must have distinct $A_2$ values. Tuple $p_3'$ is of type 2 because the $A_2$ value of the unknown tuple represented by $p_3$ may be the same as that of tuple $t_1$. Table $W$ is the result of applying a selection formula $A_2 > 55$ to table $V$; both tuples $p_1'$ and $p_2'$ have a chance of satisfying the formula, and are therefore included in $W$ as type 3.

r

| | $A_1$ | $A_2$ |
|---|---|---|
| $t_1$ | 4 | 53 |
| $t_2$ | 4 | 56 |
| $t_3$ | 5 | 41 |
| $t_4$ | 3 | 41 |

U

| | $A_1$ | $A_2$ | TYPE |
|---|---|---|---|
| $p_1$ | 4 | [50,59] | 1 |
| $p_2$ | 4 | [50,59] | 1 |
| $t_1$ | 5 | 41 | 0 |
| $p_3$ | 3 | [40,49] | 1 |

V

| | $A_2$ | TYPE |
|---|---|---|
| $p_1'$ | [50,59] | 1 |
| $p_2'$ | [50,59] | 1 |
| $t_1'$ | 41 | 0 |
| $p_3'$ | [40,49] | 2 |

W

| $A_2$ | TYPE |
|---|---|
| [50,59] | 3 |
| [50,59] | 3 |

**Figure 3.** Relation r, Table U, and Operations on Tables.

Clearly, as the type of a table tuple increases from 0 to 3, it becomes less informative. For example, a type 1 tuple is more informative than a type 3 tuple. In a practical environment, users may perhaps be interested in only type 0 and type 1 tuples. However, even if the base tables in a relational database contains only type 0 and type 1 tuples, the relational algebra operators introduce type 2 and type 3 tuples.

The semantic difference between type 2 and type 3 tuples is as follows. Consider a type 1 tuple p of a table U. When a set union of U with another table is performed, p in the resulting table may represent the same unknown relation tuple (i.e., p becomes type 2); but p will not become a type 3 (i.e., a maybe) tuple. On the other hand, as illustrated in example 1, when a selection on U is performed, the unknown tuple represented by p may or may not be selected in which case p becomes a type 3 (i.e., a maybe) tuple. Also, n number of type 2 tuples in a table represent at least one tuple in the corresponding unknown relation whereas n number of type 3 tuples may not represent any tuples in the corresponding relation.

By taking into consideration the known relationships among tuples from different tables of the database one can derive more information in query responses. Such relationships vary from one environment to the other. In this paper, we identify various assumptions about relationships among tuples in different tables. Then, a family of extended models *(M-1, M-2, M-3,* and *M-4)* based on these assumptions is introduced. This approach, we believe, is practical because we are able to put at the users' disposal the variety of models, one of which possibly meets the users' assumptions. We now describe the family of models.

In model *M-1*, we model an environment where errors or incompleteness are introduced at the **relation level** meaning that the same unknown tuple may be represented by different tuples in different tables. Thus, within the database, different errors may occur in different occurrences of a given tuple in different relations.

**Basic Assumption in Model M-1** *(Relation Level)*:
Consider two tuples $z_1$ and $z_2$ occurring in different tables. If the candidate tuples of $z_1$ and $z_2$ do not intersect then $z_1$ and $z_2$ represent different (relation) tuples, otherwise they may represent the same (relation) tuple.

**Example 2:** In figure 4, relation $r_2$ with tuple $t_1$ contains no errors, and is represented "as-is" as a table (with the addition of the TYPE column). On the other hand, the first digit of the second component of tuple $t_1$ of $r_1$ is unknown, and $r_1$ is represented by table $U$. Tuple $t_1$ of $r_1$ is represented by the partial tuple $p_1$ in $U$. But the occurrence of $t_1$ in $r_2$ is total. The inference we can make is that two tuples from different tables do **not** represent the same relation tuple if at least one of their tuple components do not intersect.

$r_1$

| | $A_1$ | $A_2$ |
|---|---|---|
| $t_1$ | 4 | 53 |
| $t_2$ | 4 | 56 |

U

| | $A_1$ | $A_2$ | TYPE |
|---|---|---|---|
| $p_1$ | 4 | [13,93] | 1 |
| $t_2$ | 4 | 56 | 0 |

$r_2$

| | $A_1$ | $A_2$ |
|---|---|---|
| $t_1$ | 4 | 53 |
| $t_3$ | 4 | 55 |

**Figure 4.** Table Interpretations in Model *M-1*

M-2 models an environment where errors or incompleteness are introduced at the level of attribute values at a source, before the tuples are inserted into different tables in the database. Hence the same identifier can be attached to the same unknown value wherever it occurs in the database.

**Basic Assumption in Model M-2** *(Attribute Value Level)*:
An unknown value has a unique identifier. Different occurrences of the same unknown value in the database have the same identifier and the same range value. If two identifiers for two unknown values $\tau_1$ and $\tau_2$ are the same then $\tau_1 = \tau_2$; otherwise $\tau_1$ may or may not be equal to $\tau_2$.

In *M-2*, we have the advantage that duplicate unknown tuples can be identified in base tables using identifiers, resulting in inexpensive Union and Difference operations. Clearly, in such an environment, the base tables of the database contain only total and type 1 tuples.

In *M-3*, errors are introduced at the attribute-value level as in *M-2*, but any two different identifiers are known to

represent different values.

**Basic Assumption in Model M-3** (*Attribute Value Level*):
Two identifiers for two unknown values $\tau_1$ and $\tau_2$ are the same iff $\tau_1 = \tau_2$.

*M-4* models an environment where errors are introduced at the **tuple level** before tuple insertions are made into the various tables of the database. A given tuple with unknown values is represented across the database by the same partial tuple. However, identifiers are not attached to unknown attribute values appearing in relations with different schemes. Rather, identifiers are attached to tuples.

**Basic Assumption in Model M-4** (*Tuple Level*):
Two table tuples in different tables represent the same unknown (relation) tuple iff they have the same identifiers.

In each of the models, we extend the five basic relational algebra (RA) operators. As a rule, we retain the semantics of the regular RA operators for total tuples, and extend them for partial tuples. All of the extended operators in our models are *faithful* [Maie 83] in the sense that they reduce to the usual RA operators when the tables consist of only total tuples. We show for each of the models that our extension is *sound* (i.e., no incorrect results are derivable when an RA expression is evaluated) in the Imilienski-Lipski sense [ImiL 84]. However, query evaluation is not always complete; that is, some valid conclusions may not be derivable. In [OlaO 88b], we present a version of the model M-2 in which RA operators are restricted to the cases where the query evaluation is sound and complete.

The rest of the paper is arranged as follows. Section 2 gives the terminology and definitions. Section 3 discusses the correctness notions for the extended models. In section 4, models *M-1* and *M-2* are presented. In section 5, we compare models *M-1* and *M-2* in our family of models with other approaches. Section 6 discusses the common geometric operations needed in RA operator implementations of all the models in the family. We are presently implementing a version of model M-1 in a main-memory database system [OlaO 88a, She 88]. Section 7 briefly discusses the parameters being measured in the implementation.

## 2. Terminology and Definitions

Let *tuples*(p) denote the set of total tuples contained in a partial tuple p, and $t_p$ be the unknown tuple represented by p. $t_{p_i}$ represented by a type 1 tuple $p_i$ is different from any total tuple or $t_{p_j}$, i≠j, for $p_j$ in the same table. Relational algebra operators for tables (as opposed to relations) are superscripted by *.

Tables are denoted as $U_j$'s, and relations are denoted as $r_j$'s. The letters t and p, with or without subscripts, denote total and partial tuples, respectively. The letter z is used to refer to either total or partial tuples. And, the predicate that the tuple z is in table U is written as $T^U(z)$, $T_1^U(z)$, $T_2^U(z)$ or $T_3^U(z)$ when z is total, type 1, type 2 or type 3, respectively. $U_T$, $U_{T_1}$, $U_{T_2}$, $U_{T_3}$, respectively, denote the collection of total, type 1, type 2 or type 3 tuples in table U. Different combinations of the numbers 0, 1, 2 and 3 as subscripts of T refer to a combination of tuple types; for instance, $U_{T_{023}}$ denotes total, or type 2 or type 3 tuples in U.

$U_{T_1}$, $U_{T_2}$, $U_{T_3}$ are *multisets* (i.e., sets with duplicates). Therefore, we use the predicate that two tuples $z_1$ and $z_2$ in a table U with possibly identical tuple components are both "physically" kept in U, written as $z_1 \neq_p z_2$. The usual equality predicate, =, holds (does not hold) between any two distinct tuples with identical (nonidentical) tuple components.

z[Y], the projection of a type 1 or type 2 tuple z on a set of attributes Y, is total, written as T(z[Y]), if there are no unknown values in the Y-values of z. However, z[Y] for a type 3 tuple z is always type 3.

The *intersection of two tuples* $z_1$ and $z_2$, written as $z_1 \cap z_2$, consists of those tuples occurring in both *tuples*($z_1$) and *tuples*($z_2$). $z_1$ and $z_2$ are said to *intersect* if $z_1 \cap z_2 \neq \emptyset$. Given a type 1 tuple $p_1$ in U, the *candidate tuples* for $t_{p_1}$ consists of tuples in *tuples*($p_1$) that are not in $U_T$.

Given tables U and V, the *effective intersection*, written as $p_1 \tilde{\cap} p_2$, between a type 1 tuple $p_1$ in U
(a) and a type 1 tuple $p_2$ in V is the intersection of the candidate tuples of $p_1$ and $p_2$,
(b) and a type 2 or type 3 tuple $p_2$ in V consists of the tuples in *tuples*($p_1$) ∩ *tuples*($p_2$) that are *not* in $U_T$,
(c) and a total tuple t in V is t if t ∈ *tuples*($p_1$) and t is not in $U_T$, otherwise the effective intersection is empty.
The effective intersection of tuples $z_1$ and $z_2$ has the effect of eliminating from the intersection ($z_1 \cap z_2$) those tuples that can not possibly be $t_{z_1}$ or $t_{z_2}$.

The *information content* of a table is defined [ImiL 84] by the mapping *rep* which maps a table U to *rep*(U), the set of possible relations for the unknown relation represented by U. An example of the *rep* mapping for the model *M-1* is given in example 4. Notions similar to *rep* are also used in [Gran 80] and [Bisk 83]. Let r be a relation in *rep*(U) of table U; a tuple z in U has a corresponding tuple, denoted as $(t_z)$r, in r. We call $(t_z)$r the *representative tuple* of z in r, or z is said to be mapped in r to $t_z$.

## 3. Correctness Notion

For each of the models in section 4, the soundness and completeness of a query is examined using revised versions of concepts from [ImiL 84].

**Definition 1** : Let $U_1$ and $U_2$ be two tables. Let α and θ be unary and binary operators, respectively, i.e., α ∈ {π, σ} and θ ∈ {-, ∪, ⋈}. Let α* and θ* be the extended versions of α and θ, respectively.
(a) An extended model is *sound* if, for every relation r in (*rep*($U_1$) θ *rep*($U_2$)) or α (*rep*($U_1$)), there is a relation s in *rep*($U_1$ θ* $U_2$) or *rep*( α* ($U_1$)), respectively, such that s ⊆ r.
(b) An extended model is *complete* if, for every relation s in *rep*($U_1$ θ* $U_2$) or *rep*( α* ($U_1$)), there is a relation r in (*rep*($U_1$) θ *rep*($U_2$)) or α (*rep*($U_1$)), respectively, such that r

$\subseteq$ s.
$rep(U_1)\ \theta\ rep(U_2)$ is understood to be $\{r_1\ \theta\ r_2 \mid r_1 \in rep(U_1)$ and $r_2 \in rep(U_2)\}$, while $\alpha(rep(U)) = \{\alpha(r) \mid r \in rep(U)\}$.

Clearly, in definition 1, $rep(U_1\ \theta^*\ U_2)$ is the "imperfect" representation that attempts to capture the "ideal" representation $rep(U_1)\ \theta\ rep(U_2)$. One may also think of alternative soundness criteria. Let us look at one.

**Definition 2** : An extended model is sound* if, for every relation s in $rep(U_1\ \theta^*\ U_2)$ or $rep\ (\alpha^*(U_1))$, there is a relation r in $(rep(U_1)\ \theta\ rep(U_2))$ or $\alpha\ (rep(U_1))$, respectively, such that $s \subseteq r$.

Definition 2, however, is not acceptable. We give an example.
**Example 3**: Let a given $\alpha$ and $\alpha^*$ be such that $\alpha(rep(U))$ and $rep(\alpha^*(U))$ consist of the relation sets $\{r_1, r_2\}$ and $\{s_1, s_2, s_3\}$ in figure 5 and figure 6, respectively.

$r_1$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
| 6 | 7 |

$r_2$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 4 | 5 |

**Figure 5.** An Example $\alpha(rep(U))$

$s_1$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 3 | 4 |

$s_2$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 3 | 4 |

$s_3$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 6 | 7 |

**Figure 6.** An Example $rep(\alpha^*(U))$

The $\alpha\ (rep(U))$ and $rep(\ \alpha^*\ (U))$ definitions in figures 5 and 6 do satisfy definition 2. One can also observe that when the unknown relation represented by U is $r_1$, all three "imperfect" relations $s_1$, $s_2$, and $s_3$ are contained in the unknown relation $r_1$, which is desirable. However, when the unknown relation represented by U is $r_2$, then none of $s_1$, $s_2$, and $s_3$ is contained in $r_2$, and definition 2 does not detect this problem. In contrast, the soundness criteria of definition 1 is violated in this example.

The condition (a) in definition 1 when considered separately from (b) ensures the soundness of query evaluation. However, consider the case in which $rep(U_1\ \theta^*\ U_2)$ or $rep(\ \alpha^*\ (U))$ contain an empty relation (This may occur, for example, in a set difference operation). In such a case, the soundness criterion of definition 1 will be trivially satisfied. Thus, whenever condition (a) is satisfied for a given extended operator, we have to see to it that the operator definition ensures a *nontrivial soundness*. Therefore, in the rest of the paper, we use the following definition of soundness.

**Definition 3**: An extended model is *sound* if, for every relation r in $(rep(U_1)\ \theta\ rep(U_2))$ or $\alpha\ (rep(U_1))$, there is a relation s in $rep(U_1\ \theta^*\ U_2)$ or $rep(\ \alpha^*\ (U_1))$, respectively, such that s = r. That is, $rep(\alpha^*(U)) \supseteq \alpha(rep(U))$ and $rep(U_1\ \theta^*\ U_2) \supseteq rep(U_1)\ \theta\ rep(U_2)$.

An operation $\alpha^*$ that satisfies definition 3 is said to be *adequate* for $\alpha$ [Maie 83]. Therefore, if each of the extended operator of a given model is adequate, we have a nontrivial soundness (for the query). Thus, we would like every extended operator in our models to be adequate. We can also observe that definition 3 implies condition (a) of definition 1. Let us denote the condition (b) of definition 1 by $rep(U_1)\ \theta\ rep(U_2) \subseteq^* rep(U_1\ \theta^*\ U_2)$.

In the next section, for each binary operator $\theta$ , we examine the relationship that holds between $rep(U_1\ \theta^*\ U_1)$ and $rep(U_1)\ \theta\ rep(U_2)$ with respect to definition 3 and condition (b) of definition 1. Similarly, for each unary operator $\alpha$, the relationship between $rep(\alpha^*(U))$ and $\alpha(rep(U))$ is examined.

## 4. Family of Extended Models

Based on the basic assumptions, we now define the extended model *M-1*. Due to space considerations, we briefly describe *M-2*, and omit the discussions of models *M-3* and *M-4*. The basic assumptions define the *rep* mapping; the mappings in models *M-1* and *M-2* are denoted as $rep_1$ and $rep_2$, respectively. In defining a *rep*, we use the following version of Reiter's [Reit 78] *Closed World Assumption* in [Bisk 83]: If a total tuple t is not in $U_T$ and there is no partial tuple p in U and relation r in $rep(U)$ such that $(t_p)r = t$, then t is not in the unknown relation represented by U.

### 4.1. The Model M-1

In model *M-1* we can not always recognize different occurrences of the same tuple in different tables. The relationship between any two tuples is defined solely by the intersection of their candidate tuples.

**Definition** (*rep* in *M-1*):

$$rep_1(U) = \{r \mid (\forall t)(T^U(t) \rightarrow t \in r)$$

$$\Lambda\ (\forall p)(T_1{}^U(p) \rightarrow (\exists t_1)(t_1 \in tuples(p) \Lambda\ t_1 \in r \Lambda$$

$$(\forall z)((z \in U \Lambda\ z \neq_p p) \rightarrow t_1 \neq (t_z)_r)))$$

$$\Lambda\ (\forall p)(T_2{}^U(p) \rightarrow (\exists t_1)(t_1 \in tuples(p) \Lambda\ t_1 \in r))$$

$$\Lambda\ (\forall p)(T_3{}^U(p) \rightarrow (\exists t_1)((t_1 \in tuples(p) \Lambda\ t_1 \in r)$$

$$\mathrm{v}\ (t_1 = \varnothing \Lambda\ t_1 \notin r)))\}$$

$rep_1(U)$ for a table U is defined independent of other tables in the database. Also, $rep_1(U)$ consists of a finite number of relations because (a) the set of candidate tuples for an unknown tuple is finite, and (b) the Closed World Assumption is used.

**Example 4**: In figure 7, $rep_1(U) = \{r_1, r_2, r_3\}$. In both relations $r_1$ and $r_3$, the type 3 tuple in U is mapped either to the null tuple or to tuple (1 , 2).

In order to define the *Union* and *Difference* operators of the extended relational algebra, we have to answer the question of when a tuple in one table is a duplicate in another one. The following *type-based membership rules* are used for that

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 1 | 2 | 0 |
| 1 | [2,4] | 1 |
| 1 | [2,3] | 3 |

$r_1$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 1 | 3 |

$r_2$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 1 | 4 |
| 1 | 3 |

$r_3$

| $A_1$ | $A_2$ |
|---|---|
| 1 | 2 |
| 1 | 4 |

**Figure 7.** Illustration of $rep_1(U)$

purpose.

*Rule 1*: Given a table U, a total tuple t in table V is
(a) a *duplicate* in U if there is a t′ in $U_T$ such that t=t′,
(b) *not a duplicate* in U if there is no tuple z in U such that z∩t=t,
(c) *may be a duplicate* in U if (a) and (b) do not hold.
*Rule 2*: Given a table U and a type 1 tuple p in table V, the unknown tuple $t_p$ represented by p is
(a) *a duplicate* in U if all the candidates for $t_p$ are total tuples in U,
(b) *not a duplicate* in U if for all tuples z in U, p ∩ z = ∅,
(c) *may be a duplicate* in U if (a) and (b) do not hold.
*Rule 3*: Given a table U and a (type 2 or type 3) tuple p in V, the unknown tuple $t_p$ represented by p is
(a) *a duplicate* in U if all the tuples in *tuples*(p) are in either $U_T$ or $V_T$,
(b) *may be a duplicate* in U if there is a tuple in *tuples*(p) that is not a total tuple in either U or V.

We now discuss the extended versions of the relational algebra operators in the model *M-1*. Formal definitions are given in [OlaO 88b].

### A. Union

The union of two tables U and V, denoted as U ∪* V, consists of
(a) the set of total tuples in either V or U,
(b) type 1 tuples p such that
(i) $T_1^U$(p) holds, and $t_p$ is *not a duplicate* in V, or
(ii) $T_1^V$(p) holds, and $t_p$ is *not a duplicate* in U,
(c) type 2 tuples p such that
(i) $T_1^U$(p) or $T_2^U$(p) hold, and $t_p$ *may be a duplicate* in V,
(ii) $T_1^V$(p) or $T_2^V$(p) hold, and $t_p$ *may be a duplicate* in U,
(d) type 3 tuples p such that $T_3^U$(p) or $T_3^V$(p) hold, and $t_p$ *may be a duplicate* in V or U, respectively.

Please note that the union operator in *M-1* does not migrate tuples into the type 3 class. That is, if a tuple of U or V is not type 3 then it will not be converted into a type 3 tuple in the union.

**Example 5**: Figure 8 illustrates the union operation. In the union, each of the three partial tuples in U and V is changed to a type 2 tuple because we can not express the fact that $t_{p_3}$ may be equal to either $t_{p_1}$ or $t_{p_2}$ but not both. This information loss is one source of incompleteness in query evaluation in model *M-1*.

U

| | $A_1$ | $A_2$ | TYPE |
|---|---|---|---|
| $p_1$ | 2 | [1,5] | 1 |
| $p_2$ | 2 | [2,6] | 1 |
| | 2 | 4 | 0 |

V

| | $A_1$ | $A_2$ | TYPE |
|---|---|---|---|
| $p_3$ | 2 | 4 | 0 |
| | 2 | [3,6] | 1 |

U ∪* V

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 2 | 4 | 0 |
| 2 | [1,5] | 2 |
| 2 | [2,6] | 2 |
| 2 | [3,6] | 2 |

s

| $A_1$ | $A_2$ |
|---|---|
| 2 | 4 |
| 2 | 3 |

**Figure 8.** The Extended Union

**Remark**: There exist tables U and V such that $rep_1(U) \cup rep_1(V) \not\subseteq^* rep_1(U \cup^* V)$.
*Proof*: Using the tables in figure 8, relation s is in $rep_1(U \cup^* V)$, but there are no relations $r_1 \in rep_1(U)$ and $r_2 \in rep_2(V)$ such that $s \supseteq r_1 \cup r_2$ because there are at least three tuples in any $r \in rep_1(U)$. *Q.E.D.*

Union fails to satisfy the completeness because we can not identify those type 1 tuples that originate from the same table and are changed to type 2 in the union.

### B. Difference

The difference of tables U and V, denoted as U −* V, consists of
(a) the set of total tuples t in U such that t is *not a duplicate* in V,
(b) type 1 tuples p such that $T_1^U$(p) holds, and $t_p$ is *not a duplicate* in V,
(c) type 2 tuples p such that $T_2^U$(p) holds, and, for all tuples z in V, z ∩ p = ∅ .
(d) type 3 tuples z such that $T^U{}_{0123}(z)$ holds, and $t_z$ *may be a duplicate* in V.
Please note that some tuples of U that are not type 3 may become type 3 in the difference. That is, difference operation may create a migration of tuples into the type 3 class of tuples.

**Example 6**: Figure 9 illustrates the difference. Again, we do not have any way of specifying that at most one of the two type 1 tuples in U can represent the same unknown tuple as the type 1 tuple in V. Hence both type 1 tuples in U are changed to type 3 in the difference. This example also illustrates the semantic difference between type 2 and type 3 tuples: the type 3 tuples in the difference may not exist; thus, they cannot possibly be type 2 tuples in the difference.

**Remark**: There exist tables U and V such that $rep_1(U) - rep_1(V) \not\subseteq^* rep_1(U -^* V)$.

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 1 | 2 | 0 |
| 2 | [2,4] | 1 |
| 2 | [2,6] | 1 |
| 1 | [2,6] | 2 |

V

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 2 | [2,4] | 1 |

U −* V

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 1 | 2 | 0 |
| 2 | [2,4] | 3 |
| 2 | [2,6] | 3 |
| 1 | [2,6] | 2 |

**Figure 9.** The Extended Difference

## C. Projection

The tables in figure 10 illustrate the projection operation.

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| [2,3] | [2,3] | 1 |
| [2,3] | [2,3] | 1 |
| [2,3] | [2,3] | 1 |
| 5 | [4,7] | 1 |
| 5 | [5,8] | 3 |

$\pi^*_{A_2}$(U)

| $A_2$ | TYPE |
|---|---|
| [2,3] | 2 |
| [2,3] | 2 |
| [2,3] | 2 |
| [4,7] | 1 |
| [5,8] | 3 |

**Figure 10.** The Extended Projection

p[$A_2$], the projection of a type 1 tuple p in table U remains as type 1 in $\pi^*_{A_2}$(U) if p[$A_2$]∩z[$A_2$] = ∅ for all the tuples z in U or whenever p[$A_2$]∩z[$A_2$] ≠ ∅, p and z have equal nonnull $A_1$ values. In figure 10, only the tuple (5 , [4,7]) satisfies this condition. The intervals [4,7] and [5,8] intersect, but the unknown tuples represented by tuples (5 , [4,7]) and (5 , [5,8]) -if they exist- can only have $A_1$ value of 5, and thus different $A_2$ values. Projection on type 2 and type 3 tuples remain as the same type. We now formalize these observations.

Let R(Z) be the scheme for table U such that X ⊂ Z. We say that two tuples $z_1$ and $z_2$ are *X-indistinguishable* if *tuples*($z_1$[X]) = *tuples*($z_2$[X]) and *tuples*($z_1$[X]) has a cardinality of 1.

Please note that a tuple $z_1$ and a type 3 tuple $z_2$ can be X-indistinguishable even though $z_2$[X] is a type 3 tuple (not a total tuple). Also please note that z[Y], the projection of tuple z on the Y attributes, is total (written as T(z[Y])) if z is either total, type 1 or type 2, and z[Y] does not contain unknown values. However, a projection on a type 3 tuple cannot be total.

Let X = (Attributes of U − Y), then the extended projection of U on Y, denoted as $\pi^*_Y$(U), consists of
(a) total tuples z[Y] such that T(z[Y]) holds - z can be total, type 1 or type 2,
(b) type 1 tuples z[Y] such that $T_1^U$(z) holds and whenever $z_1$[Y]∩z[Y] ≠ ∅, $z_1$ ∈ U, z and $z_1$ are X-indistinguishable,
(c) type 2 types z[Y] such that
(i) $T_1^U$(z) holds, z[Y]∩$z_1$[Y] ≠ ∅, and z and $z_1$ are not X-indistinguishable, for a tuple $z_1$ in U, or
(ii) $T_2^U$(z) holds, and there is a tuple in z[Y] which is not in $(\pi^*_Y(U))_T$.
(d) type 3 tuples z[Y] such that $T_3^U$(z) holds, and there is a tuple in z[Y] which is not in $(\pi^*_Y(U))_T$.

**Remark:** There exists a table U such that $rep_1(U)^{\pi_Y} \not\subseteq^* rep_1(\pi^*_Y(U))$, where $rep_1(U)^{\pi_Y}$ = {r | $r_1$ is a relation in $rep_1$(U) and r = $\pi_Y(r_1)$}.

The information loss in the projection operation of *M-1* occurs because the combinatorial implications arising from having finite candidate values for an unknown value are too expensive to be incorporated into the definition of projection operation.

## D. Equi-join

The equi-join is illustrated in figure 11.

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 3 | [2,5] | 1 |
| 2 | 2 | 0 |
| [6,9] | 6 | 2 |

V

| $A_2$ | $A_3$ | TYPE |
|---|---|---|
| 2 | [2,4] | 1 |
| 6 | [5,6] | 2 |
| 2 | 6 | 0 |

U ⋈* V

| $A_1$ | $A_2$ | $A_3$ | TYPE |
|---|---|---|---|
| 3 | 2 | [2,4] | 3 |
| 3 | 2 | 6 | 3 |
| 2 | 2 | [2,4] | 1 |
| 2 | 2 | 6 | 0 |
| [6,9] | 6 | [5,6] | 2 |

S

| $A_1$ | $A_2$ | $A_3$ |
|---|---|---|
| 3 | 2 | 2 |
| 2 | 2 | 3 |
| 6 | 6 | 5 |
| 2 | 2 | 6 |

**Figure 11.** The Extended Equi-Join

In general, let $R_1$(XY) and $R_2$(XZ), X∩Y=∅ and X∩Z=∅, be the schemes for tables U and V, respectively. For any two tuples $z_1$ in U and $z_2$ in V such that T($z_1$[X]) and $z_1$[X]=$z_2$[X] hold, the equi-join U ⋈* V contains $z_1$ concatenated by $z_2$. And whenever $z_1$[X]∩$z_2$[X]≠∅ and at least one of $z_1$[X] and $z_2$[X] is partial, there is a possibility that $t_{z_1}$ and $t_{z_2}$ are "joinable".

Let $z_1$ be in U and $z_2$ be in V. The equi-join of U and V, denoted as U ⋈* V, is defined using $z_1$ and $z_2$ as follows:
(a) If T($z_1$[X]) holds and $z_1$[X]=$z_2$[X], a concatenation of $z_1$ and $z_2$ is made; the resulting tuple of the equi-join is
(i) total if both $z_1$ and $z_2$ are total,
(ii) type 1 if one of $z_1$ and $z_2$ is type 1, and the other is total or type 1,
(iii) type 2 if either of $z_1$ or $z_2$ is type 2.
(b) If at least one of $z_1$[X] and $z_2$[X] is partial (i.e., contains an unknown value) and $z_1$[X]∩$z_2$[X]≠∅; the resulting tuple of the equi-join is type 3.

**Remark:** There exist tables U and V such that $rep_1(U) \bowtie rep_1(V) \not\subseteq^* rep_1(U \bowtie^* V)$.
*Proof:* In figure 11, relation s is in $rep_1(U \bowtie^* V)$, but there are no relations $r_1 \in rep_1(U)$ and $r_2 \in rep_1(V)$ such that $r_1 \bowtie r_2 \subseteq$ s. *Q.E.D.*

## E. Selection

The resulting table when a selection formula F is applied to a table U, denoted as $\sigma^*_F(U)$, consists of

(a) total tuples t such that $T^U$(t)holds, and F(t) holds,

(b) type 1 tuples p such that $T_1{}^U$(p) holds, and all the candidate tuples of p satisfy F,

(c) type 2 tuples p such that $T_2{}^U$(p) holds, and all the tuples in *tuples*(p) satisfy F,

(d) type 3 tuples p such that $T_{123}{}^U$(p)holds, and at least one $t_1$, $t_1 \in$ *tuples*(p), satisfies F, and there is a $t_2 \in$ *tuples*(p) that does not satisfy F.

Figure 12 illustrates the extended selection operation.

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 1 | 2 | 0 |
| 1 | [3,4] | 1 |
| 2 | [2,4] | 1 |
| 2 | [2,5] | 1 |
| [2,5] | 3 | 2 |

$\sigma^*_{A_2>2}(U)$

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 1 | [3,4] | 1 |
| 2 | [2,4] | 3 |
| 2 | [2,5] | 3 |
| [2,5] | 3 | 2 |

s

| $A_1$ | $A_2$ |
|---|---|
| 1 | 3 |
| 2 | 3 |

**Figure 12.** The Extended Selection

**Remark:** There exists a table U such that $\sigma_F(rep_1(U)) \not\subseteq^* rep_1(\sigma^*_F(U))$.

*Proof*: In figure 12, relation s is in $rep_1(\sigma^*_F(U))$, and there is no relation r in $rep_1(U)$ such that $\sigma_F(r) \subseteq s$. *Q.E.D.*

### • Correctness of Query Evaluation in M-1

**Lemma 1**: Let $U$ and $V$ be two tables, and $\alpha$ and $\theta$ be any unary and binary RA operators, respectively, i.e., $\alpha \in \{\pi, \sigma\}$ and $\theta \in \{-, \cup, \bowtie\}$. Let $\alpha^*$ and $\theta^*$ be the extended versions of $\alpha$ and $\theta$ in $M-1$, respectively.

(a) $rep_1(U\ \theta^*\ V) \supseteq rep_1(U)\ \theta\ rep_1(V)$, and

(b)$rep_1(\alpha^*(U)) \supseteq \alpha(rep_1(U))$

**Theorem 1**: Query evaluation in the extended model *M-1* is sound.

*Proof*: Follows from lemma 1, and a straightforward induction on the number of operators of an RA expression.

### 4.2 The Model M-2

In *M-2*, the base tables of the database contain only total and type 1 tuples. However, the projection operation introduces type 2 tuples in model *M-2*. In other words, tuples can not be uniquely identified in intermediate tables resulting from a projection operation (on a base or intermediate table). Selection is defined as in *M-1*; hence type 3 tuples are also introduced in model *M-2*.

The *identifier of a tuple* z in a table with scheme $U(A_1, A_2, ..., A_d)$, denoted by id(z), is the tuple $(i_1, ..., i_d)$ where $i_j$ is $z[A_j]$ if $z[A_j]$ is a known value, otherwise $i_j$ is the identifier for the unknown value $z[A_j]$.

$rep_2$, the *rep* mapping for *M-2*, is similar to $rep_1$ except that the same unknown-value identifier, wherever it occurs in the database, must be mapped to the same value. Hence $rep_2(U)$ is defined relative to the other tables in the database D of tables. Please note that a partial tuple can appear in intermediate tables as a different type tuple; when this is the case, a type 3 tuple in an intermediate relation obtained from a type 1 tuple, for instance, can be mapped to the null tuple. D represents the database of tables.

**Definition** (*rep* in *M-2*):

$$rep_2(U)/D = \{r \mid (r \in rep_1(U) \wedge$$

$$(\forall U_1)(\forall r_1)(\forall z_1)(\forall z)((U_1 \in D \wedge r_1 \in rep_1(U_1)$$

$$\wedge z_1 \in U_1 \wedge z \in U \wedge id(z_1[A_i]) = id(z[A_j]))$$

$$\rightarrow ((((t_{z_1})r_1)[A_i] = ((t_z)r)[A_j] \vee (t_{z_1})r_1 = \varnothing \vee (t_z)r = \varnothing)))$$

$$\}$$

The RA operators of *M-2* are similar to the RA operators of *M-1*, except that we now recognize tuples of any type with the same identifier, and keep only one copy of such tuples in the resulting table. Definitions of the RA operators of M-2 are in [OlaO 88b].

After substituting "M-2" for "M-1", lemma 1 and theorem 1 still hold. That is, similar to *M-1, query evaluation in M-2* is sound, but not complete. A special case of *M-2* for which query evaluation is sound and complete is given in [OlaO 88b].

## 5. Comparison of Models M-1 and M-2 with Other Models

*M-1* is unique in the sense that it models incomplete data environments that can not be handled by models based on Codd tables [Codd 79 and Bisk 83] where unknown values are denoted by a special null symbol, or the models based on V-tables [ImiL 84] where unknown values are represented by variables.

Let us consider relation r in figure 13. In *M-1*, r is represented as table U if the last digit of the $A_2$ value of the tuples in r are unknown.

r

| $A_1$ | $A_2$ |
|---|---|
| 4 | 53 |
| 4 | 56 |
| 5 | 41 |
| 3 | 41 |

U

| $A_1$ | $A_2$ | TYPE |
|---|---|---|
| 4 | [50,59] | 1 |
| 4 | [50,59] | 1 |
| 5 | [40,49] | 1 |
| 3 | [40,49] | 1 |

V

| $A_2$ |
|---|
| [50,59] |
| [50,59] |
| [40,49] |
| [40,49] |

**Figure 13.** Duplicate Partial Tuples in [Gran 80]

The incomplete relation would be represented in [Codd 79 and Bisk 83] by table R in figure 14, where "STATUS=d" denotes definite tuples, and $\omega$ is the null symbol. Only one occurrence of tuple (4 , $\omega$) is stored because one stored tuple *represents* several, but unspecified, number of model (unknown) tuples. In R, we are not able to represent the fact that there are exactly four unknown tuples. This has significant implications, especially when the *COUNT* of the tuples satisfying a given query is important. In *M-1*, using the tuple types, we are able to provide a tighter range for *COUNT*.

R

| $A_1$ | $A_2$ | STATUS |
|---|---|---|
| 4 | ω | d |
| 5 | ω | d |
| 3 | ω | d |

U

| $A_1$ | $A_2$ |
|---|---|
| 4 | $x_1$ |
| 4 | $x_2$ |
| 5 | $x_3$ |
| 3 | $x_3$ |

S

| $A_1$ | $A_2$ |
|---|---|
| 4 | 50 |
| 5 | 42 |
| 3 | 42 |

**Figure 14.** V-Tables and Codd Tables

[Gran 80] allows duplicate partial tuples as in *M-1*. Thus, the incomplete relation r in figure 13 is represented by table U as in *M-1*. However, in [Gran 80], the relationship among the tuples of an intermediate table can not be specified. For example, in figure 13, table V represents the projection of U on attribute $A_2$ in [Gran 80]. There is no way of stating the fact that the two occurrences of tuple ([50,59]) represent different unknown values, while the two occurrences of tuple ([40,49]) may denote the same value.

The model *M-2* is similar to the model based on V-tables in [ImiL 84], where unknown values are represented by distinct variables which take values from infinite domains. In comparison, in *M-2*, the range of an unknown value is finite and the tuple types (in a projection, for instance) distinguish variables that are known to take different values (type 1's) and those that may take the same value (type 2's).

Both Codd tables and V-tables can be used to model the *M-2* environment. Using Codd tables, every unknown value is denoted by a null symbol; but, as stated above, duplicate partial tuples are not allowed. In V-tables, the unknown values are represented by variables. An unknown value takes the same variable name wherever it occurs in the database. For instance, the incomplete relation in figure 13 is represented by the V-table U in figure 14. However, a V-table is not quite the same as a table in *M-2*. The interpretation of a V-table is such that a variable may take any value from an appropriate domain; different variables may take the same value. For example, relation s in figure 14, consisting of only three tuples, is in the set of relations represented by the V-table U.

## 6. Geometric Operations on d-rectangles

We now briefly discuss the geometric operations needed to implement the RA operators of model M-1 (and, also, the models M-2, M-3 and M-4).

A d-dimensional rectilinearly-oriented rectangle (d-rectangle) is the cartesian product of d closed intervals, one on each coordinate axis. A *partial tuple* p in a table of degree d is a set of disjoint d-rectangles in d-space. The *candidate tuples* of p is the set of total tuples contained in p.

Efficient manipulation of d-rectangles in the geometric model is central to the query evaluation process. Geometric operations that have to be performed include
(a) Finding the intersection of d-rectangles (used in all the five basic RA operations).
(b) Finding the complement of a set of d-rectangles (used in the projection operation).
(c) Testing for the containment of a d-rectangle in a union of d-rectangles (used in the selection operation).

Below we discuss these operations.

### 6.1. Intersection of d-rectangles

The intersection of two partial tuples is obtained by intersecting the corresponding d ordered sets (one per tuple component), each of which may have up to $f$ intervals. Since the intersection of two ordered sets of intervals can be determined by comparing the two sets sequentially in $O(f)$ time, the intersection of two partial tuples is performed in $O(df)$ time.

Given a partial tuple p, finding all partial tuples of a table U which intersect with p is also needed in all five RA operators. The most straightforward approach is to intersect p sequentially with each partial tuple of U, which takes $O(ndf)$ time. Using the following result from [EdeM 81, Edel 83], one can give a different time complexity for this task as $O(f \log^d nf + kf)$ where k is the number of partial tuples of U that intersect with p.

Let S be an arbitrary collection of n d-rectangles. Then, there exists a data structure which reports in $O(\log^d n + k)$ time the k d-rectangles intersecting a given d-rectangle. The data structure is a combination of *segment trees* and *range trees*( i.e, a *tree-of-trees* structure). The construction of the data structure requires $O(n\log^{d-1} n)$ space and $O(n\log^d n)$ time. An insertion and deletion of a d-rectangle from S can be accomplished in $O(\log^d n)$.

The disadvantage of the approach in [EdeM 81, Edel 83] is that a complex tree-of-trees structure has to be constructed and maintained separately, which is expensive.

Another question that arises in union, difference and projection operations is to find whether any of the total tuples of a relation (or the candidate tuples of a partial tuple) intersects with (i.e. are contained in) a given partial tuple. The following lemma gives the complexity of such a task.

**Lemma 2:** Whether there is a total tuple in a d-degree relation of size n, sorted on an attribute $A_i$, that intersects with $k_i$ of a partial tuple $p = k_1 x k_2 x \ldots x k_d$ can be determined in O(f logn) comparisons, where f is the degree of fragmentation.

The RA operator evaluations also utilize the counting problem of finding the number of candidate tuples of a partial tuple that intersect with the total tuples of a table. This operation can be implemented using the intersection operation of lemma 2.

### 6.2. Finding the Complement of a d-rectangle

We define the **complement** $C'$ of a given rectangle C as the set of rectangles that contains all the points in $(D_{A_1}, D_{A_2}, \ldots, D_{A_d})$ except those in C. The complement $C_p'$ of the region defined by a partial tuple p can in general result in a set with disjoint $(2f+1)^d - f^d$ d-rectangles. However, lemma 3 states that $C_p'$ can be represented by at most (f+1)d d-rectangles.

**Lemma 3:** Given a partial tuple represented by a set of d-rectangles $C_p$, the *complement* $C_p'$ of $C_p$ can be represented by the union of at most (f+1)d d-rectangles which are not necessarily disjoint.

### 6.3. Testing for Containment

In order to determine if a partial tuple p satisfies a selection formula Q, we test for the containment of $C_p$, the region defined by p, in $C_Q$, the query region defined by Q. For this test, one can use the concept of *d-measure* [Guy 77]. The d-measure of a single rectangle is a numerical value of the product of its intervals. For d=2 and d=3, this corresponds to area and volume, respectively. In finding the measure of a union of rectangles, points appearing in two or more rectangles are "counted" once. Thus, $C_p = C_p \cap C_Q$, and hence $C_p \subseteq C_Q$ holds only if the measure of $C_p$ is the same as the measure of $C_p \cap C_Q$. However, the query region of Q, and consequently $C_p \cap C_Q$, may consist of a union of arbitrary d-rectangles. Finding the measure of n d-rectangles may take up to $0(n^{d-1})$ time [LeeW 81]; this is quite expensive for large d.

For tables with large degrees, Q may be transformed into the Conjunctive Normal Form, and p may be tested iteratively against every conjunct. The details of this approach are in [OlaO 88].

## 7. Implementation Effort

There are two parameters that directly effect the performance of any of the gemetric incomplete database models given in this paper. The first is the **degree of fragmentation f**, i.e., the maximum number of intervals allowed in a tuple component. Clearly, there is a tradeoff between the number of intervals for an unknown value and the cost of operator evaluation. The second parameter is the **percentage of partial tuples in tables.**

In [OlaO88a], for each of the extended operators, an evaluation algorithm is given. The worst-case time costs of the algorithms are analyzed in contrast with the costs of evaluating the usual RA operators in a particular DBMS. For more accurate time cost estimates, the distribution of nulls in partial tuples and the degree of fragmentation must be taken into consideration. We have come to the conclusion that a prototype development is needed to provide a better understanding of the feasibility of the models proposed in this paper.

As a first step in evaluating the performance of the models in this paper, we are implementing a version of model M-1 within a main-memory-only DBMS [She 88]. In our implementation, we are using the basic assumption of model $M-1$, and its type 0 and 1 tuple types. Type 2 and type 3 tuples of $M-1$ are merged into a single tuple type of type 2'. This change simplifies the RA operator definitions of M-1 significantly, and allows for a more efficient implementation. The goals of the implementation are

(a) to obtain empirical information about the effects of the parameters

(i) the degree of fragmentation, and

(ii) the proportion of partial tuples in tables, on the performance of the system, and

(b) to evaluate the feasibility of the models proposed in this paper.

We directly use in the implementation the geometric operations of section 6.

## 8. References


[AbKG 87] Abiteboul, S., Kanellakis, P. and Grahne, G., "On the Representation and Querying of Sets of Possible Worlds", *proc. ACM-SIGMOD conf.*, 1987.

[Bisk 81] Biskup, J., "A Formal Approach to Null Values in Database Relations", in *Advances in Database Theory* (H. Gallaire, J. Minker and J.M. Nicolas, Eds), vol 1, 1981, pp. 299-341.

[Bisk 83] Biskup, J., "Foundations of Codd's Relational Maybe Operations", *ACM TODS* vol 8(4) 1983, pp. 608-636.

[Codd 79] Codd, E.F., "Extending the Database Relational Model to Capture More Meaning", *ACM TODS*, vol 4(4) 1979, pp. 397-434.

[EdeM 81] Edelsbrunner, H. and Maurer, H.A., "On the Intersection of Orthogonal Objects", *Inf. Proc. Letters*, 13, 1981, pp 177-181.

[Edel 83] Edelsbrunner, H., "A New Approach to Rectangle Intersections", *Int. J. of Computer Math.*, vol 13, 1983, pp 209-219.

[Gran 79] Grant, J., "Partial Values in Tabular Database Model", *Inf. Proc. letters*, 9, 3, 1979, pp. 97-99.

[Gran 80] Grant, J., "Incomplete Information in a Relational Database", *Fundamenta Informaticae III.3* (1980), pp. 363-378.

[Guy 77] Guy, R., "Research Problems", in *American Math. Monthly*, 84(4), 1977, pp. 284-285.

[ImiL 84] Imielinski, T. and Lipski, W., "Incomplete Information in Relational Databases", *JACM* vol 31(4) 1984, pp. 761-791.

[LeeW 81] Leeuwen, J and Wood, D., "The measure problem for rectangular ranges in d-space", *Journal of Algorithms*, 2, 1981, pp.282-300.

[Lips 79] Lipski, W., "On semantic Issues Connected with Incomplete Information", *ACM TODS* Sept. 1979, pp. 262-296.

[Maie 83] Maier, D., *The Theory of Relational Databases*, Computer Science Press, 1983.

[OlaO 88a] Ola, A. and Ozsoyoglu, G., "Geometric Modeling of Incomplete Relational Databases", Tech. Report, CWRU, May 1988.

[OlaO 88b] Ola, A. and Ozsoyoglu, G., "A Family of Incomplete Relational Database Models", Tech. Report, CWRU, Aug. 1988.

[Reit 78] Reiter, R., "On Closed World Databases" in *Logic and Databases* (Gallaire and Minker, Eds), Plenum Press, 1978, pp.55-76.

[She 88] Shen, Jun, "A Main-Memory Incomplete Information DBMS and its Performance", MS thesis, CWRU, in progress, 1988.

[SixW 82] Six, H.W. and Wood, D., "Counting and Reporting Intersections of d_Ranges", *IEEE TC*, vol C-31, 3, 1982, pp 181-187.

[Vass 79] Vassiliou, Y., "Null Values in Database Management: A Denotational Semantics Approach", *ACM-SIGMOD* 1979, pp. 162-169.

[Zani 84] Zaniolo, C., "Database Systems With Null Values", *JCSS*, 1984.