

15

Relational meta-answers

Amihai Motro

Computer Science Department, University of Southern California,
University Park, Los Angeles, CA 90089-0782

ABSTRACT

A *meta-answer* is a set of statements that assert various properties of an answer to a database query. These statements may *qualify* the validity or the completeness of the answer, or they may *characterize* the answer abstractly. Meta-answers add meaning and insight to standard database answers. We describe an extension to the relational database model for representing properties of the stored data, and for computing meta-answers from these representations.

1. INTRODUCTION

An important aspect that distinguishes an intelligent database system from a standard database system is the ability to recognize the intentions of users and *cooperate* with them towards realizing these intentions. There are various forms of cooperation, and a rough classification is to distinguish between cooperation that addresses user *errors* and cooperation aimed at *improving* the answers issued by the system. Two examples of the first kind of cooperation are methods for interpreting inputs that are unparseable (due to either syntactic or semantic problems), and methods for recognizing erroneous presuppositions about the domain of discourse (and either correct them or just point them out). Examples of the second kind of cooperation include methods for avoiding empty answers by providing answers to related queries, and methods that provide additional information that is assumed to be of interest to the user.

This work was supported in part by NSF Grant No. IRI-8609912 and by an Amoco Foundation Engineering Faculty Grant.

In the last example, the additional information volunteered by the system may be of two kinds: *data* and *knowledge*. When evaluating queries on a particular topic, the system may broaden them automatically to include additional data. For example, the answer to a bibliographic query on the titles of articles that concern a particular topic might also report articles that concern a related topic, and perhaps also the names of the authors (thus, in terms of the relational model, the answer was broadened to include additional rows, as well as additional columns). Alternatively, the additional information might not be new data, but some useful knowledge about the answer. For example, the answer to the above query might be accompanied by statements that qualify its completeness (e.g., point out that bibliographic information is available only for domestic publications) or characterize it abstractly (e.g., point out that the articles listed were all published prior to 1980).

The topic of this paper is cooperation aimed at improving answers to correct queries, by volunteering additional knowledge about the answer.

This paper is organized as follows. The remainder of this section outlines the general approach and discusses related research. Section 2 describes the view inference problem in relational databases, and explains how this problem can be applied towards our purpose. Section 3 details a solution to the view inference problem, and Section 4 concludes with an example and a brief summary.

1.1. The Approach

Our approach is to assume that information concerning various properties of the stored data is available (meta-information). This information is then used to infer various properties of each answer issued by the system (meta-answer). This approach is implemented by extending the relational model to represent the meta-information in special relations (meta-relations), and by extending the relational algebra operations to meta-relations. When a query is presented to the database, it is performed *both* on the relations, resulting in an answer, and on the meta-relations, resulting in a meta-answer.

The meta-information includes statements of three kinds:

1. Constraints: definitions of conditions that are satisfied by the stored data.
2. Validity: definitions of subsets of the database that are guaranteed to include only valid information.
3. Completeness: definitions of subsets of the database that are guaranteed to include complete information.

The first kind of meta-information corresponds to standard integrity constraints, which are formulas in predicate logic that assert required relationships among the data values. The other two kinds of meta-

information relate the contents of the stored data to the "real world".

Meta-answers include statements of four kinds:

1. Constraints: definitions of conditions that are satisfied by the answer.
2. Containments: definitions of subsets of the database that are contained in the answer in their entirety.
3. Validity: definitions of subsets of the answer that are guaranteed to be valid.
4. Completeness: definitions of subsets of the answer that are guaranteed to be complete.

As an informal example assume a bibliography database of journal articles, and assume that its meta-information states that every article published in ACM journals is included (completeness), that the accuracy of all articles published since 1980 has been verified (validity), that articles published in ACM journals prior to 1970 are available only on film (constraint), and that the author of all articles from Mountebank University is always Charl A. Tan (constraint). Consider now a query about the articles by Charl A. Tan that appeared in ACM journals. The answer issued by the database will be accompanied by a meta-answer stating that

1. The answer is certified to include all such articles (completeness).
2. The answer is certified to be accurate only for articles published since 1980 (validity).
3. The articles in the answer that were published prior to 1970 are available only on film (constraint).
4. The answer contains all the articles ever written by Mountebank University authors (containment).

Such a meta-answer provides assurances on the quality of the answer, as well as additional insight into its properties.

1.2. Related Work

The work described here is based on two previous works in the areas of database integrity (Motro, 1989a) and intensional answers (Motro, 1989b). We review these areas briefly with emphasis on these two works.

The primary concern of users of any information system is the *integrity* of its answers. This concern may be divided into two parts: (1) Is the answer valid? and (2) Is it answer complete? For example, when a user consults a bibliographic database for articles by Charl A. Tan, he is concerned with the validity of the information he receives (is the information provided on each article correct?), as well as about its completeness (are there any other articles by this particular author

that were not listed?). Hence, answers have integrity, if they contain *the whole truth* (completeness) and *nothing but the truth* (validity).

Until recently, all efforts to enhance the integrity of relational databases have been within the framework of *integrity constraints*. In general, integrity constraints are formulas in predicate calculus that express relationships that must be satisfied by the database (Date, 1986). Assuming that initially a database satisfies the constraints, thereafter update requests are accepted only if they do not violate any of the constraints. Such constraints *enhance* the integrity of a database (and hence the integrity of its answers), but they cannot *ensure* it. Thus, in our previous example, it is not possible to express integrity constraints that ensure that the information about articles published since 1980 is valid, or that all articles in ACM journals are included.

A new model of integrity was then introduced by this author (Motro, 1989a). This model is based on new kinds of integrity constraints, called *validity constraints* and *completeness constraints*. A validity constraint asserts that a particular subset of the database is guaranteed to be valid, and a completeness constraint asserts that a particular subset of the database is guaranteed to be complete. The validity and completeness constraints are then used to infer the validity and completeness of each answer issued by the database.

The concept of information validity is related to the "standard" integrity constraints. Indeed, standard integrity constraints are a specific kind of validity constraints (Motro, 1989a). The concept of information completeness is related to the *Closed World Assumption (CWA)* (Reiter, 1978). Under this assumption, a database contains all the occurrences of data that it attempts to model. More accurately, the CWA states that if a fact is not included in the database (and cannot be inferred from it), then it is false. The CWA is usually made on the database *as a whole*. In practice, however, this assumption is not realistic, as most databases include at least some information that is possibly incomplete. In other words, in reality the CWA can only be made on some *subsets* of the database.

An *intensional* answer to a query is a set of characterizations of the set of database values that satisfy the query (the *extensional* answer). Intensional answers provide users with additional insight into the nature of standard extensional answers. Recently, several research contributions have been concerned such issues. While each adopts a different approach, all share a common goal, which is to answer queries more abstractly.

Assuming a logic-based data model, Cholvy and Demolombe (1986) are interested in providing answers to queries that are independent of a particular set of facts; i.e., answers that are derived only from the rules. The authors define a constructive derivation process that computes such answers. One of the problems they consider is how to avoid

answers that are "irrelevant". Pirotte and Roelants (1989) follow this general approach, but distinguish between two kinds of intensional information: inference rules and integrity constraints. They then show how integrity constraints can be used to filter out inadequate intensional statements from the answer, producing simpler and more informative intensional answers. Also in the environment of logic databases, Imielinski (1987) argues that rules should be allowed to occur in the answer to the query. This is shown to be beneficial both from the conceptual and the computational point of view. As an example, assume database predicates $\text{Teach}(x,y)$, denoting that professor x can teach course y , and $\text{Group}(x,z)$, denoting that professors x and z belong to the same research group. And assume a rule that requires that professors from the same group be able to teach the same courses. Then a query "Who can teach the Database course" may be answered intensionally by "Everybody in Smith's group". Exhaustive enumeration of this answer will be performed upon request. Corella (1984) notes that while research on knowledge representation produced much work on the derivation of taxonomies of concepts, at times, concepts are also essential in responses to queries. The author defines a formal model of retrieval based using taxonomies of concepts. Roughly, a concept is a unary predicate over a given domain, and a taxonomy is a finite tree of concepts, where the concept described by each node is subsumed by the concept described by its parent, and the union of sibling concepts is equal to their parent concept. Shum and Muntz (1988a, 1988b) also note that answers that are exhaustive enumerations of individual objects are not always the most efficient or the most effective means of information exchange. In the first paper they are concerned with implicit representation of answers through concise expressions that involve both concepts and individuals. For example, an acceptable answer to the query "Who earns more than \$30,000?" is "All engineers except Smith". In the second paper they are concerned with providing aggregate responses, where preciseness is sacrificed for conciseness. An example would be an answer such as "90/120 engineers + 20/30 managers". In both cases they assume, like Corella, the availability of taxonomies that encompass all concepts and individuals. Finally, this author (Motro, 1989b) describes a model in which standard integrity constraints are used to derive intensional answers of two kinds, called *constraints* and *containments*. A constraint defines a condition that is satisfied by the entire answer. A containment defines a subset of the database that is contained in its entirety in the answer.

2. THE VIEW INFERENCE PROBLEM

2.1. Preliminaries

We assume the following definition of a relational database (Maier, 1983). A *relation scheme* R is a finite set of *attributes* A_1, \dots, A_m . With each attribute A_i a set of values D_i , called the *domain* of A_i , is associated (domains are non-empty, finite or countably infinite sets). A *relation* on the relation scheme R is a subset of the product of the domains associated with the attributes of R . A *database scheme* \mathcal{R} is a set of relation schemes R_1, \dots, R_n . A database instance D of the database scheme \mathcal{R} is a set of relations $R_1(D), \dots, R_n(D)$, where each $R_i(D)$ is a relation on the relation scheme R_i .

A *view* V is an expression in the relation schemes of \mathcal{R} that defines a new relation scheme, and for each database instance D defines a unique relation on this scheme denoted $V(D)$. In this paper we consider views that are defined by *conjunctive relational calculus expressions* (Ullman, 1982). Using domain relational calculus, expressions from this family have the form:

$$\{a_1, \dots, a_n \mid (\exists b_1) \dots (\exists b_k) \psi_1 \wedge \dots \wedge \psi_m\}$$

Where the ψ s may be of two kinds:

1. **membership:** $(c_1, \dots, c_p) \in R$, where R is a database relation (of arity p), and the c s are either a s or b s or constants.
2. **comparative:** $d_1 \theta d_2$, where d_1 is either an a or a b , d_2 is either an a or a b or a constant, and θ is a comparator (e.g., $<$, \leq , $>$, \geq , $=$, \neq).

In particular, each a and each b must appear at least once among the c s.

We refer to such views as *conjunctive views*. While this family is a strict subset of the relational calculus, it is a powerful subset. The family of conjunctive relational calculus expressions is precisely the family of relational algebra expressions with the operations *product*, *selection* and *projection* (where the selection expressions are conjunctive). The attributes that participate in the selection predicate will be called *selection attributes*, and the attributes that participate in the projection will be called *projection attributes*.

Views may have a particular *property*. In this paper we consider only properties that are *preserved* by conjunctive views. That is, if a set of views has a property, then any conjunctive view of these views will retain this property.

2.2. The Problem

Consider this general problem in relational databases: Given a query and a set of database views that possess a particular property, what views of the answer possess this property? This situation is shown in Figure 1: R_1, \dots, R_n are the database relations, V_1, \dots, V_m are the views with a property, and Q is the query. Recall that we assume that views of V_1, \dots, V_m will also have the property. Therefore, every view of Q which is also a view of V_1, \dots, V_m is a view of the answer that has the property. A is such a view.

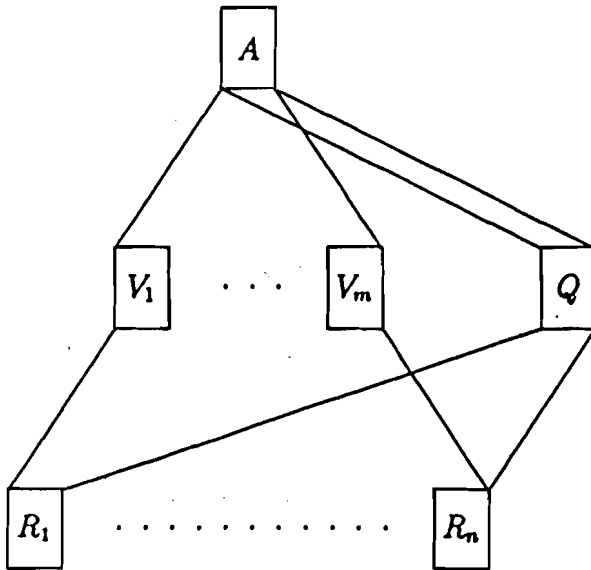


Figure 1: A General Problem

2.3. Applications

By selecting the particular property possessed by the views, different applications can be derived. In this paper we discuss three particular view properties: validity, completeness and emptiness. A fourth property, with application to database access authorization, was described elsewhere (Motro, 1989c).

The application of the view inference problem to the properties of validity and completeness is straightforward. Given a set of views that are asserted to contain only valid information, the system will infer for each answer it issues views that are guaranteed to be valid. Similarly, given a set of views that are asserted to include complete information,

the system will infer for each answer it issues views that are guaranteed to be complete.

The property of emptiness expresses standard database integrity constraints. Assume integrity constraints of the form "for all \vec{x} , if $\alpha(\vec{x})$ then $\beta(\vec{x})$ "¹. This statement may be rewritten as "the set of values \vec{x} for which $\alpha(\vec{x})$ is true and $\beta(\vec{x})$ is false is empty. Therefore, constraints may be expressed as database views whose property is that they are always empty. Given a set of views that that are asserted to be empty, the system will infer for each answer it issues views of the answer that must be empty; that is, integrity constraints that hold over the answer.

These three properties correspond to three of the four kinds of meta-answers discussed in Section 1.1. The fourth kind, containments, will be discussed later.

3. META-PROCESSING

3.1. The Approach

Our solution to the view inference problem is to represent the definitions of the given views in special relations, called meta-relations, whose structure mirrors the actual relation. Standard algebraic operations (product, selection and projection) are extended to these meta-relations.

When a query is presented to the database system, it is performed *both* on the actual relations, resulting in an answer, and on the meta-relations, resulting in definitions of views of the answer that inherit the particular property of the given views.

This approach is illustrated by the commutative diagram shown in Figure 2. The horizontal lines describe the relationships between meta-relations and relations, and the vertical lines describe query processing and meta-processing. The solid lines describe the current situation: the meta-relations R' define views of the database relations R that have a given property, and the relation A is derived from R to answer query Q . The dashed lines describe our method: query processing is extended to manipulate also R' , to yield the meta-relation A' , that defines views of answer A with the given property.

3.2. Meta-Relations

The representation of conjunctive views in meta-relations recalls the representation of QBE queries in skeleton tables (Zloof, 1977).

¹ α and β are safe relational calculus expressions with free variables \vec{x} .

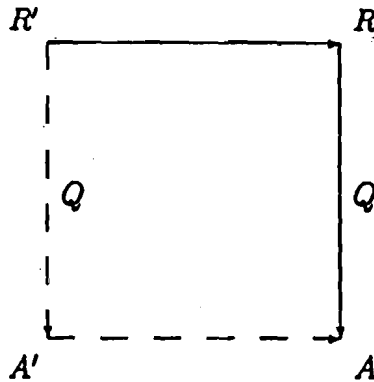


Figure 2: Meta-processing

For each relation R a *meta-relation* R' is defined. The scheme of R' is identical to the scheme of R , except for an additional attribute called PROPERTY. Also, an auxiliary relation is defined: COMPARE = $(X, \text{COMPARE}, Y)$. The meta-relations will be used to store membership formulas of views. Their tuples will be referred to as *meta-tuples*. Comparative formulas will be stored in the relation COMPARE.

Consider a view

$$\{(a_1, \dots, a_n) \mid (\exists b_1) \dots (\exists b_m) \psi_1 \wedge \dots \wedge \psi_k\}$$

A formula ψ of the kind $(c_1, \dots, c_p) \in R$ is first modified so that the cs that are as are suffixed with $*$, and the cs that are variables (i.e., as or bs) that appear only once in the whole expression are replaced with \sqcup (blank). Hence, each component of the modified formula is either a constant, or a variable, or a blank, and each may be suffixed by $*$. This tuple is prefixed with a value that indicates the property of the corresponding view, and is stored in R' . A formula ψ of the kind $d_1 \theta d_2$, where θ is not $=$, is transformed to the tuple (d_1, θ, d_2) and stored in the auxiliary relation COMPARE. If θ is $=$, then all occurrences of d_1 in the other formulas are substituted with d_2 . Finally, we assume that variable names are not shared among views.

For our examples, we assume a simple bibliographic database, whose scheme is shown in Figure 3. The semantics of ARTICLE.AUTHOR and AUTHOR.NAME are similar, and the semantics of ARTICLE.JOURNAL and JOURNAL.NAME are similar. The attribute MEDIUM takes values F (film only), P (paper only), and B (both). In the following examples, the values V, C, and E are used, respectively, to designate the properties of validity, completeness and emptiness. These examples correspond to the meta-information in the intuitive example presented in Section 1.1.

ARTICLE	=	(TITLE,AUTHOR,JOURNAL,YEAR, MEDIUM)
AUTHOR	=	(NAME,AFFILIATION)
JOURNAL	=	(NAME,PUBLISHER)

Figure 3: Scheme of a bibliographic database

Let M1 be a completeness view describing the title and author of articles published in ACM journals:

$$\{a_1, a_2 \mid (\exists b_1)(\exists b_2)(\exists b_3) \\ (a_1, a_2, b_1, b_2, b_3) \in \text{ARTICLE} \wedge (b_1, \text{ACM}) \in \text{JOURNAL}\}$$

M1 is represented with two meta-tuples:

$$(C, *, *, x_1, \perp, \perp) \in \text{ARTICLE}' \\ (C, x_1, \text{ACM}) \in \text{JOURNAL}'$$

Let M2 be a validity view describing the title, author and year of articles published since 1980:

$$\{a_1, a_2, a_3 \mid (\exists b_1)(\exists b_2) \\ (a_1, a_2, b_1, a_3, b_2) \in \text{ARTICLE} \wedge a_3 \geq 1980\}$$

M2 is represented with two meta-tuples:

$$(V, *, *, \perp, x_2, \perp) \in \text{ARTICLE}' \\ (x_2, \geq, 1980) \in \text{COMPARISON}$$

Let M3 be an empty view describing the titles of articles published in ACM journals prior to 1970 and not available only on film (i.e., articles published in ACM journals prior to 1970 are available only on film):

$$\{a \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4) \\ (a, b_1, b_2, b_3, b_4) \in \text{ARTICLE} \wedge (b_2, \text{ACM}) \in \text{JOURNAL} \wedge \\ b_3 < 1970 \wedge b_4 \neq F\}$$

M3 is represented with four meta-tuples:

$$(E, *, \perp, x_3, x_4, x_5) \in \text{ARTICLE}' \\ (E, x_3, \text{ACM}) \in \text{JOURNAL}' \\ (x_4, <, 1970) \in \text{COMPARISON} \\ (x_5, \neq, F) \in \text{COMPARISON}$$

Let M_4 be an empty view describing the titles of articles from Mountebank University by authors other than Charl A. Tan (i.e., Charles A. Tan is the author of all articles from Mountebank University):

$$\{a \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4) \\ (a, b_1, b_2, b_3, b_4) \in \text{ARTICLE} \wedge (b_1, \text{MBU}) \in \text{AUTHOR} \wedge b_1 \neq \text{CAT}\}$$

M_4 is represented with three meta-tuples:

$$(E, *, x_6, \sqcup, \sqcup, \sqcup) \in \text{ARTICLE}' \\ (E, x_6, \text{MBU}) \in \text{AUTHOR}' \\ (x_6, \neq, \text{CAT}) \in \text{COMPARISON}$$

3.3. Meta-Operations

Definition 1: Assume that R' and S' are meta-relations that define views of R and S . The product of R' and S' , denoted $R' \times S'$, is defined as follows. For every pair r and s of meta-tuples (having the same property P) from R' and S' , respectively,

$$r = (P, r_1, \dots, r_m) \\ s = (P, s_1, \dots, s_n)$$

$R' \times S'$ includes the meta-tuple:

$$q = (P, r_1, \dots, r_m, s_1, \dots, s_n)$$

Definition 2: Assume that R' is a meta-relation that defines views of R . Let λ denote a primitive selection predicate (i.e., either $R' \theta c$, or $R' \theta R'_j$). The selection from R' by predicate λ , denoted $\sigma_\lambda(R')$, is defined as follows. Consider first the case $\lambda = R' \theta c$, and let r be a meta-tuple from R' ,

$$r = (P, r_1, \dots, r_i, \dots, r_m)$$

Denote by μ the selection predicate expressed by r_i ². If r_i is suffixed by $*$, then $\sigma_\lambda(R')$ includes the meta-tuple:

$$q = (P, r_1, \dots, r'_i, \dots, r_m)$$

where r'_i represents $\lambda \wedge \mu$. Consider now the case $\lambda = R' \theta R'_j$, and let r be a meta-tuple from R' ,

$$r = (P, r_1, \dots, r_i, \dots, r_j, \dots, r_m)$$

²If r_i is blank, then μ is true.

Denote by μ the selection predicate expressed by r_i and r_j . If r_i and r_j are both suffixed by $*$, then $\sigma_\lambda(R')$ includes the meta-tuple:

$$q = (P, r_1, \dots, r'_i, \dots, r'_j, \dots, r_m)$$

where r'_i and r'_j represent $\lambda \wedge \mu$.

Definition 3: Assume that R' is a meta-relation that defines views of R . The projection of R' that removes its i 'th attribute, denoted $\pi_{R-R_i}(R')$, is defined as follows. For every meta-tuple r from R' ,

$$r = (P, r_1, \dots, r_m)$$

If r_i is \sqcup (possibly suffixed with $*$), then $\pi_{R-R_i}(R')$ includes the meta-tuple:

$$q = (P, r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_m)$$

Intuitively, the meta-product is the product of the meta-relations, with the restriction that only meta-tuples with the same property are matched. Meta-selection selects a meta-tuple only if the selection attribute is in the projected attributes (the meta-tuple is modified to incorporate the selection condition). Meta-projection retains a meta-tuple only if the projection attribute is not in the selection attributes (the meta-tuple is modified to remove the projection attribute).

3.4. Discussion

These definitions were shown (Motro, 1989a) to be "correct", in the sense that the meta-product defines views that would be obtained by applying the product to the views defined in the original meta-relations³; the meta-selection defines views that would be obtained by applying the selection to the views defined in the original meta-relation; the meta-projection defines view that would be obtained by applying the projection to the views defined in the original meta-relation.

Formally, let D be an instance of this database, let r , s , and q be as in these definitions, and let $r(D)$, $s(D)$, and $q(D)$ denote, respectively, the relations defined by the meta-tuples r , s and q . Then

$$\begin{aligned} \text{product :} & \quad q(D) = r(D) \times s(D). \\ \text{selection :} & \quad q(D) = \sigma_\lambda r(D). \\ \text{projection :} & \quad q(D) = \pi_{R-R_i}(r(D)). \end{aligned}$$

³Recall that the meta-product does not match views with different properties.

Also, these meta-operations were shown (Motro, 1989a, 1989b) to *preserve* each of the view properties that are relevant to this paper (i.e., completeness, validity, emptiness).

Formally, let D be an instance of this database, let r , s , and q be as in these definitions, and let $r(D)$, $s(D)$, and $q(D)$ denote, respectively, the relations defined by the meta-tuples r , s and q . Then

product: if $r(D)$ and $s(D)$ have a property, then $q(D)$ has this property.

selection: if $r(D)$ has a property, then $q(D)$ has this property.

projection: if $r(D)$ has a property, then $q(D)$ has this property.

These two results may be summarized as follows. Assume a database scheme \mathcal{R} and views \mathcal{V} with a property. Let Q be a conjunctive query against this database. Let S be the relational algebra expression that implements Q . Let S' be the relational algebra expression obtained from S by substituting every reference to R with a reference to R' . S operates on the actual database relations to yield the answer A . S' operates on the meta-relations to yield the meta-relation A' of views of A . Then, the views in A' possess the same property.

This result guarantees that meta-tuples in A' are views of A that have the property. However, some meta-tuples may still share variables with meta-tuples outside A' , and are therefore not expressible entirely within A' . Such views are avoided if S' is modified so that all products are performed first, and their result is pruned to retain only those meta-tuples that do not share variables with other meta-tuples. Also, it is advantageous to perform selections before projections (Motro, 1989a). Altogether, S' is transformed to a sequence of products, followed by selections, and ending with projections. This simple strategy for implementing conjunctive queries is not necessarily the most efficient. However, we note that the efficiency is not so essential for meta-relations, because they are relatively small. For the actual relations, where efficiency is essential, a different strategy may be implemented.

The result guarantees that the method for generating integrity constraints is *sound*, but it does not guarantee that it is *complete*. That is, this method generates views of the result that indeed have the property, but does not necessarily generate all such views. A method that would guarantee completeness would undoubtedly be of a different complexity altogether. However, for our purpose here, to provide knowledge about database answers, completeness is not an absolute requirement. Yet, several simple refinements were developed, that generate additional desirable views (Motro, 1989a).

Meta-processing uses validity constraints, completeness constraints, and standard constraints which are stated on the database (i.e., views of the database that are valid, complete or empty) to infer validity con-

straints, completeness constraints and standard constraints that may be stated on the answer. In Section 1.1 we mentioned a fourth type of inferred information, called containments. These are definitions of views of the database that are contained in the answer in their entirety. The process by which these views are derived is a simple extension of meta-processing (Motro, 1989a).

4. CONCLUSION

4.1. Example

In the database described in Section 3.2 consider the following query to retrieve the titles of articles by Charles A. Tan that appeared in ACM journals. This query is defined in the following view:

$$\{a \mid (\exists b_1)(\exists b_2)(\exists b_3) \\ (a, \text{CAT}, b_1, b_2, b_3) \in \text{ARTICLE} \wedge (b_1, \text{ACM}) \in \text{JOURNAL}\}$$

The meta-processing of this query will result in a meta-answer with four meta-tuples, whose interpretation is that the accompanying answer is certified to include all such articles; that it is certified to be accurate only for articles published since 1980; that the listed articles published prior to 1970 are available only on film; and that the answer contains all the articles ever written by Mountebank University authors ⁴.

4.2. Summary

We presented an extension to the relational database model, in which properties of the stored data (meta-information) are stored in a meta-database, and each standard query is also processed in the meta-database to derive meta-information that is applicable to the standard answer (meta-answer).

This model is currently being implemented as a "front-end" interface to a standard relational database system. In this system, called Panorama, users will define meta-information with appropriate view statements and the system will insert automatically the appropriate meta-tuples into the meta-relations. In response to a query statement, the user will receive the usual extensional answer, accompanied by inferred meta-information in the form of view statements. Thus, the

⁴The meta-processing of this query involved several minor refinements that were not discussed in this paper.

meta-relations and meta-tuple notation would be completely transparent, with all user-system communication done with customary query language statements.

Various problems need further investigation and we mention here two examples. Currently, the methods handle only conjunctive views (meta-information and queries). Work is underway on extensions that will handle more general views; for example, views with disjunctions and views with aggregate functions. A difficult problem is how to identify the meta-information that is *relevant* to a query. For example, consider a query to list all the articles by Charles A. Tan. While it is true that this set of articles satisfies the constraint that those published in ACM journals prior to 1970 are available only on film, this information may be irrelevant. These issues were discussed further in two previous papers (Motro, 1989a, 1989b).

REFERENCES

- Cholvy, L. and Demolombe, R. (1986). Querying a rule base. In *Proceedings of the First International Conference on Expert Database Systems* (Charleston, South Carolina, April 1-4), pages 365-371.
- Corella, F. (1984). Semantic retrieval and levels of abstraction. In *Proceedings of the First International Workshop on Expert Database Systems* (Kiawah Island, South Carolina, October 24-27), pages 91-114.
- Date, C. J. (1986). *An Introduction to Database Systems, Volume I (Fourth Edition)*. Addison Wesley, Reading, Massachusetts.
- Imielinski, T. (1987). Intelligent query answering in rule based systems. *Journal of Logic Programming*, 4(3):229-257.
- Maier, D. (1983). *The Theory of Relational Databases*. Computer Science Press, Rockville, Maryland.
- Motro, A. (1989a). Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480-502.
- Motro, A. (1989b). Using integrity constraints to provide intensional responses to relational queries. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases* (Amsterdam, The Netherlands, August 22-25), pages 237-246.
- Motro, A. (1989c). An access authorization model for relational databases based on algebraic manipulation of view definitions. In *Proceedings of the IEEE Computer Society Fifth International Conference on Data Engineering* (Los Angeles, California, February 6-10), pages 339-347.
- Pirotte, A. and Roelants, D. (1989). Constraints for improving the generation of intensional answers in a deductive database. In *Proceedings of the IEEE Computer Society Fifth International*

- Conference on Data Engineering* (Los Angeles, California, February 6–10), pages 652–659.
- Reiter, R. (1978). On closed world data bases. In *Logic and Databases*, pages 55–76, Plenum Press, New York, New York.
- Shum, C. D. and Muntz, R. (1988a). Implicit representation for extensional answers. In *Proceedings of the Second International Conference on Expert Database Systems* (Tysons Corner, Virginia, April 25–27), pages 257–273.
- Shum, C. D. and Muntz, R. (1988b). An information-theoretic study on aggregate responses. In *Proceedings of the Fourteenth International Conference on Very Large Data Bases* (Los Angeles, California, August 25–28), pages 479–490.
- Ullman, J. D. (1982). *Principles of Database Systems*. Computer Science Press, Rockville, Maryland.
- Zloof, M. (1977). Query-by-Example: a database language. *IBM Systems Journal*, 16(4):324–343.