The Problem of Optimal Approximation of Queries Using Views, and its Applications [*]

Alexander Brodsky and Amihai Motro
Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444

Technical Report ISSE-TR-95-104

May 1995

Abstract

In several recent works a common scenario is described: a relational database scheme $\mathcal{R}$ and a set of views $\mathcal{V}$ of this scheme are defined, and queries on the database scheme $\mathcal{R}$ must be transformed to equivalent queries on the views $\mathcal{V}$. Given a query Q on $\mathcal{R}$, the first problem is whether there exists a query $Q_{\mathcal{V}}$ expressed exclusively on $\mathcal{V}$ which is equivalent to Q, and, if it exists, how to find it. Clearly, an equivalent $Q_{\mathcal{V}}$ may not always exist. In such a case, it is important to *approximate* Q as best as possible with a query on $\mathcal{V}$. The problems here are to define approximations, to determine whether a best approximation exists, whether it is unique, and how to find it. In this paper we formalize and answer these questions. Specifically, we show that the following problems are decidable for a conjunctive query Q and a set of conjunctive views $\mathcal{V}$: (1) is there a conjunctive query $Q_{\mathcal{V}}$ on $\mathcal{V}$ that is equivalent to Q, and (2) is there a union Q_U of conjunctive queries on $\mathcal{V}$ that is equivalent to Q? Furthermore, we show that $Q_{\mathcal{V}}$ and Q_U are effectively computable if they exist. Moreover, the greatest lower bound of Q exists and is unique up to equivalence. This is done by introducing lower bound classifications, and proving their existence, finiteness and effective computability. This problem has many interesting applications, including physical database design (aimed at optimization of query processing), cooperative answering (where answers are annotated with some of their properties), and multidatabase query decomposition. We examine these applications, and we show how our work extends earlier results obtained in these applications.

[*]The work of Brodsky was supported in part by NSF Research Initiation Award (1994) No. 92-122, and by Office of Naval Research under prime grant No. N00014-94-1-1153. The work of Motro was supported in part by ARPA grant, administered by the Office of Naval Research under Grant No. N0014-92-J-4038.

1 Introduction

In several recent works a common scenario is described: a relational database scheme and a set of views of this scheme are defined, and queries on the database scheme must be transformed to equivalent queries on the views.

This problem has many interesting applications, including physical database design (aimed at optimization of query processing), cooperative answering (where answers are annotated with some of their properties), and multidatabase query decomposition. These applications are described in more detail in Section 2.

The basic problem is formalized simply as follows. Let $\mathcal{R} = \{R_1, \ldots, R_n\}$ denote a set of relation definitions (a *database scheme*), and let $\mathcal{V} = \{V_1, \ldots, V_m\}$ denote a set of views of these relations (a *view scheme*). Transform a given query Q on the database scheme to an equivalent query $Q_\mathcal{V}$ on the view scheme.

Obviously, not all queries Q can be transformed in this way. As a trivial example, assume a database with relations $R = (AB)$ and $S = (BC)$, and a single view $V = R \bowtie S$. Consider the query $Q = \sigma_{A=a} R$. Clearly, Q cannot be answered from the view V, because the join may have removed some of R's tuples. In situations where a query cannot be transformed, an issue of great importance is how well can Q can be *approximated*; i.e., what is the "best estimate" for Q that can be evaluated from the views?

Following the terminology of [6], every query that is contained in Q is a *sound approximation* of Q: it includes only data from Q, but possibly not all of it; and every query that contains Q is a *complete approximation* of Q: it includes all the data of Q, but possibly some other data as well. Thus, the answer to Q is "sandwiched" between a sound approximation and a complete approximation. Clearly, it is desirable to find the *greatest sound approximation (GSA)* and the *least complete approximation (LCA).* [1]

We consider views and queries that are *conjunctive*. The family of conjunctive queries has the same power as the family of relational algebra queries with the operators Cartesian product, selection (where the selection formula is a conjunction of attribute-attribute or attribute-constant comparisons) and projections. Section 3 reviews basic definitions and results concerning conjunctive queries; in particular, query containment and query equivalence.

First, we adapt a known result on the containment of conjunctive queries to queries on views, providing an effective method for determining whether one such query is contained in another.

Using lower bound concepts, we then show that the following problems are decidable for a conjunctive query Q and a set of conjunctive views $\mathcal{V}$: (1) is there a conjunctive query $Q_\mathcal{V}$ on $\mathcal{V}$ that is equivalent to Q, and (2) is there a union Q_U of conjunctive queries on $\mathcal{V}$ that is

[1] Of course, the term *least complete approximation* refers to the least approximation which is complete, not to the approximation which is the least complete!

equivalent to Q? Furthermore, we show that $Q_{\mathcal{V}}$ and Q_U are effectively computable if they exist. Moreover, the greatest lower bound of Q exists and is unique up to equivalence. This done by introducing lower bound classifications, and proving their existence, finiteness and effective computability.

These results guarantee that we can either translate Q to $Q_{\mathcal{V}}$, or find the "best" approximation for Q using $\mathcal{V}$.

When resorting to approximations, it is also important to consider answers that are incomplete because they are missing some of the attributes of Q. That is, partial answers should also include any projections of Q that are expressible in $\mathcal{V}$. This may be done by applying the above results for arbitrary projections of Q.

These results, described in Sections 4 and 5, allow us to extend the results that have been previously obtained in each of the aforementioned applications (i.e., physical database design, cooperative answering, and multidatabases). These extensions are described in the next section, which is dedicated to a review of these applications.

Section 6 concludes this paper with a summary of the contributions of this work and the issues that are still under investigation.

2 Applications

The problem of transforming a query on a given set of relations to a query on a given set of views of these relations has been encountered in at least three applications. In this section we describe these applications and we comment on the solutions that have been designed.

2.1 Physical Database Design

The first to have addressed this problem have been Larson and Yang [4, 5]. The context in which the problem was discussed is *physical database design* for relational databases. The standard approach to physical database design is to maintain a one-to-one correspondence between the relations declared in the scheme of the database and the relations that are actually stored. The authors point out that for reasons of performance it may advantageous to forgo this correspondence, and design physical relations that correspond to the actual queries that the database will process. Queries and updates on the database scheme must then be transformed to queries and updates on the stored relations.

A variant of this problem appears in *distributed database systems*, where queries may need to be transformed to queries on the *relation fragments* that are stored at specific sites. Fragments, however, are usually limited to selections and projections of a single relation [1]. In the area of *query optimization*, it has been suggested that it may be advantageous to store *intermediate results* of query evaluation, or the final results of *recently evaluated queries*, as these may speed up the processing of new queries.

The family of queries considered by Larson and Yang is considerably larger than the queries considered in the aforementioned variants. Larson and Yang assume queries and views that involve projections, selections and joins. However, several non-trivial assumptions are made, the most important of which is that a relation name may not appear more than once. This restriction makes the issue of decidability (can a query on the relations be transformed to a query on the views) straightforward.

The results obtained in this paper allow us to extend this work in two ways. First, we do not impose this restriction, and we allow arbitrary conjunctive queries. Additionally, Larson and Yang do not address the issue of approximative answers, when a query is not expressible in terms of the views. In virtually all the applications that they cite, the ability to compute an approximation of the given query from the views is highly desirable.

2.2 Cooperative Answering

In [6] it is observed that 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 sound? i.e., is the information accurate in all respects? (2) Is the answer complete? i.e., does it include all the occurrences that actually exist? Hence, answers have integrity, if they contain *the whole truth* (completeness) and *nothing but the truth* (soundness). A new model of integrity was introduced which was based on new kinds of integrity properties, called *soundness properties* and *completeness properties.*[2] A soundness property asserts that a particular view of the database is guaranteed to be sound, and a completeness property asserts that a particular view of the database is guaranteed to be complete. More specifically, a hypothetical database instance is assumed that models the real world perfectly. A database view is sound if it is contained in the corresponding real world view, and it is complete if contains the corresponding real world view. The integrity properties of a database are denoted as *view definitions.* This information is then used to infer the soundness and completeness properties of answers that are issued in response to individual queries.

Formally, assume a database scheme $R_1, \ldots, R_n$ and assume that $S_1, \ldots, S_m$ are definitions of views of this database that are declared to be sound and that $C_1, \ldots, C_k$ are definitions of views that are declared to be complete. Consider a query Q on $R_1, \ldots, R_n$. The answer to Q is sound if it can be rephrased as a view of the sound views $S_1, \ldots, S_m$, and it is complete if it can be rephrased as a view of the complete views $C_1, \ldots, C_k$. If the answer to Q is not sound or complete, it is important to determine whether any part of it is sound or complete; i.e., discover the views of Q that are views of $S_1, \ldots, S_m$ or of $C_1, \ldots, C_k$.

In [8] the context is broadened to include *arbitrary* properties of the database (not necessarily soundness and completeness). A database system is then described that allows users to store various assertions about properties of the data (meta-information), and then uses

[2]Our terminology here is slightly different from that used in [6].

these assertions to infer properties of each answer computed by the system (meta-answers). Meta-answers are offered to users along with each answer, and help them to assess the value and meaning of the information that they receive. In many ways this process emulates the cooperative behavior of humans, who, in response to questions, often volunteer additional information *about* their answers.

In both papers, all queries and views are assumed to be conjunctive relational expressions; specifically, each view or query is a Cartesian product of an arbitrary number of relations, followed by a selection which is a conjunction of attribute-attribute or attribute-constant comparisons, followed by an arbitrary projection. Let $R_1, \ldots, R_n$ denote the database relations, let $V_1, \ldots, V_m$ denote definitions of views with a specific property, and let Q denote a query. A method is described for inferring views of Q that are views of $V_1, \ldots, V_m$ (i.e., views of the answer that have the specific property).

In the terminology of the introduction, if Q itself is one of these inferred views, then Q can be transformed. Otherwise, the union of the inferred views provides a sound approximation of the query. The inference method is shown to be sound (i.e., it discovers only views that indeed have the property), but is not necessarily complete (i.e., it may not discover all such views).[3] Consequently, it cannot be argued that this approximation is the best possible (i.e., the greatest sound approximation).

The results obtained in this paper allow us to extend the work described in [6, 8]. Specifically, we provide the completeness missing from the previous results, which guarantees a greatest sound approximation. In practice, when the database system issues an answer to a user query, the user receives the most complete description of the portion of the answer that possesses a specific property.

2.3 Multidatabases

One of the most challenging and applicative areas of research in information systems is the *integration* of multiple information sources in a single virtual source that effectively encompasses the information contained in the entire environment.

In [7] an attempt is made to provide formal definitions for fundamental concepts such as a multidatabase, a multidatabase query and a multidatabase answer. These definitions are intended to extend the commonly accepted definitions for a single database environment to an environment of multiple databases, and to generalize many of the existing approaches to database integration. A brief summary of this formalization follows.

A database is a database scheme D and an instance d of the scheme D. Assume two database schemes D_1 and D_2. A *scheme mapping* (D_1, D_2) is a collection of view pairs $(V_{i,1}, V_{i,2})$ $(i = 1, \ldots, m)$, where each $V_{i,1}$ is a view of D_1, each $V_{i,2}$ is a view of D_2, and $V_{i,1}$ is

[3] Note that the soundness and completeness discussed here refer to the method of finding views that have a property; they should not be confused with soundness and completeness described earlier, which were two specific examples of properties.

equivalent to $V_{i,2}$. Intuitively, "equivalence" means that the materializations of both views are always identical. A multidatabase is

1. A scheme D.
2. A collection $(D_1, d_1), \ldots, (D_n, d_n)$ of databases.
3. A collection $(D, D_1), \ldots, (D, D_n)$ of scheme mappings.

This definition provides three important "degrees of freedom" which reflect the realities of multidatabase environments.

First, the mappings from D to the individual databases are not necessarily "total"; i.e., not all views of D are expressible in every individual database (and even if they are expressible, there is no guarantee that they are mapped). This models the dynamic situation of a multidatabase system, where some component databases might become temporarily unavailable. In such cases the corresponding mappings are "suspended", and some global queries might not be answerable in their entirety.

Second, these mappings are not necessarily "onto"; i.e., the individual databases may include views that are not expressible in D (and even if they are expressible, there is no guarantee that they are mapped). This models the pragmatism of multidatabases, which usually cull from existing databases only the information which is relevant to a specific set of applications.

Finally, the mappings are not necessarily "single-valued"; i.e., a view of D may be found in several component databases. This models the realistic situation, in which information is found in several overlapping databases, and provides a formal framework for dealing with multidatabase inconsistency.

Intuitively, the views in the first position of the mappings are "the answerable questions" at the global level, and the views in the second position describe how these answers are materialized. (Indeed, the component databases need not be relational, and the views in the second position need not be relational expressions. The only requirement is that they compute a tabular answer.)

A multidatabase query is simply a query on the scheme D. The answer to a multidatabase query is obtained by transforming it to a query over the views in the first positions of the mappings (the answerable questions). Specific solutions to this transformation problem are not discussed in [7].

The results obtained in this paper extend this work described in [7] in two ways. First, we provide accurate semantics to the concept of a multidatabase query, by showing how a multidatabase query is transformed to a query over the "answerable questions". This improves the specificity of the formal model of multidatabases presented there. Moreover, as explained above, in multidatabases it is essential to provide partial answers, when complete answers cannot be computed. The approximative answers we compute in this paper may serve as partial answers.

3 Conjunctive Queries: Containment and Equivalence

In this section we review the definition of conjunctive queries and results regarding their containment and equivalence. Most of the description here is from [10], with some changes in notation.

We consider arbitrary conjunctive queries of the form

$$\{\overline{X}_0 \mid R_1(\overline{X}_1), \ldots, R_m(\overline{X}_m), C\}$$

where $R_1, \ldots, R_m$ are predicate symbols and $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_m$ are tuples in which each position is either a variable name or a constant, and C is a conjunction of equalities of the form $x = y$ or $x = const$, where x, y are variables appearing in $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_m$. Note that $R_1, \ldots, R_m$ are not necessarily distinct. In the following sections we will consider a more general form of conditions C.

A database is a collection of relations, one for each predicate symbol in the query. Thus, each predicate symbol R_i in the query has an associated relation r_i in the database. Note that, if $R_i = R_j$ for $i \neq j$ (R_i and R_j is the same relation name), then $r_i = r_j$. The answer to the query is a relation defined by

$$\{\overline{X}_0 \mid (\exists \overline{Y})(\overline{X}_1 \in r_1 \wedge \ldots \wedge (\overline{X}_m) \in r_m \wedge C\}$$

where $\overline{Y}$ is the set of all variables in $\overline{X}_1, \ldots, \overline{X}_m$ which are not in $\overline{X}_0$.

Since constants and multiple occurrences of the same variable in $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_m$ can always be expressed by adding corresponding equalities to C, we shall assume without loss of generality that queries are *rectified*, that is, each position in tuples $\overline{X}_0, \overline{X}_1, \ldots, \overline{X}_m$ contains a unique variable name. Thus, all equalities among predicate attributes and constants must be stated in C.

Let Q_1 and Q_2 be two conjunctive queries. Q_1 is *contained* in Q_2, denoted $Q_1 \subseteq Q_2$, if for every database the answer relation to Q_1 is contained in the answer relation to Q_2. Q_1 is *equivalent* to Q_2, denoted $Q_1 \equiv Q_2$, if for every database the answer relations to Q_1 and Q_2 are identical. Note that $Q_1 \equiv Q_2$ if and only if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$.

The key to the testing of containment $Q_1 \subseteq Q_2$ is a *variable mapping* h from the set W_2 of variable names in Q_2 to the set W_1 of variable names in Q_1. This mapping extends naturally to symbolic expressions that contain variables, by "pushing" h inside expressions. That is, $h((x_1, \ldots, x_n)) = (h(x_1), \ldots, h(x_n))$, $h(R(x_1, \ldots, x_n)) = R(h(x_1, \ldots, x_n)) = R(h(x_1), \ldots, h(x_n))$, and conditions $x = y$ and $x = const$ are mapped to $h(x) = h(y)$ and $(h(x) = const$, respectively.

Let

$$\begin{aligned} Q_1 &= \{\overline{X}_0 \mid R_1(\overline{X}_1), \ldots, R_m(\overline{X}_m), C_1\} \\ Q_2 &= \{\overline{Y}_0 \mid S_1(\overline{Y}_1), \ldots, S_l(\overline{Y}_l), C_2\} \end{aligned}$$

and let W_1 and W_2 denote the variable names in Q_1 and Q_2, respectively. A *containment mapping from Q_2 to Q_1* is defined as a variable mapping h from W_2 to W_1 that maintains

1. $h(\overline{Y}_0) = \overline{X}_0$.
2. For every $1 \leq i \leq l$ there exists $1 \leq j \leq m$ such that $h(S_i(\overline{Y}_i)) = R_j(\overline{X}_j)$ (in particular, $S_i = R_j$).
3. $C_1 \models h(C_2)$; that is, for every mapping m of variables in C_1 into constants, if m satisfies C_1 then m satisfies $h(C_2)$.

The following is a result of Chandra and Merlin [2].

Theorem 1 *$Q_1 \subseteq Q_2$ if and only if there exists a containment mapping from Q_2 to Q_1.*

Theorem 1 was extended by Sagiv and Yannakakis to unions of conjunctive queries [9]. The conjunctive queries discussed so far allowed only equality comparisons. For comparisons that involve inequalities, Klug showed a condition sufficient for containment [3].

Our results in this paper hold for conjunctive queries with arbitrary conditions C for which Theorem 1 holds, and for which the following is computable: for given conditions C_1 and C_2, find the weakest condition $C_{\mathcal{V}}$ such that $C_1 \wedge C_{\mathcal{V}} \models C_2$.

4 Queries on Views: Containment, Expressibility, and Approximation

We consider views expressed by conjunctive queries. Suppose views $\mathcal{V} = \{V_1, \ldots, V_k\}$

$$V_i = \{\overline{X}_i \mid R^i_1(\overline{Y}^i_1), \ldots, R^i_{m_i}(\overline{Y}^i_{m_i}), C^i\}$$

We can define queries on the views $\mathcal{V}$, such as

$$Q_{\mathcal{V}} = \{\overline{X}_0 \mid V_1(\overline{X}_1), \ldots, V_k(\overline{X}_k), C_{\mathcal{V}}\}$$

To evaluate $Q_{\mathcal{V}}$, first the relations $v_1, \ldots, v_k$ are evaluated by the queries $V_1, \ldots, V_k$, then $Q_{\mathcal{V}}$ is evaluated on the relations $v_1, \ldots, v_k$. The definition of *containment* and *equivalence* of conjunctive queries naturally extends to the case of queries expressed on views. Note that some of the views V_i can be database relation names, so queries may mix views and relations.

A query expressed on views can also be expressed as a query on the original database. We assume now without loss of generality that the only common variables in $Q_{\mathcal{V}}$ and V_i,

$1 \leq i \leq k$, are those in $\overline{X}_i$. For a query $Q_\mathcal{V}$ on views $\mathcal{V}$, we define its *view-free* counterpart $Q^*_\mathcal{V}$ by replacing each $V_i(\overline{X}_i)$ in $Q_\mathcal{V}$ with

$$R^i_1(\overline{Y}^i_1), \ldots, R^i_{m_i}(\overline{Y}^i_{m_i}), C^i$$

which is the definition of V_i. More formally,

$$\begin{aligned} Q^*_\mathcal{V} = \{\overline{X}_0 \mid \quad & R^1_1(\overline{Y}^1_1), \ldots, R^1_{m_1}(\overline{Y}^1_{m_1}), \\ & R^2_1(\overline{Y}^2_1), \ldots, R^2_{m_2}(\overline{Y}^2_{m_2}), \\ & \vdots \\ & R^k_1(\overline{Y}^k_1), \ldots, R^k_{m_k}(\overline{Y}^k_{m_k}), \\ & C^1 \wedge C^2 \wedge \cdots \wedge C^k \wedge C_\mathcal{V}\} \end{aligned}$$

The following proposition follows directly from the definitions:

Proposition 1 $Q_\mathcal{V} \equiv Q^*_v$

Proposition 1 and Theorem 1 trivially imply

Theorem 2 $Q_{\mathcal{V}_1} \subseteq Q_{\mathcal{V}_2}$ *if and only if there exists a containment mapping from* $Q^*_{\mathcal{V}_2}$ *to* $Q^*_{\mathcal{V}_1}$.

For approximations of queries using views we need various notions of lower bounds.[4] Consider a query $Q_\mathcal{V}$ on views $\mathcal{V}$

$$\{\overline{X}_0 \mid V_1(\overline{X}_1), \ldots, V_k(\overline{X}_k), C_\mathcal{V}\}$$

Definition 1 *A query* $Q'_\mathcal{V}$ syntactically contains $Q_\mathcal{V}$, *if* $Q'_\mathcal{V}$ *can be constructed from* $Q_\mathcal{V}$ *by deleting some atoms* V_i *or replacing the constraint* C *by a* strictly weaker *constraint* C' *(that is,* $C \models C'$, *but* $C \not\equiv C'$.*)*

Definition 2 $Q_\mathcal{V}$ *is a* lower bound *of a query* Q *on* $\mathcal{R}$ *if* $Q_\mathcal{V} \subseteq Q$. $Q_\mathcal{V}$ *is a* maximal *lower bound of* Q *if (1) it is a lower bound, and (2) no other lower bound* $Q'_\mathcal{V}$ *of* Q *syntactically contains* $Q_\mathcal{V}$.

Consider now a query Q_U which is a union of conjunctive queries

$$Q_U = \bigcup_{v=1}^{n} Q_\mathcal{V}$$

Definition 3 Q_U *is a greatest lower bound of a query* Q *on* $\mathcal{R}$, *if (1) it is a lower bound, and (2) every lower bound* Q'_U *(that can be a union of conjunctive queries) is contained in* Q_U *(*$Q'_U \subseteq Q_U$*).*

[4]In this and the next section we shall use the term *lower bound* instead of the term *sound approximation* that was used in Sections 1 and 2.

We are interested in answering the following *expressibility and approximation* questions for a given conjunctive query Q on $\mathcal{R}$ and a predefined collection $\mathcal{V}$ of views $\{V_1, \ldots, V_k\}$.

1. Does a query $Q_\mathcal{V}$ exist, which is equivalent to Q and is expressed exclusively through $\mathcal{V}$? If so, how to find $Q_\mathcal{V}$?
2. If $Q_\mathcal{V}$ does not exist, does a query Q_U exist, which is a *greatest lower bound* of Q and is expressed exclusively through $\mathcal{V}$? If so, is it unique up to equivalence of queries, and how to find Q_U?
3. If $Q_\mathcal{V}$ does not exist, and P is an arbitrary projection of Q, does a query $P_\mathcal{V}$ exist, which is equivalent to P and is expressed exclusively through $\mathcal{V}$? If so, how to find $P_\mathcal{V}$?
4. If $P_\mathcal{V}$ does not exist, does a query P_U exist, which is a *greatest lower bound* of P and is expressed exclusively through $\mathcal{V}$? If so, is it unique up to equivalence of queries, and how to find P_U?

To illustrate the relevance of these questions to the applications described in Section 2, consider the example of multidatabases (Section 2.3). Q is a multidatabase query and $\mathcal{V}$ describes the local queries that are currently answerable. If the first question is answered positively, then the multidatabase query Q can be processed successfully, and the expression $Q_\mathcal{V}$ provides the necessary decomposition of the "global" query to "local" queries. The second question becomes interesting when the first question is answered negatively. If a unique Q_U exists and can be found, then a method is available for assembling from the available information the best sound approximation for Q. Intuitively, this approximation provides the largest number of tuples that satisfy the query that are currently obtainable. It represents "the best the system can do at the moment". When resorting to approximations, it is also important to consider answers that are incomplete because they are missing some of the attributes of Q. That is, partial answers should also include any projections of Q that are obtainable. The third and fourth questions repeat the first and second questions with respect to projections of Q.

To answer these questions, we first develop the useful tool of *lower bound classification.*

5 Lower Bound Classification

Two queries may be identical, except for a renaming of their variables and a reordering of their atoms. Formally, Q_1 (with variables W_1) and Q_2 (with variables W_2) are are *isomorphic* if there exists a bijective variable mapping m from W_1 to W_2, such that both m and m^{-1} are containment mappings. Clearly, the isomorphism is an equivalence relation. We say also that two sets of queries $\mathcal{S}_1$ and $\mathcal{S}_2$ are *identical up to isomorphism*, if there exists a bijective function m mapping each query in $\mathcal{S}_1$ to an isomorphic query in $\mathcal{S}_2$. Note that in this case $\mathcal{S}_1$ and $\mathcal{S}_2$ have the same cardinality.

Definition 4 *A* lower bound classification *of Q is a finite set $\mathcal{S}$ of query maximal lower bounds $Q_{\mathcal{V}}$ having the following properties:*

1. *Completeness: for every lower bound Q_l of Q, there exists $Q_{\mathcal{V}}$ in $\mathcal{S}$ containing Q_l ($Q_l \subseteq Q_{\mathcal{V}}$).*
2. *Minimality: no two maximal lower bounds in $\mathcal{S}$ are isomorphic.*

Lower bound classification, if it exists, allows answering all the expressibilty and approximation questions formulated in the previous section.

Theorem 3 *The following problems are decidable: For a* lower bound classification *$\mathcal{C}$ of a conjunctive query Q and a collection of views $\mathcal{V}$, whether there exist*

1. *a conjunctive query $Q_{\mathcal{V}}$ on $\mathcal{V}$ that is equivalent to Q and*
2. *a union of conjunctive queries Q_U on $\mathcal{V}$ that is equivalent to Q.*

Furhermore, given $\mathcal{C}$, each of $Q_{\mathcal{V}}$ and Q_U is effectively computable if it exists. Moreover, the greatest lower bound of Q exists and is unique up to equivalence.

Proof If a conjunctive query $Q_{\mathcal{V}}$ on $\mathcal{V}$ is equivalent to Q, it is a maximal lower bound of Q. In this case, we can find it in $\mathcal{C}$, which is finite. This proves the decidability of the first problem, and the effective computablitly of $Q_{\mathcal{V}}$.

Consider
$$Q_U = \bigcup \{Q_{\mathcal{V}} \mid Q_{\mathcal{V}} \in \mathcal{C}_{\mathcal{V}}\}$$
The query Q_U is a greatest lower bound of Q, since every lower bound of Q is contained in Q_U, and thus, so is every union of lower bounds.

Clearly, a greatest lower bound Q_U is equivalent to Q if and only if Q is expressible as a union of conjunctive queries on $\mathcal{V}$. This proves the decidability of the second problem and the effective computability of Q_U.

Finally, the uniqueness of the greatest lower bound up to equivalence follows from its definition. $\square$

Theorem 4 *A lower bound classification (LBC) of Q and $\mathcal{V}$ exists. Furthermore, all LBCs are identical up to isomorphism, and finite (and therefore have the same number of queries). Moreover, the LBC is effectively computable.*

Proof (sketch)
Let Q be

$$\{\overline{X}_0 \mid R_1(\overline{Z}_1), \ldots, R_m(\overline{Z}_m), C\}$$

and the views V_i be of the form

$$V_i = \{\overline{X}_i \mid R_1^i(\overline{Y}_1^i), \ldots, R_{m_i}^i(\overline{Y}_{m_i}^i), C^i\}$$

Consider an arbitrary query on $\mathcal{V}$

$$Q_{\mathcal{V}} = \{\overline{X}_0 \mid V_1(\overline{X}_1), \ldots, V_k(\overline{X}_k), C\}$$

which is a lower bound of Q. Then, by Theorem 2, there exists containment mapping h from Q (Note that $Q^* = Q$) to $Q_{\mathcal{V}}^*$, where $Q_{\mathcal{V}}^*$ is of the form

$$\begin{array}{rl} Q_{\mathcal{V}}^* = \{\overline{X}_0 \mid & R_1^1(\overline{Y}_1^1), \ldots, R_{m_1}^1(\overline{Y}_{m_1}^1), \\ & R_1^2(\overline{Y}_1^2), \ldots, R_{m_2}^2(\overline{Y}_{m_2}^2), \\ & \vdots \\ & R_1^k(\overline{Y}_1^k), \ldots, R_{m_k}^k(\overline{Y}_{m_k}^k), \\ & C^1 \wedge C^2 \wedge \cdots \wedge C^k \wedge C_{\mathcal{V}}\} \end{array}$$

The mapping h maps each R_i-atom in Q to a R_k^j-atom in $Q_{\mathcal{V}}^*$. Recall that each R_k^j-atom originates from the corresponding V_j-atom of $Q_{\mathcal{V}}$. In this case we say that the R_k^j-atom is mapped inside the V_j atom of $Q_{\mathcal{V}}$.

Clearly, the number of V_j-atoms inside which R_i-atoms are mapped is bounded by the number of atoms R_i in the original query. If we eliminate all other V_j-atoms from $Q_{\mathcal{V}}$, we create a query $Q'_{\mathcal{V}}$, which still, by Theorem 2, is a lower bound of Q, and syntactically contain $Q_{\mathcal{V}}$.

We may conclude, that the number of V_j-atoms in every maximal lower bound Q is bounded by the number of R_i-atoms in the original query Q.

The main idea of the lower bound classification construction is based on the above property. In the full paper, we build a finite number of variables mapping h from Q to a "skeleton" of $Q_{\mathcal{V}}^*$, which looks exactly as $Q_{\mathcal{V}}^*$, except it has a "?" in place of $C_{\mathcal{V}}$.

To produce a maximal lower bound for each such mapping h, we need to find the "weakest" condition $C_{\mathcal{V}}$ such that

$$C^1 \wedge C^2 \wedge \ldots \wedge C^k \wedge C_{\mathcal{V}} \models h(C)$$

In the full paper we show that by this construction we do not lose any maximal lower bounds, and that no two of them are isomorphic. Thus produces maximal lower bounds constite lower bound classification. $\square$

6 Conclusion

In this paper we have described preliminary results on the problem of optimal approximation of queries on a database scheme by queries on a given view scheme. Much work remains to be done, and we mention here two important issues.

We have mentioned in the introduction that the answer to a query Q is "sandwiched" between a greatest sound approximation, and a least complete approximation. In this paper we focused on the former (it also appears that to have more practical value). Further work is necessary on the dual estimate LCA.

An important concept behind our techniques was query containment. Query containment is based on *tuple* subset relationship: Q_1 is contained in Q_2 if every tuple of Q_1 is a tuple of Q_2. In most of the applications, it is just as useful to provide approximations that are based on a more general relationship by which Q_1 is "contained" in Q_2 if every tuple of Q_1 is a *subtuple* of some tuple of Q_2. For example, in the multidatabase application, a global query on the positions and salaries of all employees could be approximated by the union of the positions of all employees and the positions and salaries of all junior employees. We have touched upon this issue when we discussed optimal approximations for arbitrary projections. Yet, the problem of optimal approximations based on this broader notion of containment requires further work.

References

[1] S. Ceri and G. Pelagatti. *Distributed Databases: Principles and Systems*. McGraw-Hill, New York, New York, 1984.

[2] A. K. Chandra and P. M. Merlin. Optimial implementation of conjunctive queries in relational databases. In *Proceedings of the Ninth Annual ACM Symposium on the Theory of Computing*, pages 77–90, 1977.

[3] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, January 1988.

[4] P.-A. Larson and H. Z. Yang. Computing queries from derived relations. In *Proceedings of the Eleventh International Conference on Very Large Data Bases* (Stockholm, Sweden, August 21–23), pages 259–269. Morgan Kaufmann, Los Altos, California, 1985.

[5] P.-A. Larson and H. Z. Yang. Computing queries from derived relations: Theoretical foundations. Technical Report CS-87-35, Department of Computer Science, University of Waterloo, August 1987.

[6] A. Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, December 1989.

[7] A. Motro. A formal framework for integrating inconsistent answers from multiple information sources. Technical Report ISSE-TR-93-106, Department of Information and Software Systems Engineering, George Mason University, October 1993.

[8] A. Motro. Panorama: A database system that annotates its answers to queries with their properties. (submitted for publication), October 1993.

[9] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operators. *Journal of the ACM*, 27(4):633–655, October 1980.

[10] J. D. Ullman. *Database and Knowledge-Base Systems, Volume II.* Computer Science Press, Rockville, Maryland, 1989.