Information Integration and Incomplete Information

Dedicated to Witold Lipski Jr., 1949–1985

Gösta Grahne
Department of Computer Science, Concordia University
Montreal, Quebec, Canada H3G 1M8
grahne@cs.concordia.ca

Abstract

We first briefly review relational databases and incomplete information. We then describe how querying in Information Integration Systems amounts to querying incomplete databases. We also discuss query rewriting, which essentially is a (compile-time) query optimization technique used in processing queries in Information Integration systems. Finally, we review some recent advances in query rewriting.

1 Basic Definitions

Relational Databases. The Relational Model structures information in the form of tables. There are various ways to formalize this notion. The first is a direct adaptation of the mathematical concept of a relation: Let the atomic values come from a countably infinite set $\mathbf{dom}$.[1] Let k be a positive integer. A k-ary relation is then a finite subset of $\mathbf{dom} \times \mathbf{dom} \times \cdots \times \mathbf{dom}$, k-times, sometimes denoted $\mathbf{dom}^k$. In order have names for relations, such as R, R_1, R_2, etc, we assume a function I that when applied to a relation name R gives the *instance* $I(R)$ of the actual tuples from $\mathbf{dom}^k$. For a relational "schema" $\{R_1, R_2, \ldots, R_n\}$, the database instance consists of an instance $I(R_i)$ for each relation name R_i, $i \in \{1, \ldots, n\}$. Note that there are no column names in this formalization, only positions (first column, second column, etc).

In this paper we adopt a variation of this modeling influenced by the logic programming approach to databases. From the domain $\mathbf{dom}$ and the relation names we build up a (Herbrand) universe consisting of all expressions of the form $R(a_1, a_2, \ldots, a_k)$, where R is a relation name and the a_i's are values in $\mathbf{dom}$. Such expressions are called *facts*. A database instance is then simply a finite set of facts, i.e. a finite subset of the universe, such as $\{R_1(a_1, a_3), R_1(a_2, a_3), R_2(a_3, a_4)\}$. Since relation names are part of the facts, we do not need to list facts from different relations separately.

For more information on the various ways to formalize the relational model, see [AHV95].

A *query* is an expression that defines a function from all database to (usually) a single relation. In this paper we focus on a simple class of queries, namely the conjunctive queries. For this we need the concept of an *atom*. An atom is like a fact, except that we allow *variables*, taken from and infinite set $\mathbf{var}$, as well as constants. For

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] We assume for simplicity and clarity of exposition that there is only one domain from which all values are drawn.

instance, $R(a,X)$ is an atom, whereas $R(a,b)$ is a fact.[2] The variables in the atoms are placeholders, and they stand for any domain value. Variables will be denoted by uppercase letters, such as X and Y.

A *conjunctive query* φ is an expression of the form

$$head(\varphi) \leftarrow body(\varphi), \tag{1}$$

where $body(\varphi)$ is a set of atoms over relation names in the database schema, and $head(\varphi)$ is an atom over an answer relation name not used in the database. We assume that all variables occurring in $head(\varphi)$ also occur in $body(\varphi)$, i. e. that the query φ is *safe*.

A conjunctive query φ can be applied to a database $\mathbf{db}$ resulting in a set of facts

$$\varphi(\mathbf{db}) = \{\sigma(head(\varphi)) : \sigma(body(\varphi)) \subseteq \mathbf{db} \text{ for some valuation } \sigma\}. \tag{2}$$

A *valuation* σ, is formally a finite partial mapping from $\mathbf{var} \cup \mathbf{dom}$ to $\mathbf{dom}$ that is the identity on $\mathbf{dom}$.

Example 1: If the database $\mathbf{db}$ is $\{R_1(a_1,a_3), R_1(a_2,a_3), R_2(a_3,a_4)\}$, and φ is $Q(X) \leftarrow R_1(X,Y), R_2(Y,Z)$, then $\varphi(\mathbf{db}) = \{Q(a_1), Q(a_2)\}$. On the other hand, if φ is $Q(X) \leftarrow R_1(a_1,Y), R_2(Y,X)$, then $\varphi(\mathbf{db}) = \{Q(a_4)\}$. □

Incomplete Databases. Usually a database is a complete description of the state of the world modeled by it. This is actually true only if we adopt the *Closed World Assumption*, CWA [Rei78], according to which facts not explicitly stored in the database are false. For example, in the database $\{R_1(a_1,a_3), R_1(a_2,a_3), R_2(a_3,a_4)\}$, the fact $R_1(a_3,a_1)$ is false. The closed world assumption is convenient since there in general is an infinitude of false facts. The CWA is however not appropriate for modeling all situations. We might for instance have a database $\mathbf{db} = \{$*Owns(Sue,Mazda)*,*Owns(Joe,Chevrolet)*$\}$. If we consider the fact *Owns(Mary,Toyota)* we might not want to draw the conclusion that Mary does not own a Toyota, we just don't have evidence recorded about it. We then adopt the *Open World Assumption*, OWA, which regards the database as an *incomplete* description of the world: All facts stored in the database are true, the truth value of any other fact is *unknown*. Semantically this means that the stored database $\mathbf{db}$ actually is a finite description of a *set of possible worlds*, defined as

$$\{\mathbf{db}' : \mathbf{db} \subseteq \mathbf{db}'\}. \tag{3}$$

Each $\mathbf{db}'$ represents one possible complete state of affairs (or world). Thus for instance the fact *Owns(Mary, Toyota)* will be true in some possible worlds, and false in others. Facts that are true in *some* possible worlds are called *possible* facts, and facts true in *all* possible worlds are called *certain* facts.

A database under the OWA is the simplest example of an incomplete database. To go a step further, suppose we know that Mary owns a car, but we don't know the make of it. This we could represent by the atom *Owns(Mary,X)*. Here again, we have an incomplete description of the world. The world could correspond to *Owns(Mary,Toyota)*, or to *Owns(Mary,Mazda)*, or to *Owns(Mary,Drabant)*, and so on.

A very simple (and efficient) way to represent possible worlds is to allow the database to consist of atoms, as opposed to pure facts only. A set of facts is called a *tableau* [Men84]. A tableau is denoted T, as opposed to $\mathbf{db}$ which stands for a finite set of facts (not atoms).

Example 2: Let $T = \{$*Owns(Sue,Mazda)*,*Owns(Mary,X)*$\}$. This tableau contains two atoms, the first of which actually is a fact (a special case of an atom). □

A tableau T represents a set of databases. This set is denoted $rep(T)$, and it is defined by

$$rep(T) = \{\mathbf{db} \ : \ \text{there is a valuation } \ \sigma \ \text{ such that } \ \sigma(T) \subseteq \mathbf{db}\}. \tag{4}$$

The definition says that a database $\mathbf{db}$ is represented by a tableau T, if there is a valuation σ such that when all variables in T are replaced by their image under σ, the set of facts thus obtained is a subset of $\mathbf{db}$.

[2] $R(a,b)$ is of course also an atom, since a fact is a special case of an atom.

Example 3: In our tableau $T = \{$*Owns(Sue,Mazda)*, *Owns(Mary,X)*$\}$, we will have $\{$*Owns(Sue,Mazda)*, *Owns(Mary,Mazda)* $\} \in rep(T)$, $\{$*Owns(Sue,Mazda)*,*Owns(Mary,Chevrolet)*$\} \in rep(T)$, etc. In this database *Owns(Sue,Mazda)* is a certain fact since it is true in all possible worlds in $rep(T)$, and *Owns(Mary, Mazda)* is a possible fact since it is true in some possible worlds represented by T. $\square$

There are many other ways to represent sets of possible worlds, see [Gra84, IL84], and [Gra91, AHV95] for an overview.

Now what does it mean to query an incomplete database? Since an incomplete database is a set, each element representing a possible world, the natural answer to a query is the *set* of answers obtained by applying the query on each possible world. This set of answers is called the *exact answer*.

Suppose now we have a query φ on an incomplete database represented by a tableau T. The exact answer would be $\varphi(poss(T))$. But just as the original database was represented by a tableau T we would like to represent the exact answer by another tableau U, such that $rep(U) = \varphi(rep(T))$. More practically thinking, given a query φ and a tableau T, we would like to have an algorithm, or evaluation mechanism, call it $\widehat{\varphi}$, applicable on tableau, such that

$$rep(\widehat{\varphi}(T)) = \varphi(rep(T)). \tag{5}$$

This requirement can be illustrated by the following commutative diagram.

$$\begin{tikzcd}
\text{Tableau } T \arrow[r, "rep"] \arrow[d, "\widehat{\varphi}"'] & \text{Possible worlds } rep(T) \arrow[d, "\varphi"'] \\
\text{Answer } \widehat{\varphi}(T) \arrow[r, "rep"] & \text{Exact answer } \varphi(rep(T))
\end{tikzcd}$$

Is the requirement in Equation (5) achievable? The answer is yes, but only if we use a representation mechanism more complicated than tableaux (see [IL84]). If we wish to use tableaux we have to contend with a weaker form of (5), where we use coinitiality, instead of equality. Let $\mathcal{X}$ and $\mathcal{Y}$ be two enumerable sets of possible databases. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *coinitial* if they have the same $\subseteq$-minimal elements. Coinitiality is denoted $\mathcal{X} \approx \mathcal{Y}$.[3]

For tableaux, it turns out that if we for evaluation purposes treat the variables as pairwise distinct constants, and distinct from all "true" constants, we can apply standard evaluation and obtain an representation of the result that is coinitial with the exact answer. To formally explain the evaluation, we need the concept of a substitution. A *substitution* is a valuation, except that we allow variables to be mapped into variables, not only constants. Thus, a substitution θ is a function from (a subset of) $\mathbf{dom} \cup \mathbf{var}$ to $\mathbf{dom} \cup \mathbf{var}$, keeping in mind that constants have to be mapped to themselves. Then

$$\widehat{\varphi}(T) = \{\theta(head(\varphi)) : \theta(body(\varphi)) \subseteq T \text{ for some substitution } \theta\}. \tag{6}$$

The fundamental result that

$$rep(\widehat{\varphi}(T)) \approx \varphi(rep(T)) \tag{7}$$

was established in [IL84, Var86], and adapted to tableaux in [GM99, GK02]. Furthermore, in Eq. (7) the certain facts of $\varphi(rep(T))$, that is $\cap(\varphi(rep(T)))$, can be obtained from $\widehat{\varphi}(T)$ by retaining only the pure variable-free atoms.

[3]For a deeper treatment of coinitiality, see [IL84].

Example 4: Let $T = \{R_1(a,b), R_1(d,X), R_2(b,c), R_2(X,e), R_2(Y,f)\}$, and $\varphi = Q(X,Y,Z) \leftarrow R_1(X,Y), R_2(Y,Z)$. Then $\widehat{\varphi}(T) = \{Q(a,b,c), Q(d,X,e)\}$. The only certain fact in the answer is $Q(a,b,c)$. If $\varphi = Q(X,b) \leftarrow R_1(X,b)$ we have $\widehat{\varphi}(T) = \{Q(a,b)\}$. This fact is also certain. $\square$

2 Global databases: The meaning of integrated information

Information Integration systems [Ull97, Hal00, Hal01, Len02] aim to provide a uniform query interface to multiple heterogeneous sources. One particular and useful way of viewing these systems, first proposed within the Information Manifold project [LRO96], is to postulate a *global schema* (called a world view) that provides a unifying data model for all the information sources. A query processor is in charge of accepting queries written in terms of this global schema, translating them to queries on the appropriate sources, and assembling the answers into a global answer. Each source is modeled as a *materialized view* defined in terms of the global relations, which are virtual. Note the reversal of the classical model: instead of thinking of views as virtual artifacts defined on stored relations, we think of the views as stored, and the relations that the views are defined on as virtual. [4]

A question of semantics now arises: what is the meaning of a query? Since a query is expressed in terms of the global schema, and the sources implicitly represent an instance of this global schema, it would be natural –at least conceptually– to reconstruct the global database represented by the views and apply the query to this global database.

Well, it turns out that the global database actually is incomplete. In other words, there might be *several* (usually infinitely many) global databases that are consistent with the definition of, and the data in the sources.

Example 5: For a simple example, suppose we have a global relations *Owns(Person,Car)*, and two sources: Source S_1 has definition $S_1(X,Y) \leftarrow$ *Owns(X,Y)* and extension $\mathbf{s}_1 = \{S_1(\textit{Sue, Mazda})\}$. Source S_2 has definition $S_2(X) \leftarrow$ *Owns(X,Y)* and extension $\mathbf{s}_2 = \{S_2(\textit{Mary})\}$. Then it is natural to think that any database that contains at least the facts *Owns(Sue,Mazda)*, and *Owns(Mary,*a*)*, for some $a \in \mathbf{dom}$, is a possible global database for $\mathcal{S} = \{S_1, S_2\}$. $\square$

Formally, for a source collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where each S_i has definition ψ_i and extension $\mathbf{s}_i$, we call its set of possible databases $poss(\mathcal{S})$, and define it as

$$poss(\mathcal{S}) = \{\mathbf{db} \ : \ \mathbf{s}_i \subseteq \psi_i(\mathbf{db}), i \in \{1, \ldots, m\}\}. \tag{8}$$

Note that $poss(\mathcal{S})$ is infinite. Now the set $poss(\mathcal{S})$ can be conveniently represented by a tableau denoted $T(\mathcal{S})$, such that

$$rep(T(\mathcal{S})) = poss(\mathcal{S}). \tag{9}$$

Example 6: For $\mathcal{S} = \{S_1, S_2\}$ in Example 5, we have $T(\mathcal{S}) = \{$*Owns(Sue,Mazda), Owns(Mary,*X*)*$\}$. $\square$

The details of how to construct $T(\mathcal{S})$ can be found in [GM99, GK02].

In definition 8 we make a version of the OWA. The facts in the sources are considered to be *sound*, in the sense that they all are generated by facts in the global database. An alternative view would be to state that a source can contain (spurious) facts that do not stem from facts in the global database. In this case the source would be considered *complete*, but not necessarily sound. If the sources are required to be both sound and complete, they are called *exact*.[5] The situations where sources are allowed to be both sound and complete

[4] Whether the views are actually stored at each source or materialized by other means, and whether they consist of relations or semi-structured objects or files are issues irrelevant to our discussion and hidden from the query processor by appropriate wrappers.

[5] If all sources are complete, the definition of $poss$ would be $poss(\mathcal{S}) = \{\mathbf{db} \ : \ \mathbf{s}_i \supseteq \psi_i(\mathbf{db}), i \in \{1, \ldots, m\}\}$, and for exact sources we would have $poss(\mathcal{S}) = \{\mathbf{db} \ : \ \mathbf{s}_i = \psi_i(\mathbf{db}), i \in \{1, \ldots, m\}\}$.

are treated in [AD98, GM99]. The presence of complete sources however usually increase the computational complexity of decision problems related to integration, see [AD98, GM99]. In [MM01] the authors consider a generalization of the sound/complete view assumptions. Each source is is considered to be both *partially sound*, and *partially complete*. A soundness of say 80% would say that the source contains 80% facts that are derived from facts in the global database, and 20% of "spurious" facts. A completeness of 80% says that 80% of the facts derivable from the global database are in the source.

Now that we have seen that a source collection $\mathcal{S}$ actually defines an incomplete database $poss(\mathcal{S})$ that we can represent by a tableau $T(\mathcal{S})$, and that an incomplete database represented as a tableau can be queried using the $\widehat{\varphi}$-evaluation of a query φ, we have a method for computing a representation of the exact answer to a query φ posed on the global database: First compute the tableau $T(\mathcal{S})$, then apply $\widehat{\varphi}$ to obtain $\widehat{\varphi}(T(\mathcal{S}))$. Equation (7) gave us $rep(\widehat{\varphi}(T)) \approx \varphi(rep(T))$, for any tableau T, and since Equation (9) gives $rep(T(\mathcal{S})) = poss(\mathcal{S})$, we get

$$rep(\widehat{\varphi}(T\mathcal{S})) \approx \varphi(poss(\mathcal{S})). \tag{10}$$

Example 7: In examples 5 and 6, if the query φ were the identity, i.e. it asked for the facts in the *Owns*-relation, a representation of the exact answer would be the tableau $\{$*Owns(Sue,Mazda)*, *Owns(Mary,*X*)*$\}$. The certain fact in this answer is *Owns(Sue,Mazda)*. □

3 Using rewritings to answer queries on global databases

In the previous section we saw how to answer a query φ by first constructing the tableau representing all possible global databases. However, computing $T(\mathcal{S})$ might involve a lot of redundant work, since it amounts to constructing the tableau corresponding to the entire source collection $\mathcal{S}$, whereas the global relations that are in the body of φ might be mentioned in only few source definitions. Furthermore, the query might have selections and joins that could be computed directly at the sources.

A *rewriting* φ' of a global query φ is a query formulated directly in terms of the source relations S_i. It appears that rewritings were first proposed for a restricted setting in [YL87]. They were generalized in [LMSS95] and [LRO96]. The correctness criteria for rewritings were defined in terms of a certain containment between the rewriting and the original query.

Example 8: Let $\mathcal{S} = \{S_1, S_2\}$, with definitions $S_1(X,Z) \leftarrow R_1(X,Y), R_2(Y,Z)$, and $S_2(X,Z) \leftarrow R_1(X,Y)$, $R_3(Y,Z)$. Let the query be $\varphi = Q(X,Y) \leftarrow R_1(X,Y)$. According to the correctness criteria in [LRO96], the desired rewriting φ' would be the union of $Q(X,Y) \leftarrow S_1(X,Y)$, and $Q(X,Y) \leftarrow S_2(X,Y)$. □

The correctness criteria did not however state what the answer produced by the rewriting meant. It was subsequently shown in [DG98, GM99] that the rewritings actually produced the *certain answer* $\cap(\varphi(poss(\mathcal{S})))$. This is easy to see in Example 8. Source S_1 contains facts of R_1 that join with a fact in R_2, and S_2 contains facts of R_1 that join with a fact in R_3. Since the sources provide us no information about R_2 and R_3, there is a possible database where R_2 and R_3 are empty. Therefore, the facts of R_1 that are in every possible database is the union of the facts in S_1 and S_2, which is exactly what the "correct" rewriting would produce in Example 8.

The first algorithms for rewritings essentially explored the whole (finite) solution space in order to find φ'. Later more efficient algorithms were developed. These include the set-covering based algorithm in [GM99], the MiniCon algorithm of [PL00], the CoreCover algorithm [ALU01], and its extension in [ALM02]. Other extensions of the basic technique are surveyed in [Hal00, Hal01]. The cases where the queries defining the sources, as well as the user query, are allowed to be more general than conjunctive ones are analyzed in [AD98].

However, perhaps since the relationship between information integration and incomplete information had not been clearly articulated, there were no algorithms for computing the exact answer, until the tableau techniques were applied in [GM99], and [GK02] showed how to come up with a rewriting that actually produces a representation of the exact answer.

Example 9: Suppose the global schema contains two relations, *Owns(Person,Car)*, and *Lives(Person, City)*. A fact *Owns(Sue,Mazda)* means that *Sue* owns a *Mazda*, and e.g. *Lives(Sue,Edmonton)* means that *Sue* lives in *Edmonton*. We have two sources, $S_1(X, Y) \leftarrow$*Owns*(X, Y), *Lives*(X, Z), and $S_2(X, Z) \leftarrow$*Owns*(X, Y), *Lives*(X, Z). The user issues the query $\varphi = Q(X, Y, Z) \leftarrow$*Owns*$(X, Y)$,*Lives*$(X, Z)$. Using the correctness criteria in [LRO96], we get an empty rewriting, and thus an empty query result for the user. However, suppose $\mathbf{s}_1 = \{S_1(\textit{Sue,Mazda})\}$, and $\mathbf{s}_2 = \{S_2(\textit{Mary,Prague})\}$. Then $T(\{S_1, S_2\}) = \{\textit{Owns(Sue,Mazda)},\textit{Owns(Mary},X_1),$ *Lives(Mary,Prague)*, *Lives(Sue,*X_2) $\}$, and $\widehat{\varphi}(T(\{S_1, S_2\})) = \{Q(\textit{Sue, Mazda}, Y_1), Q(\textit{Mary},Y_2,\textit{Prague})\}$. This exact answer tells the user that *Sue* owns a *Mazda* and lives in an unknown city, and that *Mary* lives in *Prague* and owns a car of an unknown make. This is all the information we can deduce based on incomplete information about the the global relations provided by the sources. $\Box$

In [GK02] a methodology is given for constructing a rewriting $\widetilde{\varphi}$ of a query φ, applicable directly on the sources $\mathcal{S}$, such that

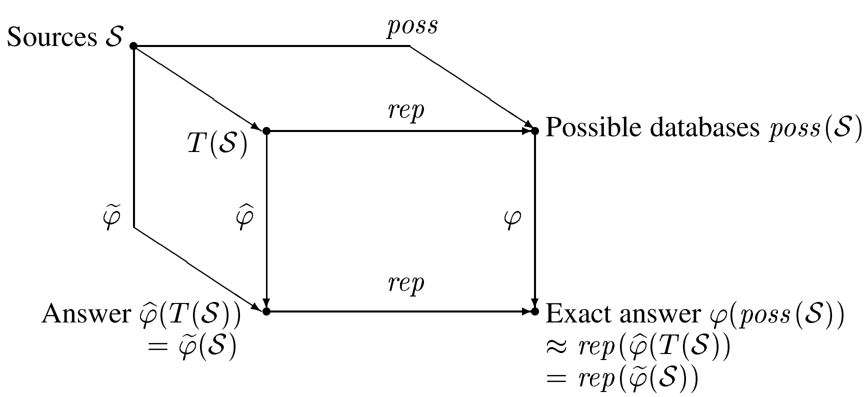
$$\widetilde{\varphi}(\mathcal{S}) = \widehat{\varphi}(T(\mathcal{S})). \tag{11}$$

Since Equation (10) says that $\mathit{rep}(\widehat{\varphi}(T(\mathcal{S}))) \approx \varphi(\mathit{poss}(\mathcal{S}))$, Equation (11) now gives

$$\mathit{rep}(\widetilde{\varphi}(\mathcal{S})) \approx \varphi(\mathit{poss}(\mathcal{S})). \tag{12}$$

Example 10: In Example 9 the method in [GK02] would produce as rewriting the union of $Q(X, Y, Y_1) \leftarrow S_1(X, Y)$ and $Q(X, Y_2, Z) \leftarrow S_2(X, Z)$. The unsafe variables Y_1 and Y_2 represent unknown values. When evaluating such unsafe queries, an unsafe variable in the head of the query essentially generates a new variable in the answer. The answer is then a tableau, with the variables representing missing values. Applying the rewriting to the sources in Example 9 will indeed produce the result $\{Q(\textit{Sue, Mazda}, Y_1), Q(\textit{Mary},Y_2, \textit{Prague})\}$. $\Box$

Query answering in Information Integration systems can now be summarized by the following diagram.



Acknowledgments

Many thanks to Victoria Kiricenko and Alberto Mendelzon for comments on an earlier version of this paper.

References

[AD98] Abiteboul S. and O. M. Duschka: Complexity of Answering Queries Using Materialized Views. In *Proc. 17th Annual ACM Symp. Principles of Databases (PODS '98)*, Seattle, Washington 1998, pp. 254–263.

[AHV95] Abiteboul S., R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, Reading Ma. 1995.

[ALM02] Afrati F., C. Li and P. Mitra Answering Queries Using Views with Arithmetic Comparisons. In *Proc. 21st ACM Symp. on Principles of Database Systems (PODS '02)*, Madison, Wisconsin 2002, pp. 209–220.

[ALU01] Afrati F., C. Li and J. D. Ullman. Generating Efficient Plans for Queries Using Views. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD '01)* Dallas, Texas 2001.

[DG98] Dushka O. M. and M. R. Genesereth. Query Planning with Disjunctive Sources. In *Proceedings of the AAAI Workshop on AI and Information Integration*, Madison, Wisconsin 1998.

[Gra84] Grahne G. Dependency satisfaction in databases with incomplete information. *Proc. of the 10th International Conference on Very Large Databases (VLDB '84).* Singapore 1984, pp. 37–45.

[Gra91] Grahne G. *The Problem of Incomplete Information in Relational Databases*. Lecture Notes in Computer Science, vol. 554. Springer-Verlag, Berlin 1991.

[GM99] Grahne G. and A. O. Mendelzon. Tableau Techniques for Querying Information Sources through Global Schemas. In *Proc. 7th International Conference on Database Theory (ICDT '99)*. Delphi, Greece 1999, pp. 332–347.

[GK02] Grahne G. and V. Kiricenko. Obtaining more answers from Information Integration Systems. *Proc. Fifth International Workshop on the Web and Databases (WebDB '02)*. Madison, Wisconsin 2002, pp. 67–76.

[Hal00] Halevy A. Y. Theory of Answering Queries Using Views. *SIGMOD Record* 29(4): 40-47 (2000).

[Hal01] Halevy A. Y. Answering queries using views: A survey. *VLDB Journal* 10(4): 270-294 (2001).

[IL84] Imielinski T. and W. Lipski Jr. Incomplete Information in Relational Databases. *J. ACM* **31**:4, 1984, pp. 761–791.

[Len02] Lenzerini M. Data Integration: A Theoretical Perspective. Invited tutorial in *Proc. 21st ACM Symp. on Principles of Database Systems (PODS '02)*. Madison, Wisconsin 2002, pp. 233–246.

[LMSS95] Levy A. Y., A. O. Mendelzon, Y. Sagiv and D. Srivastava. Answering Queries Using Views. In *Proc. 14th ACM Symp. on Principles of Database Systems (PODS '95)*. San Jose, California 1995, pp. 95–104.

[LRO96] Levy A. Y, A. Rajaraman, J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. *Proc. 22nd Int'l. Conf. on Very Large Databases (VLDB '96)*, pp. 251–262, Mumbai (Bombay), India 1996.

[Men84] Mendelzon A. O. Database States and Their Tableaux. In *ACM Trans. on Databases Systems* **9**:2, 1984, pp. 264–282.

[MM01] Mendelzon A. O. and G. Mihaila. Querying Partially Sound and Complete Data Sources. In *Proc. 20th ACM Symp. on Principles of Database Systems (PODS '01)*, Santa Barbara, California 2001, pp. 162–170

[PL00] Pottinger R. and A. Y. Levy. A Scalable Algorithm for Answering Queries Using Views. In *Proc. 26th Int'l. Conf. on Very Large Databases (VLDB '00)*, Cairo, Egypt 2000, pp.484–495.

[Rei78] Reiter R. On Closed World Databases. In *Logic and Databases*, H. Gallaire and J. Minker (Eds.), pp. 56–76, Plenum Press, New York, 1978.

[Ull97] Ullman J. D. Information Integration Using Logical Views. In *Proc. 6th International Conference on Database Theory (ICDT '97)*. Delphi, Greece 1997, pp. 19–40.

[Var86] Vardi M. Y. Querying Logical Databases. *Journal of Computer and System Sciences* 33, pp. 142–160, 1986.

[YL87] Yang H. Z. and P. A. Larson. Query Transformation for PSJ Queries. In *Proc. 13th Int'l. Conf. on Very Large Data Bases (VLDB '87)* Brighton, England, pp. 245–254.