

Intelligent Methods in Virtual Databases

Amihai Motro*, Philipp Anokhin and Jacob Berlin

Department of Information and Software Engineering
George Mason University, Fairfax, VA 22030, USA

Abstract

Considerable progress has been achieved in the area of virtual database systems, but substantial problems still persist. In this paper we discuss two current research directions: The resolution of extensional inconsistencies among databases participating in a virtual database, and the automatic incorporation of new information sources based on a process of discovery. Our approach to both problems involves to some degree what is often referred to as soft computing techniques.

1 Introduction

The integration of information from multiple databases has been an enduring subject of research for over twenty years. Indeed, while the solutions that have been advanced tended to reflect the research approaches prevailing at their time, the overall goal has remained mostly unchanged: to provide flexible and efficient access to information residing in a collection of distributed, heterogeneous and overlapping databases (more generally, the problem may involve other kinds of information sources as well).

1.1 Virtual Databases

A common approach to this problem has been to integrate the independent databases by means of a comprehensive *global scheme* that models the information contained in the entire collection of databases (for example, [7, 13, 3, 9]). This global scheme is fitted with a *mapping* that defines

*Email address for contacting the authors: ami@gmu.edu

the elements of the global scheme in terms of elements of the schemes of the member databases. Algorithms are designed to interpret queries on the global scheme. Such *global queries* are translated (using the information captured in the mapping) to queries on the member databases; the individual answers are then combined to an answer to the global query. The global scheme and the scheme mapping constitute a *virtual database*; the main difference between a virtual database and a conventional database is that whereas a conventional database contains data, a virtual database points to other databases that contain the data. An important concern is that this query processing method be *transparent*; i.e., users need not be aware that the database they are accessing is virtual.

1.2 Persisting Issues

Considerable progress has been achieved in the area of virtual database systems, but substantial problems still persist. In this paper we discuss our current research to address two such problems. Our approach to both problems involves to some degree what is often referred to as soft computing techniques.

(1) **Inconsistency resolution.** Much of the previous work on integration focused on the resolution of the semantic heterogeneity among the the member databases. The purpose was to reconcile among the different database designs, addressing problems caused by the use of different data models, different structures within the same model, different attribute (column) types, and even different semantics. However, when some of the member databases “overlap”, that is, they purport to provide the same information, it is not uncommon to receive conflicting information. We emphasize that such conflicting information is not due to different semantics or different data types, but is pure inconsistency. Inconsistencies prevent the virtual database system from constructing unique answers to global queries. In Section 3 we discuss flexible schemes that are being developed to address this problem.

(2) **Contribution discovery.** In most virtual database systems, the process of mapping the global scheme into member schemes is complex and costly. Consequently, such systems tend to be useful only when the community of member databases is small and stable. Our basic assumption is more contemporary, that virtual databases should be able to incorporate large numbers of member databases (also called here *information sources* or *data providers*); such providers might be asked to contribute only small portions of their repositories, and might be used for short durations only. Given the vast amount of information available and the cost of locating and incorporating such information into a virtual database, we are developing methods for *discovering* contributions with only limited human effort. This research direction is given preliminary discussion in Section 4.

The work described is within the environment of the Multiplex virtual database system, but is of general applicability to most such systems. We begin with a brief description of the Multiplex system.

2 Multiplex

The multidatabase system Multiplex [11] is an example of a virtual database system. The basic architecture of Multiplex is fairly simple. The virtual database consists of a *global scheme* (described in the relational model) and a *mapping table*. Each entry in this table is called a *contribution* and consists of two expressions. The first expression is a view (expressed in SQL) of the global scheme; the second expression is a query to one of the member databases (expressed in the language of that system). The result obtained from the second expression is assumed to be a materialization of the view described in the first expression. The complexity of these expressions can vary greatly: they could range from a complex calculation, to a statement that simply denotes the equivalence of two attribute names.

Four important characteristics of Multiplex are described below.

(1) **Full support for heterogeneity.** The simplicity and popularity of the relational model make it an ideal integration model, and the integrated view that Multiplex provides is indeed relational. Yet, there is no restriction on the underlying data models; the only requirement is that they communicate their results in tabular form. Consequently, the member databases in a Multiplex multidatabase may be relational, object-oriented, or, in general, stored in any software system that can respond to requests with tabulated answers.

(2) **Flexibility to model real-world situations.** The Multiplex model is distinguished by several *degrees of freedom* that allow it to model actual situations encountered in multidatabase applications. Specifically,

1. **Source unavailability.** Multiplex reflects the dynamics of multidatabase environments where member databases may become temporarily unavailable, and global queries might therefore be unanswerable in their entirety.
2. **Source inconsistency.** Multiplex accepts that requested information may be found in more than one database, and admits the possibility of inconsistency among these multiple versions.

3. **Limited integration.** Multiplex permits ad-hoc global schemes of limited scope, that cull from existing databases only the information relevant to a given application.

Intuitively, these degrees of freedom correspond to scheme mappings (from the global scheme to the member schemes) that are *neither total, nor single-valued, nor surjective*. We note that earlier approaches to global scheme integration were based on often unrealistic assumptions that existing database schemes could be integrated completely and perfectly in a single global scheme (i.e., mappings that are total, surjective, and single-valued; sometimes even one-to-one). The complexity of existing databases quite often renders this approach unrealistic. The aforementioned degrees of freedom are therefore important, as they represent a significant departure from earlier approaches.

(3) **Approximative answers.** Because Multiplex mappings are not total, global queries may be *unanswerable*; because the mappings are not single-valued and because there is no assumption of mutual consistency among the member databases, global queries may have *several candidate answers*. In both these situations, Multiplex defines *approximative answers*. The overall concern is that when a single authoritative answer is unavailable, a multi-database system should approximate the request as best as possible. The subject of approximative answers is related to inconsistency resolution and is discussed in more detail in Section 3.

(4) **Quick adaptation to evolving environments.** Present data environments may be highly dynamic; for example, newly discovered data sources may need to be incorporated, the structure of existing data sources may change, or existing data sources may need to be deleted altogether. In Multiplex such changes are easy to effect, by adding, removing or changing individual contribution entries in the mapping table.

3 Inconsistency Resolution

3.1 Intensional vs. Extensional Inconsistencies

Inconsistencies among information sources are of two kinds: intensional and extensional. *Intensional inconsistencies* (also referred to as semantic heterogeneity) focus on schematic differences. These include structural differences (e.g., telephone numbers stored in either one field or in several fields), unit differences (dollars vs. euros), type differences (e.g., numbers vs. strings), or differences in the semantics of attributes (e.g., yearly salary vs. monthly

salary). Each such intensional inconsistency requires an appropriate translation to convert one format to another (or both formats to a third format). Once a translation is defined, the resulting extensions are comparable. As already mentioned, much of the work in the area of database integration involved the resolution of this type of inconsistency (for example, [2, 5, 6]).

In contradistinction, *extensional inconsistencies* surface only *after* all intensional inconsistencies have been resolved, at a point where the participation information sources may be assumed to have identical intensional representation for all overlapping information. At this point it is possible that two information sources would provide different answers to the same query. For example, two databases may report two different dates of birth for the same person (even after their formats have been reconciled). This complementary problem of extensional inconsistencies has received much less attention.

To resolve such extensional inconsistencies we have been exploring three different approaches.

3.2 Lower- and Upper-Bound Estimation

Intuitively, this resolution method is based on *voting*. The existence of overlapping data sources implies that a global query can have multiple translations. For example, a query Q may have answers A_1, \dots, A_n , where each A_i is a set of data elements that satisfy the query Q .

Lacking any evidence of the superiority of any answer, all answers are assumed to be equally good.¹ Indeed, we assume that all answers are provided with the same guarantee of soundness and completeness [12]. That is, each answer is claimed to contain the whole truth (completeness) and nothing but the truth (soundness). The claim for soundness is interpreted as a *positive vote* issued by the answer to each of its data elements; the claim for completeness is interpreted as a *negative vote* issued by the answer to all the data elements that are not in that answer. Hence, each data element in the union of the answers receives n votes.

The data elements for which the vote is unanimously *yes* is adopted as a *lower bound* for the answer; that is, a “minimal answer” that is assumed to be included in the true answer. The data elements for which the vote was not unanimously *no* is adopted as an upper bound for the answer; that is, a “maximal answer” that is assumed to include the true answer.

Altogether, the true answer is estimated by this pair of answers; i.e., it is

¹In Section 3.4 this assumption will be relaxed.

assumed to be “sandwiched” between these two answers:

$$\langle \{d \mid d \text{ has only } \textit{yes} \text{ votes}\}, \{d \mid d \text{ does not have all } \textit{no} \text{ votes}\} \rangle$$

Consider, for example, a query on the age of employees, and these three answers:

$$\begin{aligned} A_1 &= \{(Joe, 28), (Betty, 30), (Larry, 42)\} \\ A_2 &= \{(Joe, 28), (Betty, 30), (Larry, 42)\} \\ A_3 &= \{(Joe, 31), (Betty, 30)\} \end{aligned}$$

The true answer is estimated from below by

$$\{(Betty, 30)\}$$

and from above by

$$\{(Betty, 30), (Joe, 28), (Joe, 31), (Larry, 42)\}$$

Note that although there is no apparent disagreement on the age of Larry, the assumption of completeness implies that, according to answer A_3 , $(Larry, 42)$ does not belong in the answer. Hence, this data element is not in the lower bound. These completeness assumptions may be relaxed if necessary; i.e., a source may be declared as sound but not complete, or complete but not sound (which will affect the votes).

Note also, that the “sandwich” approach constructed two answer “layers”. However, intermediate layers can also be constructed. In general, we can define answer layer j as all the answers that are supported by at least j positive votes. The two answers given earlier correspond to $j = n$ (the lower bound) and $j = 1$ (the upper bound). In this approach, the layer $j = \lceil n/2 \rceil$ corresponds to the data items that receive a majority of votes.

Lower and upper bound answers are used not only to resolve inconsistencies. More generally, lower and upper bounds can be regarded as the best answer the system can provide in the given circumstances. Consider these two examples. The best answer to a query on the names, salaries, departments and locations of all employees, could be the names and departments of *all* employees, the salaries of *some* employees, and the locations for *none* of the employees. As another example, the best approximative answer to a query on the employees who earn over 50,000, could be the employees who earn over 40,000. Note that the former approximative answer was “less” than what was requested, whereas the latter was “more” than what was requested, corresponding to “below” and “above” approximations.

3.3 Fusion

The bounding approach assumes that different data elements refer to different real-world objects, and hence does not attempt to *reconcile* among different answers. If we could conclude that $(Joe, 28)$, $(Joe, 28)$ and $(Joe, 31)$ describe the same real-world object, then we may prefer to reconcile them with $(Joe, 29)$, a data element that represents their average value. Such reconciliation is often referred to as *fusion* [1]. We define fusion as a function that maps a set of values to a single value.

A flexible fusion scheme enables users to select different fusion functions for different attributes. For example, in a query that includes the attributes telephone number, salary and age, one may want to fuse different telephone numbers with the number most frequently mentioned, different salaries with the highest, and different ages with the average.

In addition to such standard fusion functions, one should also be able to define custom functions. For example, given a set of multiple inconsistent answers, one may want to remove first answers that are considered to be improbable (e.g., statistical outliers), before fusing them using a standard function.

Fusion requires a preliminary phase of *identification*. Within the information collected from the different sources, one must identify the “descriptions” (e.g., records) that pertain to the same real-world object. This is usually accomplished with database keys: assuming that key fields are free of errors, records that have the same key value, but are not identical in other fields, exhibit inconsistency and are fused. More ambitiously, identification can also be done by *clustering* the different descriptions according to a definition of “description distances”; members of each cluster exhibit inconsistency and are fused.

3.4 Property-based Resolution

The two methods described so far treat the different information sources equally, without allowing users to express any preference. In reality, however, information sources are often associated with various *properties* that render them particularly attractive in certain circumstances. The properties we consider include:

1. **Timestamp:** A source may have an associated timestamp that indicates the time in which the information was entered. Usually, higher timestamps are preferred because they reflect more current information.

2. **Cost:** A source may have an associated cost that indicates the price of the information. It could also mean the time necessary to retrieve this information. Usually, lower costs are preferred.
3. **Quality:** A source may have an associated level of quality. This parameter may indicate the level of accuracy or completeness of the information. Higher quality is preferred.
4. **Authority:** A source may be associated with an authority level. Information provided by source of a higher authority is usually preferred.

In other words, the multiplicity of sources creates a conflicting information environment from which *many different answers could be constructed*. This third method allows users to describe the answer they prefer, based on properties; for example, the cheapest answer, the most recent answer, and so on.

Note that it is also possible to express mixed preferences. In the previous example of a query that includes the attributes telephone number, salary and age, one may want to retrieve telephone numbers from the most recent source, salaries from the source with highest authority, and ages from the cheapest source (possibly reflecting the lesser importance of this attribute).

Once an answer is created, the system also computes its derived properties from the properties of its components. The new properties are reported to the user, and can also be used in case this information is involved in further transactions.

4 Discovery of Contributions

4.1 Specification vs. Discovery

A time consuming task in a virtual database system such as Multiplex is the incorporation of new information sources into a virtual database. This involves an examination of both the global and member schemes and the definition of expression pairs: a global view and an equivalent query to the source.

Early virtual integration systems were developed under the assumption that the community of member databases in a virtual database is both *small* and *stable*. That is, the number of member databases is relatively low, and databases are not added or removed. Under this assumption, scheme mapping is a one-time activity, performed at the time the virtual database is designed.

As described earlier, the environment assumed by Multiplex is considerably different, since the community of member databases is neither small nor stable. It is assumed that the number of member databases could be rather large, and that frequent changes in this community take place, with new databases added and existing databases either modified or deleted altogether.

To facilitate the incorporation of new databases, a process is being developed that will automate the incorporation of a new database into the virtual database.

The basic idea is to subject each new “candidate database” to a *discovery* process, in which the database is scanned for possible contributions to the virtual database. We shall also refer to this discovery process as *recruitment* of new contributions.

The discovery process assumes that the virtual database already includes several member databases and hence there are several *examples* of previously mapped contributions. In other words, the virtual database is launched manually, but after some time can be made to grow autonomously.

Each contribution example includes two components: a global view and a member query. The member query, however, includes two components: a definition (an intensional description) and an extension.

A simplified description of this process is given below. This version assumes that all views (both global and local) are selection-projection expressions (in particular, they involve a single relation only). The environment consists of

1. A virtual database scheme R_1, \dots, R_n .
2. A mapping table C_1, \dots, C_m (i.e., a set of contribution examples).
3. Each contribution example C_k specifies
 - (a) A global relation scheme R_i .
 - (b) A selection-projection expression α on relation R_i .
 - (c) A member relation scheme S_j .
 - (d) An extension s_j of relation S_j .
 - (e) A selection-projection expression β on relation S_j (the number of columns produced by α and β is identical).

Let T be new member relation with extension t . The task is to discover views of T that are likely contributions to some relation R_i .

Discovering a new contribution in T amounts to finding a selection - projection view of T that is similar to the examples that have already been mapped to R_i . Furthermore, the view that we find must exceed a predefined threshold of acceptability. We handle this task in two steps:

1. **Projective transformation:** Discovering the *columns* of the new contribution.
2. **Selective transformation:** Discovering the *rows* (the selection specification) of the new contribution.

We find the most suitable projective transformation of T by means of Naive Bayes (NB) text classification. The basic approach for NB text classification is discussed in [8]; however, substantial modifications are necessary for the task at hand. We present here a three-step overview of our approach to projective transformations.

In the learning phase, the NB classifier examines the extensions of the contribution examples C_1, \dots, C_m and derives the probabilities of the data occurring within the mapped attributes. Next, given the new relation T , the classifier considers each column of T and each column of R_i and determines the probability that the former is an “instance” of the latter. The output of the classifier is a weighted bipartite graph which matches columns of the new relation T with columns of the global relation R_i , with each edge carrying the corresponding probability. Edges with weights below a threshold are deleted, and resulting disconnected nodes are deleted as well (these represent columns of T and R_i that are “projected-out”). Finally, an $O(n^3)$ algorithm for maximum weighted matching in bipartite graphs [4] is used to find an optimal matching.

At this point, if successful, we have found a projection of the candidate relation and a projection of the global relation that offer the best “match”. The matching process considered each column independently, without considering any “relationships” among the columns; roughly speaking, the matching of a column of T with a column of R_i is, in effect, a conclusion that the two columns are defined over similar domains of values. Hence, we have found a projection of T that conforms to previous contributions with respect to the “types” of its columns.

The rows of this projection now have to be pruned to retain only those rows that “resemble” rows in the relevant examples. However, because the new contribution should be applicable even after the candidate relation is updated, an *intensional* description of the desirable rows is needed (not simply the rows themselves). Note that it is possible that in this second phase the contribution would be rejected altogether, because the new rows do not

resemble rows in the relevant examples (although they are formed over appropriate columns).

The selection process combines two techniques. First, it may be possible to learn from the intensional descriptions of the relevant examples. If a correspondence between the columns of the new relation and the columns of a previous example can be established, then the selection specification of that example may suggest a possible selection specification for the new relation. It should be noted, however, that a selection specification of an example may use columns unique to that example, in which case the intensional description is not useful. An example is a contribution that involves a table with columns A, B, C, D , a selection specification $D = x$, and a projection on columns A, B, C , and a candidate table T with columns that are “equivalent” to A, B, C , but without an equivalent to column D .

A second technique applies machine learning methods to learn the *horizontal patterns* of the examples. Tuples of the candidate table are selected if approved by the resulting classifier. An intensional description of the approved rows can be generated by various intensional answering techniques [10].

The process of recruiting contributions to a virtual database involves uncertainty. In the various stages of discovery the likelihood of a successful discovery is estimated, and eventually all discovered contributions are tagged with an appropriate level of confidence. In this regard, we observe a connection between the two research directions discussed in this paper. A discovered contribution that overlaps with previous contributions may introduce inconsistencies. The level of inconsistencies for overlapping data may be used as a level of affirmation of the validity of this contribution.

5 Summary

The virtual integration of heterogeneous information sources has been the subject of continuous research for over twenty years. In this paper we described two current research directions, both involving some measures of soft computing: The resolution of extensional inconsistencies and the automatic incorporation of new information sources based on a process of discovery.

References

- [1] M.A. Abidi and R.C. Gonzalez, editors. *Data Fusion in Robotics and Machine Intelligence*. Academic Press, 1992.

- [2] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, Dec. 1986.
- [3] Y. Breitbart, P. L. Olson, and G. R. Thompson. Database integration in a distributed heterogeneous database system. In *Proceedings of the IEEE Computer Society Second International Conference on Data Engineering*, pp. 301–310, 1986.
- [4] Z. Galil. Efficient Algorithms for Finding Maximum Matching in Graphs. *ACM Computing Surveys*, 18(1):23–38, Mar. 1986.
- [5] D. Fang, J. Hammer, and D. McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pp. 136–143, 1991.
- [6] R. Krishnamurthy, W. Litwin, and W. Kent. Interoperability of heterogeneous databases with semantic discrepancies. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pp. 144–151, 1991.
- [7] T. A. Landers and R. L. Rosenberg. An overview of Multibase. In H.J. Schneider, editor, *Distributed Databases*, North-Holland, 1982.
- [8] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [9] A. Motro. Superviews: Virtual integration of multiple databases. *IEEE Transactions on Software Engineering*, SE-13(7):785–798, July 1987.
- [10] A. Motro. Intensional Answers to Database Queries. *IEEE Transactions on Knowledge and Data Engineering* 6(3):444–454, June 1994.
- [11] A. Motro. Multiplex: A Formal Model for Multidatabases and Its Implementation. In *Proceedings of the NGITS 99, Fourth International Workshop on Next Generation Information Technologies and Systems*. Lecture Notes in Computer Science No. 1649, Springer-Verlag, pp. 138–158, 1999.
- [12] A. Motro and I Rakov. Not all answers are equally good: Estimating the quality of database answers. In *Flexible Query-Answering Systems*, pp. 1–21. Kluwer, 1997.
- [13] M. Templeton, D. Brill, S. K. Dao, E. Lund, P. Ward, A. L. P. Chen, and R. McGregor. Mermaid — a front-end to distributed heterogeneous databases. In *Proceedings of IEEE*, volume 75, number 5, pp. 695–708, May 1987.