Multiplex, Fusionplex and Autoplex — Three Generations of Information Integration

Amihai Motro, Jacob Berlin, Philipp Anokhin
Information and Software Engineering Department
George Mason University
Fairfax, VA

Abstract

We describe three generations of information integration systems developed at George Mason University. All three systems adopt a *virtual database* design: a global integration schema, a mapping between this schema and the schemas of the participating information sources, and automatic interpretation of global queries. The focus of Multiplex is rapid integration of very large, evolving, and heterogeneous collections of information sources. Fusionplex strengthens these capabilities with powerful tools for resolving data inconsistencies. Finally, Autoplex takes a more proactive approach to integration, by "recruiting" contributions to the global integration schema from available information sources. Using machine learning techniques it confronts a major cost of integration, that of mapping new sources into the global schema.

1 Introduction

The problem of integration of multiple information sources is almost a quarter of a century old: early works include [21, 5, 15]; more recent systems include [8, 10, 11, 7]. Over the years this problem has been redefined repeatedly to address new, contemporary challenges (e.g., source heterogeneity, source independence), and has been attacked with new methodologies (e.g., software wrappers and mediators, intelligent agents, neural networks, machine learning).

With the evolution of the Internet, new challenges have surfaced, of which we mention here four: (1) *scale*: integrating information from a very large number of participating sources, (2) *dynamics*: integrating information in data environments that are evolving continuously, (3) *inconsistency*: detecting and resolving mutual inconsistencies among different information sources, and (4) *automation*: automating some of the more laborious tasks in the integration process.

These challenges motivated research on three generations of information integration systems, called Multiplex, Fusionplex and Autoplex. Of these, the latter two systems make increased use of "semantics", which we interpret here liberally as information *about* the individual sources. In Fusionplex, it is information on the *quality* of the content, and it is *provided*; in Autoplex, it is information on its *meaning*, and it is *discovered*.

2 Multiplex

Multiplex [16] defines a formal model for integrating a collection of databases (the *local* databases) by means of an integration schema (a *global* schema). The integration schema and the collection databases are related by means of a *schema mapping*. A schema mapping is a set of view pairs, where the first element of each pair is a view of the global schema, and the second element is a view of a local database. The view in the second position (the local view) materializes the view in the first position (the global view). Hence, the global schema defines the information that is *potentially available*, whereas the global views define the information that is *actually available*. Consequently, queries against the global schema must be translated to queries over the global views.[1] Figure 1 shows a high level overview of Multiplex.

As the Multiplex model is the basis for the two subsequent systems, it is fitting to discuss briefly some of its main features.

(1) *Formal assumptions of integrability*. Most integration methods operate under tacit assumptions regarding the mutual consistency of the schemas and instances of the underlying databases. Multiplex distinguishes between two kinds of inconsistency, intensional and extensional, and formulates two assumptions of integrability: the Schema Consistency As-

[1] This approach was subsequently categorized as GLAV [9].

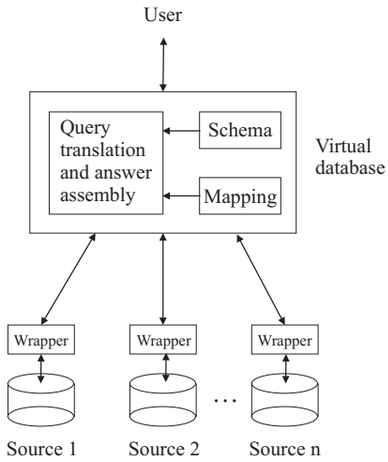

Figure 1: Multiplex architecture.

sumption and the Instance Consistency Assumption. These explicit assumptions, absent from previous work, provide an unambiguous framework for integration. They postulate the existence of a hypothetical database (schema and instance) that models the real world perfectly, and they govern the relationships between actual databases and this ideal database.

The Schema Consistency Assumption states that all database schemas are *derivatives* of the real world schema. That is, in each database schema, every relation schema is a view of the real world schema. The meaning of this assumption is that the different ways in which reality is modeled are all correct; i.e., there are no *modeling errors*, only *modeling differences*. To put it in yet a different way, all intensional inconsistencies among the independent database schemas are reconcilable. Reconciliation is achieved by the aforementioned schema mappings.

The Instance Consistency Assumption states that all database instances are *derivatives* of the real world instance. That is, in each database instance, every relation instance is derived from the real world instance. The meaning of this assumption is that the information stored in databases is always correct; i.e., there are no factual *errors*, only different *representations* of the facts. Any such differences are attributable solely to schema differences, and they are reconciled as a consequence of the schema reconciliation.

Multiplex assumes that the Schema Consistency Assumption *holds*, meaning that all differences among database schemas are reconcilable. Schema integration and reconciliation are achieved simultaneously by creating an integration (global) schema (which is yet another derivative of the ideal database schema) and defining mappings between it and the schemas of the local databases. On the other hand, Multiplex assumes that the Instance Consistency Assumption does *not* hold, allowing the possibility of irreconcilable differences among database instances. This means that the database instances are *not* assumed to be derivatives of the real world instance.

(2) *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 therefore relational. Yet, there is no restriction on the data models of the collection databases; the only requirement is that they communicate their results in tabular form. Thus, the member databases may be relational, object-oriented, or, in general, stored in any software system that can respond to requests with tabulated answers. A view in the second position of a mapping pair is therefore any expression that is understood by a local database and results in a table of the same number of columns as its corresponding global view.

(3) *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 multi-database applications. Specifically,

1. **Source unavailability**. Multiplex reflects the dynamics of multi-database 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. **Ad-hoc integration**. Multiplex permits ad-hoc global schemas of limited scope, that cull from existing databases only the information relevant to a given application.

Intuitively, these degrees of freedom correspond to mappings (from the global schema to the member schemas) that are *neither total, nor single-valued, nor surjective*. These degrees of freedom represent a significant departure from earlier approaches to global schema integration, which were based on often unrealistic assumptions that existing database schemas could be integrated completely and perfectly in a single global schema (i.e., mappings that are total, surjective, and single-valued; sometimes even one-to-one). The complexity of existing databases quite often renders these approaches unrealistic.

(4) *Approximate 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 *approximate answers*. The overall concern is that when a single authoritative answer is unavailable, a virtual database system should approximate the request as best as possible. For example, 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 answer to a query on the employees who earn over 50, could be the employees who earn over 40. Note that the former answer was "less" than requested, whereas the latter was "more" than requested, corresponding to "below" and "above" approximations.

(5) *Quick adaptation to evolving environments.* Present data environments may be highly dynamic: newly discovered data sources may need to be incorporated, the structure of existing sources may change, or existing sources may need to be deleted altogether. In Multiplex such changes are easy to effect. The integration consists of providing pairs of equivalent views: a view on the global database ("the information offered"), and a view on a member database ("how it is materialized"). The complexity of these views can vary greatly: they could range from a complex calculation, to a statement that simply denotes the equivalence of two attribute names.

(6) *Integration server.* The software architecture of Multiplex provides for a flexible *integration service*. A remote user wishing to integrate several information sources (possibly, sources from this user's own enterprise), logs into the server, provides it with the appropriate definitions (i.e., a global schema and some view pairs), and can begin using its integration services right away. Future updates are fairly simple; for example, a new source may be incorporated with as little as a pair of views, and a structural change in an existing source requires only that the views it contributes be redefined.

3 Fusionplex

Multiplex detects data inconsistencies and attempts to resolve them. However, inconsistency is approached at the "record level": inconsistency occurs when a global query results in two or more different *sets of records*. Multiplex then proceeds to construct an *approximation* of the true set of records, with a *lower bound* set of records (a *sound* answer) and an *upper bound* set of records (a *complete* answer). The two estimates are obtained from the conflicting answers through a process similar to voting. The lower bound set is contained in the upper bound set, and the true answer is estimated to be "sandwiched" between these two approximations.

A limitation of this approach to inconsistency resolution is that Multiplex regards two different records as describing entirely different objects, even if they are almost identical (e.g., identical in all but one "minor" field). Consequently, when two such records are suggested by two information sources, there is no attempt to recognize that these might be two descriptions of the same object, and therefore no attempt to reconcile their conflicting values. The two records are simply both relegated to the upper bound estimate.

Fusionplex [1] seeks to rectify this limitation. The main principle behind Fusionplex is that "all data are not equal." The data environment is not "egalitarian," with each information source having the same qualifications (as assumed by Multiplex and other systems). Rather, it is a diverse environment, in which information providers have their individual advantages and disadvantages. Some data is more recent, whereas other is more dated; some data comes from authoritative sources, whereas other may have dubious pedigree; some data may be inexpensive to acquire, whereas other may be costlier. To resolve conflicts, Fusionplex looks at the qualifications of its individual information providers. Thus, it uses meta-data to resolve conflicts among data.

Every Internet user is often confronted with the need to choose between alternatives: Which is the most trustworthy source? Which is the most reliable download site? Which is the least expensive newswire service? Some of these meta-data may be provided by the source itself (e.g., date of last update, cost), other meta-data may be obtained informally from other Internet users, and there are also Web sites that are dedicated to calculating the quality of information and services provided by other sites (often through the evaluations of fellow users). So it is not far-fetched to assume that in the near future, given the Internet's continuing, fast-paced expansion, such meta-data will become commonplace, possibly even in a standard format. In a more restricted information environment, comprising perhaps only a few dozen sources (perhaps with a focus on a particular subject, such as business or medicine), it is quite conceivable that the virtual database administrator will assign meta-data scores to its sources, and will keep updating these scores.

Fusionplex's term for such meta-data is *features*. Examples of features include (1) *timestamp*: the time

when the information in the source was validated, (2) *cost*: the time it would take to transmit the information over the network or the money to be paid for the information, (3) *accuracy*: probabilistic information that denotes the accuracy of the information, (4) *availability*: the probability that at a random moment the information source is available, and (5) *clearance*: the security clearance level needed to access the information.

The other guiding principle of Fusionplex is that inconsistency resolution is a process in which users must be given a voice. Depending on their individual preferences, users must be allowed to decide how inconsistencies should be resolved. Decisions are made in two phases: Some decisions are made ad-hoc at query-time, other decisions are more enduring and control all subsequent queries. One important query-specific decision, implemented by means of a vector of feature weights, amounts to a subjective definition of *data quality*; it allows the system to *rank* the competing values according to their *utility* to the user. Other query-specific parameters are thresholds of acceptance with which users can ensure minimal performance of the data, excluding from the fusion process data that are too old, too costly, or lacking in authority. It also allows users to reject data that are too high, too low, or obvious outliers. Yet another decision is the particular fusion function to be used for a particular attribute; for example, *average*, *maximum*, or simply *any*. Several simple extensions to SQL are all that is needed to allow users to state these resolution parameters. Altogether, Fusionplex provides its users with powerful and flexible, yet simple, control over the fusion process.

The fusion process is multi-step. First, Fusionplex determines the global views that are *relevant* to the given query. A global view is relevant if its projection attributes intersect with the query's projection attributes, and its selection predicate does not contradict the query's selection predicate. Next, the relevant contributions are materialized and "polished": unnecessary attributes are removed, necessary but unavailable attributes are filled with nulls, etc. The union of these contributions is called a *polyinstance*. Next, the tuples of the polyinstance are clustered in *polytuples*. The members of each polytuple are different "versions" of the same information. Once data inconsistencies have been detected, they need to be resolved; i.e., every polytuple is reduced to a single tuple. This process is performed in two passes. In the first pass, the members of each polytuple are ranked and purged according to user-specified preferences (utility). In the second pass, in each polytuple, in each attribute, remaining values are purged and then fused in a single value. User preferences and attribute-specific fusion functions necessary for this phase are given in the query.

Optimal fusions. As explained above, utility, the overall value of a data item to its user, allows us to rank a set of inconsistent answers, and offer the top-ranked answer as a preferred answer. In addition, utility can determine whether a fusion value is indeed better than the initial values, by calculating its utility and comparing it to the utilities of the initial values. Moreover, common optimization methods may be used to discover the *best* fusion: the fusion formula that optimizes the utility [17].

4 Autoplex

Although the architecture of Multiplex and Fusionplex facilitates the incorporation of new information sources into the virtual database, the process of locating new sources and identifying their potential contributions to a given global schema is time consuming and costly. It may be argued that this difficulty is the main reason why virtual database systems do not scale up well to environments in which the number of sources is very large, and which are constantly changing (such as the World Wide Web). Indeed, it is commonly agreed upon in this field and the related field of data warehousing that future research should find ways to automate the integration and maintenance process [12, 20].

Given the vast amount of information available and the cost of locating and incorporating such information into a virtual database, we have been developing a system, called Autoplex [2], for *discovering* member schemas and incorporating them into the global schema with only limited human effort. Based primarily on Bayesian learning, the system acquires probabilistic knowledge from examples that have already been integrated into the virtual database (i.e., as in Multiplex). The approach thus follows a supervised learning paradigm. From the acquired probabilistic knowledge, the system can discover "content contributions" in new, previously unseen information sources. A comparable machine learning-based method for automatic discovery of schema mappings in the context of information integration is described in [6].

The challenge of Autoplex can be stated in terms of the Multiplex system as follows. Given

1. A relation schema $R = (X_1, \ldots, X_n)$. This is the virtual database. Each column X_i of R is labeled as either *required* or *optional*.
2. A set of contribution examples, each consisting

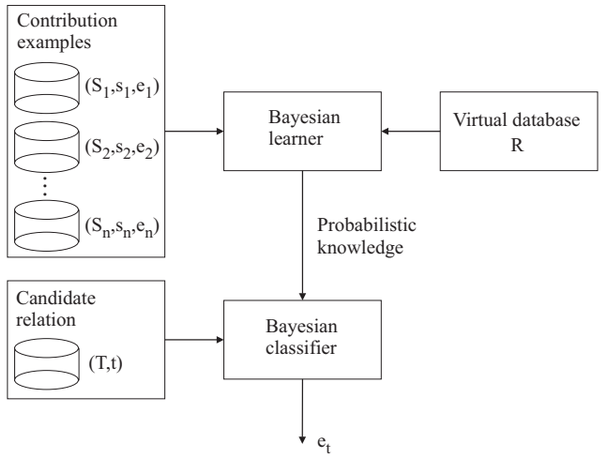

Figure 2: Autoplex architecture.

of a relation schema $S = (Y_1, \ldots, Y_k)$, a relation instance s of schema S, and a selection-projection expression e on relation S that defines a contribution to R.

3. A new, previously unseen relation schema $T = (Z_1, \ldots, Z_m)$ and a relation instance t of T. We shall often refer to T as a *candidate* relation.

Determine whether T contains an acceptable contribution to R, and if so, find the expression e_t that defines it. An acceptable contribution is one that satisfies all required columns in R and exceeds a predetermined threshold.

A high level overview of the Autoplex architecture is shown in Figure 2. This architecture includes two main components: a learner and a classifier.

The learner. The Bayesian learner is given the virtual relation schema R and a set of contribution examples. Each example consists of a local relation schema S, an instance of this schema s, and a selection-projection expression e on schema S. The extension of this expression in the instance s generates rows for R. The Bayesian learner uses this information to acquire probabilistic knowledge on features of the examples. This knowledge is stored on secondary storage in efficient data structures for future use.

The classifier. A candidate relation schema T and instance t are provided by a new local database as inputs to the Bayesian classifier. This classifier uses the acquired probabilistic knowledge to infer a selection-projection view that defines a contribution of T to R.

In Figure 2, classification is a single process. More precisely, classification comprises four different phases. Given a candidate relation T and a global relation R, in the first phase the classifier considers each column of T and each column of R and determines the probability that the former is an "instance" of the latter. In the second phase, the classifier finds an assignment of the local relation to the global relation that maximizes total column probabilities. This problem is modeled as maximum weighted matching in a bipartite graph. At this point, if successful, the classifier has found a projection of the candidate relation that offers the best "match."

The rows of this projection now must be pruned to retain only rows that "resemble" rows in the examples. In the third phase, the classifier partitions the instance t into two sets of rows: those that should be included in the contribution and those that should be excluded from it. Because the new contribution should be usable even after the extension of t is updated, an *intensional* description of the included rows is desirable. In phase four, a classification tree algorithm is used to derive a selection predicate that corresponds to the set of included rows.

The final output of this process is a selection-projection expression, or *False* in the event an acceptable contribution could not be found. Our approach is biased to search within the space of projections and selections on the candidate relation. Furthermore, our approach is greedy in that we search for a projection followed by a selection. Clearly, other approaches could produce better results. For example, a better projection may be found if some rows are first removed from the candidate. Furthermore, a better mapping may be found if we allow *transformations* of the values in the candidate (such as conversions of measurement units). Our approach represents a reasonable tradeoff between the advantage of more a powerful search (which will discover additional contributions) and the need to keep the problem tractable.

To measure performance, the discovery process of Autoplex is treated as four Boolean decisions:

1. *Column Mapping:* For each combination of a candidate column and a virtual column, decide whether or not the columns match.
2. *Table Mapping:* For each combination of a candidate table and a virtual table, decide whether or not the tables match.
3. *Tuple Partitioning:* For each row in a candidate table, decide whether to assign it to the contributing set or to the non-contributing set.
4. *Tuple Selection:* After inferring a selection predicate from the partitioned rows, decide for each row whether or not it satisfies the predicate.

Obviously, the last two decisions are made only if we have made a positive table mapping decision.

For each type of decision, the output falls into four disjoint categories: *true positives*, *false negatives*, *false positives*, and *true negatives*. The proportion of true positives among the cases thought to be positive measures the accuracy of Autoplex when it decides *True*. The proportion of positives detected by Autoplex among the complete set of positives measures the ability to detect positives. These are, respectively, the *soundness* and the *completeness* of the discovery.

An experiment involving an integration schema of 3 relations, and 15 different information sources resulted in these four decisions having soundness and completeness rates better than 80%.

TupleRank. Like all discovery processes, the Autoplex process is intrinsically probabilistic; that is, each discovery is associated with a specific value that denotes assurance of its appropriateness. Consequently, the rows in a discovered virtual table have mixed assurance levels, with some rows being more credible than others. In [3] we argue that rows in discovered virtual databases should be ranked, and we describe a ranking method, called TupleRank, for calculating such a ranking order. Roughly speaking, TupleRank calibrates the probabilities calculated during a discovery process with historical information about the performance of the system.

Automatch. The machine learning techniques used in Autoplex were also applied to the general problem of *schema matching*: finding mappings between the attributes of two semantically related database schemas. The schema matching problem[2] is an important, current issue for many database applications such as schema integration, data warehousing, and electronic commerce [14, 19, 13]. Unfortunately, schema matching remains largely a manual, labor-intensive process. Furthermore, the effort required is typically linear in the number of schemas to be matched; the next pair of schemas to match is not any easier than the previous pair. Thus, database applications that require schema matching would scale up to much larger communities of member sources if the schema matching "bottleneck" was broken by automating the matching process.

Automatch [4] automates the schema matching process. Based primarily on Bayesian learning, Automatch acquires probabilistic knowledge from examples of schemas that have been "mapped" by domain experts in a knowledge base of database attributes called the *attribute dictionary*. Roughly speaking, this dictionary characterizes different attributes by means of their *possible values* and the *probability estimates* of these values. Furthermore, the dictionary may be extended to contain any attribute meta-data that has a probabilistic interpretation (e.g. attribute names or string patterns).

When presented with a pair of "client" schemas that need to be matched (and their corresponding database instances), Automatch matches them via its dictionary. Using probabilistic methods, an attempt is made to match every attribute of one client schema with every attribute of the other client schema, resulting in individual "scores." An optimization process based on a Minimum Cost Maximum Flow network algorithm finds the overall optimal matching between the two client schemas, with respect to the sum of the individual attribute matching scores.

To overcome the problem of very large dictionaries caused by very large attribute domains, Automatch employs statistical *feature selection* techniques to learn an efficient representation of the examples. That is, each attribute is represented with a minimal set of most informative values. Thus the attribute dictionary is made human understandable through aggressive reduction in the number of values. Although the example schemas may contain many thousands of values, we are able to focus learning on a very small subset, consisting of as few as 10% of the initial values.

The results of initial experimentation with Automatch are encouraging as they show performance that exceeds 70% (measured as the harmonic mean of the soundness and the completeness of the matching process). Although the attribute dictionary was built for this particular system, we conjecture that it could be employed as a knowledge asset in other schema matching systems.

5 Conclusion

The ever-increasing amount of information collected and made available requires the continuous development of matching information management solutions. In confronting this information explosion, information integration has emerged as an important discipline.

A most pressing issue in information integration is the need to *automate* this laborious process. In this paper we described solutions to several problems associated with automatic integration, including schema matching, proactive recruitment of information for virtual databases, and resolution of inconsistencies.

Our solutions combined traditional optimization techniques with techniques based on machine learning, showing that such a mix holds promise for other issues in automatic integration.

[2] And the related problem of ontology merging [18].

References

[1] P. Anokhin and A. Motro. Fusionplex: Resolution of data inconsistencies in the integration of heterogeneous information sources. Tech. Rep. ISE-TR-03-06. Dept. of Information and Software Engineering, George Mason Univ., 2003.

[2] J. Berlin and A. Motro. Autoplex: Automated discovery of contents for virtual databases. In *Proc. of CoopIS 01, Sixth IFCIS Int. Conf. on Cooperative Information Systems*. Lecture Notes in Computer Science No. 2172, pp. 108–122, 2001.

[3] J. Berlin and A. Motro. TupleRank: Ranking discovered content in virtual databases. Tech. Rep. ISE-TR-02-03. Dept. of Information and Software Engineering, George Mason Univ., 2002.

[4] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. of CAiSE 02, the 14th Int. Conf. on Advanced Information Systems Engineering*. Lecture Notes in Computer Science No. 2348, pp. 452–466, 2002.

[5] U. Dayal and H. Hwang. View definition and generalization for database integration in a mutlidatabase system. *IEEE Transactions on Software Engineering* SE-10(6), pp. 628–644, 1984.

[6] A. Doan, P. Domingos , and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pp. 509–520, 2001.

[7] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2), pp. 117–132, 1997.

[8] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pp. 539-542, 1997.

[9] A. Y. Halevy. Answering queries using views: A survey. VLDB Journal, 10(4):270–294, 2001.

[10] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, I. Muslea, A. Philpot, and S. Tejada. The Ariadne approach to Web-based information integration. Int. Journal of Cooperative Information Systems, 10(1-2):145-169, 2001.

[11] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the 22nd Int. Conf. on Very Large Databases*, pp. 251–262, 1996.

[12] A. Levy, C. Knoblock, S. Minton, and W. Cohen. Information integration. *IEEE Intelligent Systems*, 13(5):12–24, 1998.

[13] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. of the 27th Int. Conf. on Very Large Databases*, pp. 49–58, 2001.

[14] R. Miller, L. Haas, and M. Hernández. Schema mapping as query discovery. In *Proc. of the 26th Int. Conf. on Very Large Databases*, pp. 77–88, 2000.

[15] A. Motro. Superviews: Virtual integration of multiple databases. *IEEE Transactions on Software Engineering*, SE–13(7):785–798, 1987.

[16] A. Motro. Multiplex: a formal model for multidatabases and its implementation. In *Proc. of Fourth Int. Workshop on Next Generation Information Technologies and Systems*, Lecture Notes in Computer Science No. 1649, pp. 138–158, 1999.

[17] A. Motro, P. Anokhin and A. C. Acar. Utility-based resolution of data inconsistencies. In *Proc. of Int. Workshop on Information Quality in Information Systems* (in conjunction with SIGMOD 2004), pp. 35–43, 2004.

[18] N. F. Noy and M. A. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies, 59(6):983–1024, 2003.

[19] E. Rahm and P. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft, 2001.

[20] E. A. Rundensteiner, A. Koeller, and X. Zhang. Maintaining data warehouses over changing information sources. *Communications of the ACM*, 43(6):57–62, 2000.

[21] 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 *Proc. of the IEEE*, 75(5):695–708, 1987.