Integration of Disparate Information Sources: A Short Survey

Mong Li Lee, Stéphane Bressan, Cheng Hian Goh
National University of Singapore
{leeml,steph,gohch}@comp.nus.edu.sg

Raghu Ramakrishnan
University of Wisconsin
raghu@cs.wisc.edu

March 31, 1999

Abstract

In this paper, we try and give a brief introduction to the Gestalt approach to the integration of disparate information sources and to the Context Interchange approach to the automatic identification and resolution of semantic conflicts. We also attempt present a survey of the modern solutions to the problems of query processing in Internet networked data sources, which both Gestalt and Context Interchange leverage and attempt to adapt and improve.

1 Introduction

The growth of the Internet has not only greatly simplified the access to existing sources of information but also motivated the creation of numerous new sources. Paradoxally, this growth has made difficult the tasks of finding, extracting, and aggregating relevant information. Gateways, protocols like Object Database Connectivity (ODBC), and standards like eXtensible Markup Language (XML) are gaining popularity for they potentially solve the first issues of heterogeneity and connectivity. As a result, we can anticipate for the near future that large numbers of *structured sources* will be available. The database community has much to offer to enable users to search and query this highly distributed collection of structured data over the web.

We call independent information sources, which are both distributed and potentially heterogeneous, *disparate* information sources. We assume that the disparate information sources represent and export data in a common data model. This may only be the case after the source has been complemented by a possibly ad hoc interface called a *wrapper*, which maps queries and results between the data model of the source and the data model assumed. In the sequel, for the sake of simplicity and because it is so far the most robust model for business applications, the common data model assumed is the relational model.

Roughly speaking, we can distinguish between three types of data heterogeneity leading to conflicts when attempting to integrate disparate information sources: *intensional heterogeneity*, *semantic heterogeneity*, and *schematic heterogeneity*.

Intensional heterogeneity refers to the differences in the information content of the different sources. It is a curse and a blessing. Indeed, the quantity of information available on the Internet is to a great extent only available because of the heterogeneity and number and diversity of interests and application domains. It provides us with the opportunity to leverage this wealth and provide customized information services in the form of *Gestalten*. However, and here is one of the main challenges, the integration and reuse of the information available from the intensionally heterogeneous sources requires the global comprehension of the discrepancies in the domain and constraints assumed by the disparate sources. Such a comprehension is usually constructed in the *Gestalt* design phase and takes the form of the world view (global schema, domain model). The use of domain specific ontologies developed by experts allows the resolution of intensional conflicts as new sources are joining the *Gestalt*

Semantic heterogeneity refers to the fact that data in the different systems may be subject to different interpretations, even when data types, labels, and generally schemas are identical. Three main types of conflicts can arise from semantic heterogeneity:

1. *Naming and format conflicts* arise when different (often idiosyncratic) conventions are used to report individual data elements. For instance, the name of a series in different sources can be reported differently: "LNCS", "Lecture Notes in Computer Science". Another example is the format of dates: "12-01-99", "01/12/99", etc. These differences may not always follow systematic rules and may require conversion tables to be maintained or used. The date example may also suggest that, in some cases, the differences may be inferred from the data itself. This is of course rarely the case: what date is "04.03.65"?
2. *Scaling and unit conflicts* arise from the differences in the choice of units of measurements and scales for numerical data. The examples are numerous: Centigrade, Celsius, or Fahrenheit degrees, inches, and centimeters, etc [1]. The conversions are public knowledge.
3. *Confounding conflicts* arise from the confounding of different concepts in one particular source. For instance, certain sources may provide a currency exchange rate which is an approximation (average value) that can be used for rule of thumb monetary conversions (e.g. in the "lonely planet" "facts for the visitor" table). In this average value actual exchange rates as of individual days are confounded.

A common solution to semantic conflicts is based on the choice of a reference interpretation (standard). We will argue in the sequel that this approach is heavy and may not always be practical.

Finally, *Schematic heterogeneity* has been widely studied (see, for instance, [28]). Let us give examples of various sorts of conflicts that can possibly arise from schematic heterogeneity:

1. *Data type conflicts* arise when the different systems use different implementations of the primitive types (e.g. ASCII versus EBCDIC, "big endian" versus "small endian"), or different primitive types (e.g. presence, and compatibility of types "date"). Such conflicts are usually solved by the design of an appropriate communication protocol taking care of the various necessary conversions and/or offering a standard representation for a set of basic types (e.g ODBC, XDR, etc.).

[1] curious reader can look into the "desktop reference" for a survey of commonly used units and scales for various measurements

2. *Labeling conflicts* refer to synonyms or homonyms in the components of the different schemas. Such conflicts are usually easy to solve, using views for instance.

3. *Aggregation conflicts* and *generalization conflicts* arise, in the relational model, from the differences in the design decisions (arbitrary or justified by the application's requirements). Such conflicts can again be solved by means of views.

Of course, such a classification of the different types of heterogeneity can be argued. We only proposed it to serve as a guideline for the following discussion.

A Word about the User So far we have considered the heterogeneity of "sources". In fact we should extend our definitions to take into account users, application programs, or more generally speaking "receivers". On the one hand, legacy applications already embed assumptions about the schema, the data semantics, and the application domain and therefore pose the same problems as independent information sources. On the other hand, there is a strong argument in providing new users and programmers of new applications with the flexibility to make their own assumptions and choose their preferred schema, data semantics, and interpretation of the data that fit their specific target application domain.

2 Gestalten: Leveraging Intensional Heterogeneity

Integrating disparate information sources on the Internet faces three issues: even for very specific application domains, the number of sources of potential interest is very large; each source is managed independently; new sources may appear, existing sources may change or even be discontinued. The traditional approach to federated database integration where a system integrator defines integrated views over all the data sources breaks down because there are simply too many data sources, too much autonomy and too rapid changes.

The *Gestalt* strategy [42] mooted at the University of Wisconsin-Madison provides scalable approaches to integrating, querying and managing change in Internet databases, where the number of nodes and users are expected to be orders of magnitude greater than in current federated databases. The novelty of this approach is that the integrator simply *publishes* a schema for the integrated view, referred to as a *Gestalt*, and that administrators of the relevant data sources are given the opportunity to independently *register* their sources into this view. One or more integrated Gestalt can be created using this decentralized approach. Each Gestalt can itself be registered into other Gestalten, leading to hierarchies of Gestalten.

The Gestalt approach naturally scale. On the one hand, the responsibility of maintaining the "content" of the Gestalt is distributed across the various sources. The owner of a data source only has to understand his data source and the Gestalt description, but does not have to understand other data sources. On the other hand, as the Gestalt approach naturally leads to views with many unions (as opposed to joins), it does not create interdependence across sources.

As an example, let us consider the scenario where a Gestalt for on-line book stores is built. The Gestalt exports a view of the form:

```
Book (On-line-bookstore, URL, ISBN, Title, Author, Publisher, Price, Delivery, Subject)
```

Each vendor could then define a local "My_Books" table with the same fields and registers it into the Gestalt. Customers querying the Book table can have access to the aggregated contents of each local My_Books tables.

Information and service providers have incentive to cooperate in this collaborative approach because consumers increasingly prefers a "one-stop" on-line Gestalt for news, book bargain finder, or travel agent for instance. The Gestalt service maybe provided by a third party organization, a trusted agent who proposes additional services to the customers and/or the stores themselves (e.g. delivery services, digital cash payment, etc). It could also be provided by one of the bookstore in an attempt to create a dynamic competition environment. Open and direct competition leveraging the Internet technology is neither an impossible nor futurist scenario. For example, Books.com (http://www.books.com), one of the many on-line bookstores, already offers a "price compare" option on its ordering page: "Click above to compare our price with Amazon and BarnesandNoble.com. If ours isn't already the best price we'll lower it."

It is clear at this point that the success of a Gestalt service will depend on the quality of the support it provides to the information services administrators to easily register their site and to easily maintain an up-to-date description, as well as the quality of the services offered to the end-customers. The quality of support to the information services administrators will depend on the availability of tools facilitating the integration task, in particular the resolution of the semantic and schematic conflicts naturally arising. The overall quality of service will then depend on the efficiency of the query processing techniques supported by the Gestalt in a very dynamic environment such as the Internet where communication costs are difficult to predict.

3 Context Interchange: Semantic Integration

The *Context Interchange* [18] strategy is an approach invented at MIT to facilitate the integration of information sources available on the Internet and the World Wide Web into practical applications. The Context Interchange network is a collection of tools and software enabling the development integration services and their deployment in desktop applications (cf. figurecontext-arch, architecture of the MINT prototype [4]). Although the Context Interchange project at MIT and its partner project at the National University of Singapore are in practice concerned by all aspects of information integration, from a research point of view, they have primarily focussed on the issue of semantic heterogeneity.

Consider the two following example (from [5]) drawn from an actual application of the Context Interchange network prototype, MINT: "The NASDAQ stock exchange recently (1997) decided to report stock quotes in fractions at the 32nd (previously at the 16th). Most non-American stock exchanges, on the other hand, chose to report stock prices in decimals."

Most sources exhibiting heterogeneity (and potential conflicts) are easy to comprehend and resolve when taken individually. For example, prices advertised by the French online bookstore Bol (http://www.bol.fr a branch of the European http://www.bol.com) are explicitly reported in French Francs. These problems often reduce to scaling factors, unit conversions, or naming conventions. Furthermore, for a given application domain, the different types of data heterogeneity can reasonably be modeled by the use of a shared vocabulary – which we call the *domain model* – consisting of the set of common concepts manipulated by the information sources and the various applications.

Although the reconciliation of the conflicts can often be manually performed by the user or the administrator registering a new information source, this task may become unbearably cumbersome with increasing number of potential conflicts. Furthermore,it is often not the role of the user o source administrator to comprehend some of details of the necessary conversions. The alternative solution consisting in using a standard representation in the global schema is not sustainable either as it introduces an additional tier without really tackling the problem.

Let us take the simple example of currency conversion: we believe it should, in most cases, suffice that the user or information provider indicates the currency or currencies in which data should be or is expressed and that they need not have to take care directly of the details of the conversion (which rate, from which source, as of which date).

Our research stems from the premise that applications and users ought to focus on specifying what information is needed and how the results should be presented, *but should not have to worry about different representation and interpretation of the same information in disparate systems nor about the details of the necessary conversions.* We suggest that this latter task can be taken care of by a *Context Mediator*. *Context mediation* refers to the process whereby a query formulated under one *context* (the set of assumptions concerning how data is represented or how it ought to be interpreted) is transformed to a semantically equivalent query for accessing information present at sources having disparate contexts, such that all potential conflicts can be identified and resolved. To this end, we propose a deductive and object-oriented data model (COIN) and language (COINL) for the definition of contexts. COIN and COINL are specialized variants of F-Logic [27]. We show that context mediation can be automatized provided that a *domain model* and *context definitions* for the sources and receivers are available. The procedure used is presented in [6] and is based on an abductive reasoning [25]. The overhead of defining the domain model for an application domain and of defining contexts for the sources and receivers is justified by the observation that the information sources made accessible under this Context Interchange framework remain loosely coupled to one another. This means that new information sources can be added (or existing ones withdrawn) without any adverse impact on the other systems. As we have pointed out elsewhere [18], this approach allows us to achieve interoperation on a large-scale despite the rapid pace of changes that take place on the Internet.

4 Support for Resolving Schematic Conflicts and Query Processing

Traditional projects on the integration of heterogeneous sources [2, 28, 3, 23, 35, 51, 47, 30, 31] focus on integrating relatively small numbers of structured databases. Most of these works resolve the semantic and schematic heterogeneity that arise upon integration. The solutions are based on semantically rich data models that are able to represent the semantic connections between the distributed data. With the growth of information repositories on the Internet, a new generation of projects (e.g. HERMES [48], Lore and TSIMMIS [38, 11, 16], Information Manifold [33]) has focused on the integration of potentially large amounts of sources that may not necessarily be structured databases (i.e. with a known schema expressed in a strictly defined data model) but a collection of ASCII files, HTML documents and forms, or data in a spreadsheet format, for instance.

As we have seen, an important component for the effective resolution of schematic conflicts are the availability of standard communication protocols enabling the first level of connectivity, or the appropriate support for the definition of views and the processing of queries. Given the new requirements posed by the Internet, these tools and techniques need to be adapted to scale and to handle the non standard data. In the sequel we survey the major trends and results concerning: the definition of views, the handling of limited query processing capabilities of the sources, the definition and application of query folding techniques, and the definition of both cost-based and semantic query optimization techniques.

4.1 Views

Views can be defined in essentially two ways.

The world view or Gestalt schema can be defined as a global view of the federation of sources from which in their turn, users' views are defined. The five layer architecture of Sheth and Larson [45] constitutes a guideline for the schematic integration from this perspective. This approach acknowledges the fact that the data is stored (primarily managed) by the sources and that the world and users' view are composed of virtual relation (i.e. relations defined in intension). Query processing consists of the optimization and evaluation of queries against views.

However, views are only definitions and do not need to parallel the actual existence of extensional relations. Indeed, alternatively, both the users' and sources' relations can be considered views on a set of relations constituting the world view. This world view may not be materialized by any other means than by the materialization of some of its views in the sources. Here query processing consists of the rewriting of users'queries against views to obtain a query in terms of the world view and the subsequent attempt to answer the world view queries using the sources as materialized views. These techniques have recently been studied under the name "answering queries using views" (e.g. [15]) and previously as "query folding".

Regardless of the approach to defining views, there are four issues involved in the processing of queries in Gestalten and Gestalt-like systems such as [52, 14, 33]:

- source capabilities: sources can only process certain binding patterns and accept only certain constraints, queries can only be decomposed accordingly if at all.
- query folding: also known as query rewriting which is the activity of deciding if and how a query can be answered using a given set of resources. These resources might be materialized views, cached results of previous queries, or even queries answerable by other databases.
- physical optimization: the independence of the remote sources and of the network itself only allows one to develop statistical cost models for the remote processing of s' queries and the

communication. Despite their inherent weaknesses such models are essential for the efficient processing of queries and, in practice, for the effectiveness of the entire system.

- semantic optimization [29, 7]: in traditional database systems, the relative accuracy of the cost models and consequently the amount of optimization achieved by physical optimization has always consigned semantic optimization to a tertiary role. However, in the integration of disparate information sources over the Internet, the role of semantic optimization should be at least equal to that of physical optimization because the reliability of semantic knowledge here is higher than that of cost models.

4.2 Source Capabilities

As opposed to the traditional databases in a federated database infrastructure, information sources and services on the Internet and the World Wide Web may have different capabilities.These capabilities must be known to the Gestalt as they determine the type of queries the source can answer. Indeed, a source may be implemented using ad-hoc scripts and flat files, have very limited, or no query processing capability and serve HTML documents. On the other end of the spectrum, a source could be powered by a full-fledge relational database management system, export all of the advanced query capabilities, and serve XML data. The varying source capabilities will affect the cost of retrieving answers from these sources.

Traditional system mediators focus their attention on the contents of the sources and their relationship to the integrated views provided to the users. They do not keep track of the capabilities of sources to answer queries. This may result in query plans with source queries that cannot be answered by the sources.

In our example of on-line bookstores, Amazon proposes a search mechanism where constraint on both the author and the title can be posed. Books.com allows constraint exclusively on the author or the title. Books-a-million (http://www.booksamillion.com), as opposed to the above mentioned on-line bookstores does not offer a search by ISBN. Most bookstores offer a list or hierarchy of categories to support browsing based search. Of course the structures for the categories may not match across sources in form and content. If a mediator issues a query to Amazon to retrieve the title and author of a book given the ISBN, then the query will fail and the mediator will be unable to answer the user query.

HERMES [48] use a mediator specification language where some literals explicitly specify the parameterized calls that are sent to the sources. However, this reduces the interface between the integration system and the sources to a limited set of explicitly listed parameterized calls. On the other hand, TSIMMIS [39, 22] use source query capability descriptions to describe infinite sets of queries. The input to the query translator is a list of rules each consisting of:

1. A query template, which is a parameterized query in the global language. Assigning values to the parameters of the template results in a query. For example, the parameterized template "`BookPrice < $1`" might be instantiated to "`BookPrice < 100`. Thus, a query template can describe an infinite number of queries.
2. A sequence of operations that must be carried out to answer any query obtained by instan-

tiating the template.

The Information Manifold [33] focus on the capabilities description of sources found on the Web. As a result, it does not consider recursion and the expressive power of its capabilities describing mechanism is less powerful. The DISCO system [53, 54] describes the capabilities of the sources using context-free grammars which are augmented with actions. DISCO initially ignores the limited wrapper capabilities to generate possible query plans. It then checks the query plans against the wrapper grammars and rejects plans containing unsupported queries. This strategy can be very expensive compared to capabilities-based rewriting as the latter ensures that the queries issued to the wrappers are answerable by the source.

The Garlic system [21] combines capabilities-based rewriting with cost-based optimization. It assumes that all the variables in a query are always available from the wrapper. This compromises the expressiveness of the description language but greatly simplifies the proposed algorithm. Moreover, the capabilities descriptions are given in terms of plans supported by the wrappers. Additional assumptions are made at this point regarding the class of plans that can be described. The information integration system, Infomaster [17], has a facilitator that dynamically determines an efficient way to answer the user's query by using as few sources as necessary. Rules and constraints are used to describe information sources and translations among these sources. These rules and constraints are stored in a knowledge base.

4.3 Query Folding/Query Rewritings

Query folding is important for the following reasons:

1. The base relations referred to in a query might be stored remotely and is too expensive to access.
2. The base relations might not be available for access because of temporary network disconnection.
3. The base relations might be conceptual relations only and is not existent physically.

Query folding has applications in query optimization in centralized databases, [10], query processing in distributed databases [26], and query answering in federated databases [32, 34]. In centralized databases, materialized views are often maintained, and a query can be answered by accessing views instead of base relations if the query can be folded using the views [10]. In multiple query answering, the result of a query can be used to at least partially answer another query if the second query can be folded using the first one. Query folding is even more crucial in distributed environments. In client-server applications, views and queries might be cached at the client site. Client queries can be answered more efficiently if they can be folded using the cached data [26]. In the event of a disconnected network, a query can still be answered at least partially if it can be folded using views and queries maintained at available sites. In a federated environment containing multiple heterogeneous autonomous and legacy data sources, a data source might be capable of answering only limited kinds of queries [32]. The job of the federated query processor is to determine if and how federated queries can be answered using these views, or equivalently, to determine if and how federated queries can be folded using these views.

Recall our example of on-line bookstores. Suppose source Amazon gives the *Title, Author, Subject* information of books while source Books-a-million gives the *ISBN, Title, Publisher* information. A user query on the Book Gestalt to retrieve the titles and subjects of all the books written by "Alain Stone" and published by "McGraw-Hill" will be processed as follows:

1. Send a source query q1 to Amazon to retrieve the titles and subjects of all the books authored by "Alain Stone"
2. Send a source query q2 to Books-a-million to retrieve the titles of all the books published by "McGraw-Hill".
3. Join the results of q1 and q2 to get the answer to the user query.

Note that the feasibility of this plan is dependent on the source capabilities.

Several works have considered the problem of implementing a query processor that uses the results of materialized views [56, 43, 46, 12, 55, 10]. [10, 55, 56] developed solutions to query folding or rewritings for conjunctive queries and resources with built-in predicates. These solutions are based on either finding syntactic or 1-1 mappings from the view to the query. Their algorithms use exhaustive search strategies that are exponential-time in complexity, and often use unnecessary pruning conditions that do not guarantee to find all foldings. Furthermore, these algorithms compute only strong and complete foldings.

[32] observe that the problem of answering queries using views is closely related to the problem of query containment, a special case of query folding. In order to determine whether a query is contained in another query, we determine instead whether the second query can be answered using a view defined by the first query. Query containment problem for conjunctive queries is known to be NP-complete [9]. [32] consider the problem of computing answers to queries using materialized views. In particular, they consider the problem of finding a rewriting of a query that uses the materialized views, the problem of finding minimal rewritings, and finding complete rewritings (that is, rewritings that use only the views). They show that all the possible rewritings can be obtained by considering containment mappings from the views to the query, and that the problems they consider are NP-complete when both the query and views are conjunctive and don't involve built-in comparison predicates. There are two sources of complexity: the number of possible containment mappings, and the complexity of deciding which literals from the original query can be deleted. [41] built on the results of [32] and considered the problem of finding rewritings using query templates (views) that are restricted to use specific binding patterns.

[40] investigated the query folding problem in the context where queries and resources are conjunctive queries. A hypergraph is used to represent a conjunctive query (cyclic and acyclic after removing "ears" from the graph) and skolemization to derive folding rules. Contribution in [40] includes an exponential-time query folding algorithm to compute a set of complete or partial foldings of query Q using a set of resources R which is similar in spirit to arc consistency algorithms for constraint satisfaction problems [37], and a polynomial-time algorithm that tests the existence of complete or partial foldings for the large and natural subclass of acyclic conjunctive queries.

[20, 19] enhance the power of query folding by taking into consideration integrity constraints that are known to hold on base relations. [20] describes a technique which utilizes inclusion dependencies to generate foldings that might otherwise be overlooked.

4.4 Physical/Cost-Based Optimization

Query processing and optimization in mediator systems that access distributed autonomous sources is different from traditional distributed query optimization and pose many novel problems. An extensive discussion of the differences is given in [36]. Cost-based optimization is hard because the mediator do not have access to source statistics information and it is not easy to model the source's performance. Furthermore, querying remote sources may be expensive because of high connection overhead, long computation time, financial charges, and temporary unavailability.

[13, 58] investigate the behavior of a non-proprietary source by probing it with carefully organized sample queries and applying regression techniques to estimate various parameters of a predetermined cost model. While this method is effective, it is not applicable when we do not have a predetermined cost model which is the case with many unconventional sources such as face databases, video repositories, databases of plans for transportation logistics etc.

[1] propose a cost-based optimization technique that caches statistics of actual calls to the sources. Estimates of he cost of possible execution plans are then obtained based on the cached statistics. Experimental results show that their method apply to heterogeneous databases which are both traditional as well as non-traditional sources.

[57] use a cost model that focus on the main costs (sending queries to sources and getting answers back) incurred by large join queries in mediation systems. Under this cost model, two algorithms are developed to sequence source queries. The algorithm that is based on a greedy strategy produces optimal plans in some scenarios while the algorithm that is based on a partitioning scheme produces optimal plans in more scenarios. The problem remains NP-hard.

4.5 Semantic Query Optimization

An alternative approach to optimize query plans generated by information mediators is to use semantic query optimization [24, 44, 49, 8, 50]. *Semantic Query Optimization* as described in [7], is the process of optimizing (increasing the potential for an efficient evaluation) of database queries using the semantic information contained in the integrity constraints. The essential idea is to use semantic knowledge (such as known integrity constraints) about data to reformulate a query into a more efficient but *semantically equivalent* query. The advantage of this approach is that the optimizer can use semantic knowledge to derive information about intermediate data. This effectively reduces unnecessary data transmission and eliminates more unnecessary joins.

There are two major phases in semantic query optimization. In the first phase, the optimizer locates applicable semantic knowledge and proposes a sequence of one or more reformulation operations such as to delete a literal or insert a new literal that preserves the semantics of the query. The second phase is to evaluate the proposed reformulations and apply the best reformulation based on a cost model of query execution.

A query may be answered without accessing the database if enough knowledge is contained in the integrity constraints. For instance, a query trying to find cheaper copies book than a paperback copy may be answered immediately, there is no such copy, against a source supplying only hard-cover copies, with the knowledge that hard-covers are always more expensive than paperback. When accessing the database can not be avoided, the potential for the evaluation of a query can still be improved by the introduction or elimination of constraints.

[29] has identified six types of transformations that can result from the use of integrity constraints knowledge:

1. Index Introduction: the specialization of one element of an accessed relation that can be used to improve the indexed access;
2. Join Elimination: the elimination of a redundant access to a relation (typically in the case of an inclusion dependency);
3. Scan Reduction: the introduction of a constraint which may optimize the scanning of the relation;
4. Join Introduction: the introduction of an access to a relation which may be used to accelerate the elimination of tuples (e.g. a join index);
5. Detection of Unsatisfiable Conditions: cases where the result of the query will be empty regardless of the database content.
6. No Transformation: cases where no transformation is useful even if possible.

We see that introduction and elimination are reverse actions and that a strategy (based on a cost model) must be define for the utilization of the constraints. Join introduction can help solve some of the problems arising from the limited capabilities of the sources by introducing ancillary relations. For instance if a source cannot be searched by title only and not by ISBN, the introduction of a correspondence table between ISBN and titles (from another source, for instance) could resolve the capability restriction issue.

Chakravarthy et al. [7] present a method for compiling integrity constraints into residues attached to the rules (view definitions) of a deductive database program and show how to exploit the residues in at the query optimization stage to identify opportunities for the application of one of the six preceding cases. The compilation of the residue for a rule is based on a partial subsumption algorithm. The different types of transformation proposed by King are shown to correspond to different transformations decided according to the type of residue obtained. There are four different types of residue: the null clause, a goal clause, a unit clause, Horn clause with non-empty body and head.

5 Conclusion

In this paper we have attempted to demonstrate the complementarity of two major research efforts addressing issues in the integration of disparate information sources: the Gestalt project at the University of Wisconsin-Madison, and the Context Interchange project at MIT and at the National University of Singapore. While the Gestalt strategy proposes a scalable infrastructure for exploiting the natural intensional heterogeneity of the sources available over the Internet and in particular on the World Wide Web, the Context Interchange strategy focuses on proposing a scalable solution to the problem of semantic heterogeneity. Recognizing that new view definition and query processing techniques need to be developed to meet the challenges posed by a new and dynamic environment, we have also surveyed some of the major recent results in this area. Both the Gestalt and the Context Interchange project are concerned by real applications. The proposed infrastructures are being or have been prototyped and applied to real word data.

References

[1] S. Adali, K.S. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of ACM SIGMOD conference*, pages 137–148, 1996.

[2] R. et al Ahmed. The pegasus heterogeneous multidatabase system. *IEEE Computer*, 24:19–27, 1991.

[3] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18:323–364, 1986.

[4] S. Bressan, K. Fynn, C. Goh, M. Jakobisiak, K. Hussein, H. Kon, T. Lee, S. Madnick, T. Pena, J. Qu, A. Shum, and M. Siegel. The context interchange mediator prototype. In *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, 1997. (see also http://context.mit.edu/demos/sigmod).

[5] S. Bressan and C. Goh. Answering queries in context. In *Proceedings of the International Conference on Flexible Query Answering*, 1997.

[6] S. Bressan, C.H. Goh, T. Lee, S. Madnick, and M. Siegel. A procedure for mediation of queries to sources in disparate context s. In *Proceedings of the International Logic Programming Symposium*, Port Jefferson, NY, Oct 12–17 1997.

[7] U.S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Trans. on Database Systems*, 15(2), 1990.

[8] U.S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2), 1990.

[9] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive qeuries in relational databases. In *Proc. of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[10] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proc. of the 11th Int. Conf. on Data Engineering*, pages 190–200, 1995.

[11] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference*, pages 7–18, 1994.

[12] C.M. Chen and N. Roussopoulos. The implementation and performance evaluation of the adms query optimizer: Integrating query result caching and matching. In *Proc. of the Int. Conf. on Extending Database Technology*, 1994.

[13] W. Du, R. Krishnamurthy, and M.C. Shan. Query optimization in heterogeneous databases management systems. In *Proc. VLDB Conference*, pages 277–291, 1992.

[14] O. Duschka and M. Genesereth. Query planning in infomaster. http://infomaster.stanford.edu/, 1997.

[15] O. Duschka and M. Genesereth. Answering recursive queries using views. In *Proceedings of the International Conference on Principles Of Dat abase Systems (PODS)*, 1998.

[16] H. et al Garcia-Molina. The tsimmis approach to mediation: data models and languages. *Journal of Intelligient Information Systems*, 8:117–132, 1997.

[17] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proc. of ACM SIGMOD conference*, pages 539–542, 1997.

[18] C. Goh, S. Bressan, S. Madnick, and M. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *To be published in ACM Transactions on Information Systems*, 1999.

[19] J. Gryz. An algorithm for query folding with functional dependencies. In *Proc. of the 7th International Symposium on Intelligent Information Systems*, pages 7–16, 1998.

[20] J. Gryz. Query folding with inclusion dependencies. In *Proc. of the 14th Int. Conference on Data Engineering*, pages 126–133, 1998.

[21] L. Haas, D. Kossman, E. Wimmers, and J. Yang. An optimizer for heterogeneous systems with non-standard data and search capabilities. *Special Issue on Query Processing with Non-Standard Data, IEEE Data Engineering Bulletin*, pages 37–43, 1997.

[22] J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, and R. Yerneni. Template-based wrappers in the tsimmis system. In *Proc. of ACM SIGMOD conference*, 1997.

[23] J. Hammer and D. McLeod. An approach to resolving semantic heterogeniety in a federation of autonomous, heterogeneous database systems. *Int. Journal of Intelligient and Cooperative Information Systems*, 2:51–83, 1993.

[24] C.N. Hsu and C.A. Knoblock. Semantic query optimization for information mediators of heterogeneous databases, 1997.

[25] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6), 1993.

[26] A.M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. In *Proc. of the Int. Conf. on Parallel and Distributed Informtion Systems*, 1994.

[27] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 4:741–843, 1995.

[28] W. et al Kim. On resolving schematic heterogeneity in multidatabase systems. In *Distributed and Parallel Databases*, pages 251–279, 1993.

[29] J.J. King. *Query Optimization by Semantic Reasonning.* PhD thesis, Stanford University, 1981.

[30] M.L. Lee and T.W. Ling. Resolving structural conflicts in the integration of entity-relationship schemas. In *Proc. of the 14th Int. Conference on Object-Oriented and Entity-Relationship Modelling (OOER95)*, pages 424–433, 1995.

[31] M.L. Lee and T.W. Ling. Resolving constraint conflicts in the integration of entity-relationship schemas. In *Proc. of the 16th Int. Conference on Conceptual Modeling*, pages 394–407, 1997.

[32] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1995.

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

[34] A. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Int. Journal of Intelligient Information Systems*, 1995.

[35] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22:267–293, 1990.

[36] H. Lu, B.C. Ooi, and C.H. Goh. Multidatabase query optimization: Issues and solutions. In *Proc. RIDE-IMS*, pages 137–143, 1993.

[37] A.K. Mackworth. The logic of constraint satisfaction. *Journal of Artificial Intelligience*, 58(1):3–20, 1992.

[38] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *IEEE International Conference on Data Engineering*, pages 251–260, 1995.

[39] Y. Papakonstantinou, A. Gupta, and L. Haas. Capabilities-based query rewriting in mediator systems. In *Proceedings of the Intl. Conf. on Parallel and Distributed Information Systems*, 1996.

[40] X. Qian. Query folding. In *IEEE International Conference on Data Engineering*, pages 48–55, 1996.

[41] A. Rajaraman, Y. Sagiv, and J.D. Ullman. Answering queries using templates with binding patterns. In *Proc. of the 14th ACM Symposium on Principles of Database Systems*, pages 105–112, 1995.

[42] R. Ramakrishnan and A. Silberschatz. Scalable integration of data collection on the web. Technical report, University of Wisconsin-Madison, 1998.

[43] T. Sellis. Intelligient caching and indexing techniques for relational database systems. *Journal of Information Systems*, pages 175–185, 1988.

[44] S.T. Shenoy and Z.M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Trans. on Data and Knowledge Engineering*, 1(3):344–361, 1989.

[45] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[46] M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos. On rules, procedures, caching and views in database systems. In *Proc. of ACM SIGMOD conference*, 1990.

[47] S.Y.W. et al Su. Ncl: A common language for achieving rule-based interoperability among heterogeneous systems. *Int. Journal of Intelligient Information Systems*, pages 171–198, 1996.

[48] V.S. et al Subrahmanian. Hermes: A heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes/overview/paper, 1996.

[49] W. Sun and C.T. Yu. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Trans. on Data and Knowledge Engineering*, 1(3):362–375, 1989.

[50] W. Sun and C.T. Yu. Semantic query optimization for tree and chain queries. *IEEE Trans. on Data and Knowledge Engineering*, 6(1):136–151, 1994.

[51] G. et al Thomas. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22:237–266, 1990.

[52] A. Tomasic, R. Amouroux, Ph. Bonnet, O. Kapitskaia, H. Naacke, and L. Raschid. The distributed information search component (DISCO) and the w orld-wide web. In *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, 1997.

[53] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of disco. Technical report, INRIA, 1995.

[54] A. et al Tomasic. The distributed information search component (disco) and the world wide web. In *Proc. of ACM SIGMOD conference*, pages 546–548, 1997.

[55] O.G. Tsatalos, M.H. Solomon, and Y.E. Ioannidis. The gmap: A versatile tool for physical data independence. In *Proc. of Int. Conference on Very Large Databases*, pages 367–378, 1994.

[56] H.Z. Yang and P.A. Larson. Query transformation for psj-queries. In *Proc. of Int. Conference on Very Large Databases*, pages 245–254, 1987.

[57] R. Yerneni, C. Li, J. Ullman, and H. Garcia-Molina. Optimizing large join queries in mediation systems. In *Proc. of Int. Conference on Database Theory*, 1999.

[58] Q. Zhu and P.A. Larson. A query sampling method for estimating local cost parameters in a multidatabase system. In *IEEE International Conference on Data Engineering*, pages 144–153, 1994.