Query Consolidation: Interpreting a Set of Independent Queries Using a Multidatabase Architecture in the Reverse Direction

Aybar C. Acar
Department of Computer Science
George Mason University
4400 University Drive
Fairfax, VA
aacar@gmu.edu

Amihai Motro
Department of Computer Science
George Mason University
4400 University Drive
Fairfax, VA
ami@gmu.edu

ABSTRACT

We introduce the problem of *query consolidation*, which seeks to interpret a set of disparate queries submitted to independent databases with a single "global" query. This problem has multiple applications, from improving database design to protecting information from a seemingly innocuous set of apparently unrelated queries. The problem exhibits attractive duality with the much-researched problem of *query decomposition*, which has been addressed intensively in the context of multidatabase environments: How to decompose a query submitted to a virtual database into a set of local queries that are evaluated in individual databases. We set the new problem in the architecture of a canonical multidatabase system, using it in the "reverse direction".

1. INTRODUCTION

Consider an individual who submits a set of queries to different databases, and then, off-line, consolidates the information obtained in a "big answer" of some sort. Because the information this user requires is dispersed over multiple databases, the user is forced into a laborious process of submitting individual queries to different databases and then correlating and assembling the information off-line. Discovering a single interpretation for his entire query set may help suggest how information could be *reorganized* to facilitate similar tasks in the future. Indeed, the main argument for constructing virtual databases has always been to provide in a single source all the information necessary for a particular task [16]. Thus, discovering interpretations for distributed query sets may suggest useful reorganizations and consolidations, either physical or virtual.

As an analogy, consider a shopping center with multiple stores, and assume that an analysis of sale records shows that within a small time interval, the same customer purchased a box of candy in one store, gift-wrapping paper in another, and a greeting card in a third. A global interpretation of this local information may suggest that service could be improved if these three items where to be sold in the same store. Similarly, query consolidation may suggest redesign of available information sources to correspond more efficiently to popular information needs.

A different, though not entirely unrelated, reason for interpreting query sets, in either the distributed or centralized cases, is *user inexperience or ignorance*. In the distributed case, the user might be submitting a set of queries to different databases and correlating them off-line, when the same goal could be achieved by accessing a single database. In the centralized case, the user might be submitting a set of small queries and assembling them off-line, when the same goal could be achieved with a single query, perhaps using a feature of which the user is not aware. A query consolidation analysis may suggest flaws in the way the system is advertised or in the training of its users. This application is reminiscent of other systems that track user behavior and suggest improvements, such as office software or on-line stores.

Returning to the analogy of the shopping center, the reason for the individual purchases could be that the customer may be trying to *hide* his overall purpose. Accordingly, a possible application of query consolidation is surveillance and security: A consolidated query discloses the intentions of the user posing the queries. While the elucidation of these intentions from consolidated queries is a task for human experts, a query consolidation system can do the preparatory work. Since there could be a large number of users each with multiple queries, the function of the query consolidator will be to sift through the logs, compile likely consolidations, and present them to the expert for judgement. A variety of options are available: The expert can focus on a single user and get a listing of interests during a time period. Alternatively, trends can be analyzed across many sources looking for intentions shared by a group of users. Query consolidation can also be useful as a detection mechanism when the possible intentions and the global queries that imply them are known in advance. Then, an operator can set up the system so that certain information is on a watch-list and any consolidation of queries that significantly overlaps that information is flagged automatically by the system, along with the users who posed these queries. An earlier attempt at security-inspired query consolidation, albeit using a dif-

VLDB '08, August 24-30, 2008, Auckland, New Zealand

ferent approach, can be found in [2].

We propose to address the problem of interpreting distributed sets of queries, by using the well-researched architecture of *virtual databases* [17]. A query submitted to a virtual database (based on the global scheme) is *decomposed* into queries against the local databases, and the corresponding answers are assembled in an answer to the original query. The entire process is transparent to the user.

Query decomposition is the re-writing of a global query into component queries which are in terms of the local databases. These component queries are answered by the respective sources and the answers are recombined to generate an answer that is equivalent to the answer of the global query. For our purpose here of interpreting a set of local queries, we adopt the same architecture, but consider a process that is the *reverse* of query decomposition, and which we name *query consolidation*: Given local the queries (and their answers), the problem is to find the global query that would decompose into the former. The main obstacle here is that whereas query decomposition is usually a *function* (it is a deterministic process in which each global query generates a unique decomposition), it is not *injective*. That is, there could be multiple global queries that the query decomposition procedure will decompose into the same set of local queries.

Our approach to this new problem can be sketched as follows. We assume that the independent databases to which queries are submitted have been incorporated into a virtual database system. Under assumptions of sufficiency (the given query set includes all the information necessary to achieve the goal) and necessity (it includes only information necessary to achieve the goal) we "reverse" the query decomposition process. The process incorporates two steps where multiplicity of solutions must be considered: At one point the system must infer the most likely set of *equi-joins* for a set of relations; at another point it must discover the most likely *selection constraints* that would be applied to a relation. In each case we develop a procedure that ranks solutions according to their perceived likelihood. The final result is therefore a ranked list of suggested consolidations.

The focus of this paper is on the definition of the new problem and its applications, its setting in a virtual database architecture; the methodology of its solution, detailed discussions of the algorithms, the software prototype, and results of experimentation are largely omitted, for reasons of space. The paper is organized as follows. Section 3 provides the formal framework for this work and Section 4 outlines the general approach to solving the issues mentioned. Finally, Section 5 concludes with a brief summary and discussion of work in progress. We begin with a brief review of related work.

2. BACKGROUND

The work presented in this paper draws from the subjects of information integrating systems (multidatabase systems) and query decomposition.

An information integration system combines information from a heterogeneous collection of autonomous information sources. The integrating site is often referred to as *global*, and the individual sources are termed *local*. There have been many different models and architectures for information integration systems. Of interest to us here are systems that follow the architecture of *virtual databases*. A virtual database has a database scheme (a *global* scheme), but no database instance. Instead, it has information that maps the global scheme into schemes of local databases. The materialization of a global query is done by translating the query into multiple subqueries to be materialized at the local sources and shipped back to the integrator for assembly. Virtual databases can be classified by the type of their global-local associations [7]. This classification distinguishes between architectures in which the local database schemes are defined as views of the global scheme (termed Local-as-View or LAV), and architectures in which the global scheme is defined as views of the local schemes (termed Global-as-View or GAV). An example of the former type are The Information Manifold [9]. Examples of the latter type are SIMS [3], TSIMMIS [6] and HERMES [21]. The architecture of Multiplex [17] is more powerful in that it associates views of the global schema with views of the local schema's. This hybrid approach earned the term GLAV.

A primary concern in virtual database systems is the process of query decomposition: The translation of a global query to a set of local queries. The main problem here is the need to rewrite queries defined over relations to queries over views of these relations (this is especially difficult for LAV systems) [7]. Optimization is also challenging because statistical information on local data is often unavailable. Finally, the decomposition procedure may have to account for temporary unavailability of some data, or multiple, inconsistent copies of other data [19, 18].

3. FORMAL FRAMEWORK

The formal framework for this research consists of three parts: (1) A statement of the problem, (2) a description of a "generic" virtual database architecture and query decomposition procedure, and (3) assumptions on the sufficiency and necessity of the given queries for the overall goal.

3.1 The Problem

A virtual database architecture consists of a set of *local* databases $D_1, \ldots, D_n$, a global database scheme D, and a mapping of the global scheme into the local databases. The main service of this architecture is *query decomposition*:

> Given a global query Q, find local queries $Q_1, \ldots, Q_n$ and expression E such that $Q = E(Q_1, \ldots, Q_n)$.

Query decomposition can be viewed as a *function* that assigns each query Q a *unique* set of queries $Q_1, \ldots, Q_n$ and suitable assembly expression E.

The problem of *query consolidation*, which is the subject of this paper, is defined as the *reverse* of the query decomposition problem:

> Given local queries $Q_1, \ldots, Q_n$, find global query Q and expression E such that the query decomposition procedure will decompose Q into $Q_1, \ldots, Q_n$ using E, so that $Q = E(Q_1, \ldots, Q_n)$.

The solution to the problem as stated is not unique. That is, there could be multiple global queries $Q^1, \ldots, Q^m$ and corresponding expressions $E^1, \ldots, E^m$, such that a query decomposition procedure would decompose Q^i into $Q_1, \ldots, Q_n$ using E^i (for $1 \leq i \leq m$). We address this issue in Section 3.3.

3.2 The Multiplex Model for Virtual Databases

To solve the query consolidation problem we must adopt a virtual database model. Many different architectures have been proposed for virtual databases, and we adopt the Multiplex architecture [17]. The advantages of Multiplex that are attractive include its simplicity and generality. Simplicity is due to the fact that Multiplex assumes that all databases are in the well-known relational model, without introducing any new concepts or structures. Generality is achieved by the method in which the global and local databases are associated, namely by arbitrary view pairs.

We begin by defining the language for all queries and views. We assume the subset of the relational algebra defined by the operators selection, projection and Cartesian product (SPC)[1], with selections that are purely conjunctive. Although this family of queries is a restricted subset of the algebra (i.e., it excludes union, difference, non-conjunctive selections), it is often considered adequately expressive for the large portion of queries used in the real world [12]. In has been shown that any expression in this language can be written in the form[2]:

$$Q = \pi_A \sigma_C (R_1 \times R_2 \times \ldots \times R_n) \quad (1)$$

Assuming expressions in this form often simplifies discussions and proofs.

A Multiplex database is:

1. A global database scheme D,
2. A set $D_1, \ldots, D_n$ of local database schemes, and their associated database instances $d_1, \ldots, d_n$, and
3. A set $(V_1, U_1), \ldots, (V_m, U_m)$ of view pairs, where each V_i is a view of the global scheme D, and each U_i is a view of one of the local schemes.

Thus, the global database scheme D has no associated database instance. Instead, there is a collection of views of this scheme that are materialized using the corresponding local views, i.e., the instance of the global view V_j is materialized by the instance of the view U_j (in the appropriate local database).

An essential part of the Multiplex model is its query decomposition algorithm. Multiplex employs its own algorithm which is very similar to the well-known Bucket Algorithm [11]. Although the Multiplex algorithm has added provisions to handle data inconsistencies, the two can be considered equivalent for the purposes of this paper.

3.3 Assumptions on Sufficiency and Necessity

We interpret the consolidating query Q as the *goal* of the user in submitting the queries $Q_1, \ldots, Q_n$. This assumes that the user is not using information obtained elsewhere to achieve his goal. In other words, we adopt a principle of *sufficiency*: The information in the local queries $Q_1, \ldots, Q_n$ is sufficient to achieve the goal, and hence can be approximated by an appropriate consolidation.

Recall that we characterized query decomposition as a procedure with a unique outcome. Consider a simple global query that retrieves a single value such as a person's age. Obviously, there could be multiple correct decompositions. For example, the local query Q_i could retrieve just the person's age; or it could retrieve that person's entire tuple, and the expression E would project the age; or it could retrieve the tuples of multiple persons and E would select that person's tuple and project the age. The guiding principle of the query decomposition procedure is to retrieve from the local databases as little as possible, taking advantage of the local system's query processing capabilities. This reduces possible costs charged by the local database, as well as the costs and time of transmitting the data. Hence, the decomposition adopted is one that *optimizes* the process.

A similar principle will guide our query consolidation procedure. In the previous example, assume a given local query Q_i that retrieves tuples of multiple persons. From this, one could conclude a global query Q that needs all this information; or one that selects a single tuple from the set; or one that extracts the age of a particular person. A principle that guides the query consolidation procedure is that of *necessity*: All the information given in the queries is assumed to be necessary for the global query. The consolidation necessity principle is similar to the decomposition optimality principle: both assume that *all* information extracted from local databases is necessary, either to answer Q (decomposition) or to conclude Q (consolidation).

We note that both assumptions are at times unjustified. The user may have some additional information that may be instrumental in achieving his goal. Or he may submit non-optimal queries that retrieve unnecessary information (or he may be dishonest, attempting to hide his true goals). Such violations of our assumptions require further research.

4. APPROACH

In rough strokes, our overall approach may be sketched as follows. We assume a virtual database is available that integrates local databases $D_1, \ldots, D_n$ in a global scheme D. Given local queries $Q_1, \ldots, Q_n$, we follow a procedure that essentially reverses query decomposition. The global relations that are (partially) populated now need to be assembled into a join-selection-projection expression.

There may be multiple ways these relations can be joined. The join presents us with the first manifestation of the aforementioned non-unique nature of the problem. The problem of finding the best join is the familiar *join inference* problem [13, 22, 15, 8, 14]. Note that each spanning tree over the join graph in this situation is a possible join. Roughly, we approach this problem by weighting each of the edges in the graph (i.e., each possible equi-join) with a score based on the information gain[3] given the join. We then rank these possible solutions in order of decreasing weight.

The second manifestation of non-uniqueness arises from tuple selections that are done after the relations have been joined. These selections always involve comparisons of attributes from different sources, as this is the only type of comparisons that must be performed at the integration site (due to the assumption of optimality, other types of selections, namely comparisons to constants or comparisons of attributes in the same source, are assumed to be performed in the local databases). We have approached this issue by a form of association rule learning from previous query examples.

[1] Or, equivalently, selection projection, join, rename (SPJR).
[2] See [1] for proof.
[3] Kullback-Leibler divergence [10].

5. CONCLUSION

We described a new problem, which we termed *query consolidation*. Query consolidation seeks to interpret a set of disparate queries that were submitted to independent databases with a single global query: A query that expresses the ultimate goal of the user. Setting the problem in the architecture of a virtual database, it exhibits attractive duality with the much-researched problem of *query decomposition*.

We assumed that the independent databases to which the component queries are submitted are "monitored" by means of a virtual database. Since the same set of queries could be consolidated in different global queries (all of which will decompose back to the same component queries), our solution *ranks* the possible consolidations. The rankings are derived from our own treatment of the problem of join inference. We further enrich the possible consolidations with added selection constraints where previous experience shows that such constraints are likely, given the columns requested.

The assumption that the databases had been integrated previously in a virtual database implied the existence of a global scheme. This scheme provided semantic associations among the individual queries, and thus simplified the task of consolidation. A more challenging situation, which is the direction of our future work, is when such a virtual database had not been constructed. In this situation the extensions must be analyzed to infer their semantic associations, a task reminiscent of the well-known scheme-matching problem [20, 4, 5].

6. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*, chapter 4, pages 55–56. Addison-Wesley, 1995.

[2] A. C. Acar and A. Motro. Why is this user asking so many questions? Explaining sequences of queries. In *Proceedings of DBSEC 04, 18th IFIP Annual Conference on Data and Applications Security, Sitges, Catalonia, Spain*, pages 159–176, 2004.

[3] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2/3):99–130, 1996.

[4] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of CAiSE 02, 14th Conference on Advanced Information System Engineering, Toronto, Canada*, pages 452–466, 2002.

[5] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. pages 509–520, 2001.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

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

[8] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proceedings of VLDB 02, 28th International Conference on Very Large Data Bases, San Fransisco, CA*, pages 670–681, 2002.

[9] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, Palo Alto, CA*, pages 85–91, 1995.

[10] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[11] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. *Proceedings of VLDB 96*, pages 251–262, 1996.

[12] A. Y. Levy, A. Rajaraman, and J. D. Ullman. Answering queries using limited external query processors (extended abstract). In *Proceedings of PODS 96, 15th ACM Symposium on Principles of Database Systems, Montreal, Canada*, pages 227–237, 1996.

[13] D. Maier, J. D. Ullman, and M. Y. Vardi. On the foundations of the universal relation model. *ACM Transactions on Database Systems*, 9(2):283–308, 1984.

[14] T. Mason and R. Lawrence. INFER: A relational query language without the complexity of sql. In *Proceedings of CIKM 05, 14th ACM Conference on Information and Knowledge Management, Bremen, Germany*, pages 241–242, 2005.

[15] A. Motro. Constructing queries from tokens. In *Proceedings of ACM SIGMOD 86, International Conference on Management of Data, Washington, DC*, pages 120–131, 1986.

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

[17] A. Motro. Multiplex: A formal model for multidatabases and its implementation. In *Proceedings of NGITS 99, Fourth International Workshop on Next Generation Information Technologies and Systems, Zichron Yaacov, Israel*, Lecture Notes in Computer Science No. 1649, pages 138–158. Springer-Verlag, 1999.

[18] A. Motro and P. Anokhin. Fusionplex: Resolution of data inconsistencies in the integration of heterogeneous information sources. *Information Fusion*, 7(2):176–196, 2006.

[19] F. Naumann, U. Leser, and J. C. Freytag. Quality-driven integration of heterogenous information systems. In *Proceedings VLDB 99, 25th International Conference on Very Large Data Bases, Edinburgh, Scotland*, pages 447–458, 1999.

[20] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[21] V. S. Subrahmanian, S. Adali, A. Brink, R. Emery, J. J. Lu, A. Rajput, T. Rogers, R. Ross, and C. Ward. HERMES: A heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes/publications/abstracts/hermes.html, 1994.

[22] J. A. Wald and P. G. Sorenson. Resolving the query inference problem using Steiner trees. *ACM Transactions on Database Systems*, 9(3):348–368, 1984.