Selectively Materializing Data in Mediators by Analyzing Source Structure, Query Distribution and Maintenance Cost

Naveen Ashish, Craig A. Knoblock and Cyrus Shahabi
Information Sciences Institute, Integrated Media Systems Center and
Department of Computer Science
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{nashish,knoblock,cshahabi}@cs.usc.edu

Abstract

We present an approach to selecting data to materialize in Web based information mediators by analyzing multiple factors. An issue in building Web based information mediators is how to improve the query response time given the high response time for retrieving data from remote Web sources. We had earlier presented a framework for optimizing the performance of information mediators by selectively materializing data. In this paper we describe our approach for automatically selecting the portion of data that must be materialized by analyzing a combination of several factors, namely the distribution of user queries, the structure of sources and the update cost.

1 Introduction

There are several projects focusing on building information mediators, the representative systems include TSIMMIS [8], Information Manifold [9], The Internet Softbot [6], InfoSleuth [5], Infomaster [7], DISCO [12], HERMES [1], SIMS [2] and Ariadne [10]. Many of these systems allow database like querying of semi-structured Web sources through *wrappers* around pre-specified Web sources, and they also provide integrated access to multiple data sources. The query response time for information mediators, particularly Web based mediators is often very high, mainly because to answer most queries a large number of Web pages must be fetched over the network. For instance consider a mediator that provides integrated access to Web sources of information about countries in the world. For all mediator applications the set of Web sources from which the mediator will extract and integrate information is pre-specified and fixed. For the country mediator the set of sources is:

- The CIA World Factbook[1] which provides interesting information about the geography, people, government, economy etc. of each country in the world.
- The NATO homepage[2] from which we can get a list of NATO member countries.
- The InfoNation[3] source which provides statistical data about UN member countries.

Without any kind of optimization i.e., assuming that all data must be fetched from the Web sources in real time, a typical query to this mediator such as *"Find the defense expenditure and spending on education of all countries that have a national product greater than 500 billion dollars"* can take several minutes to return an answer. This is because for this particular query the mediator must retrieve the pages of all countries in the CIA World Factbook to determine which ones have a national product greater than $500 billion, which takes a large amount of time. The query response time can be greatly improved if frequently accessed data is materialized at the mediator side.

We presented an approach to improving performance in mediators by selectively materializing data in [3]. There are two key issues that must be addressed in designing a performance optimization system based on materialization. They are:

- Designing the overall materialization framework. This addresses the issue of how we represent and use the materialized data. We describe our approach in [3] and will briefly review it in the following section.

[1]http://www.odci.gov/cia/publications/factbook/country.html
[2]http://www.nato.int/family/countries.htm
[3]http://www.un.org/Pubs/CyberSchoolBus/infonation/e_infonation.htm

- Selecting data to materialize. In [4] we also stress the need to materialize data in a *selective* fashion. This leaves us with the issue of how to select the portion of data that is most useful to materialize. Our approach to the issue of selecting what data to materialize is the primary focus of this paper.

2 Overall Materialization Framework

In [3] we presented a framework for representing and using materialized data in an information mediator. The basic idea is to locally materialize useful data and define it as another information source for the mediator. The mediator then also considers using the materialized data instead of retrieving data from remote sources to answer user queries. The reader is referred to [3] and [4] for details.

3 Selecting Data to Materialize

We mentioned in [4] the need for materializing data in a *selective* manner. The brute force approach of simply materializing all the data in all the Web sources being integrated is impractical for two main reasons. First, the sheer amount of space needed to store the data could be very large. The other reason is that data can get updated at the Web sources and the maintenance cost for the materialized data becomes very high. There is thus the issue of how to identify the portion of data that must be materialized. We present an approach where we consider primarily the following three factors:

- The distribution of user queries. From the query distribution we can identify what classes of data are frequently queried by users.
- Structure of sources. We may be able to predetermine expensive classes of queries that could be asked of a source and prefetch and materialize data to improve response time for those queries.
- Updates at Web sources. The maintenance cost for the materialized data is also taken into consideration when materializing data.

We now describe our work in progress on analyzing each of the above factors and also how we propose to combine various factors in deciding what data to materialize.

3.1 Analyzing the Distribution of User Queries

In [4] we describe in detail our work on analyzing *one* of the above factors, namely the distribution of user queries to select what data to materialize. We have developed an algorithm known as the CM (Cluster and Merge) algorithm which identifies useful classes of information to materialize, by extracting patterns in user queries. A key feature of this algorithm is that it outputs a *compact* description of the patterns extracted. A compact description is necessary from a query planning perspective. We define a new information source in the mediator for each class of data materialized. A fragmented description of the patterns extracted will result in a large number of new sources created. Now the problem of query planning in a mediator is combinatorially hard and having a very large number of sources will cause performance problems for the planner. With a compact description of the patterns we need to materialize fewer classes of data and thus define just few new information sources.

The CM algorithm first classifies queries by analyzing constraints in the queries to determine what classes of data the user is interested in. We construct an ontology of such classes that users are interested in. For instance users may be interested in *European Countries* or *Democratic Countries* etc. For each such class we then determine what groups of attributes are frequently queried. For instance for European Countries users may be primarily interested in say the *economy* and *national product*. We try to merge together attribute groups queried with approximately the same frequency to make the description more compact. Finally we can further make the description more compact by merging classes based on class covering relationships. This algorithm thus tries to compactly describe classes of data that are of interest to users given a query distribution. For instance the algorithm may extract patterns such as users are interested in *"the national product and economy of all European countries"*.

3.2 Analyzing the Structure of Sources

In Web based mediators we provide database like querying access to semistructured Web sources by building wrappers around the sources. Often the Web sources have limited querying capabilities. As a result, certain kinds of queries can be very expensive as the query functionality not provided by the source is provided by the wrapper. It would thus be useful if in a mediator application for each Web source being integrated we could determine in advance what could be the expensive classes of queries that could be asked in the application. We can then materialize data that could improve the response time for the expensive kinds of queries.

In our approach, we start with a specification of the query interface to a mediator application that defines exactly the kinds of queries a user could ever ask of that mediator application. We then estimate the costs of all different classes of queries using a cost estimator which is part of the mediator. The purpose is to identify in advance the expensive kinds of queries that could be asked in a particular mediator application. Then using heuristics based on the kind of query, source or sources used to answer the query and also the different data processing operations that are performed to answer the query, we prefetch and materialize data that can improve the response time for the expensive query.

For instance consider a mediator application that includes amongst other sources an online *geocoder* that accepts street addresses and returns the latitude and longitude of the place. Now geocoding (converting street addresses to latitudes and longitudes) a set of addresses using this source is an expensive query since the source is structured such that we can only geocode one address at a time. Further the mediator application might be such that we can only geocode a fixed set of places every time (for instance the set of restaurants in LA). In such a case we should materialize the result of geocoding this set of places even before analyzing any set of user queries to the mediator.

3.3 Updates

Finally we must address the issue of updates at Web sources. First the materialized data must be kept consistent with that in the Web sources. It may be that the user is willing to accept data that is not the most recent in exchange for a fast response to his query (using data that is materialized). Thus we need to determine the frequency with which each class of data materialized needs to be refreshed from the original sources. Next the total maintenance cost for all the classes of data materialized must be kept within a limit that can be handled by the system.

We have developed a language for describing the update characteristics and frequency of updates for various source classes and attributes and also the user's requirements for freshness of each domain class and attributes. We illustrate the kinds of characteristics we can specify using an example. Consider an information source for movies which we model as a source relation MOVIE_SRC having attributes such as theatre, showtimes, actors, director, review etc. Table 1 shows the update specification for the movies relation and some attributes, the specification is implemented as a database relation.

It states that for the MOVIE_SRC source class, the MEMBERSHIP= 'A' i.e., arbitrary (instances of the class can be added or deleted), CHANGE = 'Y' i.e, yes (values of objects or class members can change), TIME_PERIOD is 1 week so the data changes once every week and the TIME of change is every friday. The attributes CHANGE, TIME_PERIOD and TIME also characterize each attribute of a source class. By default for each attribute in a source class the values of the update characterization attributes are propagated from the source relation they are part of. So for instance the attribute 'theatre' has the characteristics CHANGE= 'Y', TIME_PERIOD= 1 week etc. These defaults can be overridden by explicitly stating the new values. So for instance we could specify the characteristics for the 'actors' attribute as shown in Table 2 since the value of actors for MOVIE_SRC does not change.

For each domain class we also specify for each attribute the user's requirements for currency of data, actually stated in terms of how stale he can tolerate the data to be. Consider a domain class called MOVIE where we integrate information from several sources about movies.

Table 3 shows the user's tolerance for various attributes of the domain class MOVIE. It states that the value of attributes such as 'theatre' and 'showtimes' must be current (having a tolerance of 0) whereas the 'review' can be upto 6 weeks old.

We have developed an algorithm that using the above specifications can determine the frequency with which attributes in each materialized class must be updated to be consistent with the user's requirements and thus the maintenance cost for that class. The maintenance cost is considered in two ways. First, for each individual class if the maintenance cost is very high we may decide not to materialize it. For instance a class having a *stock quote* as one of its attributes which is updated every 5 minutes has a very high maintenance cost and it is better not to materialize such a class at all. Second, the total maintenance cost for all the classes of materialized data should be kept within a limit that can be handled by the system.

3.4 Considering Multiple Factors for Materialization

The decision of what classes to materialize is based on a combination of the different factors mentioned above i.e., distribution of queries, structure of sources and maintenance cost. We are working on developing a solution to the problem of how to combine different factors when deciding to materialize data. We have identified some of the issues that need to be addressed. For instance one issue is in what order (if any) should the various factors be analyzed ? The maintenance cost must always be taken into account for any class of data materialized. In fact the maintenance the cost for a particular data item may be very high and thus we may decide not to materialize any class containing that data item. Thus the update costs should be analyzed before the source structure or query distribution. Based on source structure analysis and with estimates of update costs we may decide to materialize a particular class even before looking at the user query distribution. Thus source structure should be analyzed before the query distribution. We are developing a systematic framework that defines exactly how various factors are analyzed, how they affect one another and how various

CLASS	MEMBERSHIP	CHANGE	TIME_PERIOD	TIME
MOVIE_SRC	A	Y	1 week	week:friday

Table 1: Characteristics of updates for a source class

ATTRIBUTE	CHANGE	TIME_PERIOD	TIME
actors	N	-	-

Table 2: Characteristics of updates for an attribute

factors are combined to determine what classes of data are to be materialized.

4 Admitting and Replacing Materialized Classes

Finally there is the issue of having a policy for admitting and replacing classes of materialized data. Each class of materialized data uses two resources, space needed for local storage and maintenance cost in case of updates. For any mediator application it is reasonable to assume that we would have a fixed limited space for storing all the materialized classes and also a maximum total maintenance cost that can be borne for keeping the materialized data consistent. We are developing a strategy similar to the one described for data warehouse cache management in [11] where the execution cost of queries in a class, space occupied by the class and maintenance cost are taken into account for deciding in what order of priority classes of data are to be materialized.

5 Initial Results

We have implemented an optimization system for the Ariadne information mediator using the approach of defining the materialized data as another information source described above. In this initial implementation the query distribution is analyzed using the CM algorithm to decide what classes of data to materialize. Source structure analysis and the update issue were not addressed in this initial implementation. The primary objectives of this set of experiments were:

- To determine whether the CM algorithm is indeed successful in extracting patterns that may be present in queries.
- Performance improvement (measured by reduction in average query response time) provided by our optimization scheme. We compare the performance improvement with Ariadne without any kind of optimization and also with Ariadne with an optimization system based on existing schemes such as page level caching.

5.1 Experimental Results

The experiments were conducted for the countries information mediator, an Ariadne application that integrates information from multiple sources of information about countries in the world and that we had made available online. We measured the total query response time against two query sets to the mediator without any optimization system, with our optimization system and with an optimization system based on page level caching. The first query set Q1 is one that we generated and in which we introduced some distinct query patterns (detailed description of query set omitted for lack of space). The second query set Q2 is the set of actual user queries to the countries mediator that we had made available online.

Table 4 shows the response time (total time for the entire query set) without any optimization, with page level caching and with our system for both query sets. First, the CM algorithm does appear to be effective in extracting patterns from user queries. It successfully extracted the patterns that were deliberately introduced in the query distribution Q1 and also extracted patterns from the actual user query set Q2. We materialized these frequently queried classes of data to improve performance. Also our system appears to be effective in improving performance. For Q1 (200 queries) with a limited space for materialized data our system system not only provides significant improvement in query response time but also the optimization is an order of magnitude better than with page level caching with the same local space. We also show query response against the set of actual user queries Q2 (182 queries). Again with our optimization system the performance is much better than no optimization or with page level caching.

6 Conclusions

We have presented an approach to automatically selecting data to materialize in information mediators. We reviewed our overall approach for optimizing performance in information mediators by selectively materializing data and defining it as an auxiliary information source. We presented an approach where multiple factors are taken into account to select data to materialize in an optimal fashion. Our approach has been developed with a particular focus on semistructured Web sources. As certain kinds of queries can be very expensive given the limited query capabilities of Web sources we have presented an approach for prefetching data to

ATTRIBUTE	TOLERANCE
theatre	0
showtimes	0
actors	0
director	0
review	6 weeks

Table 3: Staleness tolerances for attributes of a domain class

Query set	Response Time (No optimization)	Response Time (Page level)	Response Time (Our system)	%improvement (Page level)	%improvement (Our system)
Q1	9661 sec	8695 sec	1536 sec	10 %	84 %
Q2	3742 sec	3255sec	2245 sec	13 %	40 %

Table 4: Query response times

improve response time for such queries. Also we take into account the fact that Web sources can get updated and that the user needs to be provided data consistent with his requirements for freshness. We expect that this work will ultimately provide an effective solution to the performance issue in Web based information mediators.

References

[1] S. Adali, K. S. Candan, Y. Papakonstantinou, and V.S.Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, 1997.

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

[3] N. Ashish, C. A. Knoblock, and C. Shahabi. Intelligent caching for information mediators: A kr based approach. In *Knowledge Representation meets Databases (KRDB)*, Seattle, WA, 1998.

[4] N. Ashish, C. A. Knoblock, and C. Shahabi. Selectively materializing data in mediators by analyzing user queries. In *Fourth International Conference on Cooperative Information Systems (CoopIS)*, Edinburgh, Scotland, September 1999.

[5] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Semantic integration of information in open and dynamic environments. Technical Report MCC-INSL-088-96, MCC, Austin, Texas, 1996.

[6] O. Etzioni and D. S. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7), 1994.

[7] M. Genesereth, A. Keller, and O. Duschka. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, 1997.

[8] J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Information translation, mediation, and mosaic-based browsing in the tsimmis system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Jose, CA, 1995.

[9] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution engine for data integration. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, June 1999.

[10] C. A. Knoblock, S. Minton, J.-L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. Philpot, and S. Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, Madison, WI, 1998.

[11] P. Scheuermann, J. Shim, and R. Vingralek. Watchman: A data warehouse intelligent cache manager. In *Proceedings of the 22nd VLDB Conference*, Mumbai(Bombay), India, 1996.

[12] A. Tomasic, L. Raschid, and P. Valduriez. A data model and query processing techniques for scaling access to distributed heterogeneous databases in disco. In *Invited paper in the IEEE Transactions on Computers, special issue on Distributed Computing Systems*, 1997.

A COUNTRY Relation

Relation: COUNTRY
Attributes: (geography location map_references region area total_area land_area comparative_area land_boundaries coastline maritime_claims international_disputes climate terrain natural_resources land_use irrigated_land environment note people population age_structure population_growth_rate birth_rate death_rate net_migration_rate infant_mortality_rate life_expectancy_at_birth total_fertility_rate nationality ethnic_divisions religions languages literacy labor_force government names digraph type capital administrative_divisions independence national_holiday constitution legal_system suffrage executive_branch legislative_branch judicial_branch political_parties_and_leaders other_political_or_pressure_groups diplomatic_representation_in_US us_diplomatic_representation organization flag economy overview national_product national_product_real_growth_rate national_product_per_capita inflation_rate_consumer_prices unemployment_rate budget exports imports external_debt industrial_production electricity industries agriculture illicit_drugs economic_aid currency exchange_rates fiscal_year transportation railroads highways inland_waterways pipelines ports merchant_marine airports communications telephone_system radio television defense_Forces branches manpower_availability defense_expenditures)