Optimizing Information Agents by Selectively Materializing Data *

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 for optimizing the performance of information agents by materializing useful information . A critical problem with information agents, particularly those gathering and integrating information from Web sources is a high query response time. This is because the data needed to answer user queries is present across several different Web sources (and in several pages within a source) and retrieving,extracting and integrating the data is time consuming. We address this problem by materializing useful classes of information and defining them as auxiliary data sources for the information agent. The key challenge here is to identify the content and schema of the classes of information that would be useful to materialize. We present an algorithm that identifies such classes by analyzing patterns in user queries. We describe an implementation of our approach and experiments in progress. We also discuss other important problems that we will address in optimizing information agents.

Introduction

There is currently great interest in building information agents that can integrate information from multiple data sources. The representative systems include TSIMMIS (Hammer *et al.* 1995), Information Manifold (Kirk *et al.* 1995), The Internet Softbot (Etzioni & Weld 1994), InfoSleuth (Bayardo *et al.* 1996), Infomaster (M.R.Genesereth, A.Keller, & O.Duschka 1997), DISCO (Tomasic, Raschid, & Valduriez 1997), SIMS (Arens, Knoblock, & Shen 1996) and Ariadne (Ambite *et al.* 1998) – a descendant of SIMS focusing on integrating Web sources. Such systems allow database like querying of semi-structured Web sources through *wrappers* around the Web sources, and also provide integrated access to multiple data sources. The query response time for information agents is often very high, mainly because to answer typical queries a large number of Web pages must be fetched over the network. For instance consider an agent that provides integrated access to Web sources of information about countries in the world[1]. Specifically the sources we provide access to are:

- The CIA World Factbook[2] which provides interesting information about the geography, people, government, economy etc. of each country in the world.
- The Yahoo listing of countries by regions from where we can obtain information such as what countries are in Asia, the Pacific Rim etc.
- The NATO homepage from we can get a list of NATO member countries.

Assuming that all data must be fetched from the Web sources at real time, a typical query to this agent such as *"Find the defense expenditure of all countries that have a national product greater than $500 billion"* can take as much as around 10 minutes to return an answer. This is because for this particular query the agent 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 agent side.

We present a performance optimization approach for information agents by selective materialization of data. We describe the overall approach including how we represent and use the materialized data and also present an algorithm for determining what classes of information are useful to materialize. We also discuss several other issues we will address in order to provide effective performance optimization for information agents.

Our Approach to Optimization

We first provide a brief overview of the SIMS information agent, in fact the SIMS architecture is typical of many of the other information agent systems we mentioned. The Ariadne architecture also borrows heavily from that of SIMS. SIMS is used to integrate information

Copyright ©1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved. This research has been supported by USC's Integrated Media Systems Center (IMSC), an NSF Engineering Research Center.

[1]We provide access to a working version of such an agent at http://www.isi.edu/ariadne

[2]http://www.odci.gov/cia/publications/factbook/country-frame.html

from mainly database systems whereas Ariadne integrates information from semi-structured Web sources. We then provide a high level description of our optimization approach.

SIMS Architecture

In the SIMS system we use the LOOM (MacGregor 1988)knowledge representation language (we can also perceive this as a data model) for modeling data. The user is presented with an integrated view of the information in several different sources which is known as the *domain model.* We describe the contents of the individual information sources in terms of the domain model. A simple example is shown in Figure 1 (a). The white circle labeled COUNTRY represents a domain *concept* (equivalent of a class in an object-oriented model) and the shaded circles represent sources. The arrows on the circles represent attributes of the concepts. In this example the domain concept COUNTRY provides an integrated view over two sources of information about countries – FACTBOOK-COUNTRY and COUNTRY-HEADS. The user queries the integrated view i.e., concepts in the domain model and the query planner in the agent generates plans to retrieve the requested information from one or more sources. Please refer to (Arens, Knoblock, & Shen 1996) for a more detailed description of SIMS.

Overall Optimization Approach

Our approach to optimization is based on an idea described in (Arens & Knoblock 1994) where we identify useful classes of information to materialize, materialize the data in these classes in a database local to the agent and define these classes as auxiliary information sources that the agent can access. For instance in the countries application suppose we determined that the class of information - *the population and national product of all European countries* was frequently queried and thus useful to materialize. We materialize this data and define it as an additional information source as shown in Figure 1 (b). Given a query the agent prefers to use the materialized data instead of the original Web source(s) to answer the query.

Defining the materialized data as another information source provides two benefits. First, we can provide a semantic description of the contents of the materialized data (in LOOM). Second, the query planner in the agent considers the materialized data source also when generating plans to retrieve data to answer a user query. The SIMS query planner is designed to generate high quality plans and will generate plans that attempt to use the materialized data sources to the maximum extent. We use the materialized data in essentially the same manner as in a *semantic* caching (Dar *et al.* 1996) system. Our approach of defining the materialized data as another information source allows us to use the information agent's knowledge representation system and query planner to address two important problems that arise in any semantic caching system, namely providing a description of the materialized or cached data and doing containment checking i.e., determining if all or some portion of data in a query is already cached.

There are several advantages in using a semantic caching approach as opposed to a non-semantic approach such as caching at the level of individual Web pages. First, a semantic approach provides more flexibility in specifying what information to materialize. We provide structured query access to information on Web pages and users may be only interested in a small fraction of the information on a page. With page level caching we have to cache the entire Web page but with semantic caching we can specify exactly what attributes on the page we would like to cache. Second, semantic caching allows more effective reasoning about the contents of the data in the cache. For instance suppose we have materialized all attributes of all the European countries. Now assume that there is a query asking for say the national product of all European countries. With page level caching we still have to scan through the pages of all countries in the factbook to determine which ones are in Europe, thus having materialized the data does not save us any time. With a semantic approach however we can reason that all the European countries are in the cache and we do not need to retrieve the page for any country from the actual Web source. Finally semantic caching also allows us to formulate *remainder* queries when part of the requested data is materialized. For instance assume that we have materialized all attributes of European, Asian and African countries. Given a query asking for say the national product of all countries in the world, the query planner will plan to retrieve the national product of European, Asian and African countries from the cache and only the remainder data i.e., the national product of North and South American and Australian countries from the actual Web source.

The key problem in our approach is to determine what classes of information are useful to materialize. We present below an algorithm that determines such useful classes by analyzing patterns in user queries.

RM Algorithm

We provide a description of the RM (Relaxed Merge) algorithm which identifies useful classes of information to materialize. There are three main steps in the algorithm. First we analyze user queries to determine the classes of information that are frequently queried. Next for each such class we try to group together attribute sets that are queried with about the same frequency. Finally we attempt to merge sets of several classes into one whenever possible, in order to reduce the number of classes materialized. This is because it is important that the number of materialized sources be kept small from a query processing perspective. The general problem of query planning to retrieve information from a number of integrated sources is combinatorially hard and having a very large number of information sources created due to the materialized data will create performance problems for the query planner.

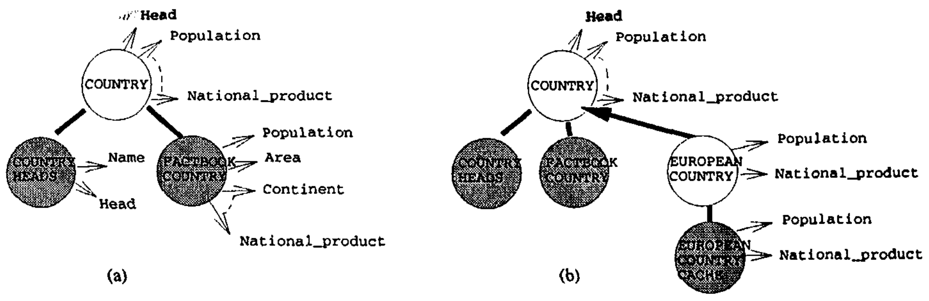

Figure 1: Information modeling in SIMS

The RM algorithm analyzes a large number of previous queries in each pass. We now describe the steps in the algorithm in more detail.

1. Identifying classes of interest. We maintain an ontology in LOOM of classes of information that are queried by users. Initially the ontology is comprised of classes in the domain model and in the individual sources being integrated. We then add subclasses of the existing classes in the ontology by analyzing constraints in the queries. Assuming an SQL syntax for user queries we analyze the WHERE clause in the queries. For instance (for the concept COUNTRY) a query such as:

```
SELECT Population, Area
FROM Country
WHERE Continent = ''Europe'';
```

indicates that the user is interested in the subclass of European countries (i.e., a Country with the Continent attribute bound to Europe). We add the subclass of EUROPEAN COUNTRIES to the ontology if it does not already exist there. For instance for a set of queries on the concept COUNTRY in which the WHERE clauses have constraints on the attributes Continent (bound to a value such as Europe, Asia etc.) or Government Type (bound to a value such as Republic, Monarchy, Communist etc.) or both, we would create an ontology such as shown in Figure 2.(The arcs in the figure represent coverings of groups of subclasses for the superclass COUNTRY).

2. Finding attribute clusters in each class. For each class we record what attribute sets have been queried and with what frequency. Frequency information is maintained for attribute sets rather than for individual attributes because it is important to consider attribute groupings when materializing. If a particular group of attributes is queried in several queries, materializing only part of these attributes does not provide any benefit at all as the system must still query the remote Web source to get the remaining attributes. Thus we should materialize either all or none of the attributes in an attribute group.

We also attempt to merge together attribute sets with similar frequencies in order to reduce the number of sets for each class that we have to consider. Attribute sets are merged together if the relative difference of their frequencies is within a certain predefined threshold value.

3. Merging classes. We mentioned earlier that it is important to keep the number of classes of information materialized small from a query processing perspective. Consider the following classes of information [3] - (EUROPEAN-COUNTRY,{Population,Area}), (ASIAN-COUNTRY,{Population,Area}), (AFRICAN-COUNTRY,{Population,Area}), (N.AMERICAN-COUNTRY,{Population,Area}), (S.AMERICAN-COUNTRY,{Population,Area}) and (AUSTRALIAN-COUNTRY,{Population,Area}). We could replace the above six classes by just one class (COUNTRY,{Population,Area}) which represents exactly the same data. In general thus a group of classes of information of the form (C1,A), (C2,A), ... ,(Cn,A) may be replaced by one class i.e., (S,A) if C1,C2,..,Cn are direct subclasses of S and form a covering of S. As the ontology of classes is maintained in LOOM, we use LOOM to determine groups of classes we can merge based on class/subclass relationships.

In fact we also allow for a kind of 'relaxed' merge where we may merge a group such as (C1,A1), ..., (Cn,An) to (S,A) where the *Cis* are direct subclasses of S as above. However A1,..., An need not be exactly equal sets rather they just need to overlap well, and A is the union of A1,...,An. The disadvantage in this case is that the merged class of information will contain some extra data i.e., data not present in any of the classes of information merged to form the merged class and we will waste space to store this extra data. However we trade some space occupied by extra data in exchange for the advantage of reducing the number of classes materialized. The criteria of how much extra space can be wasted in exchange for a reduction in number of classes should be determined carefully, there are disadvantages if it is either too strict or too relaxed. This criteria can be varied by changing the value of a configurable parameter in the materializing system whose value corresponds to how strict or relaxed the merging can be.

After merging all possible sets of classes of information as described above, the RM algorithm outputs the

[3] A "class of information" is identified by the pair (C,A) where C is a class and A is a set of attributes

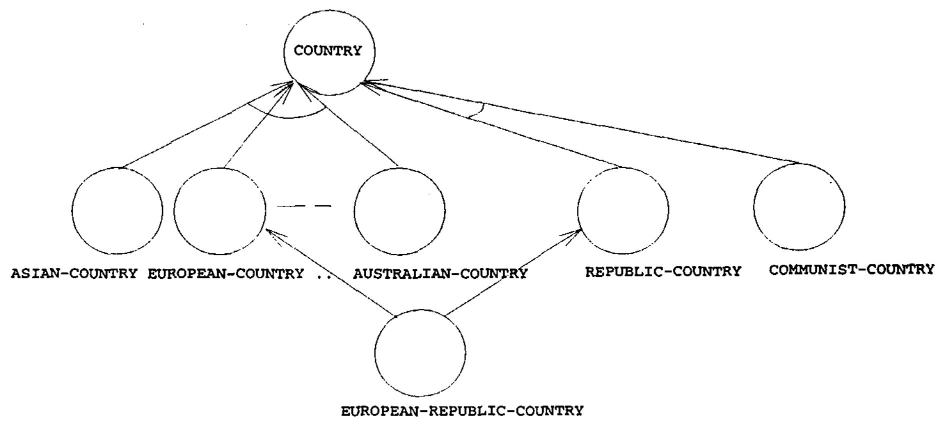

Figure 2: Ontology of subclasses of COUNTRY

resulting classes of information sorted by number of queries to each class. We run the RM algorithm periodically. An appropriate time period is chosen for each agent application. We flush the existing materialized data and then begin to materialize classes of information output by the RM algorithm sorted by number of queries. We materialize classes till all available space for the materialized data is used up or all classes output by the RM algorithm are materialized.

Implementation and Experiments

We have implemented an optimization system for the Ariadne information agent using the approach and RM algorithm described above. Currently we are engaged in performance analysis of the optimization system. The initial set of experiments will be run for the countries application domain described earlier. We would be using actual user queries to the application available online[4] as well as generated queries which follow some identifiable patterns.

Specifically we are interested in estimating the following:

- Whether the RM 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 as compared to existing optimization schemes such as caching data at the level of individual Web pages.
- The effect of varying the values of the various configurable parameters of the RM algorithm on system performance. This will give an idea of what the optimal values for the parameters should be.

Related Work

Improving performance by materializing or caching data is a topic that has been extensively investigated in the context of client–server database systems, operating systems and more recently for proxy servers for the Web. Also although there are several efforts in progress on building information integration agents, very few have addressed the issue of improving performance by materialization or other means.

Most work on client-server database systems caching and operating systems caching has focused on tuple level or page level caching. Only recent approaches are based on semantic level (Dar *et al.* 1996) or *predicate based* (Keller & Basu 1996) caching. A problem with the semantic caching approach is that that the containment checking problem i.e., determining exactly what portion of the data requested in a user query is present in the cache, is hard and having a large number of semantic regions creates performance problems. A solution proposed in (Keller & Basu 1996) is to reduce the number of semantic regions by merging them whenever possible. This is in fact an idea we have built on. In the RM algorithm we have presented an approach for systematically creating new semantic regions to consider for materializing and merging them when possible. We have also proposed a relaxed merging of semantic regions in addition to exact merging. Also we make use of a dynamically constructed ontology of subclasses of information of interest that helps us decide what groups of regions we can merge as opposed to a brute force approach of checking all combinations of semantic regions.

Caching schemes have also been proposed for Web proxy servers (Chankunthod *et al.* 1995) to reduce the amount of data that must be shipped over the internet. However these approaches are based on caching at the level of individual Web pages, which we argued is not optimal in our environment. In fact we hope to show through experiments that our approach does indeed outperform caching or materializing at the level of entire Web pages.

There is some work on caching that has been done for the information integration environment. An approach to caching for mediator systems is described in (S.Adali *et al.* 1997). The focus of their work how-

[4]At http://www.isi.edu/ariadne/

ever is caching for a mediator environment where we integrate information from sources that may not be traditional database systems. Their contribution is a caching approach based on first estimating the cost of accessing various sources based on statistics of costs of actually fetching data from the sources. In their approach reasoning about cache contents is done through the notion of *invariants* which are basically expressions that show possible substitutions or rewritings of queries. This approach provides quite limited semantic reasoning capabilities about the contents of the cached data as compared to our approach in which we are able to perform more powerful reasoning of the materialized data contents through LOOM.

Another approach to caching for federated databases is described in (Goni *et al.* 1997). Theirs is also a semantic caching approach where the data cached is described by queries. There are explicitly cached queries (queries directly corresponding to data cached) and also implicitly cached queries i.e, queries that can be answered if the data in the explicitly cached queries is cached. At all times the system maintains the set of explicitly and implicitly cached queries and containment reasoning is done against this set for each user query to determine what portion of the requested data is cached. They also define some criteria for choosing an optimal set of queries to cache. Finding the optimal set is an NP-complete problem and they use an A* algorithm to obtain a near optimal solution. A limitation of their approach is that the cached classes can only be in terms of classes in a predefined hierarchy of classes of information for a particular application. Our approach is much more flexible in that we dynamically propose new subclasses of information of interest by analyzing constraints in the queries.

Discussion

We have described an approach for optimizing the performance of information agents. We presented the RM algorithm which is used in determining what classes of information are useful to materialize as auxiliary data sources. However there are several other important issues that remain to be addressed and which we will be working on.

One problem arises due to the fact that the data materialized may get updated at the original Web source. Developing an effective solution to the update problem requires us to consider several factors such as with what frequency is the data in the original Web source updated, is the time of update known in advance and also how critical it is for the user to have the absolute latest data. Also the materialized data may not exactly correspond to some data item in the individual Web sources, rather it might be aggregated over several data items. In this case the updates to the individual data items must be propagated intelligently to the aggregated materialized data. The above is not an exhaustive list of the issues involved in the update problem and we believe that it is an important problem that must be addressed in an optimization system based on materializing data developed for any Web based information integration environment. The update problem also arises in the context of materialized views and solutions have been proposed although primarily in the context of data sources that are relational databases. We would use the solutions that are applicable to our problem from research on materialized views and work on issues that remain to be addressed for a Web based environment such as ours.

Another important feature that must be present in an integration environment is to be able to materialize data in terms of classes that are close to the user queries. These may be domain level classes that are integrated views over several sources or classes in individual sources. The problem is complicated because the domain and individual source classes cannot be analyzed independently for materialization.

A wealth of data from the internet is accessible via information agents. However the costs for answering typical queries are prohibitive if all data must be fetched over the internet in real time. By augmenting the information agents with effective performance optimization systems we hope to reduce the query response time so that it may be acceptable for practical applications.

References

Ambite, J.-L.; Ashish, N.; Barish, G.; Knoblock, C. A.; Minton, S.; Modi, P. J.; Muslea, I.; Philpot, A.; and Tejada, S. 1998. Ariadne: A system for constructing mediators for internet sources. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.*

Arens, Y., and Knoblock, C. A. 1994. Intelligent caching: Selecting, representing, and reusing data in an information server. In *Proceedings of the Third International Conference on Information and Knowledge Management.*

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

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

Chankunthod, A.; Danzig, P. B.; Neerdaels, C.; Schwartz, M. F.; and Worrell, K. J. 1995. A hierarchical internet object cache. Technical Report 95-611, Computer Science Department, University of Southern California, Los Angeles, CA.

Dar, S.; Franklin, M. J.; Jonsson, B. T.; Srivastava, D.; and Tan, M. 1996. Semantic data caching and replacement. In *Proceedings of the 22nd VLDB Conference.*

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

Goni, A.; Illarramendi, A.; Mena, E.; and Blanco, J. M. 1997. An optimal cache for a federated database system. *Journal of Intelligent Information Systems* 1(34).

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

Keller, A. M., and Basu, J. 1996. A predicate-based caching scheme for client-server database architectures. *The VLDB Journal* 5(2):35–47.

Kirk, T.; Levy, A. Y.; Sagiv, Y.; and Srivastava, D. 1995. The information manifold. In *Working Notes of the AAAI Spring Symposium on Information Gathering in Heterogeneous, Distributed Environments.*

MacGregor, R. 1988. A deductive pattern matcher. In *Proceedings of AAAI-88, The National Conference on Artificial Intelligence.*

M.R.Genesereth; A.Keller; and O.Duschka. 1997. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.*

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

Tomasic, A.; Raschid, L.; and Valduriez, P. 1997. 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.*