Intranet Mediators: A Prototype

M. Saelee, S. Beitzel, E. Jensen, D. Grossman, O. Frieder
Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
{*lee,steve,ej,grossman,frieder*}*@ir.iit.edu*

Abstract

We describe a mediator designed specifically for an intranet rather than an internet. Our intranet mediator allows data reconciliation and integration to be completed long before the query is executed. This is because information about the data sources is known and a combined schema can be defined prior to the query. On the internet, source schemas are not available so integration cannot occur before the query. In our approach, a data warehouse is used for reconciliation of structured data. We propose a new architecture and describe an initial prototype built to test the architecture.

1. Introduction

Mediators provide unified access to disparate data. The idea is that a query entered by the user is sent to the mediator which sends the query to a variety of sources and combines the results. Typically, within a given enterprise a single intranet provides a vast resource for electronic information. Using an intranet mediator, one-stop-shopping can be provided for this information.

Much work has been done on internet mediators where sources are reconciled through form-based interfaces. A query that asks for information about books may well be sent to the Amazon and the Waldenbooks web sites. The corresponding form-based interfaces are then used to access data at each site. The mediator reconciles any conflicts (e.g., one site may not have a publication date, another may not have the ISBN number) and presents a result set to the user that appears to have come from a single data repository. The key here is that all data reconciliation is done at query time, as the internet mediator does not have access to the actual source databases — only the form based web sites.

On an *intranet*, schema integration can be done in a data warehouse long before a query is submitted. This allows us to focus on the question of how to identify the best data sources for a given natural language query. To explore this question, we developed an architecture that specifically targets the intranet problem and implemented a prototype (currently about 10,000 lines of Java source code). Section 2 gives relevant background on internet and intranet mediators. Section 3 presents prior work. Section 4 describes the architecture with details of our prototype and Section 5 contains a summary and directions for future work.

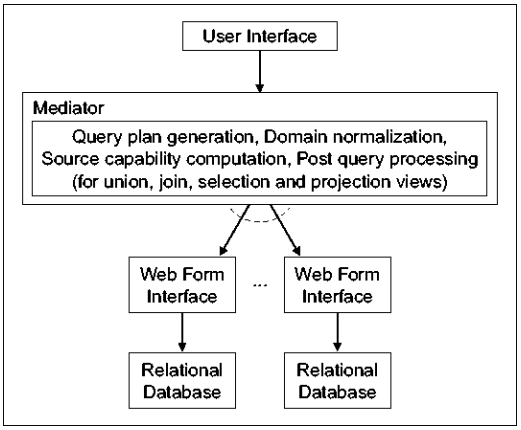

Figure 1: Typical Internet data integration architecture

2. Background

A typical architecture for internet data integration systems is shown in Figure 1. The mediation unit in internet data integration systems is complex, as much work needs to be done during query time to ensure schema integration.

Intranet mediators, on the other hand, use data warehouses to address schema integration. In this way, problems that plague the internet mediator disappear, as we now have a single clean source for structured data. Almost every Fortune-500 company has or has plans to build a data warehouse [21, 22].

The presence of a data warehouse might seem to obviate the need for a mediator, as queries going to the data warehouse would be facilitated by existing database access and query tools. [21, 22]. However, a stand-alone data warehouse only interprets data in structured sources. Migrating terabytes of unstructured data (e.g. text, images and video) into a warehouse is not feasible [4, 5]. Our mediator uses a data warehouse for storing and integrating its structured

data, while accessing unstructured sources simultaneously.

3. Prior Work

Prior work can be broken down into research on data integration in a data warehouse and internet mediation and metasearch. Work has also been done on natural language concept identification and automated query generation.

Our group has worked on fusion of information retrieval [29], where separate retrieval strategies are used together to improve retrieval effectiveness. Prior work in this area is described in [1, 34, 32, 13, 26, 29, 31]. Previously, we have shown how to integrate structured and unstructured data within a single data warehouse [18, 19, 28]. A survey of work done in the integration of distributed, heterogeneous database systems is found in [12]. Research has also gone into migrating data into common formats (e.g. XML) that support easier mediation [2, 6].

A number of popular metasearch systems of web based information retrieval engines are in widespread use. Popular metasearch engines include www.inquirus.com, www.mamma.com, www.dogpile.com, www.savvy-search.com, and www.metacrawler.com. Result combination [10, 20, 24, 23] and caching strategies [24, 23] are essential to data integration in metasearch engines. In addition, specialized indexes that develop summaries of source collections may be used to effectively choose which source system is most likely to contain the results [15, 16].

Work has also been conducted on semantic query processing for the automation of executable query generation and phrase and concept recognition. This aids in the development of natural language based query interfaces. [27] discusses issues related to concept identification in queries and [11, 33, 35, 9] examine natural language parsing and query generation.

4. Architecture

This section provides an overview of the architecture of the mediator engine as illustrated in Figure 2. An object oriented approach was adopted in the design of the mediator.

There are three high-level components to the engine: the user interface, the core mediator, and the query modules. The user interface consists of a simple text form and an optional selection box of available structured sources. The mediator's responsibility is to propagate information in the query to relevant query modules. The query modules, in turn, pose queries to structured and unstructured sources.

The mediator, which carries out the primary functions of the engine, is similarly subdivided into several modules. A query processor object performs a "translation" of a user's query into a machine decipherable tree of concepts and Boolean relations. A series of target source decision making modules are then responsible for selecting data sources relevant to a query and sending it to the query modules paired

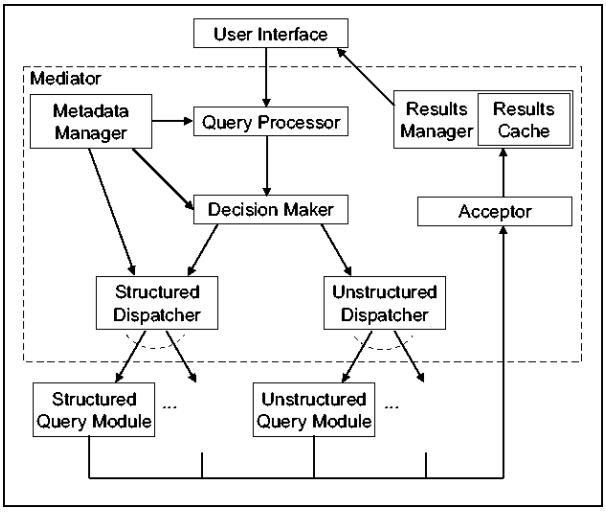

Figure 2: Architecture of the intranet mediator engine

with those sources. The source selection process is aided by a repository of metadata that we maintain about each source, stored in the Metadata Manager component of the mediator.

Results from the various query modules will be sent back up to the mediator, which contains two additional objects: the acceptor and results manager. They accept results from the various running modules and combine them into an aggregate packet to be sent back to the user interface. Incorporating results from all modules requires interpreting relevance ranks given by the disparate sources and deciding which sources are more relevant for the given query.

4.1. Query Processing and Dispatching

This section details the requirements and structuring of components in the engine that carry out the query processing and source dispatching actions.

4.1.1. Metadata Requirements. A primary item of concern was an optimal structuring of metadata related to sources that would allow for efficient decision making as to the relevance of a source per posed query. The metadata resource also contains stopword and synonym lists.

We propose a scheme wherein, for each structured source available, a series of relations between the source's attributes and relation names and a list of keywords or "synonyms" are constructed. If a match occurs between query concepts and synonym relation items it increases the relevance of the related source to the query. This structure allows easily automated insertions, as distinct individual record values for source relation domains would generally qualify as synonyms for its attribute name.

Oracle system tables are used to create relations for targeted sources within the data warehouse, which resides in an Oracle database. Synonym lists were compiled and entered together with the addition of sources to the engine.

4.1.2. Query Processing. When a query arrives from the user interface, it is translated into a machine decipherable concept tree using relational Boolean operators. The engine is capable of building an expression tree from a natural language query, joining nodes of the tree with OR operators, except where explicit Boolean expressions are specified. No automatic phrase processing or stopword removal is done here; the latter is left as an option for the lower level query modules upon sending queries to specific sources.

A survey of several packages for part-of-speech tagging and object identification within queries has been conducted, and initial work has been done to integrate natural language syntactic representations of queries into the system.

4.1.3. Target Source Selection and Dispatching. After query processing is complete, the query is passed to the decision making components of the mediator. Concepts extracted from the query expression tree and synonym data are used to decide if any of the available concepts indicate possible related sources of structured data. If such evidence is found, the query is passed to the structured dispatcher, which then uses metadata to target specific structured sources. The query is always passed to the unstructured dispatcher, even in the event of no structured sources being chosen. Modules are subsequently instantiated for each source selected.

4.2. Result Accumulation and Ranking

As the query modules run in separate threads of execution, the results they return to the mediator have no inherent ordering. The acceptor object, which contains open pipes to all active query modules, accumulates the result packets from the modules asynchronously. It then passes the accumulated results to the Result Manager object for reconciliation of relevance rankings and placement into the Results Cache. The query modules are detailed below.

4.2.1. Unstructured Result Retrieval. Each unstructured data source has an associated module responsible for low-level query formulation, query posing, and result gathering. We have implemented a number of these modules for popular web search engines.

The concurrent nature of the executing query modules presents us with a conflict between two of the major goals of the system: providing a well-unified result set and presenting the result set to the user quickly.

Alternatives we considered for result management were:

- Purely asynchronous result gathering
- Asynchronous result gathering and final reranking
- Asynchronous result gathering and dynamic reranking
- Synchronized completion of result gathering
- Our approach: Synchronized completion of gathering a *threshold* number of results from all sources

A purely asynchronous approach displays results as soon as they arrive in the acceptor, but slower performing source applications might return high ranked results after results have already been displayed. Final reranking proves both inconvenient and unresponsive, as the delay for the final ranking corresponds to the time for gathering all results. Dynamic reranking, demonstrated in Inquirus 2 [14], requires arriving results to be directly propagated to the user interface, which needs a coexisting facility for dynamic updates. This can be confusing to the user as the results display is constantly being refreshed. The synchronized completion of all modules ensures optimal result unification, but reduces the engine to worst link driven performance.

We use a results management scheme that compromises between purely asynchronous result propagation and synchronized completion of result gathering from each source. The algorithm we propose is given in Figure 3. We begin by submitting the query to our unstructured sources and queuing the asynchronously arriving results within the acceptor for processing in the results manager. This prevents a single slow or inoperative source from affecting total system performance. As results are obtained from the acceptor and placed in a cache, duplicates are removed. Not until a threshold of D results are cached for the current display projection are results ordered and sent to the user interface. In the meantime, more results are being asynchronously collected and cached in the acceptor and results manager until a preset threshold of R total results has been cached.

In practice, we find that varying the threshold of results obtained before results combination beyond the size of the result set displayed in the user interface allows for more control over the result source distribution.

4.2.2. Structured Result Retrieval. The backend for our data warehouse was implemented as a set of relational tables in an Oracle RDBMS. As such, the structured query modules were required to convert the query posed into an equivalent SQL statement. This process is aided by the synonym data contained in the metadata manager. More detailed concept matching is performed on the query, and relevant attribute names are retrieved. These attribute names are used as the basis for the construction of the SQL query. In the future, we intend to provide increased support for natural language queries, and we will use prior work on converting natural language to SQL [30, 3, 8]. Once the appropriate SQL statement is constructed, the mediator's structured query modules use the Java Database Connectivity (JDBC) API to query the database and retrieve result sets. Once the result sets are retrieved, they are stored in Result objects as described above, and returned to the mediator via the acceptor object.

4.2.3. Unified Result Ranking and Caching. When the acceptor receives result objects from the various sources it compiles them into a result packet which will be passed to

```
ALGORITHM Metasearch (Q, D, R)
Begin
   Query Q
   /* number of results to display per screen */
   DisplayThreshhold D
   /* total number of results to retrieve */
   ResultThreshhold R
   ResultSet S ← {}
   HistorySet H ← {}
   k ← 0
   Asynchronously submit Q to
   unstructured sources
   loop until k ≥ R
   Begin
      Retrieve result r from asynchronous
      queue
      if r ∉ H
      Begin
         k ← k + 1
         S ← S ⋃ {r}
         H ← H ⋃ {r}
      End
      if |S| ≥ D
      Begin
         Sort S by normalized rank
         Display S to user
         Wait for request of D results[a]
         S ← {}
      End
   End
End
```

[a] continue filling asynchronous queue during the wait

Figure 3: Metasearch results accumulation algorithm

the results manager for analysis. The results manager maintains a cache of available results which removes duplicates and sorts the results by their normalized ranks. Results returned from structured sources are taken to be of greater relevance than those returned from unstructured sources, and they are ordered first in the result packets.

4.3. User Interface

A Java servlet is used to generate HTML pages and present results to the user. A singleton tuple returned from the structured sources is displayed as an "answer", while a larger number of results emanating from structured sources are displayed in a table. Related links to unstructured source information on the Internet are also displayed.

5. Summary and Directions for Future Work

We have described an initial architecture for an intranet mediator. Our mediator is fundamentally different from an internet mediator because it performs schema integration using a data warehouse prior to the query, allowing us to focus on the problem of deciding which sources are relevant to a given query. Internet mediators must attempt schema integration at query execution time.

Structured sources need to be added to the existing framework, as this will allow more testing of individual modules and decision making components. Unstructured sources that deal with items such as imagery and video will also be integrated into the system.

An additional repository of metadata describing unstructured sources will be created. This repository will be used to determine which unstructured sources are appropriate for a given query. For example, a query about audio/video clips would be sent to a search engine which specializes in searching for documents related to multimedia. The new metadata could also be used to judge the relevance of unstructured vs. structured sources.

The process of adding synonyms and keywords to the metadata manager will be automated, and a rules database implemented to allow users to enter selection criteria for a particular source. Feedback from the user in such a tool would aid the development of machine learning algorithms based on manual exclusion or selection of structured sources for user queries.

Finally, the intranet mediator presents a new framework within which relevance feedback can be provided to the user — relevance feedback items from structured sources could potentially have much higher relevance than those obtained from unstructured sources.

6. Acknowledgments

We wish to thank Evridiki Efstathiou, Won Kim, and Neelu Sharma, for their help in implementing the components of the intranet mediator engine.

References

[1] Bartell, B. T., G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. *Proceedings of the Seventeenth Annual ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, 1994.

[2] Baru, C., A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, V. Chu. XML-Based Information Mediation with MIX. In *Proceedings of the Special Interest Group on Management of Data (SIGMOD '98)*, 1998, pp. 597-599.

[3] Cercone, N., P. McFetridge, F. Popowich, D. Fass, C. Groeneboer, G. Hall. The SystemX Natural Language Interface: Design, Implementation and Evaluation. Simon Fraser University. CSS-IS Technical Report: 93-03.

[4] Chan, S.G. and F. Tobagi. Scalable services for video-on-demand. *Technical Report, CSL-98-773*, Computer Systems Laboratory, Stanford University, December 1998.

[5] Chan, S.G., and F Tobagi. Providing distributed on-demand video services using multicasting and local caching. In *Proceedings of IEEE Multimedia Applications*, Services, and Technologies, (Vancouver, Canada), 7 June 1999.

[6] Cluet, S., C. Delobel, J. Simeon and K. Smaga. Your Mediators Need Data Conversion. In *Proceedings of the Spe-*

cial Interest Group on Management of Data (SIGMOD '98), pp. 177-188, 1998.

[7] Chang, C., H. Garcia-Molina. Mind Your Vocabulary. Query Mapping Across Heterogeneous Information Sources. In *Proceedings of the Special Interest Group on Management of Data (SIGMOD '99)*.

[8] The Cooperative Database Project. http://www.cobase.cs.ucla.edu/index.html.

[9] Damerau, F. J. Problems and Some Solutions in Customization of Natural Language Database Front Ends. Accepted for publication in *ACM Transactions on Office Information Systems, Vol. 3, No. 2, April 1985, Pages 165-184.*

[10] Dreilinger, D. and A. E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*. 15(3):195-222, 1997.

[11] Feldman, Susan. NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval. Accepted for publication in *Online, Inc., 1999*.

[12] Florescu, D., A. Levy and A. Mendelzon. Database Techniques for the World Wide Web: A Survey. *SIGMOD Record*, 27 (3), pp. 59-74, 1998.

[13] Gauch, S., G. Wang and M. Gomez. ProFusion*: Intelligent fusion from multiple, distributed search engines. *The Journal of Universal Computer Science*, 2 (9): 637-649, 1996.

[14] Glover, E. J., S. Lawrence, W. Birmingham, C. Lee Giles. Architecture of a Metasearch Engine that Supports User Information Needs. In *Proceedings of the Eighth International Conference on Information Knowledge Management (CIKM99)*, ACM, 1999.

[15] Gravano, L. and H. Garcia-Molina. Generalizing G1OSS to Vector-Space databases and Broker Hierarchies. *In Proceedings of VLDB Conference*, 1995.

[16] Gravano, L., C. K. Chang, H. Garcia-Molina and A. Paepcke. STARTS: Stanford proposal for Internet metasearching. In *Proceedings of the 1997 ACM SIGMOD Conference*, 1997.

[17] Grossman, D. A., M. C. McCabe, C. J. Staton, B. Bailey, O. Frieder and D. Roberts. Application-Level Performance Testing: A case study of a large finance application. *IEEE Software*., September, 1996.

[18] Grossman, D. A., O. Frieder, D. O. Holmes and D. C. Roberts. Integrating Structured Data and Text: A Relational Approach. *Journal of the American Society of Information Science*, 48 (2), February 1997.

[19] Grossman, D. and O. Frieder. Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, ISBN 0-7923-8271-4, 1998.

[20] Hawking, D. and P. Thistlewaite, Methods for information server selection. *ACM Transactions on Information Systems (TOIS98)*, April 1998.

[21] Inmon, W., Building the Data Warehouse, John Wiley and Sons, 1993.

[22] Kimball, R., The Data Warehouse Toolkit, John Wiley and Sons, 1996.

[23] Lawrence, S. and C. L. Giles. Text and Image Metasearch on the Web. *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 99)*, CSREA Press, pp. 829-835, 1999. Copyright CSREA Press.

[24] Lawrence, S. and C. L. Giles, Inquirus, the NECI meta search engine, in: H. Ashman and P. Thistlewaite (Eds.), In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, Computer Networks and ISDN Systems, pp. 95-195, 1998.

[25] Lawrence S., C. Lee Giles. Context and Page Analysis for Improved Web Search. In *IEEE Internet Computing*, IEEE, 1998.

[26] Lee, J. H. Analysis of multiple evidence combination. In *Proceedings of the Twentieth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR '97)*, 1997.

[27] Liddy, E. D., Enhanced Text Retrieval Using Natural Language Processing. *Bulletin of the American Society for Information Science*. Vol. 24, No. 4, 1998.

[28] Lundquist, C., O. Frieder, D. Holmes, and D. Grossman. A Parallel Relational Database Management System Approach to Relevance Feedback in Information Retrieval. *Journal of the American Society of Information Science*, 50 (5), April 1999.

[29] McCabe, M. C., A. Chowdhury, D. Grossman, O. Frieder. A Unified Environment for Fusion of Information Retrieval Approaches. In *Proceedings of the 1999 Conference on Information and Knowledge Management (CIKM99)*, ACM, 1999.

[30] Meng, F. and W. Chu, Database Query Formation from Natural Language using Semantic Modeling, UCLA CS-Technical Report: 990003.

[31] Shaw, J. A. and E. A. Fox. Combination of Multiple Searches. *The Third Text Retrieval Conference (TREC 3)*, 1995. National Institute of Standards and Technology Special Publication.

[32] Thompson, P. A., Combination of Expert Opinion Approach to Probabilistic Information Retrieval, Part I: The Conceptual Model. *Information Processing and Management*, Vol. 26 (3), 1990.

[33] Thompson, C. A., R. J. Mooney, L. R. Tang. Learning to Parse Natural Language Database Queries into Logical Form. Accepted for publication in *1997 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*.

[34] Vogt, C. and G. Cottrell. Predicting the Performance of Linearly Combined IR Systems. *Proceedings of the* 21^{st} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[35] Yang, C., O'Shaughnessy, D. Development of the INRS ATIS System. Accepted for publication in *ACM Intelligent User Interfaces, 1993*.