A Hybrid Approach for Ontology Integration

Ahmed Alasoud
Concordia University
1455 De Maisonneuve Blvd. West
Montreal
Canada
ahmed_a@cse.concordia.ca

Volker Haarslev
Concordia University
1455 De Maisonneuve Blvd. West
Montreal
Canada
haarslev@cse.concordia.ca

Nematollaah Shiri
Concordia University
1455 De Maisonneuve Blvd. West
Montreal
Canada
shiri@cse.concordia.ca

Abstract

The high proliferation of information on the World Wide Web (WWW) has made it necessary to make this huge information not only available to humans, but also to machines. Ontologies are widely being used to enrich the semantics of web, and corresponding technology developed to exploit them. Certainly, extracting information from various ontologies created independently is an important challenge for answering queries from web. In this paper, we propose a framework for ontology integration which is a hybrid of materialized (data warehouse) and virtual views. We have developed a prototype of the proposed framework. While much work is still ahead, our experiments so far indicate that the ideas used in this work are promising which may result in significant theoretical as well as practical contributions.

1. Introduction

The rapid increase in the number of multiple information sources requires efficient and flexible frameworks for integration of these sources. Such frameworks should provide a way for extracting, transforming, and loading data from these sources, and be represented to the user in some appropriate way. There are two major approaches for integration of information: (1) the data warehouse (DW) or materialized approach and (2) virtual approach (mediator based). In DW approach, huge amount of historic data is stored in the DW. In the virtual approach, on the other hand, the data is not materialized, but rather is globally manipulated using views. Each of these approaches is suitable in some kinds of applications.

DW is a powerful tool for decision support and querying the data because it explicitly stores information from heterogeneous sources locally. However, some external data, such as new product announcements from opponents and currency exchange rates, may be needed to support the accuracy of the business decisions. We should not neglect the importance of such data to avoid the problems of incompleteness, inexact, or sometimes wrong results. Warehousing huge and frequently changed information is a big challenge for the following reasons. Firstly, since the data in the DW is loaded in snapshots and the DW is a huge information repository. Secondly, as the data sources change frequently, the maintenance becomes a complicated and costly issue.

The other approach of the information integration is to provide a virtual integrated view. In this approach, the actual data resides in the sources, and queries against the integrated 'virtual' view will be decomposed into sub queries and posed to the sources. This approach is preferred over the materialized approach DW when the information sources change very often. On the other hand, the DW approach may be desired in case of a quick query answer is required and the information sources change rarely.

In this paper, we consider a third approach which is a hybrid between fully materialized and fully virtual approaches. This inherits the advantages of both. The primary contributions of this paper are twofold:

1. We implemented a general framework for supporting ontology integration. By general we mean the integrated view could capture the main concepts of modeling used in databases and software engineering, such as Entity Relationship, UML class diagrams, etc.
2. By efficient we mean faster response time. This is achieved by materializing the most frequent queried data, and making the business decision more precise by collecting some extra information from other sources as in the virtual approach for answering queries. We deal with

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

the situation where the information sources are expressed over ontologies, and we need an integrated view in order to extract information from these sources.

To reduce the query response time, some frequently queried data should be materialized. However, some queries need not be answered totally from the materialized views. In order to support more complex analyses of the materialized data, some queries must be answered directly from the sources.

To the best of our knowledge, no existing work has used a hybrid approach as a framework to support ontology integration. We present our ongoing research on integration of source ontologies expressed as integrated view which are partially materialized.

2. Related Work

The Squirrel project [7] uses a hybrid approach to develop a general and flexible mediator framework, but concentrates only on relational or object-oriented data sources. A framework for warehousing the web contents has also been discussed in [14], which uses a hybrid approach in order to integrate DW data with the "required" web-based information. It considers ontologies to express domain knowledge related to web sources and logical model of the data warehouse. Moreover, an ontology engine is being deployed as an intermediate layer by defining the mapping rules between web data and attributes of DW in the ontologies to aid the DW structure and repairing requirements. In [14], some web data are selected for materialization. However, some queries might not be answered using only materialized data in the DW. The main difference between our framework and the one in [14] is that they consider data sources as web data only, whereas in our work, data sources could be ontologies, web data, or any form of structured or semi-structured data sources. In addition, we also deal with situations where the OWL-DL Ontology Web Language with correspondence to description logics (DL) is used as formalism for the integrated view (IV) and the source ontologies. In [4], a framework for ontology integration has been introduced based on the fully virtual approach. They construct the integrated 'virtual' view based on the mapping approaches between the local ontologies and the global ontology. The meaning of the mapping here is that a concept of one ontology corresponds to a query over other ontologies. Then, a query will be posed on the global 'virtual' ontology and based on the mapping between the concepts in the global and the local onotologies, the query will be unfolded and the answers retrieved from the sources. The proposal in [2] extends the ontology integration framework by using the Description Logic DL-Lite for expressing the global schema, and using the Local as View mapping approach between the local 'databases' and the global ontology schema. There are some remarkable differences between our work and theirs. First, we consider the integrated view being partially materialized to improve query answering time, and to take advantages of fully materialized and fully virtual approaches. Second, in our work, data sources considered could also be expressed in ontologies.

The rest of the paper is organized as follows. In section 3 we motivate our approach using some examples. The framework is then introduced in section 4. The mappings between schema of the integrated view and the source ontologies are introduced in section 5. Section 6 reports our implementation experience. Concluding remarks and discussion of future work are provided in section 7.

3. Motivating Examples

Let us consider the following examples. Consider ontology of an enterprise "A", which offers different types of electronic products. For simplicity, we consider only two products, which are PCs and laptops. Fig. 1 introduces these items as an ontology at enterprise "A". It includes the concept COMPUTER which represents the desktop and laptop computers. Other concepts such as MONITOR, PROCESSOR, and PRICE, etc. in this ontology represent the specifications of the computers. Also, there is an external data source for the same products, which is enterprise "B" shown in Fig. 2. The following three examples will illustrate how the integrated view could be supported as;

- Fully materialized: Where the posted query needed to be answered only from materialized views,
- Fully virtual: Where the answers to these queries are not materialized. This could be because of queries being infrequent or the data being changed frequently, and hence the query should be answered from the source ontologies.
- A hybrid of both: Where the answers to queries are retrieved from materialized as well as virtual views.

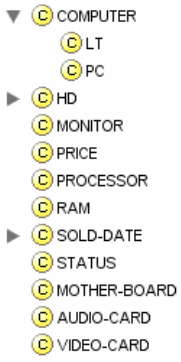

Figure1. Ontology of enterprise A

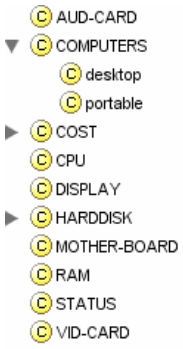

Figure2. Ontology of enterprise B

For simplicity, we assume the ontology of an enterprise "A" is part of a larger ontology restricted to only one branch of this enterprise. Moreover, the analysts can create an integrated view for this enterprise in which the client can query all branches efficiently. In addition, they want to get external information about their competitor companies, such as enterprise "B". This integrated view is shown in Fig. 3.

Furthermore, designing an integrated view depends on many factors such as: (1) which data are very frequently queried and rarely changing, and (2) which are not repeatedly queried and/or changing very often. In some cases, however, analysts may not materialize frequently queried data if it is very frequently changing. For example, in bank applications in generals, the account balance of the customer is not only very frequently queried but also changes frequently. Thus, materializing such data may not contain up-to-date status of the clients. Instead, it would be more efficient if the client uses the virtual view in which he/she can get an up-to-date statement from the sources. All in all, these factors should be taken into consideration during the design phase.

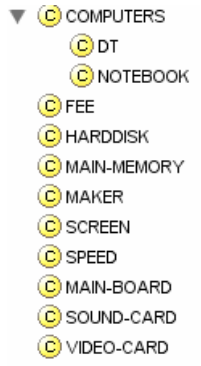

Figure 3. The integrated view

Example 1.

In this example, we illustrate the case where the end user query is answered from the materialized view only. First of all, deciding which data of the enterprise "A" should be materialized is decided by the analysts based on the aforementioned reasons. To avoid the complexity, we will assume the following:

- The most frequently queried and infrequently changed data are the prices of the computers,
- The price of the computer is affected by some important specifications such as the processor type and speed, the size of hard disk, the size and kind of the main memory, and the monitor type and size.
- Other specifications such as types of the main board, sound card, mouse, video card, etc., will not affect much the price and are occasionally visited by the user.

According to these assumptions, the concepts of FEE, SPEED, HARDDISK, MAIN-MEMORY, and SCREEN should be fully materialized, and COMPUTERS concept could be partially materialized. In partially materialized concepts, only one model of each category of computers with the same specifications of speed, hard disk size, main memory, and screen will be materialized.

Whenever the user asks for the price of a computer with particular specifications for the maker "A", the query answer will be retrieved from the materialized data only.

Example 2.

Let us consider the situation where important information about products by other enterprises is required in order to make right business decisions. We therefore need to query some selected ontologies of which may not be in materialized form, such as the enterprise "B" ontology in our running example. This process of answering such queries is done fully virtual by querying the sources through the integrated view. In other words, the integrated view is used as an intermediate layer to decompose the queries into sub queries and get the answers to each sub query from the relevant sources.

Example 3.

In this example, we consider that the integrated view is partially materialized for frequently accessed data and leave some to be answered from the virtual sources. In our example, we could leave the main board type, case material, sound card specifications, and video card type, or any other types of data sources which is occasionally visited by the users, so the infrequent accesses to data will not significantly affect the over all performance of the framework. Another case is where enterprise "A" needs to compare the prices of its computers with these produced by enterprise "B". Such queries would be answered partially from the materialized prices for "A" and the virtual prices for "B".

4. An Architecture to Support Ontology Integration

As shown in Fig. 4, our framework is based on the Integrated View (IV) and a set of wrappers. In this framework, IV follows a Local as View (LAV) approach [2] to represent the mapping between the concepts in the source ontologies and the integrated view. We will discuss the mapping in next section. The Transformation Processor (TP) transforms the data from the data source model to the materialized data model. In our implementation, we consider the materialized data being represented as an ontology model. During the maintenance of the materialized view, according to the update occurring in the data sources, the Incremental Maintenance Processor (IMP) will determine which data in the materialized view are going to be updated. After the IMP receives the integrated data from IV, it will compare to decide which parts need to be updated.

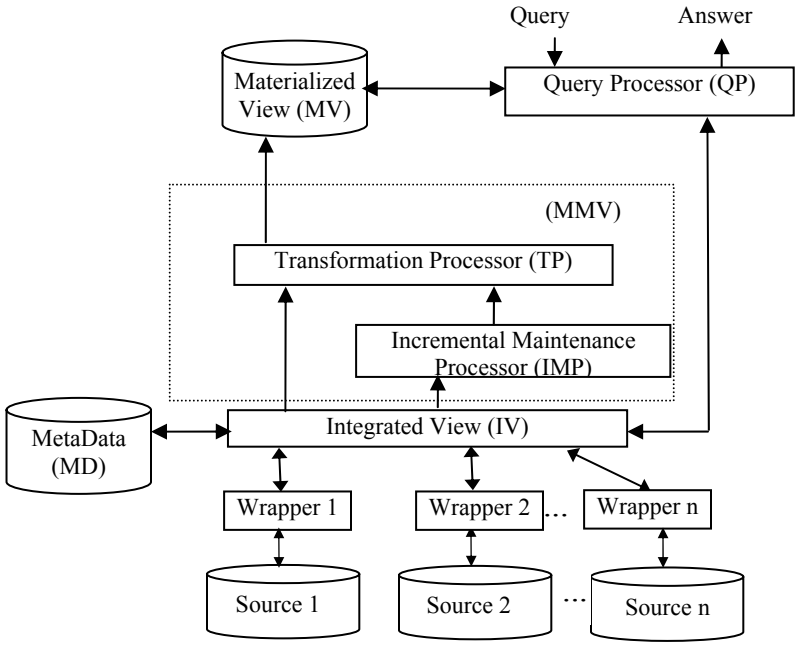

Figure 4. The architecture of the framework

The two modules TP and IMP in the dashed box in the figure form the Maintenance module for the Materialized View (MMV). The task of the Query Processor QP in our architecture is to determine if the query could be answered from the materialized view (MV), virtual view, or both. If the query needs actual data i.e., data from the sources, then the query should be decomposed and rewritten based on the mapping of the concepts between the integrated view and the data sources. As soon as the QP gets the query answer back from the data sources and the materialized view, QP "merges" it and returns it to the user. The MetaData (MD) module is a repository for the mapping terms for the concepts, roles, and individuals used by both the IV and the data sources.

5. Mapping Between Integrated View (IV) and Local Schemas

We can think of the integrated schema as divided into two main parts. The first part is the materialized part, and is responsible for updating the materialized view whenever updates are required. So, this part is a mirror of the materialized view. The second part of the integrated schema is the virtual one, and is responsible for providing the extra needed information from the sources which are not materialized, or partially materialized. In other words, it collects answers to queries that could not be answered from the materialized view only. For doing so, different approaches has been studied in [8] as a model for the integrated schema, such as, Global as View (GAV), Local as View (LAV), and Global-Local as View (GLAV). In GAV approach, each concept of the global schema is mapped to a query over the data sources. In other words, when the user poses his/her query over the integrated schema, the data corresponds to a concept in the integrated schema, which can actually be answered from the data sources through a specific query. The query processing in GAV is easy, since it just (unfold) each concept in the integrated schema in the user query with the associated query over the sources, but this approach does not help much when the sources change or grow very often, since these factors affect the mappings and require restricting the integrated schema. In contrast, the LAV approach defines the mapping in the other way around; each concept in the data sources is defined in terms of a query over the integrated schema. This makes query processing more difficult, since now the system does not know explicitly how to reformulate the concepts in the integrated view expressed in the user query in terms of the data sources. On the other hand, changes or incremental growth in the sources will not lead to reconstruction of the integrated schema, and need only to modify the mappings. The GLAV approach is the combination of the GAV and LAV approaches, where there are unrestricted mappings, in which the restrictions on the direction of the association between integrated and local schema are overcome. Query answering in this approach is largely unexplored, mainly because it combines the difficulties of the other ones.

Regardless of its difficulties, many researchers show [2, 3, 4, 8] that the LAV approach better supports a dynamic environment, where data sources can be added to the system or removed without restructuring the integrated schema. Therefore, recent research work on data integration has followed this approach.

6. Implementation

This section describes general ideas about the implementation of a prototype of our framework. This includes modules for mapping, query processing, query rewriting, and query answering.

We have followed the LAV approach for representing the mapping between the virtual Integrated View and the data sources. This mapping is expressed in nRQL (New RACER Query Language) [15] as follows:

$$\text{Enterprise} - \text{A}(x) \subseteq (\text{retrieve}\,(?x)\ (?x\,|\ \text{Enterp - A}\ ||\ \text{has - maker}\ |))$$

$$\text{Enterprise} - \text{B}(x) \subseteq (\text{retrieve}\,(?x)\ (?x\,|\ \text{Enterp - B}\ ||\ \text{has - maker}\ |))$$

In the above mapping, we are representing the concepts of data sources over the integrated view using the semantic query. This means, we map the products of enterprises A and B to queries over the integrated view by defining the concept MAKER and adding the relation has-maker between the two concepts COMPUTERS and MAKER in the integrated view. This mapping gives the key for the query processor module QP to determine where the query should be sent.

Moreover, the Semantic Web deals with diverse types of query answering with access to information represented in different formats. To allow complex queries to the integrated view, mapping of these concepts to the integrated view is essential. We are using nRQL as the query language. We use RACER [16] together with nRQL to support flexible construction of queries. Also, we use OWL-DL Ontology Web Language with correspondence to description logics (DL) as formalism for the integrated view (IV) and source ontologies. We use Protégé 3.0 as editor and knowledge representation.

The following are three query examples written in nRQL syntax. Each example shows a different scenario for query processing, rewriting, and answering. Besides, the examples will show different cases for answering queries, such as answering it using materialized view only, virtual view only, or both.

The following query scenario shows a case where answer to the query is generated from the materialized view only.

Consider the query "list the price of all laptops made by enterprise A, with the specifications: hard disk = 40 GB, screen type LCD with size = 21", main memory type is SD with size =560 KB, and processor type is Pentium-4 with speed = 2 GB".

Considering the integrated view provided, this query would be formulated in nRQL as follows:

```
(retrieve (?y) (and (?y ?c |price-of|)
                    (?c  |pnt4-2| |has-speed|)
                    (?c  |hard-disk-40| |hard-disk|)
                    (?c |lcd-21| |has-screen|)
                    (?c |sd-560| |has-ram|)
                    (?c |notebook|)
                    (?c |A| |has-maker|)
                )
)
```

Based on the mappings, the QP can easily conclude that the query should be sent and evaluated at enterprise A. Then, considering our previous assumptions that the concepts FEE, SPEED, HARDDISK, MAIN-MEMORY, and SCREEN are materialized, QP will compute the answer using the materialized view only. After that, the answer will be sent to the end user.

The second query is similar to the first, except that here we are asking the answers to be retrieved from the fully virtual view. In this case, the query answers should be retrieved from data sources, i.e., ontology of enterprise B "(?c |B| |has-maker|)", or the non-materialized concepts in the ontology for the enterprise A, such as main board type, case material, sound card specifications, and video card type, etc. For such queries, QP will rewrite them according to virtual sources (views), and send them to IV module. After the answers are obtained, QP merges the results and sends them back to the user.

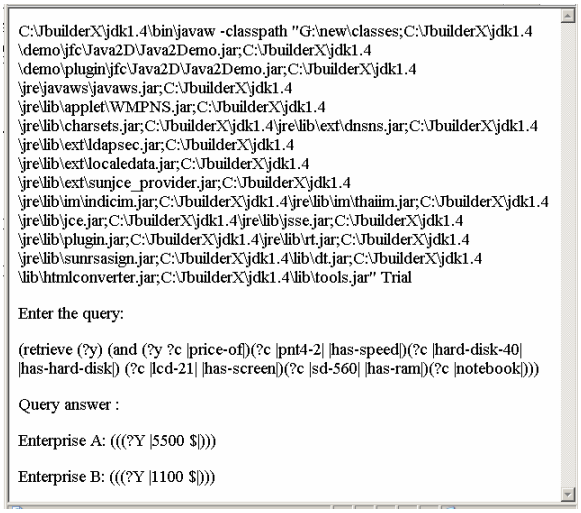

Figure 5. nRQL query with its answer

The third query scenario shown in Fig. 5 is a combination of the previous two scenarios. Here the query is the same as the first one except that we want to compare the prices of the laptop made by A and B, with the specifications mentioned above. In formulation of this query, we might not wish to specify the has-maker relation. Therefore, QP will decompose the original query, rewrite it, and sends the query to both materialized and virtual views for evaluation. After receiving the results from both types of sources, QP merges them and sends the final answer to the user.

7. Conclusion and Future Work

In this paper we presented a novel framework for supporting of integrating information contents from the data sources as ontologies for Decision Support System (DSS). Further, we also discussed the formalism used for integrating ontologies, and how we used the hybrid approach (a combination of a virtual and materialize approaches) in order to take the advantages of both for decreasing the query response time.

At present, we are improving our framework and doing further research on the following aspects;

- Finding formalism for the integrated view in an ontology integration environment which gives the best expressive power and complexity of reasoning.
- We are looking into some interesting issues for example, how to select an appropriate set of data from different source ontologies represented in different formalisms for materialization.
- The work is underway to develop an interface using which the users can browse the schema of the integrated view. This allows more general users to interact with the integrated view and query it.
- We would like to carry out more experiments with the prototype developed using several ontologies created independently by different groups of students in a graduate course in Semantic Web at Concordia University.

References

[1] A. Cal`ı, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In Proc. 20th Int'l Conf. on Conceptual Modeling (ER 2001), pages 270–284, 2001.

[2] Diego Calvanese and Giuseppe De Giacomo. Data Integration: A Logic-Based Perspective. AI Magazine, 26(1), 2005.

[3] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical reasoning for rich DLs. In Proc. Description Logic Workshop (DL 2004). CEUR Electronic Workshop Proceedings, http://ceur- ws.org/Vol-104/, 2004.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, editors, The Emerging Semantic Web — Selected Papers from the First Semantic Web Working Symposium, pages 201–214. IOS Press, 2002.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In Proc. 6th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'98), pages 2–13, 1998.

[6] F. Baader and W. Nutt. Basic description logics. In Baader et al. [2], chapter 2, pages 43–95.

[7] Hull, R., Zhou, G.: A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches, In Proc. ACM SIGMOD '96, Montreal, Canada, 1996.

[8] J. D. Ullman. Information integration using logical views. In Proc. of the 6th Int. Conf. on Database Theory (ICDT'97), volume 1186 of Lecture Notes in Computer Science, pages 19–40. Springer, 1997.

[9] M. Lenzerini. Data integration: A theoretical perspective. In Proc. 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), pages 233–246, 2002.

[10] V. Haarslev and R. M¨oller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel H. Levesque, editor, International Joint Conference on Artificial Intelligence (IJCAI'2001), August 4-10, 2001, Seattle,Washington,USA . Morgan-Kaufmann, August 2001.

[11] V. Haarslev and R. M¨oller. Description logic systems. In Baader et al. [2], chapter 8, pages 282–305.

[12] V. Haarslev and R. M¨oller. RACER system description. In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001), volume 2083 of Lecture Notes in Artificial Intelligence, pages 701–705. Springer, 2001.

[13] W.H. Inmon. Building the Data Warehouse, 2nd Ed.

[14]Yan Zhu: A Framework for Warehousing the Web Contents. ICSC 1999: 83-92.

[15] nRQL user Guide, http://www.cs.concordia.ca/%7Ehaarslev/racer/racer-queries.pdf

[16] RACER, http://www.cs.concordia.ca/~haarslev/racer/.

[17] The Protégé Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/.