Object-oriented Data Warehouse for Information Fusion from Heterogeneous Distributed Data and Knowledge Sources

L.L. Miller, Vasant Honavar, Johnny Wong and Sree Nilakanta
Iowa State University
Ames, IA, 50011

Abstract— **Tools for accessing, transforming, organizing, and assimilating data and knowledge from multiple heterogeneous, distributed data and knowledge sources in a form that is suitable for further analysis constitute some of the key enabling technologies for translating recent advances in automated data acquisition, digital storage, computers and communications into fundamental advances in organizational decision support, data analysis, and related applications. The design, implementation, and evaluation of a data warehouse using object-oriented views and mediator agents for bridging the mismatch between data sources and clients has been a major focus of research as part of the Distributed Knowledge Networks Project at the Iowa State University Artificial Intelligence Research Laboratory over the past two years. This paper describes the design, implementation, and population of a Data Warehouse component of Distributed Knowledge Networks.**

I. Introduction

Today's computing environments are characterized by increasing heterogeneity, distribution, and cooperation. These characteristics increase the complexity of advanced applications that require integration and interoperation of heterogeneous and distributed hardware and software systems. On the other hand, the technology of modern computing industry has established the infrastructure to host complex applications that were implausible just a few years ago.

The topic of integrating data from multiple heterogeneous data sources has been extensively studied. However, the changing technology of the modern computing industry demands better and more realistic solutions. In this paper, we describe a object-oriented data warehouse system based on an object-oriented view system.

The view system is an extension of the Zeus View System(ZVS) [17], [18]. The current view system has been extended to handle semistructured and unstructured data (e.g. text and images) in addition to structured data (e.g. relational databases). In addition changes have been made to enhance operations like composition of views.

The roots of our work are in the multidatabase environment. However, we see the notion of a data warehouse as being a natural outgrowth of a multidatabase system. Many of the underlying issues are the same. For example, the semantic mapping required to integrate data sources is a fundamental aspect of both environments. In our case, the notion of a warehouse became important as we looked into analyzing complex data (e.g., biological data) from heterogeneous data sources. When we looked at the data warehouse literature for object oriented warehouses it was apparent that most of the existing work is aimed at either relational or dimensional models and would not be rich enough for our applications. Since we were interested in testing a wide variety of analysis tools on the same collection of data, we needed an environment that was capable of creating and storing different "views" of the data set chosen for the tests. It was clear that we would benefit from having the tools and the data in the same package.

The main contribution of this paper is the design and implementation of the object-oriented data warehouse. The warehouse has been designed to allow both the search and the necessary transformations to be incorporated into the views that define the warehouse classes.

II. Related Work

While data warehouses are becoming common place in industrial circles, research literature on data warehouses has only recently appeared. The work has focused on using materialized views as a means of constructing the warehouse. The Stanford data warehouse project points out that object oriented views would be a useful way of accessing heterogeneous data sources [5], [16], but their publications primarily make use of the relational view. The majority of the published work has been focused on the self maintenance of materialized views [2], [4], [6], [12], [19] and concurrency control [8], [9].

Fundamental research in multidatabases includes modeling issues, mapping methodologies and semantic issues. The modeling issues center around the creation of a common data model for modeling integration and interoperation of mul-

0-7803-9914-5/98/$10.00 © 1998 IEEE

tiple database systems. Sheth and Kalinichenko [13] looked at information modeling issues in environments that range from multidatabase manipulations to multisystem applications. Su et al. [14] proposed an object-oriented rule-based approach to providing a common data model that maps a local database schema to a global schema.

The notion of view objects was introduced in [15]. A sophisticated view mechanism is introduced in [1] to help restructure data or integrate databases. Object-oriented views are used to integrate heterogeneous information systems in [3], [7].

III. An Extensible View System

We use a view mechanism for integration of data from heterogeneous structured and unstructured data sources. In this section we briefly discuss the view mechanism on which the warehouse population algorithms are based.

A. Overview

The view has abstract object semantics. It does not have a stored state nor does it provide any global resources. It is only used to describe available services and resources such that the information can be recalled by the view system to coordinate the sharing of the services and resources. We use an extensible object model (EOM) as an intermediate model to facilitate the representation of different data model constructs and application objects. We also extend the EOM to provide an integration model. The views are described by a view definition language (VDL) which provides portable syntax and semantics for view construction and sharing. The EOM and VDL create a uniform interface to hide the heterogeneity and distribution of participating systems. The view is created to define: how the local services and resources are exported, how the global services and resources are imported, or how the services and resources are constructed and presented. The view mechanism supports the creation and management of views, the processing of views and the coordination of the computations resulting from views.

B. An Extensible Object Model

Our view system is based on an extensible object model (EOM) to provide a common framework for modeling real-world and conceptual entities in a variety of application domains. We adopt the terminology used in [10] to present the EOM. The EOM is composed of a core object model and a set of components. A component is a compatible extension of the core object model. A profile is a set of selected components. *A profile along with the core object model provide the modeling facility for a particular application domain.* The core object model has been influenced by standardization efforts [10], and research [11].

C. Mapping Methodology

A mapping methodology is required to ensure that data brought in from the data sources to the warehouse/multidatabase match the types needed for the warehouse/multidatabase applications. Supporting such a mapping is complicated by the continued growth of the number of new data types that must be considered. The mapping methodology must be able to handle data from files, various database management systems, document retrieval systems, as well as, other forms of semi or unstructured data. In addition it must be able to easily adapt to new data types. Our approach is to make use of the view object concept as the basis of the mapping methodology. We define the view object type as being an extension of the object model (EOM). Our use of views is an extension of our work on the Zeus View Mechanism given in [17]. The view type consists of public attributes, private attributes, a derivation section and a method section. The public attributes, private attributes, and method sections have the normal meanings. The derivations section is used to generate the public and private attributes of each object instance created through a view.

Three levels of views have proved useful in our system:

Local Interface - The individual data sources are expected to have local control. The local interface is a view type object that is used by the local data administrator to provide a mechanism to make his/her data accessible to the warehouse environment.

Base views - The base view is used to do the necessary conversion of the data. The derivation section of a view is used to provide the mechanism for both data selection and conversion. The base view can either be a user defined view or a system view. The only real difference is that the system base views are provided for expected conversions. Two examples of system base views are views that convert object instances to a set of relations and a a view that turns a set of textual data into a set of term vectors.

Global views - Whereas base views only make use of local interfaces, global views can be composed of either local interfaces, base views or other global views. The main reason for global views is

to provide a higher degree of flexibility in creating different "views" of the data.

The composition algorithm allows global views to be based on any number of local interfaces, base views, or global views. For runtime simplicity, our current view system prototype maintains a table of precomposed views. It would be easy to incorporate the composition at runtime, but we have chosen to do the composition at the time the view is integrated into the system.

IV. Data Warehouse

The object-oriented warehouse in our approach consists of a set of classes where each class is a materialized object view. The resulting object classes are either virtual or imaginary classes using the terminology in [1]. The views used to generate the materialized view can be at any level, i.e. local interfaces, base views or global views. The choice of view level depends on the amount of preprocessing and/or preselection that is being done to create the class for the warehouse.

Virtual classes are added to the warehouse by making direct use of the local interface. We denote these materialized views as virtual classes, since the local administrator has introduced the class to the system as a set of objects. Our usage of the virtual class is somewhat looser than that used by Abiteboul and Bonner [1], since the actual data source may or may not be object oriented. For example, the actual data source may have been a set of text documents that the local data administrator mapped to the object data format.

Imaginary classes make use of either base views or global views. Global views are used to generate a materialized view, whenever it is necessary to use composition of views to generate the imaginary class. Base views are used to provide any desired mapping (preprocessing) or preselection of the data from the data sources.

There are several advantages to this approach. The main advantage of our approach over existing warehouse systems is the fact that the object format created by our view system has far richer semantics than current systems. We are able to make use of our views to provide different looks to the same set of data. For example, with the view system it is possible to store a class that contains textual documents, while another class consists of weighted term vectors for the same set of documents. This means that we can use different tools to analyze the same data set. The warehouse provides an environment for making clean reliable data available for analysis.

Our warehouse prototype has been implemented using the POET object oriented database. POET gives us a platform on which the data for the materialized views can be stored and the data analysis tools can be integrated as part of the same system environment.

Figure 1 shows a block diagram of the current prototype. A user that wishes to create a materialized view for the warehouse first adds the view (object class) definition to the POET database (warehouse) and then executes an application using the view via the view system. The view definition can either be created using POET commands or by making use of our view definition language and view generation software. Either approach results in a class being added to the POET database used to support the data warehouse.

Once the view defintion is in place (as a class), the view can be materialized (populated) in two ways. Either the user can make use of the multidatabase system based on the InterLanguage Unification (ILU) software from Xerox or a mobile agent environment can be activated. In the latter case the mobile agents use the view definition to retrieve data from the heterogeneous data sources and insert it into the POET class used to represent the materialized view.

Once the materialized view is available in the warehouse, the user can apply any of the data analysis tools to the class (materialized view) that are appropriate for the structure of the object instances.

Data analysis tools can be added to our current prototype in two ways. First, the individual in charge of the source code (e.g., a vendor) can add tools by including them as methods to the warehouse class. Users of the data warehouse can add tools to the warehouse by using the "add tool" option in the data warehouse menu. In this case the user is able to attach a tool to the list of executables that are available in the menu. The user does not see a difference between the tools entered by the two different approaches.

V. Conclusion

An object oriented data warehouse based on the view mechanism has been presented. The warehouse is being integrated into our distributed knowledge network environment to provide the capacity for data analysis of the data farom a wide variety of heterogeneous data sources.

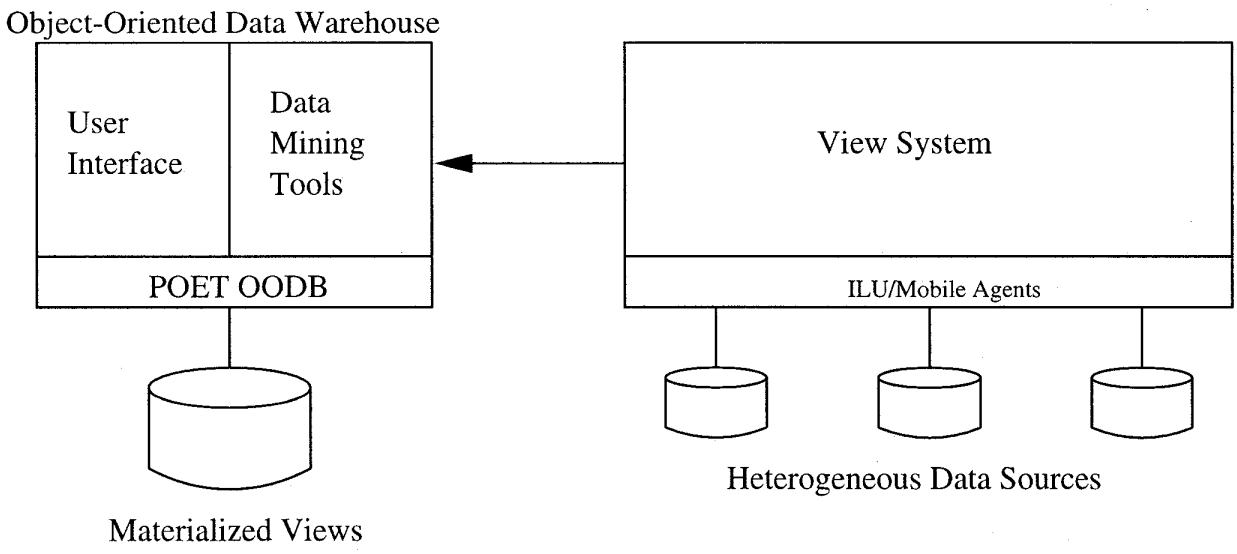

Fig. 1. **The block design of the current prototype.**

References

[1] Abiteboul, S., and A. Bonner (1991). Objects and views. *ACM SIGMOD*, 238–247.

[2] Baekgaard, L. and N. Roussopoulos (1997). Efficient refreshment of data warehouse views. Technical Report, Department of Computer Science, University of Maryland, URL: http://www.cs.umd.edu/TRs/authors/Nick_Roussopoulos-no-abs.html.

[3] Czejdo, B., and M. C. Taylor (1992). Integration of information systems using an object-oriented approach. *The Computer Journal*, 35, 5, 501–513.

[4] Gupta, H. (1997). Selection of views to materialize in a data warehouse. *Proceedings of the International Conference on Database Theory*, Athens, Greece.

[5] Hammer, J., H. Garcia-Molina, J. Widom, W. Labio, Y. Zhuge (1995). The Stanford data warehousing project. In *IEEE Data Engineering Bulletin.*

[6] Huyn, N. (1996). Efficient view self-maintenance *Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications*, Montreal, Canada.

[7] Kaul, M., K. Drosten and E. J. Neuhold (1990). ViewSystem: integrating heterogeneous information bases by object-oriented views. *IEEE Data Engineering*, 2–10.

[8] Kawaguchi, A., D. Lieuwen, I.S. Mumick, D. Quass, K.A. Ross. (1997). Concurrency control theory for deferred materialized views. *Proceedings of the 1997 ICDT Conference.*

[9] Kawaguchi, A., D. Lieuwen, I. Mumick, D. Quass, K. Ross (1997). Concurrency control theory for deferred materialized views. *Proceedings of the International Conference on Database Theory*, Athens, Greece.

[10] *OMG Architecture Guide Chapter 4: The OMG Object Model* (1992). Object Management Group.

[11] Peters, R. J., M. T. Qzsu and D. Szafron (1992). TIGUKAT: An object model for query and view support in object database systems, TR 92-14, U. of Alberta.

[12] Quass, D., A. Gupta, I. S. Mumick, and J. Widom (1996). Making views self-maintainable for data warehousing. *Proceedings of the Conference on Parallel and Distributed Information Systems*, Miami Beach, FL.

[13] Sheth, A. and L. Kalinichenko (1992). Information modeling in multidatabase systems: beyond data modeling. *Int'l Conference in Knowledge Management*, 8–16.

[14] Su, S., S. Fang, and H. Lam (1993). An object-oriented rule-based approach to data model and schema integration. *Technical Report, U. of Florida,.*

[15] Wiederhold, G. (1986). Views, objects, and databases. *IEEE Computer*, 37–44.

[16] Wiener, J.L. , H. Gupta, W.J. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom (1996). A system prototype for warehouse view maintenance. *Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications*, Montreal, Canada, 26-33.

[17] Yen, Cheng-Huang, L. Miller, and S. Pakzad (1994). The design and implementation of the *Zeus View System. 27th Hawaii International Conference on Systems and Sciences*, 206-215.

[18] Yen, C.H. and L.L. Miller (1995). An extensible view system for multidatabase integration and interoperation. *Integrated Computer-Aided Engineering*, 2. 2. 97-123.

[19] Zhuge, Y. , H. Garcia-Molina, J. Hammer, and J. Widom (1995). View Maintenance in a Warehousing Environment. *Proceedings of the ACM SIGMOD Conference*, San Jose, California.