CHANGE MANAGEMENT IN DATA INTEGRATION SYSTEMS

Rahee Ghurbhurn, Philippe Beaune
Génie Industriel et Informatique, Ecole des Mines de St Etienne
158 cours Fauriel 42000 St Etienne, France

Hugues Solignac
STMicroelectronics, zi Peynier Rousset 13790 Rousset

Keywords: Data Integration, Change management, Multiagent Systems, Ontology.

Abstract: In this paper, we present a flexible architecture allowing applications and functional users to access heterogeneous distributed data sources. Our proposition is based on a multi-agent architecture and a domain knowledge model. The objective of such an architecture is to introduce some flexibility in the information systems architecture. This flexibility can be in terms of the ease to add or remove existing/new applications but also the ease to retrieve knowledge without having to know the underlying data sources structures. We propose to model the domain knowledge with the help of one or several ontologies and to use a multi-agent architecture maintain such a representation and to perform data retrieval tasks. The proposed architecture acts as a single point of entry to existing data sources. We therefore hide the heterogeneity allowing users and applications to retrieve data without being hindered by changes in these data sources.

1 INTRODUCTION

Due to economic globalisation, organizations now evolve in a highly competitive environment where having the right information at the right time is the key of success. In the 90s, companies have invested massively in information systems so as to be able to store and analyse data through the use of data warehouses. The result was one or more information systems for each department. To derive higher value added from the stored data, departments started to share the information found in their systems. Organization wide data sharing was rationalized through the use of Enterprise Application Integration solutions. From a point to point architecture we obtained an organized global architecture composed of heterogeneous applications communicating through message buses. The problem is that in an economy where the key word is flexibility the information system rationalization created a rigid architecture.

Indeed, the tight coupling of applications and data sources makes it difficult to adapt them to technological, organizational or economical changes. Application or data source evolution are difficult to realize as it impact the whole system. Moreover, users having access to several data sources owned by different departments may find it difficult to identify the appropriate data for their analysis. Indeed, domain concepts may be found in several data sources but their semantic may differ from a domain to another. In some cases it may be interesting to use data collected by another department to make an analysis more complete.

Our proposal consists in modelling different user domain knowledge and associating them to their corresponding data found in heterogeneous distributed data sources. The user domain knowledge is expressed in the form of one or several ontologies (Gruber, 1993) that can be shared across the organization or between several organizations. To take into consideration the changing aspect of an organization's environment and the distributed nature of data sources, the knowledge representation is manipulated and maintained by a Multi-agent system (MAS)(Nwana, 1996; Sycara et al., 1996).

In this paper, we are going to consider our motivations and existing approaches in section 2. Section 3 deals with our proposition in terms of knowledge representation and in terms of MAS

Ghurbhurn R., Beaune P. and Solignac H. (2007).
CHANGE MANAGEMENT IN DATA INTEGRATION SYSTEMS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 268-273
DOI: 10.5220/0002376102680273

architecture. Lastly section 4 presents a brief conclusion and some perspectives.

2 MOTIVATIONS AND EXISTING APPROACHES

Our work is being carried out in an industrial environment composed of several independent heterogeneous and distributed data sources. These data sources are accessed by several applications for production analysis, production planning and follow up, decisional and reporting activities. Some applications are old legacy applications with direct access to the data sources other communicate through message buses and other by means of flat files. These applications and data sources belong to different services and are therefore independent. This implies that the data sources can be modified at any time, by a service, without any notification to the others.

Our objective is to design and implement a mechanism that allow functional user to explore the existing knowledge found in different data sources and retrieve the associated data without having to know the underlying data sources' structures. This data exploration and retrieval should be possible even if changes have been operated on the data sources. Moreover, we also consider the fact that if the data sources are changed the applications accessing them are also impacted. We propose to isolate these applications from this type of changes by creating a single point of entry common to all data sources.

The problem of integrating distributed data sources has been addressed by several research communities namely the database, artificial intelligence and the knowledge representation community. Several solutions have been proposed.

Federated databases (Busse, 1999) consist in defining a canonical schema to map the data sources logical schemas. Federated data sources create a tightly coupled global information system and it is therefore difficult to make each components of the system evolve without impacting the global structure.

Multi-database query languages allows data sources to be loosely integrated as there is no global schema, but the main problem is that the semantic heterogeneity is not dealt with by the system but by the users. Indeed the user has to know the physical and logical characteristics of each data sources to be able to take into account the data heterogeneity. There are several implementations of multi-database query languages namely MSQL (Litwin et al., 1989), SchemaSQL (Lakshmanan et al., 2001) and FRAQL (Sattler and Saake, 2000). These languages are all SQL extensions.

Data integration through mediation (Wiederhold, 1992; Levy, 1999) consists in defining for each data source a local schema describing the content of these data sources. These local schemas are then mapped to a global schema that describes the relationship between the content of the local schemas. The mapping between the local and the global schema can be specified by either the global-as-view or the local-as-view approach. This approach allows loosely coupled data integration in the sense that the data sources remain independent but it creates a tight coupling between the local and global schemas. Indeed when ever a data source changes the mappings have to be updated and this may prove to be a tedious task. Some projects based on the mediator approach are SIMS (Arens et al., 1993), TSIMMIS (Chawathe et al., 1994), Infosleuth(Bayardo et al., 1997). Recently two new approaches global-local as view and both-as-view have been proposed by (Friedman, 1999) and (McBrien and Poulovassilis, 2003) respectively. These approaches aim at combining the advantages of global-as-view and local-as-view. That is combining the ease of transforming queries and the ease of adding new data sources.

Ontology driven integration consists in defining a local ontology for each data sources and link them either by defining a global ontology containing concepts that subsumes the local concepts or by defining inter-ontology mappings. Ontology driven integration presents the same inconvenient as the mediation approach. There is a tight coupling between the local schemas and the global schema. Moreover, it may be difficult to reconcile different ontologies to establish inter-ontology mappings. The main projects base on ontology driven integration are OBSERVER (Mena et al., 2000) and KRAFT (Peerce et al., 2000).

All the above mentioned approaches use database views to enable data integration via a global schema. According to us these solutions are difficult to deploy in an environment where the data sources are autonomous. By autonomous we mean that each company or service administers its own data sources and can therefore modify their structure without any notification. Moreover, database view administration may be a complex task if the modelled domain evolves rapidly. Indeed changes in the domain may require the deletion of attributes or

relations in the data sources invalidating the views used for data integration (Bellahsene, 2002, Amy et al, 2002).

Another point is that the above mentioned approaches do not provide any backup mechanism to allow for query answering if one data source is offline. Indeed to the best of our knowledge we did not find any mechanism that redirect query to backup data sources if one them fails.

Lastly, these solutions have been built for human use and do not take into account the fact that some planning or decisional applications need to access these data. In fact when ever a data source is modified, it impacts not only the users but also the applications using the data source. The applications have to be updated either by updating the queries the application uses or by updating the source code for legacy applications.

3 PROPOSITION

Our proposition consists in expressing the content of the data sources in terms of users' domain knowledge. This knowledge representation takes the form of one or several ontologies, each representing a different domain, expressed in OWL DL. The ontology does not only express the relation between concepts that are present in the data sources but also their localization. That allow us to reuse a same data source in different domain ontologies. These relationships are used to replace views and are used to reformulated user queries. In the following we are going to explain our knowledge modelling and why do we need a MAS to maintain it.

3.1 Knowledge Modelling

As we said previously, our knowledge representation concern domain knowledge. The main reason is that modelling domain knowledge offers greater stability as compared to data source logical models. Moreover, logical schema presents several mechanisms used for data storage and retrieval optimization and therefore not embodying any domain knowledge that may interest functional users.

Thus we are going to describe concepts like "integrated circuit", "equipment", "production engineer" and for each of them, their associated properties. For example, an equipment can have as properties, a serial number, a localization, a date of purchase, a production load and so on. After having identified the concepts and their properties, their semantic is specified by defining the appropriate relationships between each concept.

In this form, the ontology allows users to browse through the knowledge contained in the data sources but it does not allow them to retrieve any associated data. We therefore have to add some description about their localization. For the time being, we consider only databases but our model can easily be extended to the description of any other data source like XML files or object databases.

To have a complete description of a concept, we consider that properties can be expressed as a function of one or several data source attributes found in the same or different data sources. For example, a production engineer and an equipment maintenance engineer will be interested in different aspects of an equipment. But both of them would be interested in indicators coming from another domain like the production load (production domain) or the maintenance actions performed on an equipment (maintenance domain). Indeed the maintenance engineer may use the production load indicator to plan and determine future maintenance actions. The production engineer will perhaps want to know the recent maintenance action performed in case high production loss. The problem is that this information may not be found in the same data source. For these reasons we associate to a property one or more data sources. Our proposal is not based on views, the queries over the data sources are formed at runtime based on the selected concepts and properties. The figure below is a UML representation of the basic concepts of our ontology.

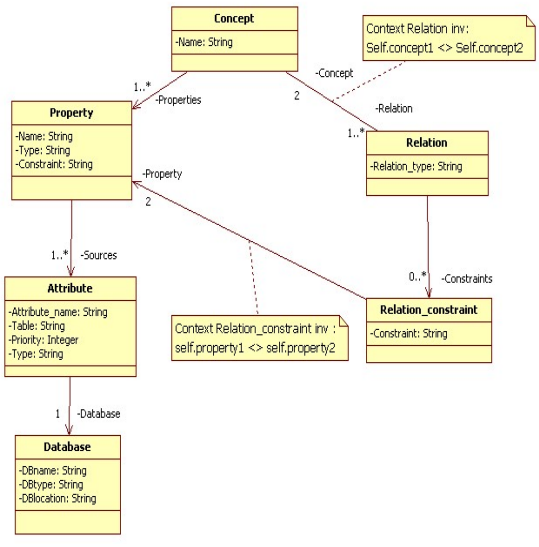

Figure 1: Ontology model.

Figure 1. represents the UML model corresponding to the structure of our ontology. A concept is composed of properties and relations. Each property can have one or more attributes referencing the corresponding data found in the data sources. Relations define the type of relationship existing between two concepts.

Having linked the knowledge representation to the data sources, users can now browse through the knowledge contained in the data sources and ask the system to retrieve the desired data. But before being able to present the retrieve data to the users, we have to firstly convert the queries expressed in terms of domain knowledge into queries understandable by the data sources. Domain queries are composed of concepts, properties and constraints on selected concepts and properties. The query transformation mechanism is explained in section 3.2.2. Secondly, we have to check for the data structure consistency and harmonize them. Another issue is the coherence of the ontology with the data sources. As we said earlier, we assume that the data sources are independent and can be modified at any time. In this case, we must be able to detect the data source changes, evaluate the impact on the ontology, make the necessary updates and deal with queries that have been issued just before the data source change.

In the next section, we propose to address all these issues through a MAS. The latter will have as main objective to convert queries expressed in terms of domain knowledge into queries understandable by data sources, retrieve the corresponding data, and harmonize them before presenting them to the users. The MAS must also be able to detect any change in the database structure, evaluate its impact on the ontology and take corrective measures to keep the representation coherent and deal with issued queries.

3.2 Multi-agent Architecture

In the past years, MAS have been used in a variety of applications like distributed computations, simulation, computer games, information gathering, data mining, production and supply chain planning. The use of MAS paradigm is motivated by the various interesting properties it presents.

Software agents are autonomous, this property allows them to act on behalf of the user. They can contain some level of intelligence, which is either hard coded through fixed rules or acquired using learning engines. This so-called intelligence allow them to interact and adapt to changes in their environment.

Agents are able to communicate with users, other systems and other agents as required. Communication is an important characteristic, in the sense that it allows several agents, having no complete information, to cooperate and execute complex tasks or achieve complex objectives.

The motivations behind the use of MAS as a potential solution to our distributed and heterogeneous data retrieval problem are based on organisational and architectural aspects. Organisational aspects include the facts that MAS can easily auto-adapt to changing environment and re-synchronise the elements composing the system. Architectural aspects concern the robustness and flexibility in terms of administration or operation. That is MAS can be partly administered or updated without having to switch the whole system offline.

The MAS architecture that we propose has three basic functions; ontology construction assistance, information retrieval and ontology coherence maintenance.

3.2.1 MAS and Ontology Construction Assistance

The data sources being distributed, we decided to affect a resource agent to each data source. After having identified the concepts of a user domain and their corresponding attributes in the data sources, the system administrator can ask the agents to retrieve the corresponding meta-data to instantiate our “database” and “sources” concept in our ontology. The association of the domain concept to the sources is done manually by the administrator. The administrator must also define the links that exists between different data sources. These links will be used when converting ontological queries into SQL queries and when fusing the result sets corresponding to the queries. The instantiation and mapping is done through the use of an ontology agent.

3.2.2 MAS and Information Retrieval

The information retrieval mechanism, that we propose, *does not* use views for query reformulation and checks for the availability of the data sources before retrieving data. If the data source is not available the mechanism is able to determine the accessibility of the data sources participating in the query and retranslate the user query based on backup data sources.

Once the ontology built, it allows functional users to formulate queries in terms of domain knowledge to retrieve data without having to know the data

sources' underlying structures. The main problem is that these domain knowledge queries are not understood by the data sources. The ontology agent translates domain knowledge queries into queries understandable by data sources by using the information found in the ontology. The steps in our query conversion mechanism are:

- From the selected concepts and properties find the corresponding data source attributes
- From the relevant attributes find the relevant data sources and data sources' tables
- Compose the queries for each data source
- From the relevant sources find the applicable links between them, from the ontology and automatically include the necessary attributes and tables in the appropriate queries (defined in the previous step)
- Query the data sources

As we previously said, we stored information regarding the data sources for each data source attribute added to the ontology. From this information we can find out to which table of which data source the attribute belongs. This information allows us to fill the SELECT and FROM part of an SQL query. Concerning the WHERE part and more precisely the join clauses, it can be built from a directed graph.

This graph is built by using the meta-information regarding the attributes forming the SELECT part of the query. The nodes of the graph represent the data source tables forming the WHERE clause of the query. The join clauses are determined by finding a path from one node to another. If several paths are found we choose the one that minimises the join cost.

An ontological query can be broken into several SQL queries concerning several data sources. The links, between the data sources, defined by the administrator are used to automatically include additional attributes and tables to the queries obtained in the previous step.

The queries are addressed to their respective data sources. On receiving the queries each resource agent checks their consistency, that is if the elements composing the queries are found in the data source. If an inconsistency is found, the resource agent notifies the ontology agent who asks the other resource agent to cancel the data source querying and restarts the query reformulation steps but this time using backup attributes found in the ontology.

If the data sources are successfully queried, the result is sent to the users via the task agent responsible of data harmonization. If the data source querying is unsuccessful then an alert is sent to the administrator and a notification sent to the user.

As with any data repository, query response time must be as short as possible. To reduce the query response time, we defined a conversion matrix composed of all the properties contained in our ontology and their corresponding attributes found in the data sources. Their equivalence is specified by a "1" at the intersection of the row and a column respectively. Therefore whenever a query translation has to be performed, no inference is made on the ontology; instead the conversion matrix is used, speeding up the translation.

As compared to approaches based on views the query transformation is done at runtime and therefore only existing attributes and online data sources are selected. Moreover the burden of views administration and maintenance is considerably reduced. Indeed, the only views created are views generally created by each local data source administrator to speed up query answering. These views can therefore be administered locally without any impact on the global data integration system.

3.2.3 MAS and Ontology Coherence

As we mentioned in section 2, we consider that the applications and the data sources are independent and can therefore be modified at any time by the system administrator. This poses serious problems as regards to the coherence of our ontologies with the data sources. Indeed, if the system administrator removes some attributes, tables or even a data base without any notification the users will not be informed and will continue to send queries, containing the removed attributes, to the data sources.

We propose to use the resource agents to monitor data sources and automatically detect any change, operated by the administrators, in the data sources' structures. When a change occurs, a notification is sent to the ontology agent. Using the conversion matrix, the latter is able to identify the impacted concepts and evaluate the impact on the ontology. The impact is measured by determining the number of relations that the impacted concept has. After having determined the impact of the changes on the ontology, the ontology agent can either, in case of attributes deletion, prevent the users from selecting the properties of the concepts that have changed or inform the system administrator that new data have been added to the data sources. The administrator can then decide if the new data should be associated to a concept or not.

Moreover, our proposition must also deal with queries that have been issued by user unaware of the changes. In this case, the system tries to return a partial answer and an explanation for the partial result.

4 CONCLUSION

We presented in this paper our current work on a MAS for information systems semantic interoperability. The aims of such an architecture is, firstly, to be able to explore and share knowledge present in heterogeneous distributed data sources within and between organization. Secondly isolate users and applications from changes in the data sources while allowing them to retrieve any data from any data sources at any time. We have partly implemented our MAS architecture and developed a small ontology describing the knowledge contained in three data sources. Some future perspectives would be to develop an database annotation mechanism that allows our agents to automatically add new data sources to the ontology

REFERENCES

Gruber, T.R, 1993. Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kluwer Academic Publishers. The Netherlands.

Nwana, H.S. 1996, Software Agents: An Overview. In *Knowledge Engineering Review*, *vol. 11, pages 205-244*

Sycara, K., Pannu, A., Williamson, M., Zeng, D. and Decker, K. 1996, Distributed Intelligent Agents. In *Intelligent Systems and their Applications, vol. 11, pages 36-46.* IEEE Expert.

Busse, S., Kutsche, R., Leser, U. and Weber, H., 1999, Federated Information Systems: Concepts, Terminology and Architectures, In *Technische Universitat Berlin., Technical Report* .

Litwin, W., Abdellatif, A., Zeroual, A., Nicolas, B., and Vigier, P. 1989, MSQL: a multidatabase language. In *Information sciences, vol. 49, pages 59-101.* Elsevier Science

Lakshmanan, L.V.S., Sadri, F. and Subramanian, S.N., 2001 SchemaSQL: An extension to SQL for multidatabase interoperability. In *ACM Transactions on Database Systems, vol. 26, pages 476-519.* ACM Press

Sattler, K., Conrad, S. and Saake, G. 2000. Adding Conflict Resolution Features to a Query Language for Database Federations. In *Proc. 3nd Int. Workshop on Engineering Federated Information Systems* Akadem. Verlagsgesellschaft

Wiederhold, G. 1992. Mediators in the Architecture of Future Information Systems. In *Computer, vol. 25, pages 38-49,* IEEE Computer Society Press

Levy, A. 1999. Combining Artificial Intelligence and Databases for Data Integration. In *Lecture Notes in Computer Science.* Springer.

Arens, Y., Chee, C.Y., Hsu, C. and Knoblock, C.A. 1993 Retrieving and Integrating Data from Multiple Information Sources. In *International Journal of Cooperative Information Systems.*

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D. and Widom, J., 1994. The TSIMMIS Project: Integration of heterogeneous information sources. In *Proceedings 10th Anniversary Meeting of the Information Processing Society of Japan*

Bayardo, R. J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D., 1997. InfoSleuth: agent-based semantic integration of information in open and dynamic environments. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data.*

Mena, E., Illarramendi, A., Kashyap V. and Sheth, A.P, 2000 OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. In *Distributed and Parallel Databases, vol. 8, pages 223-271.* Springer

Preece, A.D., Hui, K., Gray, W.A., Marti, P., Bench-Capon T.J.M., Jones, D.M., and Cui, Z. 1999 The KRAFT architecture for knowledge fusion and transformation. In *Expert Systems,* Springer, Berlin

Bellahsene, Z. 2002. Schema evolution in data warehouses. *Knowledge Information System vol. 4, pages 283-304*

Lee, A. J., Nica, A., and Rundensteiner, E. A. 2002. The EVE Approach: View Synchronization in Dynamic Distributed Environments. *IEEE Transactions on Knowledge and Data Engineering, vol 14, issue 5, pages 931-954*

Miller, L., Seaborne, A. and Reggiori, A., 2002 Three Implementations of SquishQL, a Simple RDF Query Language. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web.*

McBrien, P. and Poulovassilis, A., 2003, Data integration by bi-directional schema transformation rules. In *19th International Conference on Data Engineering.*

Friedman, M. and Levy A. and Millstein T. 1999, Navigational plans for data integration. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, American Association for Artificial Intelligence.*