A Semantic Information Integration Tool Suite

Jun Yuan Ali Bahrami Changzhou Wang Marie Murray Anne Hunt
Mathematics & Computing Technology
Boeing Phantom Works
P.O. Box 3707, M/C 7L-70
Seattle, Washington, 98124, U.S.A
{jun.yuan, ali.bahrami, changzhou.wang, marie.o.murray, anne.j.hunt}@boeing.com

ABSTRACT

We describe a prototype software tool suite for semantic information integration; it has the following features. First, it can import local metadata as well as a domain ontology. Imported metadata is stored persistently in an ontological format. Second, it provides a semantic query facility that allows users to retrieve information across multiple data sources using the domain ontology directly. Third, it has a GUI for users to define mappings between the local metadata and the domain ontology. Fourth, it incorporates a novel mechanism to improve system reliability by dynamically adapting query execution upon detecting various types of environmental changes. In addition, this tool suite is compatible with W3C Semantic Web specifications such as RDF and OWL. It also uses the query engine of Commercial EII products for low level query processing.

1. INTRODUCTION

Information integration has been recognized as a critical enabler for building large scale business applications such as enterprise-wide decision support systems. Industry has invested tremendous effort in integrating existing data systems across different programs and business units, customers, suppliers, government agencies, etc. With the recent explosion of the Internet and web-based resources, information integration and the related semantic interoperability are becoming an even greater concern.

To address such business needs, several Commercial-Off-the-Shelf (COTS) Enterprise Information Integration (EII) products have emerged [1] [2]. These products all provide a very important capability, namely, answering queries against a variety of information sources across the networks with a single unified query interface. This is a great help in resolving non-semantic heterogeneity including platform heterogeneity, network heterogeneity, data model heterogeneity, and more importantly, query capability heterogeneity. However, to our best knowledge, almost all COTS EII products in the market are limited in, or totally lack, the capabilities of semantic interoperability and dynamic adaptation upon changes [1] [3].

Specifically, the common global schema in a COTS EII product is usually represented using a low level (non-semantic) data model, e.g. a relational model or XML schema/DTD files. The semantics of information content are usually implicitly embedded in schemata, expressed by application logic, or worse, captured only in users' minds. System integrators have to spend a lot of energy, during the integration process, in discovering these hidden semantics. However, the discovered semantics cannot be explicitly represented by the low level data model in current COTS EII products. This intensive discovery work often needs to be repeated in the evolution of the information integration process.

With the low level data models, information can only be retrieved using a non-semantic query language such as SQL or XQuery. This requires users to have adequate database knowledge in understanding both the schema and the query language itself. Such a query language is perhaps preferable for application developers, but not user-friendly enough for normal business users, who usually don't have enough database knowledge but possess plenty of domain knowledge.

In addition, these COTS EII products usually fail to execute a query when any data element involved in the query becomes inaccessible. Users only get error messages without any information about other accessible data elements. In extreme cases, the failure of one local data system will cause the whole integrated system to cease functioning well. This leads to poor system reliability, especially in a complex environment where changes are expected to occur very often.

We present a prototype software tool suite that helps users build integrated information systems. The tool suite continues the work reported in [4] and is based on a framework proposed in [5]. Our approach takes full advantage of COTS EII products; for example, we use their optimized distributed query engines, and provide value-added features including dynamic querying adaptation and semantic interoperability. The tool suite keeps data in existing/legacy data systems and integrates information based upon its semantic equivalence. It uses a domain ontology to explicitly describe the semantics of global information content. To resolve semantic heterogeneity, the tool suite maps local models onto the domain ontology. It also leverages Semantic Web standards. For example, both the domain ontology and the mapping knowledge can be exported into the Resource Description Framework (RDF) or the OWL Web Ontology Language (OWL) documents.

2. TOOL SUITE ARCHITECTURE

Figure 1 describes the overall architecture of the tool suite. The metadata repository persistently stores the domain ontology. A

VLDB '06, September 12–15, 2006, Seoul, Korea.

COTS EII product connects to each local system and produces an EII common global schema to reflect its relationship with each local system. Its underlying query engine executes distributed queries at a low level. The COTS low level query engine supports basic data operations in a distributed environment, for example, distributed joins, selections and projections. With the help of a graphical mapping tool within the tool suite, a system integrator can create mappings between the ontology and local metadata (thus the EII common global schema) and save them into the metadata repository.

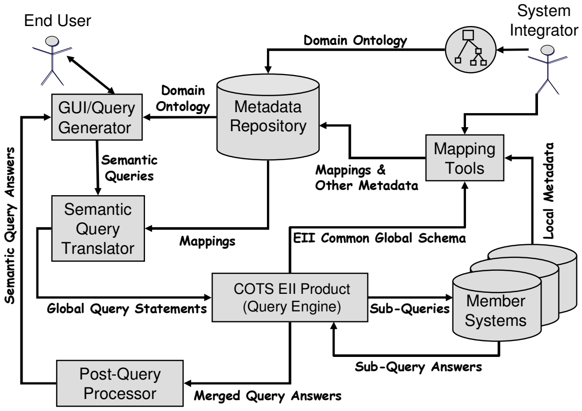


Figure 1. Semantic Information Integration Tool Suite Architecture

The mappings play an important role in translating a user-defined semantic query into one or multiple global query statements that are executable by the underlying EII query engine. The query generator allows users to define semantic queries. Upon receiving a semantic query, the semantic query translator converts the original query into one or more query statements, and then submits them to the underlying COTS EII query engine. The post-query processing module transforms the query answers from the COTS EII query engine into instantiations of the domain ontology, and presents them to the users via a GUI.

In this architecture, semantic information integration and interoperability tasks are performed in two different layers. The tasks of resolving non-semantic heterogeneity are pushed down into the COTS EII product. Various types of semantic heterogeneities are resolved using both the domain ontology and the mappings. This two-layered approach enables us to take full advantage of many distributed query optimization strategies provided by COTS EII products.

3. SEMANTIC QUERY PARADIGM

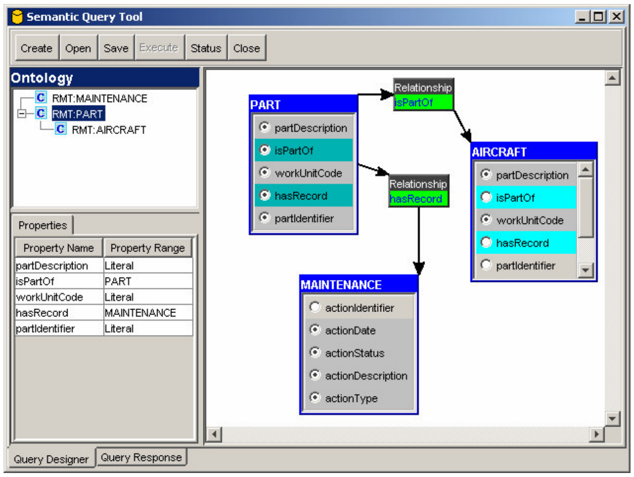
We propose a Navigational Semantic Query Paradigm (NSQP) for querying ontologies. Based on the two basic ontological elements, *concepts* and *relationships*, NSQP is neutral to any specific ontology representation standard. NSQP uses two basic constructs, concept node and relationship node. A concept node references exactly one concept in the domain ontology. A relationship node references exactly one relationship in the domain ontology. A relationship node represents a navigation step from one concept node to another under the referenced relationship. A relationship node may be connected to one or many concept nodes, and these concept nodes refer to the ending points of the navigation. A concept node may be connected to zero or many relationship nodes to indicate either the query targets or the navigations.

Concept nodes are very similar to variables in a conventional query language. They are placeholders for occurrences of concepts in the query. When a relationship/attribute value is requested, it must be bound to a specific occurrence of a concept. This enables us to only maintain the bindings for concept nodes and to easily calculate the requested relationship/attribute values on demand. As a result, merging query answers from multiple sub-queries becomes easy.

NSQP also supports query constraints and aggregations attached to either concept nodes or relationship nodes. Currently supported query constraints are in the form of a single comparison function or multiple simple comparisons connected with logical operators.

An important metric for a semantic information access methodology is the semantic understandability of query answers. NSQP represents query results as instantiations of the domain ontology, i.e., a collection of instances of concepts and relationships, and hence easily captures the semantics of query answers.

We have implemented a graphical semantic query tool for NSQP. When defining a query, a user typically picks a concept as a starting point, and simply drags-and-drops the concept into the designing panel. Then she/he selects appropriate relationships (and the tool will add corresponding concepts). In this way the user navigates the ontology from concept to concept under the relationships.



Property Name	Property Range
partDescription	Literal
isPartOf	PART
workUnitCode	Literal
hasRecord	MAINTENANCE
partIdentifier	Literal

Figure 2. Screenshot of Semantic Query Tool

Figure 2 presents a screenshot of the semantic query tool. To compose the displayed sample query, a user first selects concept *PART*, and drags-and-drops it into the query design panel. As a result, a concept node named *PART*, shows up. Then the user selects *hasRecord* (by clicking the radio button) in the *PART* concept node, a relationship node *hasRecord* and a concept node *MAINTENANCE* (references the range concept of relationship *hasRecord*) pop up, implying a navigation from the *PART* concept to the *MAINTENANCE* concept under the *hasRecord* relationship. Likewise, the user selects *isPartOf* to navigate from *PART* to *AIRCRAFT*. Note that there is a minor variation for the second navigation. The range concept of relationship *isPartOf* is replaced by *AIRCRAFT*, which is a sub-concept of *PART*.

4. METADATA & MAPPING

There are three major types of metadata in our information integration framework: domain ontology, local metadata, and mappings between them. The scope of the domain ontology encompasses the semantics of all the domain knowledge implicit in the sources. Local metadata describes the logical or physical schemata for each individual local system. Mappings encapsulate the correlations and bridge the conceptual gap between the domain ontology and different local schematic elements.

Identifying, developing, and managing mappings are critical to the operation of an integrated information system. However, little work has been done to create a generic formal methodology for this problem. Our mapping strategy addresses this challenge from the perspective of translating NSQP queries into low level queries and transforming retrieved low level query answers into instantiations of the domain ontology.

Pieces of data from different sources need to be integrated mainly because they do share common semantics. The common semantics here does not only mean 'absolute equity', but also refers to 'approximate equivalence'. In the second case, data transformations usually involve some conversion methods. Our mapping strategy correlates multiple local schematic elements, for example tables and columns, by means of mapping them onto the same ontological concepts or relationships.

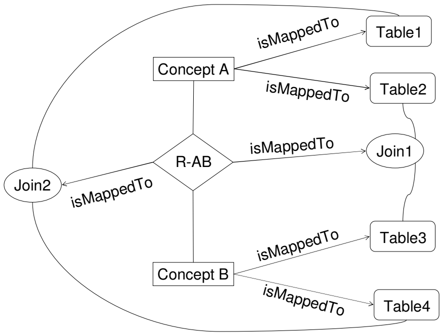


Figure 3. A Simple Illustration of Mapping Mechanism

Figure 3 illustrates our mapping mechanism. We omit the detailed mapping structure due to space limitation. In Figure 3, *ConceptA* is mapped to *Table1* and *Table2*, while *ConceptB* is mapped to *Table3* and *Table4*. The Relationship *R-AB* is mapped to two join operations, i.e., *Join1* and *Join2* respectively. *Join1* is between *Table2* and *Table3,* while *Join2* is between *Table1* and *Table4*.

The semantic navigation operation, one of the major semantic query operations, is usually denoted by a relationship node in NSQP. It can thus be evaluated by performing a relational join operation. The mapping does not pre-define a fixed type of join operation for any semantic navigation. Instead, an appropriate join type will be chosen during runtime. NSQP supports two types of navigation: optional navigation and mandatory navigation. An optional navigation from *ConceptA* to *ConceptB* via *R-AB* indicates the user's intention to obtain instances of *ConceptA* even if there is no related instance of *ConceptB* under *R-AB*. A mandatory navigation means that users are interested in getting instances of *ConceptA* only if there is at least one related instance of *ConceptB* under *R-AB*. Obviously, an optional navigation is similar to a relational outer join operation, while a mandatory navigation is similar to that of a relational inner join operation.

In order to integrate information from multiple local systems, users often need to invest significant effort extracting implicit semantics from local metadata. This is often a manual process and can hardly be automated with guaranteed accuracy. To avoid repeating this costly process, the tool suite persistently stores all extracted semantics explicitly by using ontological representations for metadata.

In addition, query processing sometimes requires sophisticated mapping knowledge be derived from basic mapping facts. For example, if the domain ontology uses concept inheritance or OWL axioms like "*sameAs*" or "*equivalentClass*", sophisticated mappings are needed. They are often derived from basic mappings on the requested concept, its super-/sub-concepts, and its equivalent concepts. These derivations may be as simple as calculating transitive closure or as complex as evaluating an advanced logic program. This imposes a requirement of inference capability on mappings. With an ontological mapping representation, such derivations can be pre-defined by rules. An appropriate inference engine can fire these rules so as to expand the basic mapping facts into more sophisticated ones.

5. DYNAMIC ADAPTATION

The second major aspect/contribution of our tool suite addresses how to cope with changes related to information access that commonly happen in a fairly complex environment. For example, the accessibility and capability of each individual information source may change; new data systems may be added or existing data systems may be removed; the underlying data model of some information sources may be modified; and network connectivity and bandwidth may change significantly. These changes may prevent users from getting the requested information. Therefore, it is necessary to actively monitor changes in real time and to automate reactions upon detection of these changes. In other words, it is necessary to re-configure the system in real-time for information integration.

Different types of changes may have different semantic implications for existing mappings and hence the information integration tasks. Automatically deriving such implications obviously enables the automatic re-configuration. Unfortunately, a fully automatic solution may not be viable yet. Nevertheless, we can work towards this goal by automatically modifying the state of the pre-defined mappings to incorporate detected changes. These modifications include: removing invalid mapping components, adding new mapping components, decomposing complex mapping objects, and regrouping mapping structure.

Using this real-time automatic modification strategy, our approach improves system robustness. Traditionally, information is integrated in a purely static manner. Given a query request with the static information integration strategy, only one static query execution plan is generated in any circumstance. If the aforementioned changes occur after the static integration, the execution often fails totally. Our approach is to generate a dynamic query execution plan according to the current state of available mappings. This allows users to get partial answers in accordance with their tolerance to approximation or incompleteness in query results.

The tool suite uses a novel query rewriting technique to perform dynamic adaptation upon detecting changes. In particular, the query rewriting technique eliminates inaccessible elements from the query execution. For instance, if a mapped relational table is inaccessible, either the table is removed from the translated query statement, or the entire translated query statement is deleted from the query execution. As a result, instead of error messages, users are able to get partial information back. Users can also be informed about the incompleteness of the query answers. However, the semantic distance between the original and the rewritten queries is not calculated in a quantitative manner in the current version.

6. DEMONSTRATION

In this demo, we will show the major features of our semantic information integration tool suite, including two graphical software tools, namely, the metadata management tool and the semantic query tool. The tool-suite is developed purely in Java, as standalone Java applications. The Jena API [6] is used to manipulate RDF/OWL compatible ontologies.

The graphical metadata management tool allows users to import local metadata through either standard interfaces such as JDBC/ODBC or proprietary APIs for Product Data Management (PDM) systems. The imported metadata is then converted into an internal ontological representation, and can be exported as RDF/OWL documents. The types of supported data sources include relational databases, XML documents, Microsoft Excel spreadsheets, and table structured files like Comma Separated Value (CSV) files. Due to availability, we are unable to demonstrate any proprietary data system, for instance a PDM system in manufacture.

The metadata management tool takes domain ontologies in RDF/OWL format. We have also developed a utility tool to accommodate ontologies written in a First-Order-Logic based language, in order to show that our technology is independent of any specific ontology language.

Mappings between the domain ontology and local metadata can be entered through the GUI of the metadata management tool. The demo will show how local schematic elements can be mapped appropriately onto basic ontological constructs.

The semantic query tool presents a GUI for NSQP. The ontological representation of NSQP allows a user-defined query to be saved as an XML serialization of RDF/OWL document. Query answers are returned as RDF/OWL documents too. We have also implemented utilities to transform the query answers into two other different formats, i.e. a tabular format (similar to conventional tables) and a tree-like format (similar to an XML document).

The demonstrated technology has been successfully applied in building a pilot web-based aircraft heath management system [7]. The system integrates several existing relational databases from different programs inside the company, some key customers, and suppliers. The tool suite has also been adopted in a demonstration to show interoperability among many oceanographic observation data sets under a government contract. Our experience shows that the performance of this tool suite is comparable to that of many COTS EII products, mainly because we have pushed down most of the query evaluation into the COTS EII query engine. The transformation of query answers into an ontological format requires additional processing. However, the performance overhead is not significant according to our observation.

7. FUTURE WORK

Dynamic changes may occur in not only information producers but also information consumers. We plan to utilize process models and user workflows to model both the information consumers and the information producers, along with their capabilities. With these models, we can not only monitor but also predict changes in regards to the consumption of information and make adaptations accordingly.

8. ACKNOWLEDGMENTS

This work is supported by Boeing Phantom Works Enabling Technology fund. The authors acknowledge Chuck Klabunde for his tremendous support. We thank the members of Boeing Information Management and Collaborative Technology group, particularly Dave Jones and Mike Uschold, for their helpful suggestions. We also thank the anonymous reviewers for their great comments and suggestions on the draft paper.

9. REFERENCES

[1] Halevy, A., et al. *Enterprise Information Integration: Successes, Challenges, and Controversies.* In Proceedings of SIGMOD 2005, pages778-787.

[2] Macvittie, L. *Enterprise Information Integration Suites—Don't Fear the Data.* Network Computing Magazine, Issue on Sept. 16, 2004. http://www.nwc.com/showitem.jhtml?docid=1518f2

[3] Uschold, M. and Gruninger, M. *Ontology and semantics for seamless connectivity*. SIGMOD Record 33(3), 2004, pages 58-64.

[4] Barrett, T., et al. *Applying Semantic Web technology to the integration of corporate information*. International Journal of Web Engineering and Technology, Vol. 2, No. 2/3, 2005, pages 231-247

[5] Yuan, J. *Semantic-Based Dynamic Enterprise Information Integration.* In Database Modeling for Industrial Data Management Emerging Technologies and Applications. Chapter VI. Idea Group Publishing, 2006, pages 185-216.

[6] Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/

[7] Wilmering, T., Yuan, J., and VanRossum, D. *A metadata architecture for mediated integration of product usage data*. In proceedings of AUTOTESTCON 2003, IEEE Systems Readiness Technology Conference, 2003, pages 564-575.