December 2004

A Methodology for Implementing Ontology-Driven Information Systems Using Higher Order Semantic Database Models

Samuel Conn
Regis University

A Methodology for Implementing Ontology-Driven Information Systems Using Higher Order Semantic Database Models

Samuel S. Conn, Asst. Professor
Regis University School for Professional Studies
MSCIT Program
sconn@regis.edu

ABSTRACT

The eventuality of the Semantic Web offers multiple attractive features to the enterprise infrastructure in the area of improved Knowledge Management and information retrieval. Current relational database models adequately perform in the transactional processing of operational data, but do little toward the advancement of knowledge foundations for innovation. The movement from an information economy to a knowledge economy mandates that successful organizations must innovate at a faster rate than their competitors. To this end, Information Systems must extend technological infrastructures away from relational transaction systems toward higher order semantic models that can accommodate schemas based on Ontologies. This report details a methodology for mapping relational data systems to a higher order semantic model that can implement an ontology-driven information system. Additionally, this report provides conclusions about the application of ontology-driven information systems to the corporate environment.

Keywords

Information System Ontologies, Ontology-driven Information Systems, Methodology, Semantic Data Models, Implementing Ontologies

INTRODUCTION

Ontology-based Information Systems will provide much of the needed infrastructure for companies to succeed in the Knowledge Economy[1]. One of the critical success factors in the Knowledge Economy is the enterprise's ability to innovate at a faster rate than its competitors (HBR, 1998). Many organizations know that the ability to foster innovation is coupled to their ability to implement knowledge management systems[2]. The natural progression of the Information System value proposition of data to information to knowledge through OLAP[3] systems, BI[4] systems, and DSS[5] remain a principle focus for companies desiring to utilize vast amounts of data to competitive advantage (Cook, 1996). However, as we move from the Information Economy to the Knowledge Economy[6], we find distinctions of design and purpose between these knowledge management systems, and information systems that serve as a catalyst for innovation (HBR, 1998). Such systems are evolving in concept as Ontology-driven Information Systems, and will be able to work in concert with next generation Web concepts such as the Semantic Web (Goldfarb and Prescod, 2004).

For some period of time the "impedance mismatch" that occurs between back-end system relational design, and front-end system object-oriented design has caused frustration and a lack of progress in evolving to higher order semantic data models (Cook, 1996). Higher order semantic data models are able to more adequately reflect the enterprise business model, and thus are desirable because the challenge of Information Systems is provide infrastructure for operational-transactional systems,

[1] The Knowledge Economy was predicted and described early on by Peter Drucker.
[2] Innovating at a faster rate is generally considered to be a critical success factor in the Knowledge Economy.
[3] Online Analytic Processing
[4] Business Intelligence
[5] Decision Support Systems
[6] The transition that started around the year 2000.

data analysis systems, and future innovation management systems (Orbst, 2003)[7]. Since the release of the SQL3 standard in 1999, database vendors are implementing the standard to create object-relational database engines with XML support capability. This database technology creates the platform for implementing a methodology to transform relational schemas to object-relational schemas, and then convert query output to XML for the purpose of providing data transport and ontological query.

The modeling primitives used in the relational to object-relational transform offer a basis for association to specific Ontologies (Goldfarb and Prescod, 2004). Aggregation of class objects into a base ontology allows the full use of object-relational database class and subclass architecture, and user defined datatypes. Mapping object-relational schemas to Information System Ontologies then becomes an easier task[8], since Ontologies rely on class and sub-class structures. The real value of the ontology then becomes the ability to interconnect and interact with other Ontologies in order to achieve "forward" knowledge[9] capability for the system user (Braga, Mattoso, and Werner, 2001). Just as TCP/IP networks have provided a physical layer means for seamless connectivity among local area network segments, and local and wide area networks to the Internet[10], Ontologies offer a logical layer means of interconnect and discovery between the same networks using intelligent inferential agents that can interact with machine readable knowledge content (Pinto and Martins, 2002). The implementation of these ontology-driven Information Systems is still reliant of high functioning database management systems and a higher order semantic data model (Kashyap and Sheth, 1996).

METHODOLOGY AND DEVELOPMENT

The approach to migrating the relational schema toward the ontology is to begin with creating a common layer of terminology and taxonomy for the data (Maedche, Motik, and Stojanovic, 2003). This is accomplished by grouping common attributes and creating object types for the literals of the attributes. A business enterprise may want to utilize a portal to communicate with its customers, employees, and business partners. A typical implementation would be to use database role-based security models in the portal[11]. So security and transactions with these entities that have some relationship with the enterprise are recorded and stored in the database. As a member of each entity logs on to the portal, the database recognizes the member of the entity in a respective role and serves up the appropriate interface or application for that role (Cook, 1996). But there are no inter-relationships recorded or stored because the modeling primitives for the relational model are focused on the cardinality of the relationships between the entities and not the business relationships between the entities. A typical schema for this would be to have relational tables that store information about each of the entities. This might take the (schema) notational form of:

CUSTOMER (CUST_ID, FNAME, MI, LNAME, ADDRESS, CITY, STATE, ZIP, EMAIL, PHONE, URL)
EMPLOYEE (EMP_ID, ADDRESS, CITY, STATE, ZIP, PHONE, EMAIL,)
BUSINESS_PARTNER (PARTNER_ID, NAME, PARTNER_TYPE, TYPE_ID)
SUPPLIER (TYPE_ID, NAME, CONTACT, ADDRESS, CITY, STATE, ZIP, EMAIL, URL)
SHIPPER (TYPE_ID, NAME, CONTACT, ADDRESS, CITY, STATE, ZIP, EMAIL, URL)
CHANNEL_PARTNER (TYPE_ID, NAME, CONTACT, ADDRESS, CITY, STATE, ZIP, EMAIL, PHONE, FAX, URL)

Entity-Relationship Modeling (ER Modeling) would create a normalized logical representation of the physical storage structures that essentially stores the same information (attributes) about each of the entity types. So we start to see common areas for storage of literal data values within the schema. But the ER Model is focused on the cardinality of the relationships between the entities, and the key constraints that allow for query paths between the storage structures to accommodate business rules for data reporting. Moreover, the basic design is about translating data to information via SQL reporting, and ensuring that the schema supports the ACID[12] theory properties for a transactional system. The logical schema could appear as the instantiation of relations that mirror the business relationships of the enterprise. Figure 1 illustrates a likely composition of relations to support the instance values for business relationships in an enterprise transactional system. The

[7] Note that one goal of 1st, 2nd, and 3rd generation database models/systems has been to improve the semantics of the model and thus more accurately reflect and adequately support the business model.
[8] Really just a natural extension of the process that may require some reverse engineering to accommodate additional classes and sub-classes.
[9] As opposed to "backward" knowledge that only allows analysis of previous events, rather than the synthesis of knowledge required to achieve innovation.
[10] A result of consolidation to TCP/IP driven by the ubiquitous proliferation of the Internet.
[11] Considering that portals are now evolving from 2nd generation transaction portals to 3rd generation process portals.
[12] Atomicity, Consistency, Isolation, and Durability are considered to be required features of transactional systems.

first step in the methodology of mapping the relational model to an object-relational model is to decide what attributes can be represented as objects, where the objects are user defined types (UDTs) that represent a taxonomy for how information about an entity is stored. So in this example, the attribute values across the entities looks like a good candidate.

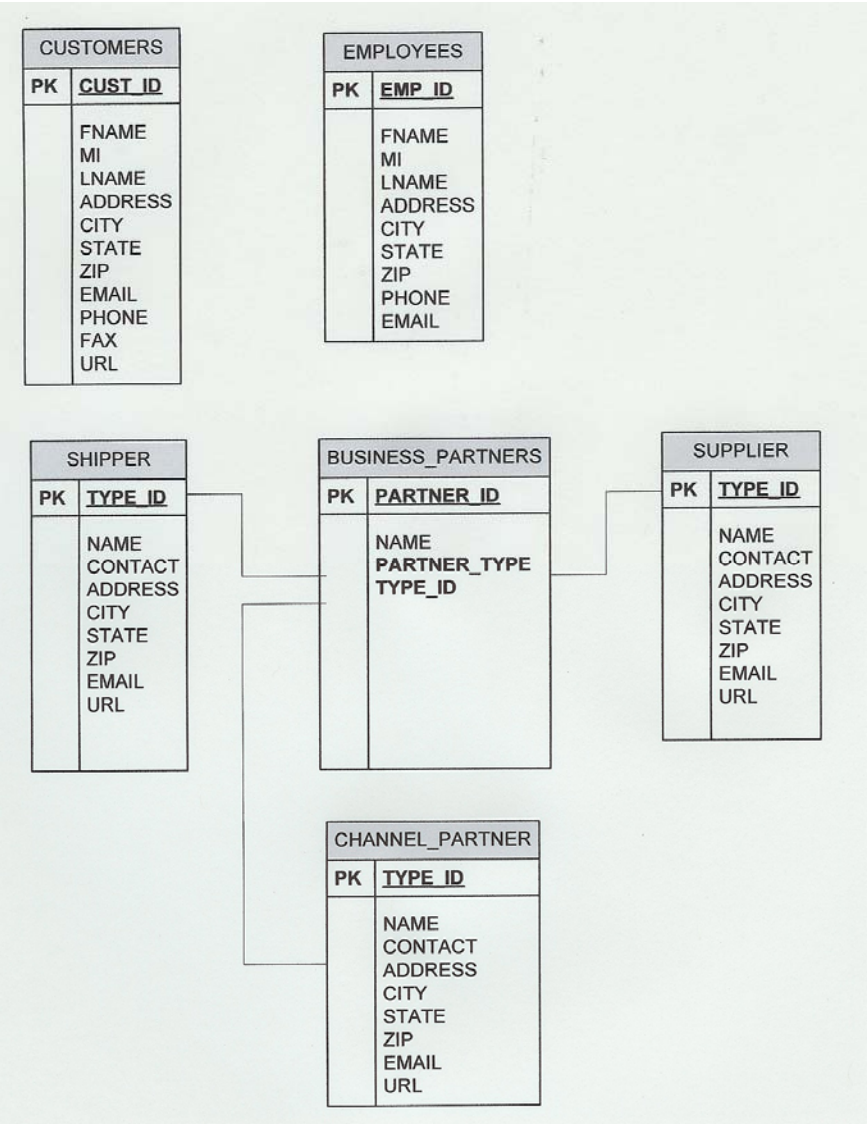

Figure 1: Relational Schema in Support of a Business Enterprise

The relations have a high degree of commonality among their attributes, but some have more attributes than others. For example, the EMPLOYEES relation does not have a URL attribute because most employees do not maintain their own web site, and if they did it would probably not be relevant in contacting the employee. There are also some attributes in some relations that would not make sense in other attributes. For example, the CUSTOMERS and EMPLOYEES relations have a MI (middle initial) attribute that would not make sense in a business partner's name. So the likely candidates for UDTs in this example are the attributes that compose the contact information for all of the entities, and the contact names for the CUSTOMERS and EMPLOYEES relations.

Relational to Object-Relational Schema Transforms

The first step in the transform is to create the two UDT objects for the aforementioned candidates. An object type (using VARRAYS[13] as the implementation methodology) for the contact information would include the following common attributes: ADDRESS, CITY, STATE, ZIP, EMAIL, and PHONE. And an object type (using VARRAYS as the implementation methodology) for the FNAME, MI, and LNAME attributes is created for the CUSTOMERS and EMPLOYEES relations.

The implementation of the object-relational construct of using UDTs in the schema would use the following SQL syntax[14]:

```
SQL*Plus: Release 9.2.0.1.0

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production

SQL> CREATE TYPE NAME AS OBJECT (
  2   FNAME   varchar2 (15),
  3   MI          varchar2 (1),
  4   LNAME   varchar2 (15) );
  5  /
Type created.

SQL> CREATE TYPE NAME_LIST AS VARRAY(31) OF NAME;
  2  /
Type created.

SQL> CREATE TYPE CONTACT_INFO AS OBJECT (
  2         ADDRESS  varchar2 (15),
  3   CITY   varchar2 (12),
  4   STATE   varchar2 (2),
  5   ZIP   number (5),
  6   EMAIL   varchar2 (20),
  7   PHONE   number (10) );
  8  /
Type created.

SQL> CREATE TYPE CONTACT_LIST AS VARRAY(64) OF CONTACT_INFO;
  2  /
Type created.

SQL> CREATE TABLE SUPPLIER (
  2   TYPE_ID  NUMBER (5),
  3   Name   VARCHAR2 (50),
  4   FAX   NUMBER (10),
  5   URL   VARCHAR2 (20),
  6   CONTACT_INFO CONTACT_INFO ) ;
Table created.
```

[13] For larger volumes of data it is best to use nested tables as a primary implementation methodology since the literals are stored external to the primary table.
[14] Spooled from the buffer of an Oracle 9.2 instance

```
SQL> CREATE TABLE SHIPPER (
  2   TYPE_ID  NUMBER (5),
  3   Name   VARCHAR2 (50),
  4   FAX   NUMBER (10),
  5   URL   VARCHAR2 (20),
  6   CONTACT_INFO CONTACT_INFO ) ;
Table created.

SQL> CREATE TABLE CHANNEL_PARTNER (
  2   TYPE_ID  NUMBER (5),
  3   Name   VARCHAR2 (50),
  4   FAX   NUMBER (10),
  5   URL   VARCHAR2 (20),
  6   CONTACT_INFO CONTACT_INFO ) ;
Table created.

SQL> CREATE TABLE CUSTOMERS (
  2   CUST_ID  NUMBER (5),
  3   Name   NAME,
  4   CONTACT_INFO CONTACT_INFO ) ;
Table created.

SQL> CREATE TABLE EMPLOYEES (
  2   CUST_ID  NUMBER (5),
  3   Name   NAME,
  4   CONTACT_INFO CONTACT_INFO ) ;
Table created.

SQL> CREATE TABLE BUSINESS_PARTNERS (
  2   PARTNER_ID  NUMBER (5),
  3   NAME   VARCHAR2 (50),
  4   PARTNER_TYPE VARCHAR2 (12),
  5   TYPE_ID  NUMBER (5) );
Table created.
```

Records can now be inserted into and retrieved from the object-relational structures, as seen in the following SQL syntax:

```
SQL> INSERT INTO SHIPPER (TYPE_ID, NAME, FAX, URL, CONTACT_INFO) VALUES
  2   (123, 'West Coast Shipping', 6245551234, 'www.wcs.com',
  3   CONTACT_INFO ('456 Maple St', 'Denver', 'CO', 80021, 'Bob@wcs.com', 3035550909));

1 row created.

SQL> select * from shipper;

TYPE_ID   NAME                 FAX          URL             CONTACT_INFO (ADDRESS, CITY, STATE, ZIP,
EMAIL, PHONE)
-------------------------------------------------------------------------------------------------------------------------------------
 123          West Coast Shipping   6245551234   www.wcs.com  CONTACT_INFO ('456 Maple St', 'Denver', 'CO', 80021, 'Bob@wcs.com', 3035550909)
```

This positions the structure for the next step in the mapping from relational to object-relational schema. The relationships of the object-relational schema are those of class and sub-class. So by changing the modeling primitives, we see that entities become the classes and sub-classes, the attributes become properties, the relationships migrate from a focus on cardinality to a focus on class and sub-class, and methods can be associated to the classes and sub-classes to provide transaction (operations) functionality against the literals and to provide an interface at the application layer. The schema notation then starts to look more like an object-relational logical structure. Figure 2 illustrates the mapping of the relational schema to the

object-relational schema. We can see that class and sub-class relationships can now be expressed, as in "SHIPPER IS A BUSINESS PARTNER" and "SUPPLIER IS A BUSINESS PARTNER". This higher order semantic data model allows for inferential reasoning, a principle consideration within the Information System ontology[15] (Daconta, Orbst, and Smith, 2003).

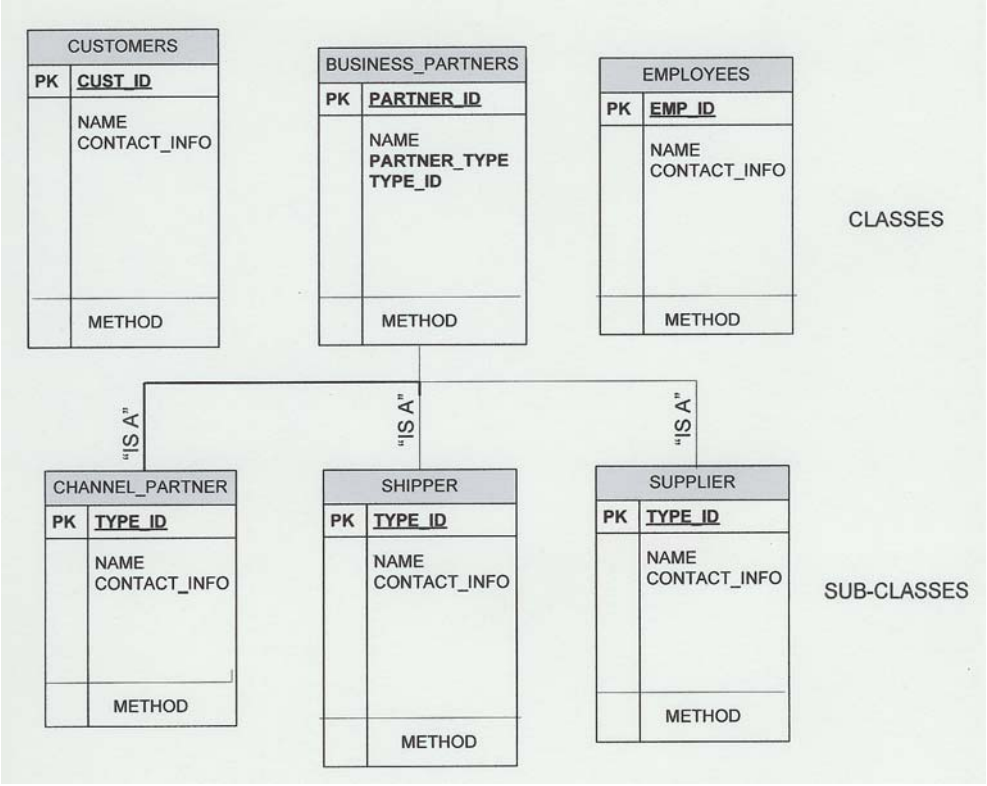

Figure 2: The Object-Relational Equivalent Schema

ONTOLOGY DEVELOPMENT BASED ON HIGHER ORDER SEMANTIC DATA MODEL

The next step in the migration from the object-relational schema to the Information System ontology is to use the higher order semantic data schema as a basis for ontological development. By creating a superclass called "Corporate Affiliations", the CUSTOMERS, BUSINESS_PARTNERS, and EMPLOYEES classes become subclasses. A channel partner can now be viewed as a subclass of business partners, which is a subclass of corporate affiliations. The ontology for the corporate domain of the enterprise can now be completed by adding any additional structured data components, and all of the unstructured data components. Additional structured data components might include a "COMPETITORS" subclass, a "MARKETING PARTNERS" subclass, a "MANAGEMENT" subclass, and a "CORPORATE STRATEGIC ALLIANCES" class. Approximately 20% of corporate data is structured data, and 80% is unstructured data[16]. Unstructured data can be found in emails, intranet site pages, memoranda, spreadsheets, etc. The ontology is used as linkage between the structured and unstructured data in the corporate enterprise. In this example, massive amounts of data and information can be found in historical information repositories. Many times corporations make the same mistake over and over again because new management comes into play and there is no corporate memory of historical events (both good and bad) that lead to previous successes and failures. Much of the corporate memory is stored in unstructured data that is not accessible by current structured data search engines[17]. The intent of the corporate information system, aside from transacting business with customers in the marketplace, is to advance the Information Systems value proposition from data to information, and then from information to knowledge. With an estimated 80% of the corporate enterprise's data and information in unstructured

[15] Meaning inferential engines are a key construct in the Semantic Web architecture and should also apply to domain ontology searching within the enterprise.
[16] According to a recent survey by the Gartner Group.
[17] Primarily referring to relational DBMS engines using SQL.

data repositories, the advancement of information to knowledge is impossible without a "bridge" between the two worlds (Sugumaran and Storey, 2003). The Corporate Domain Information System Ontology is the bridge between structured and unstructured data and information. Figure 3 illustrates how the information system ontology for a corporate domain could be structured. The next step in this methodology is to map the structured and unstructured data by using XML as a base technology to achieve higher semantics with the data and information, and achieve a basis for implementation of an ontological language. Since key structures between structured and unstructured data is difficult to achieve, and indexing unstructured data still results in low context searching, the common field of play between the two worlds is in the area of semantic searches using a search engine (Sciore, Siegel, and Rosenthal, 1994). In effect, a meta-model is created that integrates the object-relational model and the meta-data model created by manipulating unstructured data into an ontological framework using XML (Pinto and Martins, 2002). This integrated meta-model is the bridge between the structured and unstructured data worlds, and is not necessarily limited to the domain of corporate data and information, but is extensible to semantic (Web) models via migration of XML schemas to RDF[18] and RDFS[19] (Metamodel.com, 2004). This report develops one methodology for moving the structured relational data into the corporate domain meta-model by using the DBMS capability to transform structured data at the object level to a XML document. The object instances can be transformed to an XML document in the database and then appropriately manipulated to RDFS for semantic searches within the corporate domain ontology (McKinnon and McKinnon, 2003). Figure 3 also illustrates the evolution of the corporate domain ontology from the class diagram by including additional relationship classes of structured data, and generalized association with unstructured data repositories.

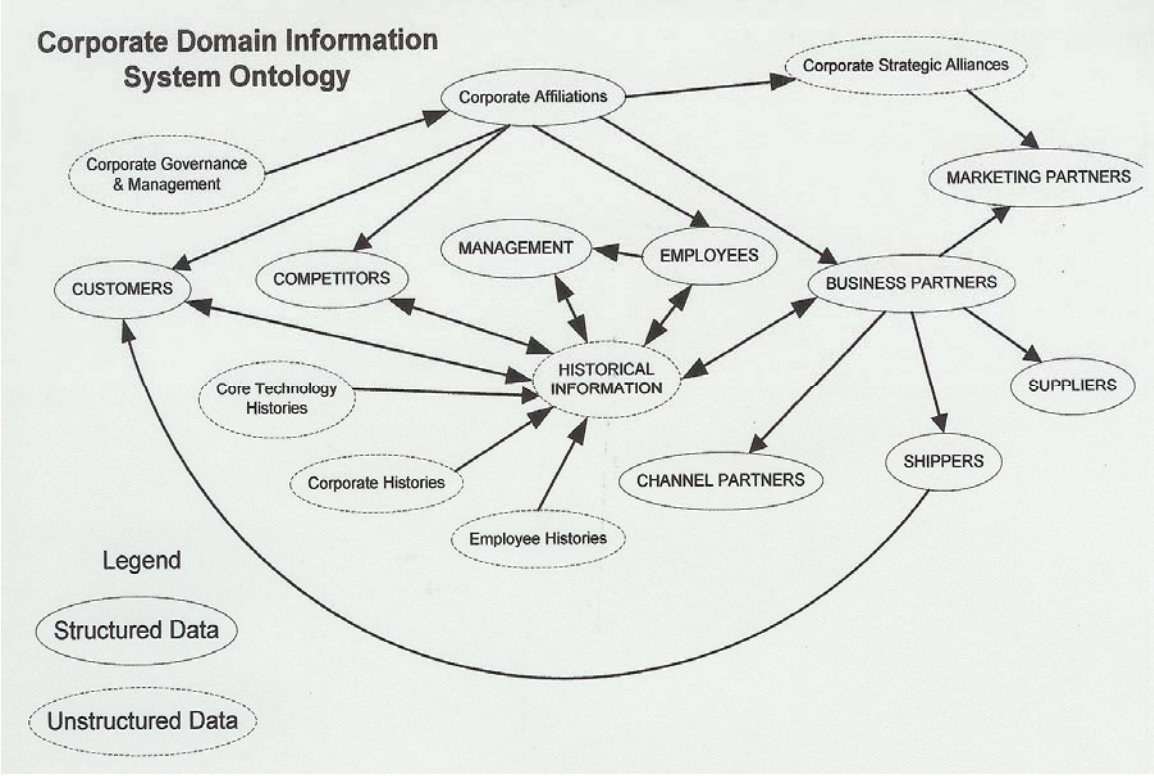

Figure 3: An Ontology for a Corporate Domain

USING XML FOR DATA TRANSPORT AND ONTOLOGICAL SEARCH CAPABILITY

[18] Resource Description Framework
[19] Resource Description Framework Schema

Any object instances are available for transform to XML type data at the database level. In the example here, we have a class SHIPPER where 3 object instances of CONTACT exist relative to 3 instances of the property type_id and name. A SQL query gives the values for the type_id and name of these properties:

```
SQL> select type_id, name from shipper;

   TYPE_ID   NAME
----------   --------------------------------
       123   West Coast Shipping
       345   East Coast Shipping
       974   Southern Shipping
```

The object CONTACT_INFO[20] can be associated to either type_id or name using basic SQL query as:

```
SQL> select name, contact from shipper;

NAME
--------------------------------------------------
CONTACT(ADDRESS, CITY, STATE, ZIP, EMAIL, PHONE)
--------------------------------------------------------------------------------
West Coast Shipping
CONTACT_INFO('456 Maple St', 'Denver', 'CO', 80021, 'Bob@wcs.com', 3035550909)
East Coast Shipping
CONTACT_INFO('789 Maple Ave', 'Atlanta', 'GA', 20236, 'Sue@ecs.com', 3035550909)
Southern Shipping
CONTACT_INFO('321 Pine St', 'Dallas', 'TX', 65321, 'Jason@ss.com', 2055550909)
```

To begin the transform of the data structure to one that supports an ontological search, a DBMS SYSTEM function can be used to convert the property and object instances into an XML document (Shah, Finin, Joshi, Cost and Mayfield, 2002). In this example, the SQL syntax[21] to generate the XML transform of the instances in the CONTACT_INFO object directly from the database is:

```
SQL> select SYS_XMLGEN (contact) as XML_CONTACT_INFO_DOC from shipper;

XML_CONTACT_INFO_DOC
--------------------------------------------------------------------------------
<CONTACT>
  <ADDRESS>456 Maple St</ADDRESS>
  <CITY>Denver</CITY>
  <STATE>CO</STATE>
  <ZIP>80021 </ZIP>
  <EMAIL>Bob@wcs.com</EMAIL>
  <PHONE>3035550909</PHONE>
<CONTACT>
  <ADDRESS>789 Maple Ave</ADDRESS>
  <CITY>Atlanta</CITY>
  <STATE>GA</STATE>
  <ZIP>20236 </ZIP>
  <EMAIL>Sue@ecs.com</EMAIL>
  <PHONE>3035550909</PHONE>
<CONTACT>
  <ADDRESS>321 Pine St</ADDRESS>
```

[20] Previously created
[21] And associated output

<CITY>Dallas</CITY>
<STATE>TX</STATE>
<ZIP>65321 </ZIP>
<EMAIL>Jason@ss.com</EMAIL>
<PHONE>2055550909</PHONE>

This methodology uses SQL queries to generate XML documents. Moreover, this XML document with the data values is available for transform to RDFS and storage in a searchable repository in the corporate domain ontology infrastructure. Alternatively, the XML document can be generated outside of the database, but there are some strong benefits to this method, the most notable being performance. In this method, the XML generation (transform) is accomplished close to the data at the server level, and all the data is retrieved in one roundtrip (Wyke, Rehman, and Leupen, 2002). Many times the XML document will contain elements or properties that have complex data structures. By using SQL query to derive the XML document, the object-relational class extensions in SQL3 have the ability to capture the structure of the data in the object type, object reference, and collection. There are two basic methods for storing the data structure in the XML document using the object-relational class as the structure (Goldfarb and Prescod, 2004). The first is to store the properties of the elements in a relational table and define object views to capture the structure, and the second is to store the structured XML elements in an object-relational table. Since the transform from structured relational data to structured object-relational data has already taken place in the methodology, the data in object-relational form can be easily updated and queried by SQL. Base data can be updated via SQL DML, and the appropriate method operates on the class properties and objects to accomplish the transform into the XML document and transport of the document to the meta-model repository. Views can also be used to combine structured and unstructured XML data (Wyke, Rehman and Leupen, 2002). Structured data can be stored in one physical space within an object-relational schema and unstructured data can be stored within CLOBs[22] or BLOBs[23] in another physical space. To integrate the data you create a view object by using type constructors in the view's SELECT definition. This allows the retrieval of the constructed data from the view as a single XML document.

INTEGRATING SEMI-STRUCTURED AND UNSTRUCTURED DATA WITH STRUCTURED DATA

What we have learned about information retrieval and searching semi-structured data and unstructured data on the World Wide Web has application to searching data within the construct of the domain (business) ontology. A goal of the Semantic Web is to improve context searching by the association of meta-data to the data values. The Ontological Web Languages are evolving through a hierarchy of improved semantics beginning with XML. The evolutionary track begins with XML or XML schema, improves in semantics and efficiency through Resource Description Framework Schema (RDFS), adds further ontological characteristics via DAML+OIL, and will most likely be standardized with the World Wide Web Consortium's (W3C) Web Ontology Language (OWL). So the focal point for the integration is XML. Since (as seen previously) structured data can be converted into hierarchical XML formats from object-relational databases, semi-structured data is generally in HTML and XML (tagged language) formats, and unstructured data can be converted to XML, then the process of integrating all data into a native XML repository has a basis. Transform of this integrated XML data into an ontological language will add an intelligent layer beneath the user's presentation layer. Using current Web search and meta-search technologies now becomes possible internally in the organization, just as the conceptual Semantic Web is structured. Integrating unstructured data, such as email, memoranda, and notes with semi-structured business data such as HTML pages and XML documents, and structured data such as relational data and spreadsheets in a federated data environment allows for non-indexed searching within the domain (business) ontology. Autonomous corporate databases and disparate data sources are required to integrate much like the integration that occurs in an operational data store for OLAP architectures. Alternatively, the integration of these diverse corporate data sources can be integrated directly in the federated XML repository that supports the domain ontology. As the object-relational transform of the relational data adds hierarchical relationships to the data, the design of the ontology will bring hierarchical relationships to the semi-structured and unstructured data. This again is implemented and organized in the federated XML repository. With this in place, reactive searching using search and meta-search engines, and proactive searching using intelligent search agents, can be accomplished across the data federation. Inferential engines have the data relationship hierarchies expressed in the ontology from which to "mine" the data for actionable inferences. The future direction of this research will be to develop the architecture for the federated XML repository, and a compatible methodology for the transform of the unstructured data into an XML format.

SUMMARY AND CONCLUSIONS

[22] Character Large Objects
[23] Binary Large Objects

Ontologies can be used in Information Systems to create a bridge between structured and unstructured data. Structured enterprise data is most often found in relational data structures and lacks the higher order semantics of an object-relational structure. By mapping relational data to object-relational structures and the using XML generation methods to transform the instances of the objects and properties to XML documents, a methodology for achieving Ontologies using higher order semantic data models can be implemented. The ontology defined in this report uses the object-relational classes to form associations between classes, and class and sub-class relationships, between structured and unstructured data. Corporate domain Ontologies of this type can yield knowledge generation in ways that no structured data (only) system can do. Inferences among the relationships between all corporate affiliations, along with the application of unstructured data and information, could result in competitive intelligence previously unavailable (Maedche, Motik, Stojanovic, Studer and Volz (2003). However, this methodology is generally opposed by the thinking that the bridge between structured and unstructured data is not in the domain of semantic order, but in the development of key constraints and indexing between the structured data and the repository of unstructured data (Kim, 2002). But that methodology is not synchronous with the current strategy of the semantic web and the general problem of poor context searching on the Web due to the lack of a meta-model approach. The generation of XML from the meta-model in this methodology forms the foundation for the transform to RDF and RDFS as an ontological language layer to search the composite data. ACID theory properties in the object-relational DBMSs real-time data is preserved through SQL DML capability, yet XML generation methods within the classes can provide updated XML feeds into the repository hosting the integrated structured and unstructured data. The object-relational schema provides the modeling primitives and data hierarchy within classes to support the development of local and extended domain Ontologies. This ontology-driven approach to implementing an information system uses the object capability of SQL3 and the higher order semantics found in object-relational design. As a result, it is possible to achieve domain Ontologies that provide a foundation for knowledge creation and management, and thus provide the enterprise with the needed fuel to innovate at a faster rate.

REFERENCES

1. Braga, R. M. M., Mattoso, M., & Werner, C. M. L. (2001). The User of Mediation and Ontology Technologies for Software Component Information Retrieval. *ACM 1-58113-358-8/01/0005*.

2. Cook, M. A. (1996). *Building Enterprise Information Architectures*. Upper Saddle River, NJ: Prentice-Hall PTR.

3. Daconta, M. C., Orbst, L. J., & Smith, K. T. (2003). *The Semantic Web*. Indianapolis, IN: Wiley Publishing.

4. Goldfarb, C. F., & Prescod, P. (2004). *XML Handbook* (5th ed.). Upper Saddle River, NJ: Prentice-Hall PTR.

5. HBR. (1998). *Harvard Business Review on Knowledge Management*. Cambridge, MA: Harvard Business School Press.

6. Kashyap, V., & Sheth, A. (1996). Semantic and Schematic Similarities Between Database Objects: A Context Based Approach. *The VLDB Journal, 1996*(5), 276-304.

7. Kim, H. (2002). Predicting How Ontologies for the Semantic Web Will Evolve. *Communications of the ACM, 45*(2), 48+.

8. Maedche, A., Motik, B., & Stojanovic, L. (2003). Managing Multiple and Distributed Ontologies on the Semantic Web. *The VLDB Journal, 2003*(12), 286-302.

9. Maedche, A., Motik, B., Stojanovic, L., Studer, R., & Volz, R. (2003). An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. *ACM 1-58113-680-3/03/0005*.

10. McKinnon, A., & McKinnon, L. (2003). *XML*. Boston, MA: Thomson Course Technology.

11. Metamodel.com. (2004). *Metamodeling*. Retrieved January 22, 2004, from the World Wide Web: http://www.metamodel.com

12. Obrst, L. (2003). Ontologies for Semantically Interoperable Systems. *CIKM 2003, ACM 2003 1-58113-723-0/03/0011*.

13. Pinto, H. S., & Martins, J. P. (2002). A Methodology for Ontology Integration. *Communications of the ACM 1-58113-380-4/01/0010*.

14. Sciore, E., Siegel, M., & Rosenthal, A. (1994). Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. *ACM Transactions on Database Systems, 19*(2), 254-290.

15. Shah, U., Finin, T., Joshi, A., Cost, R. S., & Mayfield, J. (2002). Information Retrieval on the Semantic Web. *CIKM 2002 November 4-9, 2002, ACM 1-58113-492-4/02/0011*.

16. Sugumaran, V., & Storey, V. C. (2003). A Semantic-Based Approach to Component Retrieval. *The DATABASE for Advances in Information Systems, 34*(3).

17. Wyke, R. A., Rehman, S., & Leupen, B. (2002). *XML Programming*. Redmond, WA: Microsoft Press.