Semantic Enterprise Optimizer and Coexistence of Data Models

P. A. Sundararajan[1], Anupama Nithyanand[2], S.V. Subrahmanya[3]
[1,2,3] Infosys Technologies Ltd., United Kingdom

Abstract

The authors propose a semantic ontology–driven enterprise data–model architecture for interoperability, integration, and adaptability for evolution, by autonomic agent-driven intelligent design of logical as well as physical data models in a heterogeneous distributed enterprise through its life cycle. An enterprise-standard ontology (in Web Ontology Language [OWL] and Semantic Web Rule Language [SWRL]) for data is required to enable an automated data platform that adds life-cycle activities to the current Microsoft Enterprise Search and extend Microsoft SQL Server through various engines for unstructured data types, as well as many domain types that are configurable by users through a Semantic- query optimizer, and using Microsoft Office SharePoint Server (MOSS) as a content and metadata repository to tie all these components together.

1. Introduction

Data models differ in their structural organization to suit various purposes. For example, product and organization hierarchies yield well to the hierarchical model, which would not be straightforward to represent and access in a relational model (Table 1).

Data-model name	Purpose
Hierarchical	Complex master-data hierarchies (1:M) Very high schema-to-data ratio
Network	Complex master-data relationships (M:M) Spatial networks, life sciences, chemical structures, distributed network of relational tables
Relational	Simple flat transactions. Very low schema-to-data ratio
Object	Complex master-data relationships, with nested repeating groups
XML	Integration across heterogeneous components; canonical; extensible
File systems	Structured search
Record-oriented	Primary-key retrieval—OLTP—sequential processing
Column-oriented	Secondary-key retrieval; analytics; aggregates; large data volume, requiring compression
Entity-attribute-value	Flexibility; unknown domain; changes often to the structure; sparse; numerous types together

Table 1. Data models for various purposes

The model is decided by following factors:

1. Ease of representation and understandability of the structure for the nature of data
2. Flexibility or maintainability of the representation
3. Ease of access and understanding the of retrieval, which involves the query, navigation, or search steps and language
4. Ease of integration, which is an offshoot of maintainability and understanding
5. Performance considerations
6. Space considerations

Depending on the requirement—be it a structured exact search or a similarity-based unstructured, fuzzy search—we can have a heterogeneous mix of structured, semistructured, and unstructured information to give the right context to enterprise users.

While the relational database helped with the sharing of data, metadata sharing itself is a challenge. Here, enterprise ontology is a candidate solution for metadata integration, and it leverages such advances for stable Enterprise Information Integration (EII) and interoperability, in spite of the nebulous nature of an enterprise.

Ontologies are conceptual, sharable, reusable, generic, and applicable across technical domains. They contain explicit knowledge that is represented as rules and aids in inference. Also, they improve communication across heterogeneous, disparate components, tools, technologies, and stakeholders who are part of a single-domain enterprise.

2. Evolution of Enterprise Integration

It is interesting to note the evolution of enterprise integration over periods of time, when there were simple applications for each specific task in the past, to the applications on the Web that can communicate with each other—achieving larger business functionality, and blurring the boundaries of intranets, Internet, and enterprises:

1. Data file sharing in file systems
2. Databases
3. ERP, CRM, and MDM
4. ETL—data warehouse
5. Enterprise Information Integration (EII)
6. Enterprise Application Integration (EAI)—service-oriented architecture (SOA)
7. Semantic Web services

With respect to information, too, the lines between fact and dimension, data and language, and structured and unstructured are blurred when a particular type of data morphs over time, with volume and its importance to and relationship with its environment. A normal transaction data model can progress from business intelligence and predictive data mining to machine learning toward a knowledge model and becomes actionable in language form, where it can communicate with other systems and humans.

Enterprise data needs a common vocabulary and understanding of the meaning of business entities and the relationships among them. Due to the variety of vendors who specialize in the functions of an enterprise, it usually is a common sight to see heterogeneous data models from disparate products, technologies, and vendors having to interoperate.

Data as a service with SOA, enterprise service bus (ESB), and canonical data models have tried to address this disparity in schema or structure, but have not addressed the seamless semantic interoperability until the advent of Semantic Web services.

Semantic enterprise integration through enterprise Web Ontology Language (OWL) can be a solution for the seamless semantic interoperability in an enterprise.

3. Motivation for This Paper: Industry Trends

3.1 Accommodating and Coexisting with Diversity

Storing all the data in a row-oriented, third normal form (3NF) relational schema might not be optimal. We see many trends, such as various types of storage engines, that are configurable and extensible in that specific domain data types can be configured and special domain indexes built, and the optimizer can be made aware of them to cater to heterogeneous data-type requirements. These object- oriented semantic extensions are built as applications on top of the database kernel, and there are APIs for developers to customize and extend to add their own unstructured or semistructured data types. This is used in spatial, media, and text-processing extensions that come with the product. In some cases, native XML data types are also supported.

Microsoft Enterprise Search is an example of disparate search from e-mail in Microsoft Exchange Server to user profiles in Active Directory to Business Data Catalog in Microsoft SQL Server RDBMS and Microsoft Office documents.

Prominent players have addressed unstructured data in the form of content-management systems, which again have to be accommodated in the proper context with traditional enterprise structured data—both metadata- and content-wise.

3.2 Offload to Auxiliary Units

Many database systems support a row-oriented OLTP store for updated rows and columnar-compressed store for read-only store or disk-based write-only store and memory-based read-only store— moving them across frequently, according to their life-cycle stages and the characteristics that they exhibit. Offloading some load to auxiliary processors that specialize in SQL processing are also some of the practices that we can observe in data-warehouse appliances.

3.3 Intelligent Autonomic Design

Many systems optimally design or recommend based on the following inputs:

- Logical schema
- Sample data
- Query workload

Some systems have abstracted the file-handling parts that must be handled at the OS level.

Oracle Query Advisor and Access Advisor offer best plans, based on statistics.

3.4 Impedance Mismatch and Semantic Interoperability

Object-Relation mapping (ORM) is a classic pattern in which there are two different tools that would like to organize the same data, in two different ways. Here, the same enterprise data has to be represented by using an inheritance hierarchy in object-oriented design, whereas it can be one or more tables in a relational database. LINQ to SQL and LINQ to Entities (Entity Framework) are some tools that address this space.

What if the database is a hierarchical database, or a plain entity- attribute-value model?

If the language happens to be a functional programming model, and the database that we use happens to be an object-oriented database, a different sort of impedance mismatch might emerge.

So, wherever there are different paradigms that must communicate, it is better to have an in-between intermediary.

In this case, the authors propose that the domain model (not an object model) be represented in an enterprise-wide ontological model—complete with all business logic, rules, constraints, and knowledge represented. For each system that must communicate, let it use this ontological model as a common denominator to talk to systems.

Another area of impedance mismatch is the one between a relational SQL model and the OLAP cube Multidimensional Expressions (MDX). MDX is a language in which the levels of hierarchical dimensions are semantically meaningful, which is not the case with tuple-based SQL. Here again, a translation is required.

Instead of going for a point-to-point solution, we might benefit from a common ontology.

The application requests a semantic-data-services provider, which translates the query appropriately to the enterprise ontology model and—depending on the data-source model—federates the query in a modified form that is understood by the specific data model of the data source.

The domain model is conceptual and could replace or reuse the conceptual entity-relationship or UML class-object structural diagrams.

3.5 Data-Flow Architecture for the Semantic Enterprise Model

The following sections explain the Semantic enterprise optimizer that can bridge the gap between the various disparate data models that can coexist and provide the data services intelligently (see also Figure 1).

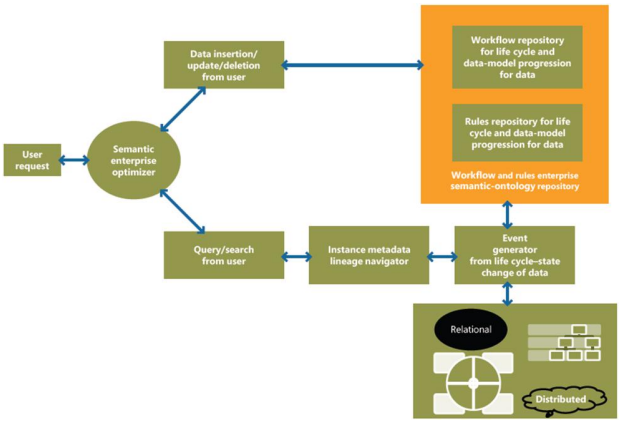

Figure 1. Semantic enterprise optimizer and coexistence of data models

3.6 Autonomic Evolutionary Logical and Physical Design

Depending on the usage, volume of data, and the life-cycle stage, we have a proposal for automatic logical and physical data-model design.

Initially, when a domain is not known with concrete requirements, an entity-attribute-value model is always good to start with. Here, the structure is defined by an administrative screen with parameters, which takes the definition of the record structure and stores the metadata and data by association.

After a periodic interval, there is an agent that watches over the record types over a period of time, as well as the actual data in the record values, to see if the changes have settled down and the structure has become relatively stabilized. When the structure is stabilized, the analyzer now qualifies which type of structure is suitable for the entity—keeping into account the queries, the data and its statistics, the changes to the structure and its relationship with other entities, and its life-cycle stage.

4. Component Model of the Semantic Enterprise Optimizer

4.1 Semantic Enterprise Optimizer

The Semantic enterprise optimizer consults the workflow and rules repositories in case of an insert or state change event, to find out which data model should accommodate the incoming or state- changed data item. In case of a query, it consults the instance metadata lineage navigator to locate the data. Accordingly, it federates the query to online or archived storages, and across heterogeneous models and products. Here, SOA-based data and enablement of metadata as a service is helpful.

4.2 Semantic Data Services

The Semantic Data Services extend the features of the data-service object to enable ontology-driven semantics in its service. The interface services consult the enterprise ontology for interaction.

4.3 Workflow and Rules Enterprise Semantic-Ontology Repository

For each type of data that is classified, we can define the lineage that it should follow. For example, we can say that an employee-master record in an enterprise will be entering as master data. It will follow the semantic Resource Description Framework (RDF) model, in which relationships for this employee with others in the organization are defined. The employee master will also be distributed to have the attendance details in the reporting location; however, salary details will be in the central office, from where disbursements happen. The employee record will be maintained in the online transaction systems till the tenure of the employee with the enterprise. After the employee has left, the employee record will remain for about one year for year- over-year reporting, before it moves into a record-management repository, where it is kept flattened for specific queries. Then, after three years, it is moved into archival storages, which are kept in highly compressed form. But the key identifying information is kept online in metadata repositories, to enable any background/asynchronous/ offline checks that might come for that employee later throughout the life of the enterprise.

All these changes at appropriate life-cycle stages are defined in the workflow repository, together with any rules that are applicable in the rules repository.

4.4 Event-Generator Agent

Based on the preceding workflows and rules, if a data item qualifies for a state change, an event is generated by this component, which alerts the optimizer to invoke the routine to check the appropriate data model for the data item to move into after the state change.

4.5 Instance Metadata Lineage Navigator

Every data instance has metadata associated with it. This will involve attributes such as creation date, created by, created system, the path that it has taken at each stage of its life-cycle state change, and so on. It will also contain the various translations that will be required to trace that data across various systems. This component helps locate the data.

4.6 Data-Model Universe

This is the heterogeneous collection of data models that is available for the optimizer to choose when a data item is created and, subsequently, changes state:

1. Master and reference data—largely static; MDM; hierarchy; relationships; graph; network; RDF
2. OLTP engine—transaction; normalized
3. OLAP cube engine—analytics; transaction life cycle completed; RDF analytics for relationships and semantic relations
4. Records management or file engine—archived data; for data mining, compliance reporting
5. Object and object-relational databases for unstructured information—image databases; content-based information retrieval
6. Text databases for text analytics, full-text search, and natural- language processing
7. XML engines for integration of distributed-transaction processing
8. Stream processing—XML
9. Metadata—comprises RDF, XML, hierarchy, graph integration of heterogeneous legacy databases in terms of M&A, and partnering for providing collaborative solutions

Here, the query must be federated, and real-time access has to be enabled with appropriate semantic translation.

When the life cycle of the data changes, there are sensors or machine-learning systems that are programmed to understand the state when the life-cycle stage changes. When such changes are detected, the record is moved accordingly from transaction management to OLAP or data mining, or archival location, as per the lineage.

So, when information is requested, the optimizer, based on the business rules that are configured, is able to find out which engine should be able to federate that query, based on the properties of the search query, and appropriately translate it into hierarchical, OLAP, or file-system query.

In regular applications that are developed, we might not see much of an advantage, but where there are changes in existing flows in life- cycle states or changes in new data types.

5. Conclusion

We see the enterprise scene dominated by a distributed graph network GRID of heterogeneous models, which are semantically integrated into the enterprise; also, that enterprise data continually evolves through its logical and physical design, based on its usage, origin, and life-cycle characteristics.

Various data models that have been found appropriate or any combination thereof can coexist to decide the heterogeneous model of an enterprise. The relational model emphasized that the user need not know the physical structure or organization of data. In this model, we propose that even the logical model need not be known, and any enterprise-data resource should be reusable across operating systems, database products, data models, and file systems.

The architecture describes an adaptable system that can intelligently choose the data model as per the profile of the incoming data. The actual models, applications and life-cycle stages that are supported themselves are illustrative. The point is that it is flexible enough to accommodate any future model that might be invented in the future. Adaptability and extensibility are takeaways from this architecture.

Also, dynamic integration of enterprise boundaries will lead to more agility and informed decisions in the increasing business dynamics.

Acknowledgments

The first two authors are thankful to the third author, S. V. Subrahmanya, Vice President at E&R, Infosys Technologies Ltd., for seeding and nurturing this idea, and to Dr. T.S. Mohan, Principal Researcher at E&R, Infosys Technologies Ltd., for his extensive and in-depth review.

The authors acknowledge and thank the authors and publishers of the papers, textbooks, and Web sites that are referenced. All trademarks and registered trademarks that are used in this article are the properties of their respective owners/companies.

References

- [1] Larson, James A., and Saeed Rahimi. *Tutorial: Distributed Database*
- [2] Management. Silver Spring, MD: IEEE Computer Society Press, 1985.
- [3] Hebeler, John, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez, and Mike Dean (foreword). *Semantic Web Programming*. Indianapolis, IN: Wiley Publishing, Inc., 2009.
- [4] Powers, Shelley. *Practical RDF*. Beijing; Cambridge: O'Reilly & Associates, Inc., 2003.
- [5] Chisholm, Malcolm. *How to Build a Business Rules Engine: Extending Application Functionality Through Metadata Engineering*. Boston: Morgan Kaufmann; Oxford: Elsevier Science, 2004.
- [6] Vertica Systems. Home page. Available at http://www.vertica.com (visited on October 16, 2009).
- [7] Microsoft Corporation. Enterprise Search for Microsoft. Available at http://www.microsoft.com/enterprisesearch/en/us/default.aspx(visited on October 16, 2009).
- [8] G-SDAM. Grid-Enabled Semantic Data Access Middleware. Available at http://gsdam.sourceforge.net/(visited on October 18, 2009).
- [9] W3C. "A Semantic Web Primer for Object-Oriented Software Developers." Available at http://www.w3.org/TR/sw-oosd-primer/(visited on October 18, 2009).
- [10] Oracle. Oracle Exadata. Available at http://www.oracle.com/database/exadata.html(visited on October 21, 2009).

P. A. Sundararajan is a Lead in the Education & Research Department with ECOM Research Lab at Infosys Technologies Ltd. He has nearly 14 years' experience in application development and data architecture in Discrete Manufacturing, Mortgage, and Warranty Domains.

Anupama Nithyanand is a Lead Principal in the Education & Research Department at Infosys Technologies Ltd. She has nearly 20 years' experience in education, research, consulting, and people development.

S. V. Subrahmanya is currently Vice President at Infosys Technologies Ltd. and heads the ECOM Research Lab in the Education & Research Department at Infosys. He has authored three books and published several papers in international conferences. He has nearly 23 years' experience in the industry and academics. His specialization is in Software Architecture.