2009 IEEE International Conference on Services Computing

Ontology-based Information Sharing in Service-Oriented Database Systems

TszYan Chow*, Wei-Tek Tsai*+, Janaka Balasooriya*, Xiaoying Bai+
*Department of Computer Science and Engineering, School of Computing and Informatics
Arizona State University, Tempe, AZ, USA
+ Department of Computer Science and Technology, Tsinghua University, Beijing, China
Wei-Tek.Tsai@asu.edu, janakab@asu.edu

Abstract

This paper presents a novel information sharing framework using Service-Oriented Database System (SODB) for service registry and repository that facilitates data integration. A SODB is a database architecture composed with reusable services to support searching, querying, deleting, and storing data. As SODB is organized in a service-oriented manner, SODB can be easily changed or maintained by reusing different services. Thus, it can be used to share information in the cloud. The Information Sharing System (ISS) component of the SODB employs domain ontology to share data sources. The domain ontology utilizes mathematical equivalence relations to map data sources into appropriate domain ontology. It also facilitates dynamic query composition across data sources. This paper also presents the design, implementation, and performance evaluation of the ISS component. Our implementation is based on a real application, the Arizona Healthcare Cost Containment System (AHCCCS). This application demonstrates that the ISS can facilitate complex information integration. Finally, this paper presents the performance of the ISS using WAPT (Web Application Testing) for Microsoft Windows XP, and the response time consistently fall in between 0.1 to 0.15 second for each request.

1. Introduction

Information sharing is important in a networked environment where organizations exchange data extensively [2, 3]. Lack of data integration in SaaS (Software as a Service) is a roadblock in service computing [32]. Current solutions need ad-hoc, custom-made, "hand coding" for each data integration requirement. In [32], authors pointed out that data integration can be solved by providing a) "modular software that can be meshed, branded, and delivered as part of SaaS provider's application environment." and b) "on demand service delivery." Recently, many SaaS applications are being moved to cloud computing as a cloud infrastructure provides virtualized computing and storage services with self-healing, self-monitoring, resource registration and discovery [33, 36].

Data integration in cloud computing can be agile and configurable as new services may be added into the system without stopping the application, and new applications can be composed from the existing services by altering the application specification without changing the underlying framework. One issue in cloud computing is to provide versatile components that present resources to the cloud including a dependable and fully functional service broker that provides a reliable and an efficient service hosting and discovery [33, 35]. Consider a registry and repository system that can support searching, querying, storing and deleting services. A system with these capabilities can be considered as a Service-Oriented Database System (SODB). A SODB is a brokerage system in which,

a) Each database component can be classified according to their function types such as data operations (insert, delete, and update), concurrency control, caching, and data formatting, and each function can be modeled as a service. Thus, a SODB is a federated service registry repository. As these services work collectively by communicating with each other through an ESB (Enterprise Service Bus), adding, replacing, or removing database services may not require bringing the database system down.
b) The information hosted on a SODB can be tied to services, not to any specific database system. The idea of storing and representing data using services provides a flexible mechanism in data storage as the information is no longer subject to the schema structures. This provides the capability to access and integrate data from heterogeneous sources.
c) The built-in self-configuration mechanism allows individual services in this framework to be combined, configured, and/or reconfigured at runtime.

Section 3 presents an SODB architecture. An SODB needs to facilitate different organizations to combine information from multiple data providers into one logical form. Thus, representing, accessing, maintaining, managing, analyzing, and integrating data across heterogeneous possibly unstructured data sources is critical in an Information Sharing System (ISS).

1.1 Need for Information Integration

Many information sharing applications such as those in healthcare often face issues in integrating and sharing data [1, 3]. One example is the Arizona Healthcare Cost Containment System (AHCCCS), and it has been created to handle application and eligibility data for programs such as Arizona Long Term Care System and KidsCare programs. Like AHCCCS, Department of Economic Security (DES) and Health-e-Arizona (HEA) also determine eligibility for non-medical related programs, such as Temporary Assistance for Needy Families (TANF) and Food Stamps. In many cases, eligibility for such non-medical benefit programs accompanies eligibility for Medicaid or other medical programs administered by AHCCCS. For example, a family that applies for TANF may have an income too high to qualify for that program but low enough to be eligible for the AHCCCS KidsCare program. Similarly, an AHCCCS eligibility worker may discover during a KidsCare eligibility interview that a family's income level is low enough to qualify for one of the DES assistance programs. Note that the information systems of these agencies are not structured to diffuse with each other because each agency has its own eligibility requirements and interests, the data extraction process must be custom-made for each participated agency. The need for information sharing is not only confined to the Healthcare sectors. The Department of Defense (DoD) has developed a Net-Centric Data Strategy Guideline [18] to address information sharing. As stated in the guideline, data interoperability and visibility can be achieved by having each of the data providers or COIs (Community of Interests) to: (i) use a common platform for exchanging their information; (ii) publish shared information that can be searched; (iii) develop an agreement on semantic and structural format on the shared information; and (iv) use domain-specific vocabulary and taxonomy for unambiguous communication.

The proposed ISS addresses these issues. It can be seen as a third-party agent that manages the shared information that

This research was sponsored by Department of Education FIPSE, and the opinions expressed are of the authors not of the federal government.

a) Supports data publishing by utilizing an ontology service that resides in a SODB to transform local business information into commonly-agreed terms.
b) The shared information can be queried across multiple domains using the standardized terminology and the results of the query are available for others.
c) Supports real-time access to the source data.

Section 4 describes the proposed ISS.

2. Related Work

Data integration in cloud is a new research direction with many issues. Yahoo PNUTS [33, 34], informatica's SaaS data integration platform [41], and IBM cloud computing initiative [35] provide initial data integration frameworks in the cloud. ***Databases with SOA features:*** In traditional database systems, every database function is implemented in the system. In contrast, components in the SODB can be implemented remotely as services providing flexibility. SODB can be integrated into a cloud infrastructure. Table 1, summarizes and compares traditional databases and SODBs.

Microsoft SQL Server 2005 [23] applied SOA technology in a database management system. In SQL Server 2005, native service access is made available via a kernel-mode HTTP driver, which allows SQL Server to register a portion of the URL namespace without requiring the need for a web server as an intermediary.

Table 1: Traditional DB vs. SODB

	Traditional Databases	**Service-Oriented Databases**
Database Functions Implementing methods	Implemented in the management system	Service functions may be implemented locally and remotely
Database Extensibility & Portability	Version upgrades or package installations require the system to be taken offline or restarted	Services work collectively by communicating through ESB. Adding, replacing, removing services from the system may not require the system to be brought down.
Data Orientation	Data is closely tied to a relational or object-oriented system	Data may be tied to a domain-specific service and not to any specific database system
Data Storage	Data is stored either in tables (by rows and columns) or in objects	Data may be stored in tables, objects, or in terms of services
Database Schema	Need to define schema before usage	Schema may be defined at runtime as new data arrives
Concurrency Control	Implemented locally in the database system	Possibly two concurrency control (CC) mechanisms: <u>*High level:*</u> CC among services using policy <u>Low level:</u> CC among the native database functions
Database Architecture style	It can be centralized, distributed, or federated	Mostly distributed and federated.
Entity Relationship Model	<u>*Relational DB:*</u> Use Primary and Foreign Keys to define data relations <u>*Object-Oriented DB:*</u> Use Inheritance or Reference Pointers	Data relations may be specified using service ontology or policies

Oracle Database 10g [24] comes with an enterprise SOA suite that offers two service modes: a service consumers and providers. Amazon SimpleDB™- Limited Beta [16] is a service that provides simple query functions, creates and stores multiple data sets and provides data look up over a schema-less database storage. DBNet [9] is another system that provides SOA-based architecture. Web Service Management System (WSMS) [26] proposed by Stanford University and Duke University aims at enabling clients to query a collection of web services in a transparent and integrated fashion. They focus on query optimization for select-project-join queries spanning multiple web services. Table 2 summaries and compares various features. As shown in table 2, currently self configurability and service broker features are not available in most of the current systems.

Information Sharing: Information retrieval system that uses ontology for data integration has prevailed recently. These systems include TSIMMIS [14], InfoMaster [5] and InfoSleuth [27]; they all use ontology to define database schema objects and their constraints for the data source. The data-source specific domain ontology will then be advertised at the broker agent so that it can perform reasoning on the advertised ontology and provide references to the appropriated data source. VISPO, is another Web-based framework that supports knowledge sharing using ontology [1].

IBM presented a service-based infrastructure for enabling information sharing [20]. Spatial data integration in geographic information system (E-GIS) using ontology and service-based method was explored in [22]. MetaMatrix [17] integrates data from multiple data sources (such as XML, services, databases, spreadsheets, and media databases) and provides many data services such as data query services, content search and discovery services. WS02 [29] adopts W2-policies to control data access, data caching and security for the underlying data sources such as MySQL, Oracle, DB2, MS Excel, CSV, and LDAP. Although WS02 data services offer a convenient way to access data originated from many different places, service queries cannot span over multiple data sources [19]. Data exposed in WS02 data services is through the direct exposure of the underlying databases objects, such as tables, service functions and store procedures. Thus if these database artifacts are modified, then the data services will fail.

Table 2: Databases with SOA Features: A Comparison

	SODB	**SQL SERVER**	**DB-NET**	**WSMS**	**Amazon**	**Oracle**
Service Broker Feature (To facilitate Service discovery & matching)	Yes	Can be added on top of the existing brokerage system.	Can be added	Can be added	Can be added	Yes
Loosely-Coupled System Organization	Yes	No	Yes	n/a	Yes	No
Flexible Data Storage	Yes	Yes	Yes	Yes	Yes	Yes
Web Service Query Optimization	Yes	Can be added	Can be added	Yes	Can be added	Can be added
Self Configurable	Yes	Can be added	can be added	Can be added	Can be added	Can be added

3. Service-Oriented Database (SODB)

An SODB architecture is shown in Fig. 1 and it is similar to WebStrar [12] and IBM WebSphere Reference Architecture [20]. WebStrar (Web Services Testing, Reliability Assessment, and Ranking) is a collaborative infrastructure that supports the development and verification of services. Services are connected to an ESB to cooperate and communicate with each other. The infrastructure allows users to publish, discover and search verification mechanisms such as test cases, test scripts, and reliability models, as well as to rank services and verification mechanisms. In the IBM WebSphere Reference Architecture, the system is composed using connectivity services and a service bus such as ESB. A service bus provides a communication channel for services to interact with each other. The architecture has two main categories of services connected to the ESB: business logic services, and application and control services. The business logic services provide the capabilities needed to execute the business logic as a combination among existing applications, as newly implemented services, or as connections to services provided by other systems. Business data and services interactions are control by interaction services, process services and information services. The interaction services are used to transport data and functions in proper formats and protocols to users.

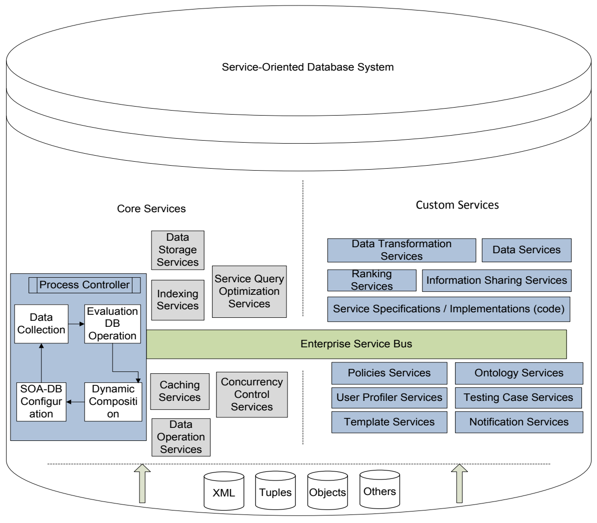


Figure 1: Service Oriented Database Architecture

Process Controller: This is an adaptive process with a feedback loop to ensure system availability and service visibility. The "Evaluation of DB Operations" service in the process controller determines if service adjustment is needed by calculating the probability of the service failure rate as well as the expected service request processing time. The evaluation of both reconfiguration criteria can be based on Bayesian analysis [25].

In an SODB, service functions are classified into two main groups: (i) core and (ii) custom services. Core services, such as caching management, service query optimization, indexing, query execution, service usage analysis and concurrency control, are vital to the brokerage system and they are the foundation units. These services are implemented locally. Custom services provide additional functionalities, and they can be customized to fit applications. Sample custom services are service ranking, application templates, and service testing and policy services.

3.1 Core Services

Data Storage Services: Information and service data can be stored in an SODB according to the user's preference. The data storage service contains query logic for retrieving data that resides locally in the SODB, in a remote database system, in data streaming arising from sensor devices or data from other web services that may contribute to the discovery process.

Caching Services: Multi-caching, in respect to service domain, location, time period and special events, provides efficient services by reducing round trips to the server.

Data Collection Services: To adjust the system performance, the SODB need to assess each service performance within the repository system. This service collects repository usage data for evaluation.

Indexing Services: Services registered in the service registry system can be indexed locally in SODB according to its domain or keywords from the service description / specification to facilitate the service discovery process.

Concurrency Control Services: This controls the concurrent access or modification to the ontology data and service data saved in SODB. An optimistic concurrency control can be used to reduce dependencies between service components and provide flexibility to alter the registered services.

Service Query Optimization: An SOA application usually involves multiple services working with each other to accomplish a task. The collaboration between service callers and receivers can be arranged to optimize performance.

3.2 Custom Services

Data Transformation Services: When sending or requesting data using service protocols, it may be necessary to transform input or output data first prior to using the service. This service provides a convenient way to transform different service data into a user-specified data format either in real time or in a batch mode. For example, XML publishing (transforming tuples to XML) can be done by stream-reading each selected data row and annotating each data value with the corresponding table column name. XML shredding (converting XML to tuples) can be accomplished by using techniques in [19]. Converting XML data to strings can be achieved by using XLST or XQuery, which requires the data service to generate an XML schema based on the selected XML file and use it to produce the stylesheet or XQuery script for extracting XML data.

Data Operation Services: These services support update/add/delete operations within SODB. Service artifacts stored in SODB may require update or change. For instance, ontology definition may need to include a new terminology in that specific domain; a service function may need to update its specification to reflect the latest change.

Policy Services: Policies specify and enforce system constraints [11] such as reliability, availability, security, performance and communications. In an SODB, policies are being used to control concurrent access to registered services, to regulate the subscription right on the published data, and to specify the service relations.

Ontology Services: These services provide a common reference for a domain as well as enables users to identify services in the domain. Service consumers can subscribe the ontology definition to create their data store [4] or use it to facilitate information exchange through semantic mapping among different data sources.

Template Services: These services explore reusable assets to facilitate the development of SOA solutions. Templates are pre-built components ready to bind to an existing application. For example, the workflow template can be used to streamline business processes within an organization by specifying the detail relations among the service functions and the application template can be used to create hierarchical application structure.

User Profiling Services: These services track service preferences (e.g., service publisher, service location, service ranking,), service usage patterns and the account information for each user. The information collected can be used to provide feedbacks for the service providers or to create collaboration agreement for service consumers and providers.

Ranking Services: Services can be ranked according to the service consumer's evaluation and service provider's creditability. Service ranking may fluctuate depending on the service subscription volume and the consumer's feedback. This service can be combined with the caching service to cache only highly ranked services in the server to increase service discovery performance.

Test Case Generation Services: These services provide a means to generate test cases for the published services based on the service specification and/or implementation. Consumers can subscribe and reuse published test cases to evaluate published, or to be published, services.

Notification Services: These services can be used as autonomous agents that continuously monitor the registered queries within an SODB. For instance, a group of users may want to subscribe certain information published in an SODB. They can register their request with this service and it will notify the consumer once the information is available. Another example, the SODB indexing service may have locked certain records in the repository system for index update. When the update is finished, it will notify other services (through the ESB) that the affected records are ready for access.

Data Services: These services provide a unified way to access heterogeneous data sources from a single access point. A query can be run against different data sources such as XML, Oracle, SQL Server, DB2, packaged applications and Web Services. In fact, numerous data services with these capabilities are available today. For example, MetaMatrix Enterprise Data Service Platform [21], Composite Application Data Service, Sybase Real-Time Data Service, Oracle's BEA AquaLogic [24, 2] and Microsoft's ADO.NET Data Service [23].

Information Sharing Services (ISSs): An ISS provides a way for different business organizations to combine information from multiple data providers into a logical form. Semantic mapping between standardized ontology and shared information is required to fracture the semantic barriers. This service can be combined with policy services to restrict different data access levels.

4. Information Sharing Service (ISS)

An ISS can be divided into three functional areas: (i) Data access profile setup, (ii) Information publishing (data mapping) and (iii) Information discovery (data querying).

Figures 2, 3 and 4 show the workflows for these areas.

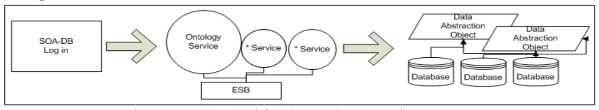


Figure 2: Identify domain ontology

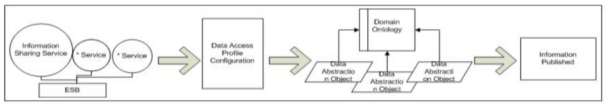


Figure 3: Publish information

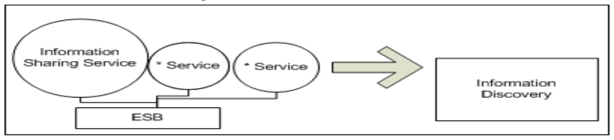


Figure 4: Discover shared information

The workflow involved four steps: (i) identify, (ii) conform, (iii) publish and (iv) discover. First, the data provider *identifies* the ontology needed. Second, the data provider creates a data service object and *conforms* their data to the selected ontology model. Third, the data provider maps the schema attributes from the data service objects to the target domain ontology and *publishes* their data on the network. Finally, users across different levels and domain areas can search and *discover* the shared information within the SODB.

1. Becoming a member. The data provider must register with the SODB before using any services provided.
2. Discover domain ontology. After login, the data provider can search for a suitable ontology through the ontology service to depict their data (Fig. 2).
3. Building the data service. The abstracted data can come from different sources, e.g., the information may be gathered from database functions, store procedures, other database systems, or even from another data services. The data service must contain data attributes that can be mapped to the selected ontology model (Fig. 2).
4. Initialize the ISS. When the abstraction layer is ready, the data provider can initiate the information sharing service to start the data sharing process.
5. Enable data access on-demand. To enable run-time discovery and subscription of the source data, the data provider must be willing to provide remote database server information (e.g., Server IP address, database name, and authentication information). This enables the ISS to access their data during runtime.
6. Mapping data elements through the web interface. The procedure for mapping the local schema attributes to the remote ontology must be done by the data publisher through the web interface provided in this framework.
7. Data publishing. Once the mapping between the source attributes and ontology has been completed, the mapping references and database access information will be stored in the ISS repository. The framework can discover and use the information published by the data source providers.
8. Data or data source modification. Data providers have full control over the mapping references stored in the information sharing service. They can add or delete mapping attributes or update the database access information to access different database server. These operations can be performed through the web interface.
9. Publish new information. As needs evolve, the associated data may change or be replaced. The data provider can reiterate steps 2 – 7 to update or add mapping references.

Note the domain ontology is reused throughout the process. It was first used during the development phase and again at runtime for data mapping. During the development phase, the data provider subscribes to the ontology, analyzes it and then determines if the ontology can be used to represent their data. If the ontology fits the publisher's needs, they can create data service objects according to the selected domain model and advance to the data mapping step. Before the data mapping process occurs, the data provider will call the same ontology service to establish direct mapping references between elements from the abstraction objects and the domain terms. After completing these processes, the information is published in the SODB and will be available for subscription. To subscribe to this published data, the ontology will be used as a query language for retrieving information in the SODB.

4.1 Domain Ontology

Ontology can be specified in XML, RDF or OWL, The proposed ISS use equivalence relations to specify relationship.

Basic Terminology and Definitions:

Let E = {e_i|e_i is the ontology that represent domain i} be ontology definitions

DSe_i be the ontology scope of the ontology e_i

And R = {parent-of (Rp), sibling-of(Rs), child-of (Rc)} be possible relationships among ontology systems.

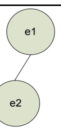
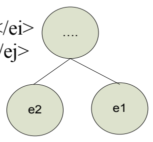
Where Rp , Rc , and Rs relationships are defined as follows

$DSe_i < DSe_j \Rightarrow e_i \text{ Rp } e_j$ and $e_j \text{ Rc } e_i$

$e_i \text{ Rp } e_j$ and $e_i \text{ Rp } e_k \Rightarrow e_i \text{ Rs } e_k$

The above relationship between domains and domain's attributes within the ontology are defined through the XML segment hierarchy as shown in table 3.

Table 3: *Ontology Domain and Its Attribute's Relationship*

Node(s)	Sample structure	Node Relationship 1	Node Relationship 2
$E_i < E_j$ (Parent of)	`<ei>` `<ej></ej>` `<ei>` 	Ej is a domain attribute/attribute value of ei	N/A
Ei = Ej (sibling of)	`<…>` `<ei> </ei>` `<ej> </ej>` `</…>` 	ei and ej represent two separate sub domain area within a domain	ei and ej can be sub-domain or domain attribute/attribute values under another common domain

The summary of the domain concept is captured at the root level element. All non-root elements can represent four entities: (i) domain attributes, (ii) sub-domains, (iii) sub domain attributes or (iv) attribute value type. Suppose the depth of the domain ontology model is greater than one and the node level for the root element also starts at one. Assume that there exist two nodes, e_i and e_j, in the ontology file. If ei is smaller than ej, then it is said that e1 is a parent of e_j. In this scenario, e_j can be a domain or sub-domain attribute of e_i, or e_j is a specific attribute value type that belongs to ei. In another scenario, if e_i and e_j are at the same node level, which is often referred as siblings, then e_i and e_j can be sub-domains that reside under the same domain, or they can be the attributes under the same sub-domain, or domain or they can be the attribute type under a specific domain attribute. The node relationship is summarized in Table 3.

Domain Correlations: The domain correlations have been defined using the equivalence relation. A binary relation between two entities in ontology can be grouped together as being equivalent.

Reflexive relation*:*

$e_i \sim e_i \ \forall \ | e_i \in E$ where ~ denote the reflexive relationship

Symmetric relation*:*

$e_i, e_j \in E$ and $Rs \in R$

$e_i \text{ Rs } e_j \Rightarrow e_j \text{ Rs } e_i$

Transitive relation*:*

$e_i, e_j, e_k \in E$ and $Rp \in R$

$e_i \text{ Rp } e_j$ and $e_j \text{ Rp } e_k \Rightarrow e_i \text{ Rp } e_k$

Fig. 5 illustrates these equivalence relations for AHCCCS. Here, each node in is labeled using its domain ontology name.

Symmetric: The domain "AZ State Eligibility Programs" umbrellas multiple eligibility programs, one of which is the state "Healthcare Eligibility". One can say the Healthcare eligibility program / domain is housed under the State's eligibility program domain establishing parent-of (Rp) relationship. Also, the "Demographics" or "Eligibility_Information" segments under "Healthcare Eligibility" are indirectly relates to the "Unemployment Benefit" domain ontology. So, "Healthcare Eligibility" and "Unemployment Benefit" ontology systems show *symmetric* relation.

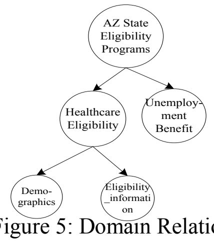


Figure 5: Domain Relations (Sample)

Transitive: The "Demographics" domain segment is listed under "Healthcare Eligibility", which in turns is also nested under the "AZ State Eligibility Programs" domain. This implies accessing domain information in "Demographics" is also possible under the State's eligibility program domain.

4.2 Information Sharing Architecture

Fig. 6 depicts the information sharing repository structure. To make the domain ontology useful for data mapping, the domain model must be parsed and saved in the ISS repository. In our prototype we have used SQL SERVER 2008 to implement the ontology. Here, the combination of *Ontology, Ontology_Main* and *Ontology_Detail* tables represent the domain ontology in a relational form. The *Ontology* table stores the ontology name and number of sections. *Ontology_Main* stores the selected ontology structure (as shown in figure 6and *Ontology_Detail* stores the enumerated values for a specific domain element.

The data mapping procedure involves three relational tables (*MappingReference_Main, MappingReference_Detail*, and *MappingReference_DataValue)* which will be shared among different data providers for various domain ontology mapping use. Finally, as there will be at least one data access profile associated with each of the data providers, the ISSrepository cooperates with User Profiling Service to maintain the data access information for each providers using three tables (*Business, Business_Detail, and Business_Connection)*. *Business* table stores each business entity's ID which is assigned upon registering their name with the SODB. *Business_Detail* stores information about the business organization and the *Business_Connection* table is used to store the enterprise database connection information.

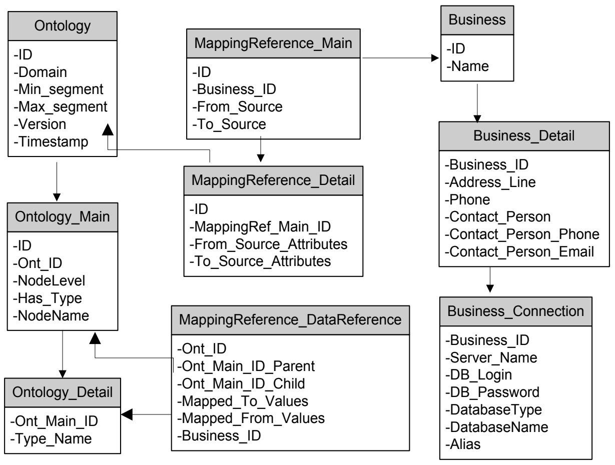


Figure 6: The Information Sharing Repository structure

4.3 Data Querying

Using ontology as a query language is not new, e.g., OntoDB [4, 6] use ontology to create database schemas as well as a query method to access information stored. This paper adopts a semantic-based query approach to access the shared data resources. The chosen querying method requires the data consumer to identify the information domain (e.g., domain ontology) and the associated domain attributes to compose the data query. The semantic-based querying method has an advantage over other querying mechanisms (e.g., keyword-based) as it can effectively capture the semantic definition for the published information.

Query Syntax. The data query mechanism consists of two main parts: (i) projection and (ii) evaluation. Projection (the "select") is used to project domain elements in the result data set. Evaluation (the "where") is used to filter information that should be included in or excluded from the result data set. A simple data query consists of one or more sub domain elements followed by their associated attribute names. Each sub domain and their attribute name pair is connected by a single dot ("."). A comma (",") is used to distinguish two separate sub domain fields. All elements in the data query are case insensitive. The query syntax for the *projection* is almost as same as the *evaluation*. The only different between the two is that statements appearing in the *evaluation* always include an operator followed by a specific value. Currently, the evaluation process can support *equal*, *not equal*, *less than*, and *greater than* operators.

If the selected ontology structure is known in advance, data consumers can directly execute the querying service in their application program. Otherwise, they can utilize the web interface tool to compose the data query to access the shared information. When using the data query web page, the data consumer can select the domain area and the data providers they are interested in for information retrieval. As it is vital for data consumers to use the right domain terminology to access the shared information, the content of the selected domain model can be dynamically displayed on the interface as a look-up service to assist in the data query composition process.

Dynamic Query Composition: The ISS can dynamically compose query statements based on selected domain attributes and selection criteria such as filter conditions specified by the consumers and the mapping references given by the data providers upon request. Dynamic query composition algorithm is shown in Fig. 7. A database query can be divided into three parts: *select*, *from* and *where* clauses. The *select* clause is built based on the projection statements whereas the *where* clause is assembled according to the evaluation statements. The *from* clause is composed using the unique combination of projection and evaluation sub-domain elements (Fig. 7). Each sub-domain and its related components (such as attributes, operators, and domain values) are parsed and stored in a customized data structure object (P_List and E_List in Fig. 7) according to its sub-domain type.

After parsing and storing the data query elements in the customized data structure, each domain components will be inserted into a temporary table within the information sharing repository for mapping translation. The mapping translation process accepts the data provider's business ID as a parameter to retrieve the corresponding mapping reference that was given by the data provider. If the mapping reference exists in the information sharing repository, then it will be returned to the service caller. Otherwise, the original data query will not be translated. As data services can be added / modified dynamically, the dynamic query composition techniques provides flexible and "on-the-fly" data integration based on available data sources.

Dynamic Query Composition:

```
Parse Projection Query
Parse Evaluation Query

For each selected Data Provider source
{
        For each Projection attributes in a selected Sub Domain
        {
                Append [original source].[original attribute name] to the SELECT clause
                Record each distinct mapping source (ex: table name, view name) in the P_List
        }
        For each Evaluation attribute in a selected Sub Domain
        {
                Append [original source].[original attribute name] [operator][value] to the AND clause
                Record each distinct mapping source (ex: table name, view name) in the E_List
        }

        // build FROM clause and WHERE clause
        If (P_List.Count = 1 and E.List.Count = 0)
        {
                FROM Clause    + = P_LIST's mapping source
                WHERE Clause  +=  ""
        }
        Else if ( P_List.Count = 1 and E_List.Count > 0)
        {
                FROM Clause   += P_List's mapping source
                For each source Item in E_List
                {
                  If the mapping source = P_List's mapping source, ignore it
                  Else,
                        // each sub domain can be joined using the segment ids
                        1) FROM Clause    + = E_LIST's mapping source
                        2) WHERE Clause   += prev_E_List's mapping source segment Id =
                                             curr_E_List's mapping source segment Id
                }
        }
        Else
        {
                For each p in P_List-1's mapping source
                {
                        If  E_List contain (p), remove p from E_List
                        FROM Clause     += P_List's mapping source
                        WHERE Clause   += Curr_P_List's mapping source segment Id =
                                          Next_P_List's mapping source segment Id
                }
                For each e in E_List-1's mapping source
                {
                        FROM Clause     += E_List's mapping source
                        WHERE Clause   += Curr_E_List's mapping source segment Id =
                                          Next_E_List's mapping source segment Id
                }

        }
        Entire Query = SELECT clause + From clause + Where Clause + And Clause

        Send Query to the data provider source

        Add Result to Data Set

        }

        Return Data Set
```

Figure 7: Dynamic Query Composition Algorithm

5. Case Study: Arizona Healthcare Cost Containment System

We have modeled and implemented the AHCCCS to illustrate the ISS.

Domain Ontology: Sample healthcare eligibility domain ontology is shown in Fig. 8. The top most element of the ontology denotes the business domain area "HealthCare - Eligibility". The "HealthCareEligibility" comprises of three sub-domain categories "Demographics", "Eligibility_ Information", and "Income_Information". Each of the sub-domains represents a distinctive unit within the domain ontology. Sub-domain segments are different from each other by the definition of their attributes. For example, the "Demographics" sub-domain segment contains all of the general information of a person record (such as SSN, Name, DOB), whereas the "Eligibility_Information" segment only contains the healthcare application related information (ex: Status, Program Name, etc) for an applicant. A finer level of sub-domain definition may be found through attributes with specific value types.

For example, "Race", under "Demographics", is used to categorize race type like American Indian, Asian or Caucasia, and "Status", under "Eligibility_Information", is used to classify current eligibility status like Approved, Denied or Discontinued. The relations mentioned earlier in the chapter can be applied to the sample ontology as follows:

- In the reflexive case, each node in the XML graph is related to itself.
- In the symmetric case, "Demographics" and "HealthCare Eligibility", "Status" and "Eligibility_Information", "ALTCS" and "MCS", and "Applicant" and "Role_Type", etc, have either a symmetrical of (i) parent of, (ii) child of, or (iii) sibling of relationships.
- In the transitive case, all leaf and intermediate nodes in the sample ontology are qualified in the transitive relationship. For example, transitive relations for "DOB" and "Demographics" and "Demographics" and "HealthCare Eligibility" is "DOB" and "HealthCareEligibility".

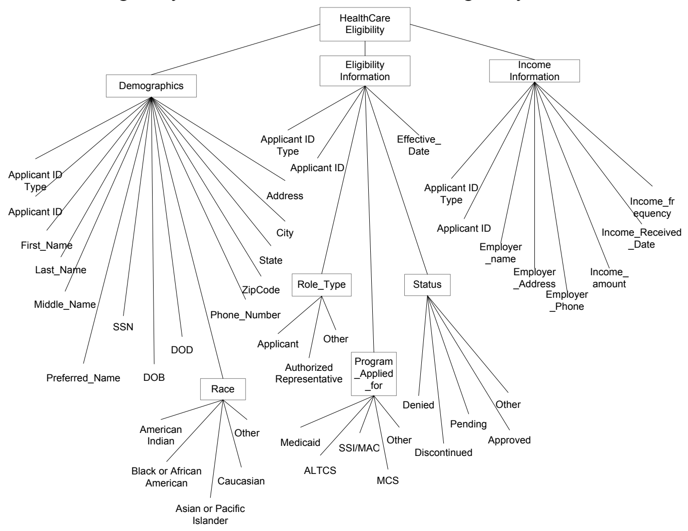
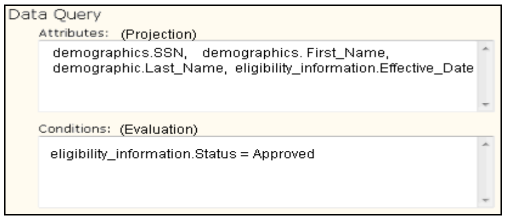


Figure 8: Health case domain ontology- tree view


Data Query

Attributes: (Projection)

demographics.SSN, demographics. First_Name, demographic.Last_Name, eligibility_information.Effective_Date

Conditions: (Evaluation)

eligibility_information.Status = Approved

Figure 9 Result of the Query Example

Query Access Demonstration. To demonstrate how to compose a data query to access the published data, the HealthCareEligibility model will be used in this demonstration.

Exampl : Suppose government agency G1 wants to obtain the SSN, Person Name, and Eligibility Effective Date of the currently active applicant information published by another agency G2. The data query should be assembled as shown in Fig. 9.

6. Performance Evaluation

To test the effectiveness of the proposed ISS discussed in the preceding sections, service performance and V&V (verification and validation) analyses have been performed to compare different service calls with respect to different data query requests, number of concurrent users and the correctness of the data result. The ISS is written in C# with ASP.NET 3.5 using Microsoft Visual Studio 2008 Professional Edition. All experiments except the simulation program were run on 2.0 GHz Intel Core 2 Duo processor with 2 GB memory and 4MB cache under Windows Vista operating system. The ISS is hosted under COX communication residential network with the maximum of 300 megabytes of traffic. To mock an actual data sharing environment, the supporting database system (SQL SERVER 2008) for the information sharing service has added multiples databases, e.g., AHC_Database, ASU_Databases, Business001_Databases, Business002_Databases to represent different data providers' sources. Testing data used in the experiments are provided by the State of Arizona training unit and is randomly distributed into one of these databases.

6.1 Service Response Time

Manual Testing (Fig. 10): The data query performance result falls within the expected time range in both manual querying and automated service testing. As the connection speed and the bandwidth may fluctuate during the day, the slowness of the internet is still the biggest barrier of the service testing. Although all experiments were carried out in non-peak hours of the day, the server response time and page load delay contributed to the internet connection speed. According to the log information within the proposed service, the query composition process was relatively fast (0.1 seconds for data source reference look-up) but when the query returned from the service back to the web interface, the page post back took almost 10 to 20 seconds to the load and display the result set on the web page.

Automated Service Testing: A multi-threaded simulation program was written to generate 500 (A), 1000 (B), 2500 (C) and 5000 (D) data queries to concurrently request data published within the SODB. To mimic the distributed environment, the simulation program was purposely run on a remote desktop that makes remote service call to the service hosting machine. To maximize the run time for each query request, the simulation program was configured to send out various queries in an alternative manner.

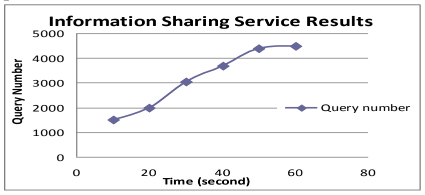
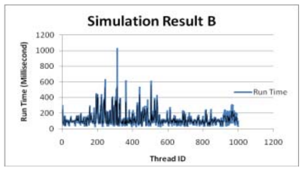


Figure 10: Query Performance Result-Manual Processing



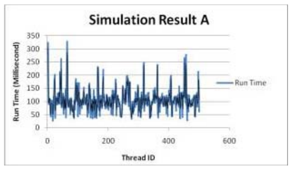
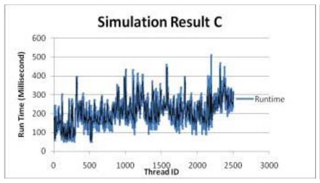


Figure 11: Simulation result with 1000 threads

Figure 12: Simulation result with 500 threads

The service response time (with 500 and 1000 threads) consistently fall in between 0.1 to 0.15 second for each request. According to the web service response time benchmark from WAPT [37], it is an ideal response time (Fig. 11, 12). In addition, to ensure the proposed ISS is good for its intended use, the various validation tests have been performed to show the proposed service meets its requirements



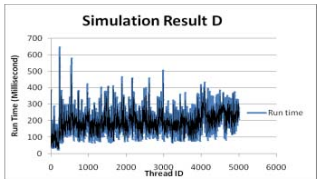


Figure 13: Simulation result with result with 2500 threads

Figure 14: Simulation 5000 threads

However, the service response time for scenario C and D (with 2500 and 5000 threads) degraded rapidly after the first 200 query requests. The response time for the subsequent requests takes as much as 0.35 to 0.6 second to complete. The slowness of the service response for simulation results was a direct result of the memory and CPU limitation of the computer running the simulation program. Fig. 15 shows the average and stand deviation of automated service testing.

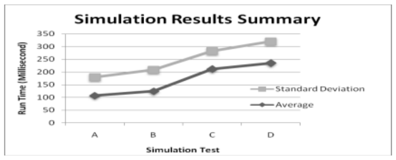


Figure 15: Query Performance Result – Simulation

7. Conclusions

This paper proposed an SODB system as a federated collection of services that facilitate information sharing. The AHCCCS case study implementation demonstrates the feasibility of the SODB . Currently, data mapping from the source to the domain ontology is done by the data service provider. In the future we plan to investigate how to employ the newly proposed probabilistic mapping method [14] to reduce the mapping cost.

8. References

[1] Bianchini, D., De Antonellis, V., Melchiori, M. (2003). Domain ontologies for knowledge sharing and service composition in virtual districts. *Proceedings of the 14thInternational Workshop on Database and Expert Systems Applications*. pp. 589-605.

[2] Campbell, D. (2005). Service Oriented Database Architecture App Server-Lite?. *Proceedings of the 2005 ACM SIGMOD international conference on Management of Data.* Pages 857-862.

[3] Dan, A., Johnson R., Arsanjani, A.(2007). Information as a Service: Modeling and Realization, *Systems Development in SOA Environments*, ICSE Workshops May 2007.

[4] Fankam, C., (2008). OntoDB2: support of multiple ontology models within ontology based database. *ACM International Conference Proceeding Series*. Volume 326, pp.21-27.

[5] Genesereth, M., Keller, A., Duschka, O. (1997). Infomaster: an information integration system. *Proceedings of 1997 ACM SIGMOD Conference*. pp. 539-542.

[6] Jean, S., Dehainsala, H., Xuan, D.N., Pierra, G., Bellatreche, L., (2007). OntoDB: It is time to embed your domain ontology in your database. *Lecture Notes in Computer Science. Springer Berlin / Heidelberg*. Volume 4442/2008. pp. 1119-1122.

[7] Jiang, H., Ho, H., Popa, L., Han, W.S. (2007). Mapping-driven XML transformation. *Proceedings of the 16th international conference on World Wide Web*. pp. 1063-1072.

[8] Rakesh, A., Dmitri, A., Ramakrishnan, S. (2004). Enabling sovereign information sharing using web services. *Proceedings of the 2004 ACM SIGMOD international conference on Management of Data*, pp. 873-877.

[9]Tok, W.T., Bressan, S. (2006). DBNet: A Service-oriented database architecture, *Proceedings of the 17th International Conference on Database and Expert Systems Applications*. Pages 727-731.

[10]Tsai, W.T., Fan, C., Chen, Y., Paul, A.R., Chung, J.Y. (2006). Architecture classification for SOA-based applications. *Object and Component-Oriented Real-Time Distributed Computing*, Volume 9, pp.295-302.

[11]Tsai, W.T., Liu, X.X., Chen, Y. (2005). Distributed policy specification and enforcement in service-oriented business systems. Proceedings of the IEEE international conference on e-business engineering. pp. 10-17.

[12]Tsai, W.T., Zhang, D., Chen, Y., Huang, H., Paul, R.A., Liao, N., (2004). A software reliability model or web services. *Conference on Software Engineering and Applications*, Volume 436, pp.144-149.

[13]Tsai, W.T., Zhou, X.Y., Chen, Y., Xiao, B., Paul, A.R., Chu, W.(2007). Roadmap to a full service broker in service-oriented architecture. *Proceedings of the IEEE International Conference on e-Business Engineering*, ICEBE 2007, pp.657-660.

[14] Tsai, W.T., Zhou, X.Y., Wei, X., (2008). A policy enforcement framework for verification and control of service collaboration. *Information System E-Business Management 6*, Volume 1, pp83-107.

[15]Uschold, M., Jasper, R. (1993). A framework for understanding and classifying ontology applications. *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods*. Volume 18, pp.11:1-11:12.

[16] Amazon SimpleDB™- Limited Beta. (2008). http://aws.amazon.com/ simpledb/

[17] Cardwell, R. (2006). Government information sharing and interoperability via data services and SOA. www.xml.gov/presentations/metamatrix/ dataservices.ppt

[18] Department of Defense, Net-Centric Data Strategy. (2003). http://www. defenselink.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf

[19]Gabhart, K. (2007). Data sources as Web Services http://www.xml.com /pub/a/2007/10/25/data-sources-as-web-services.html

[20]IBM's WebSphere. (2008) http://www-01.ibm.com/software/websphere/

[21] MetaMatrix, Inc. (2008). http://www.redhat.com/metamatrix/

[22] Microsoft .NET Data Service (2008). http://www.microsoft.com/NET/ http://www.microsoft.com/windowsserver2003/technologies/idm/uddi/

[23] Campbell D., Service Oriented Database Architecture: App Server-Lite?, TechReport, MSR-TR-2005-129 , Microsoft Research, ACM, Sep 2005.http://www.microsoft.com/sqlserver/2005/en/us/white-papers.aspx

[24]Oracle.(2008).http://searchsoa.techtarget.com/news/interview/0,,sid26_gci12 97968,00.html

[25]Seaman, S.R., Richardson, S. (2001). Bayesian analysis of case-control studies with categorical covariates. Biometrika, 88(4). pp. 10073-10088.

[26]Srivastava, U., Munagala, K., Widom, J., Motwani, R. (2006). Query optimization over web services. *Proceedings of the 32nd international conference on Very large data bases*, pp.225-366.

[27]The InfoSleuth Agent System. (2005). http://www.argreenhouse.com/ InfoSleuth

[28]Web Application Testing (WAPT) version 5.0. (2008). http://www.loadtestingtool.com/help/response-time.shtm.

[29]WS02. The open source SOA company. (2008) http://wso2.com/services/ consultancy/

[30]BEA WebLogic Server UDDI Registry (2003). http://www.bea.com/ framework.jsp

[31] Novell Nsure UDDI v3 (2007), http://developer.novell.com/uddi/

[32]Solving Cloud Data Integration: A Key Challenge for SAAS Providers, White paper, by Informatica. http://www.ebizq.net/white_papers/ 11162.html?related)

[33] An overview of cloud computing at Yahoo, Raghu Ramakrishnan , http:// us.apachecon.com/page_attachments/0000/0194/ ApacheCon-09cloud.ppt

[34] Cooper B., Ramakrishnan R., Srivastava U., Silberstein A., Bohannon P., Jacobsen H., Puz N., Weaver D., Yerneni R., PNUTS: Yahoo's! Hosted Data Serving Platform, (VLDB 2008)

[35]Seeding the Clouds: Key Infrastructure Elements for Cloud Computing, IBM white paper, Feb 2009 (ftp://ftp.software.ibm.com/common/ssi/sa/wh/n /oiw03022usen /OIW03022USEN.PDF

[36] Vouk Mladen A., Cloud Computing – Issues, Research and Implementations, Journal of Computing and Information Technology - CIT 16, 2008, 4, 235–246.