Vega Information Grid: A Suite of Toolkit for Building Information Sharing Scenario

Xiaolin Li¹, Wenlong Huang², Bo Yan²

¹Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
²Graduate School of the Chinese Academy of Sciences, Beijing 100080, China
lxl@ict.ac.cn, {huangwenlong, yanbo}@software.ict.ac.cn

Abstract. There are many common challenges to integrate and share information resources in enterprise or industrial environment, for example, heterogeneous structure of data sources, dynamical changing of source location and schema, sharing information across multiple autonomous administrative domains, fine-grained access control, and personalized requirements. Although it is possible to build a specialized solution to meet any scenario requirements, the solution, which is very suitable to the specialized scenario, may not be available to other scenarios. Based on Web Services technology, Vega Information Grid (VIG) abstracts commonality of requirements and addresses these issues by information virtualization. A suite of toolkit, which provides functionalities suitable for different scenarios, enables enterprise users to acquire and deploy information cost-effectively in a virtual grid platform, and to benefit greatly form the decrease of the full life cycle time and the cost of management and changes of sharing information. This paper introduces its information virtualization model based relation schema, and decoupled architecture supporting personalized utilization pattern.

Keywords: information grid, information virtualization, decoupled architecture, information sharing.

1 Introduction

Nowadays, enterprise’s IT group is being forced to find new and cost effective means to obtain and share the information from multiple information store sources in a wide-open network environment in order to reacting to changes in competition, market dynamics, and regulatory mandates[1], [2]. There are many critical problems for the motivation.

Heterogeneity. In most enterprises, there are a vast number of different information sources, including databases, text files, images, videos, directories, and so on. Unfortunately, the number of formats in which data is stored is as varied as the types of information stores themselves. In order for an application to deal with the variety of information sources as well as their associated means of exposing information, an application developer is forced to code specifically to each source.
Dynamical Changes. As each information source evolves and new information sources become available, the application must be built again or modified to adapt specifically to the changing of source location, data format, authorization and exposure mechanism of the information source. Although SQL provides a more general purpose mechanism through which to obtain data stored in databases, its adoption as the means to obtain information from other information sources, such as file and service, never came about.

Sharing across multiple domains. Sharing information is often required dynamically between business co-partners, user identity authentication and user authorization is necessary to keep information security. Enterprises now consume information in many different formats and from many different sources in the pattern of autonomous control. As a result, it is difficult to store in advance all forms of information in a single form of information repository, as what data warehouse did.

Personalization. With their requirements changing, users would not been bound to a consolidate information source and application. They need to share and control full life cycle of their own experiences, including deploying resource, obtaining information, managing resource and authorization, personalized profile configuration.

Problems above contain some of commonalities of requirements of information system. In order to meet these commonalities, Vega Information Grid team focuses research on these issues and has developed a suite of toolkit for building information sharing application scenario.

This paper is organized as follow: section 2 introduces some related works. Section 3 describes an information virtualization model which mainly aims to resolve the heterogeneity and dynamically changing of data source. Section 4 presents the key concepts from the perspective of utilization pattern and the decoupled architecture of the VIG platform, which enables all those components to run as an integrated system. An implementation and evaluation is presented in section 5. The last is the conclusion.

2 Related Work

In order to solve the integration challenges faced within many enterprises, commercial and research efforts have provided some comprehensive solutions spanning Web Services [3], [4] and information integration technologies [5].

OGSA-DAI[6] focuses on the issues surrounding the design of an efficient, generic, service-based interface to databases. Most previous work on interfaces has been language dependent and not service based (e.g. JDBC). Higher-level Grid data management systems, such as SRB [7] or Chimera [8], could use OGSA-DAI either to access structured data resources, or to provide access to their own metadata. European Data Grid has developed Spitfire [9], a Web Service interface to relational databases for metadata management, which allows a client to query a relational database over GSI-enabled HTTP(S). An XML-based protocol is used to represent the query, and its result. IBM Corporation provides solutions to enable distributed enterprise integration with WebSphere and the back-end DB2 Information Integrator (II). SkyQuery [10] applies the classical wrapper-mediator architecture in a service-based setting, supports querying autonomous astronomy archives over collections of
Web Services. It deploys WSs at each database store for handling metadata, performing queries, and cross matching partial results.

These solutions and technologies emphasize focus on the data source integration from the perspective of business requirements, while changing agilely for the personalization and autonomous control from the perspective of utilization pattern are often in the second place.

3 Information Virtualization

For such issues of information source as: location transparency in accessing, heterogeneous structure and schema, integrating multiple sources into a virtual view, different data format and uniform access interface, layered virtualization technique and approach are often adopted to figure out these issues.

From the point of view of user utilization, SQL provides a friendly interactive query interface although its adoption as the means can not obtain information from sources other than databases. What is the most important is that relational databases are the most popular storage of information sources, and majority of user of information system are familiar with the utilization of SQL query [11], [12].

So, Vega information grid virtualizes source by means of 3-layered architecture model (Fig. 1) [13]. The three layers, named physical relation, virtual relation and
effective relation, are respectively represented with relation schema and mapping to the lower layer relation schema. Physical relation stands for the concrete physical sources in the network which have been described with a relation-based schema, such as databases, files and directories, and services. Virtual relation is the virtualized information view which is built based on a physical relation or other virtual relations according to the business requirements of enterprise or user community [14]. While, effective relation denotes the personalized user requirements, which is built based on a virtualized relation by the user self.

The distinguished feature of the information virtualization model is that each layer would not impose heterogeneity of metadata and interface on higher layer, since each is represented with relation schema. Therefore, application code developed based on any layer can run smoothly on other layers where the requested relation schema is also available. Follow segments discuss the functionalities and features of each layer relation schema.

3.1 Physical Relation

Based on the Service-Oriented Architecture, one or multiple data sources are encapsulated as a service which serves as the container of sources. The data sources of the service are exposed to grid user by means of accessing the service. As a result, community user can browse and access data. Single data source of the service, such as database table, view, file or directory, is virtualized and represented as a physical relation of Vega information grid by describing with a relation schema. A data service can serve as the access interface for a pool of physical relations.

The key futures of virtualization in physical relation layer include:

- Data source providers can deploy or destroy autonomously any data source to the pool of a data service without any modification to the service.
- Providers can decide which table, data sets or view of data source to be shared to other grid users, since relation schema is very convenient to describe the requirement.
- Heterogeneity of data source and location transparency can be addressed transparently to higher layer by developing and deploying wrapper for the data source in the layer.

However, for the higher layer, physical relation schema is exposed instead of the data service, as what OGSA-DAI did. So the data service information, which should be implied in the description of physical relation, is necessary to access data source. The mechanism makes it easier for users to find the desired data source.

As a whole, the metadata schema of a physical relation includes two parts: the relation data schema and its data service.

Undeniably, there is an important issue needed to be resolved about how to virtualize other kinds of data source as relations, such as services or software components.
### 3.2 Virtual Relation

Virtual relation is built dynamically according to the community business requirements and schema, it is an integrated relation based on a physical relation or other virtual relations. In contrast, the definition of physical relation is from the point of view of resource provider.

There are two kinds of virtual relation: basic virtual relation and composed virtual relation. Correspondingly, two kinds of schema mapping operation exist, named reference to virtual relations and mapping to physical relation. Basic virtual relation can only map to no more than one physical relation. Composed virtual relation is built by referencing other shared virtual relation sets, including other composed virtual relations, whenever loop reference among relations should be escaped. Consequently, data sources can be integrated and fused by constructing composed virtual relations based on basic virtual relation sets, each element of which in turn maps to a shared physical relation in the network.

Distinguishing the two map operations is valuable for the user who is building business model. Mapping concerns only schema matching without any awareness of business requirements. But reference definition should be aware of business model, in which the sharing relationship and data flow of computing among virtual relations based on the schema mapping are interpreted. For example, the sum() functionality of SQL denotes computing of data flow of source based on schema definition.

The key futures of virtualization in virtual relation layer include:

- Users can build dynamically business relation data model based on relation schema. Business data schema, data format can be adapted to business requirements by changing reference to schema without any modification to application code.
- Integration of data sources can be achieved based on homogenous virtual relation schema. Integrator concerns only business data model without any awareness of heterogeneous characteristics of data source.
- Builder of virtual relation can decide sharing of virtual relation among grid virtual communities. Therefore, specifications of business data model can be shared resulting in decreasing deploy time cost of application scenario.

Generally speaking, the metadata schema of describing virtual relation includes its relation schema, mapping or reference, sharing and authorization.

### 3.3 Effective Relation

Since virtual relation is the abstraction of data model of business requirements, all community users can access and acquire consistent information view by sharing virtual relations. However, considering personal requirement, individual user may need personalized information view even if accessing the same data model. Effective relation is the personalized relation built by individual user based on a virtual relation. And it can be shared by other users too.

For example, virtual relation v provides an integrated information view of all department employees of an enterprise, however, the manager of research and
development department may only care for staff employed in the department. An effective relation built based on v can meet the manager’s requirement.

The key features of virtualization in effective relation layer include:

- Effective relation can be deployed only once and can be accessed many times.
- Effective relation enables individual user to concern only personalized requirements not needing to construct the data model.
- Effective relations which were created by individual user consist of the user’s personal workspace, but it is not needed to allocate storage for individual user to store personal data. Security of personal data can be guaranteed by access control autonomously.

The metadata schema of describing effective relation consists of its relation schema, mapping to virtual relation schema and sharing configuration.

3.4 Consistency Issue

As the schema space of relational database, virtual schema space in the virtualization model also supports constraints such as primary key, foreign key, null and unique check, and meets 3-NF constraint too. For example, if a virtual relation is the union view of multiple physical relations which in turn map to multiple data sources, primary key constraint of an attribute should keep each value of the attribute’s union result set from multiple data sources both not null and unique.

An important issue of virtualization model deals with how to keep consistency of changing relation schema, when changing or deleting an attribute of virtual schema, which does not require an overhaul of the entire schema space in pursuit of low overhead and autonomous control feature. It also deals with when and how to provide user information when exception happens in processing mapping or reference.

If attribute $A_1$ of virtual relation $V_1$ references to attribute $A_2$ of virtual relation $V_2$, and the $A_2$ is deleted, the $A_1$ would keep unchanged. During the process of parsing schema reference, the reference objective of $A_1$ should not be available. If $A_1$ has been set primary key or unique constraints, an exception should be thrown to user because the situation can not meet the requirement definition of the schema. In other cases, the default value of $A_1$ or an empty value should be returned for the attribute $A_1$ without any exception.

4 Decoupled Architecture

We defined and implemented a decoupled architecture of VIG (Fig. 2). From the perspective of organization and utilization pattern of full life cycle of the system, the architecture employs three abstractions: community, channel, and Micro-Session.

A community is a realization of the administrative domain. As a persistent construct, it maintains two sets of members, the members of users and the members of effective and virtual relations which community members have created. In addition, it maintains context and access control policies shared by the members.

For the end user, the architecture defines channel as the abstraction of user interaction with Vega Information system. Channel is a persistent construct too.
User’s requests, such as querying data, writing data to the data source, or accessing a web page, are abstracted as one or several of interaction events with a channel handle on runtime. End user can create and deploy channels in his/her individual virtual space and can share them to the other members within the same community or that of other communities.

Micro-Session is a dynamical construct on runtime, whose original intention is to improve performance and portability by sharing and supporting context of which developer did have to write codes to take care before. A Micro-Session maintains the life cycle of the context on which the request is processed. The context includes three parts. Subject part corresponds to user identity and certificate, allocated role. Object part maintains the lists of runtime instances of resources (e.g. effective or virtual relation) which have been bound to the Micro-Session for executing application logic of the request. Status part corresponds to the temporary data structure and status of processing the request of the session.

Fig. 2. Decoupled Architecture of Vega Information Grid.

When catching a request which is represented by a channel handle and corresponding event in a session cycle, the system performs in advance a parsing of channel handle and event, then checks whether the corresponding Micro-Session identified by assembling session id and channel handle has been created in the Session Container which serves as the runtime container of Micro-Session. If not, a new Micro-Session should be created to support the life cycle of the processing, and a unique identifier should be allocated to it. Otherwise, the created Micro-Session is scheduled again by the session container to support processing the current request.
The Micro-Session puts the request to the virtual relation execution engine which executes the request immediately based on current context of the Micro-Session. The engine consists of five important components. Query parser parses the request to obtain which effective relation is queried, and checks the validity of the request’s input parameters, then maps effective relation to virtual relation. If the virtual relation has then not been bound to the Micro-Session, query integrator would bind it to create an instance of the virtual relation descriptor by accessing virtual relation schema from cache, then do reference parsing to acquire referenced sub virtual relation set and bind them recursively. A virtual relation descriptor is a dynamical combo of schema and context of current Micro-Session (e.g. user identity and active role). It should be destroyed when the Micro-Session dies or expires. Subsequently, based on these instances of descriptor, integrator decomposes the query to generate sub queries which get executed on physical data sources, and stores these middle results in cache database. Finally, query integrator executes composed query based on these results, and returns final result.

For querying and writing data to data source, access control and constraints check are necessary. Access control can provide fine-grained control on relation cell of data source based on RBAC model. Constraints check can keep data consistent and avoid redundancy of data in a virtual multiple data source environment.

An execution of request may access one or multiple physical relations. Data service proxy is responsible for binding and invoking corresponding data service deployed in a Tomcat/Axis host environment. The identifier or address of data service is available in the physical relation schema.

5 Implementation and Evaluation

The virtualization model and architecture presented above have been implemented and deployed respectively in a test-bed and production mode platform in Institute of Computing Technology, CAS (ICT, CAS) [14]. Some of departments, project teams, or virtual laboratories in ICT compose the communities of the production platform. Each community is a sub information grid with multiple users allocated one or multiple role by the community administrator. It needs to integrate and share isolated applications, databases and files to provide user a single system image and agility of changing requirements of information fusion.

Since information exchanges often occur across multiple communities, it is needed to interconnect communities as an integrated information grid, in which the accessing transparently to the other community resource is allowed if resource provider has shared the resource to the community.

Some evaluation works have been done on the test-bed server, which is equipped with two 1800MHZ AMD Athlon CPUs, 1000M memory, and running tomcat4.0.6 container with Axis engine1.0, and mysql4.0.26 database.

Fig. 3 shows the evaluation result of concurrent performance. The primary concern is that the multiple layers of virtualization may increase the overload of processing a request. The result shows that Micro-Session and instance of virtual relation descriptor can maintain and share context and current status, most of which would be
initialized when processing request for the first time during a session. Consequently, this decreases evenly the overload if request for the same data source was sent continually, e.g. browsing a series of information pages, in contrast to the situation without the support of these runtime construct. Correspondingly, Fig. 4 shows the relationship of throughput with the number of concurrent requests.

In order to confirm the agility of deploying data source, the life cycle of 100 data sources deployed in the test-bed has been monitored. Fig. 5 shows that average time of user interaction is around 1.8 minutes when creating, registering, or changing a virtual relation schema by a web-based toolkit.

Fig. 3. Mean time of response(second) increases evenly with increase of concurrent user requests number.

Fig. 4. Relationship of throughput with concurrent user requests in con.
6 Conclusion

Heterogeneity, agility of changing and personalization are primary issues to integrate and share information sources across multiple administrative domains in a distributed environment. Vega Information Grid addresses these issues with information virtualization based relation schema and a loosely coupled architecture, and provides a suite of toolkit for building information system to meet commonalities of these requirements. Preliminary evaluations show that this is a feasible approach with taking care of both performance and agility of changing.

References