Implementation of Data Integration using Distributed Systems: A Review

Rakesh Jaitly

Dept. of Management Studied & Computer Applications
Amritsar College of Engineering & Technology, Amritsar, Indiarakesh_ri1@yahoo.com

Abstract

Data integration in the distributed data system is introduced to solve the problem that data model has. The data integration in the distributional systems can be supported effectively. Data conversion is still a challenge in distributed system integration. Community based system is used for distributed data integration. It comprises of three elements: community, data model and communication protocol. The integration system solves the data heterogeneous problem in production management, making users check data more transparently and conveniently. The construction of central database is a comprehensive giant engineering system, which directly serves the demands of various application subsystem developments.

Keywords: Distributed Database, Data Integration, Data Model, Data Virtualization, VDM, heterogeneous resources.

1. Introduction

Data integration and information exchangeable systems request a uniform interface that allows transfer and process share data across multiple distributed systems. Current data integration systems are built upon the traditional database systems in which queries are specified in a structured form and data is modeled in one of the traditional data models.[8] The information integration is to avoid the isomerism among different application information systems. The first step is to achieve data sharing in certain standard formats. From a microscopic point of view, a distribution application system must share with same types of data to reach data consistency and integrality, such as IEEE floating type standard. The main challenge in data integration is to solve the problems of metadata integrity in a dynamic data processing that requires namely metadata and massive data can be dynamically loaded for analysis. In order to support dynamic metadata, a reconstruction is needed. Such requirements of sharing metadata accurately gain connection information comprehensiveness. In order to support multi-application systems, all information concepts must have a high conformity, with a uniform information integration data model[1]. Data integration system frees the programmer from having to locate data sources, interact with each one in isolation and manually combine results. Data integration system cannot support applications which need quick, flexible and secure data sharing in Internet environments. [2] The goal of data integration systems (DIS) is to provide a unique, transparent and homogenous view of a set of heterogeneous data sources. Data integration is becoming even more necessary given the increasing availability of data from distributed and heterogeneous sources, as experienced in the development of the Internet. Such characteristics make it difficult to search for desired information since queries might be inappropriately answered or may have incomplete results if each data source is analyzed in isolation.[7] Integrating datasets is adopting feature matching, generalization, link, etc, which integrates multiple scale geospatial data, non-geospatial data and other geospatial datasets in other specialty department in order to form a new kind of spatial dataset.[9] Integrating datasets today does not follow standard steps or procedures and is mostly done manually with static results for urbanization in china: it is a craft. This leads to costly isolated case solutions, which are poorly or not documented at all, and not transferable to other situations. The repeated effort makes this approach even more expensive. Integrating datasets is adopting feature matching, generalization, link, etc, which integrates multiple scale geospatial data, geospatial data, non-geospatial data and other geospatial datasets in other specialty department in order to form a new kind of spatial dataset[10]. The data integration technique, teiid, which enables virtualization of various types of databases; through such virtual databases, one can access such data.
sources as relational databases, web databases, and application software such as ERP and CRM, for example, in real time. They can all be integrated for use. In fact, teiid has a unique query engine. Furthermore, the real-time data integration is accomplished by connecting business application software through the JDBC/SOAP access layer with data sources which are accessed through the connector framework[5].

2. Limitations of Data Integration

- There is not a standard data schema in many domains. So it is impossible to define a global schema to share data in these domains [2].
- Organizations require an independent data space to integrate and data in their hope to control and manage their own data integration and sharing system to keep data privacy. A centralized data integration system cannot suffice these demands.
- Centralized data integration which usually use standard JDBC/ODBC interfaces to access the data may cause security problems. For security reasons, data sources are usually secured by a firewall which only allows specific ports to be connected.

Data sources may only grant access privileges to specific hosts and users. In order to integrate the data by a data integration system outside the firewall, the firewall needs to open new ports to allow the traffic and data sources need to grant access privileges to new computers and users, which may bring security risks.

The system design, implementation and performance evaluation of Honeycomb and makes the following contributions:

- Utilize community and layered data model to implement the system. To give out the query parsing and query evaluation algorithm.
- Use Web Services as our remote access protocol between nodes. In order to decrease the overhead of web Services invocations, invent specific metadata cache and data cache techniques for our system.
- Implement a community-based system for distributed data integration and sharing and also present the performance evaluation of the system.

3. System Design

The design goals of Honeycomb are to facilitate building flexible, secure and scalable data sharing applications. Three elements to provide these actions: community, data model and communication protocol. The Elements are [2].

3.1. Community

In Honeycomb, the community is a resource container and has its own data sources. A community also has its users, roles and policies to share data. In runtime, community administrators can register data in data sources of his organization to the community; share the data in the community to other communities integrate the data from different communities. When a community instance is created, the system assigns a global identifier to the community according to community information and new community routing information will be replicated to all nodes. The nodes in Honeycomb are relative stable and the number of nodes is not very large. So it is easy to replicate community data to all nodes.

3.2. Data Model

The data model in Honeycomb is a generic relational table. In order to present heterogeneous data with different schemas, the data model contains composite schema named attributes which records table or view schemas. Other schemas of the data model are simple schemas. The data model consists of two data structures: physical relation and virtual relation. A physical relation is a storage description that specifies the data stored at a physical data source, and the attributes schema of physical relation records physical data schemas. A virtual relation is a query description that contains information including community identifier, resource type, and access policy and so on. The attributes schema of virtual relation records schemas and schema mapping information. There are two types of virtual relation: basic virtual relation and composite virtual relation. The schemas of a basic virtual relation are one-one correspondence to the schemas of physical relation. The virtual schemas of composite virtual relation can map to the schema from different virtual relations but physical relations. An example of schema mapping between virtual relation and physical relation is shown in Figure 1.

![Fig 1: An example of schema mapping](image)

3.3. Communication Protocol

Honeycomb system often needs to query metadata and data of a community which resides in remote node. So it is necessary to choose a remote access protocol to query the metadata and data. It should follow several principles to choose the remote access protocol for Honeycomb.

- The protocol should be an open and standard protocol.
- The protocol should bring minimum impacts on security configuration of system.
- The protocol should guarantee the communication is stateless.
- The protocol should be easy to be used to develop applications.

4. System Implementation

Implementation of Honeycomb, focusing on query execution and cache management.
4.1. System Structure
Honeycomb sits between applications and the underlying data sources. There are several kernel components that are loosely integrated in Honeycomb. Figure 2 illustrates the internals of Honeycomb.

![Fig 2: System structure of Honeycomb](image)

There are two types of access interfaces: metadata management interfaces and data query interfaces. Honeycomb publishes these interfaces as Web Services to permit metadata and data to be accessed by peer nodes. Metadata management component provides functions to manage metadata in Honeycomb. The component also supports functions to store metadata in metadata catalog and retrieve them from metadata catalog. The function of query execution component is to generate and executes query plan trees. The component determines if access the metadata and data locally or from peer node according to community routing information. The cache management component determines the caching and replacement policy for metadata cache and data cache which keep query results for subsequent answers.

4.2. Query Execution
When query execution component receives a query request, it first generates a query plan tree and then executes query according to the query plan tree. The process of generating a query plan tree in Honeycomb is called query parsing and the progress of executing the query plan tree is called query evaluation.

4.2.1. Query Parsing.
The root node of query plan tree is the virtual relation which is queried and called topVR in Honeycomb and the leaf nodes are basic virtual relations which are directly or indirectly referenced by the topVR. During query parsing, the query execution component determines whether to invoke local method or Web Services to retrieve virtual relations.

4.2.2. Query Evaluation.
After a query plan tree has been constructed, query execution engine executes the query according to the query plan tree. The engine first constructs tables of nodes in the query plan tree in a bottom-up approach. Then the engine executes the query and returns the result. The engine may construct the tables in a multithreading way.

4.3. Cache Management
There are three reasons why the cache is necessary and need specific cache design. First, the communities deployed in different nodes can reference resources from other communities arbitrarily. So there may be lots of Web Services invocation to occur in Honeycomb. Second, most of the queries in community data sharing system are simple queries in which Web Services invocation is the performance bottleneck. It is necessary to take actions to reduce the number of Web Services invocation. Third, because the system may expand quickly, it cannot assume that the metadata cache can hold all metadata and replacement operation in metadata cache may occur frequently.

4.3.1. Metadata Cache.
During the process of querying data, query parsing component needs to obtain metadata to construct query plan tree. Thus virtual relation objects are accessed frequently. We implement a variant of cache replacement policy for virtual relation cache. The goal of replacement policy is to reduce the possibility of replacement of virtual relations of remote communities, the number of Web Services invocation and query overhead. Figure 5 shows the detailed algorithm for replacement policy of virtual relation cache.

4.3.2. Data Cache.
After receiving query plan tree, query execution engine first decides if the data in temp table can suffice the query request. If the temp table of each virtual relation in query plan tree is not out of date, and the temp table of top virtual relation contains query items and selection conditions, the temp table of top virtual relation can suffice the query request. When one of the temp tables is out of date, the temp table should be reconstructed. When all temp tables are fresh and the temp table of top virtual relation does not contain query items or selection conditions, query execution will analyze all kinds of combination conditions to reuse the data in data cache.

5. Framework Architecture of the Data Integration System
The Framework architecture can be divided into three tiers in vertical direction: data layer, application layer and web layer. The data integration system has three modules in construction: data center, data service platform based on data center and distributed data nodes, where central database is the key link. The basic data which have nothing to do with business, including spatial data and basic information, are imported into data center. Business data is constructed and maintained by separate
data nodes. So central database is distributed maintained, while managed, serve centrally and is connected to the distributed data nodes on network to read the business data[3]. Framework architecture of the integration system is shown in Figure. 3

5.1. Data layer
Data layer consists of data center and several distributed heterogeneous databases and provides data support for data service. The basic data, which are obtained from distributed data nodes and processed according to unified standards, are brought together to the general information pools of data center and put in storage to central database by corresponding management system after data processing and data grading.

5.2. Application layer
The application service layer receives and analyzes the service requests from the web server firstly, secondly, accesses data of the data layer, and aggregates data to web layer.

5.3. Web layer
Web layer consists of web servers and targeted browses and provides interface for the interacting between end users and the system.

6. The Implementation of the Prototype System
The integration system is developed with Windows, Oracle 10g database and ASP.NET development environment, and archives the construction of data center based on hybrid integration models.

6.1. Data center and central database
Data center is the key to system design and development. The main framework of data center is shown in Figure. 4. The central database obtains processed basic data information from data nodes, and divides organized data into several application topics for application systems so as to integrate data as a topic-oriented database system.

6.2. Function module- querying data
These operations correspond with three web service interfaces: querying data interface, inserting data interface and deleting data interface, and take SQL statement, the database connection settings dataset as parameters. The web service address (URL) is recorded in data table, leaving out the course of registration. The clients invoke web service method dynamically with web service URL, and positions to targeted web service to implement relevant methods.

7. Data Environment Analysis
According to the analysis of data schema and logic construction in database in safety production, management, monitoring, controlling and selling system and analysis of the data, it has heterogeneous characters as following[4].

- **Naming difference**
  In different data schema, the same term describes various concepts or the various terms describe the same concept.

- **Value difference**
  The same concept has different data in different data source or the same figure uses different measuring unit in different data source.

- **Type difference**
  Data types are different in database. A data type of the same entity is inconsistent in different systems.

- **Schema difference**
  The same series of data demonstrate different data schema.
8. Database Virtualization

Databases of many kinds exist in terms of their associated data model differences and vendor differences. Regarding differences among data models, each has different data representation, and unique associated manipulation. Some typical examples include the table type of relational databases (RDB), XML-representation type of XML databases (XMLDB), and object-oriented databases (OODB). To have a virtualization feature, we will consider the inclusion of features to manage distributed databases of similar types, the distributed databases of different types, and provide location transparency for users, such that they notice no differences of database structure or location and become able to use databases of all kinds in a flexible fashion. Figure 5 portrays a comprehensive view of a database virtualization technique. For virtualization of ubiquitous databases in our study, we will describe the schema information of the real databases, of which more than one always happens to exist, by creating and using one common XML schema

8.1 Visualization of Homogeneous Distributed Databases

As the first step of database virtualization, is a method of building a virtual database management system for RDBs provided by different vendors. This can be achieved by three methods
- XML conversion program
- RDB schema conversion into XML
- RDB data conversion into XML

8.2 Virtualization of Heterogeneous Distributed Databases

The second step is the virtualization of modeled DBs of different types. For virtualization of different types of modeled DB, the schema information of each model using a single common schema. The common schema will use an XML Schema. Around it, will perform virtualization Figure 6 present a virtualization method for different database types.

8.3 Common Schema Generation

The common schema provides the virtualized database structure for the application programs. This schema is used to examine the syntax of query sentences and the constraints.

8.4 Query Conversion

Now under development of the query language to access the virtual databases. The planning the extension of the existing XQuery and the query conversion program from the XQuery language into SQL language or XQuery language

9. VDM System Framework

The whole VDM system is composed of five modules: client, parse engine, execute engine, resource delegate and virtual database manager. The interaction of different modules is shown in the Figure 7.
The client first logs into a parse engine. And then the client sends commands to the parse engine and finally gets the result from the parse engine. The parse engine listens to the command from clients. When the parse engine receives one command from the client, it analyzes the command and carries out corresponding work to deal with the command. Based on the type of command, the parse engine has the following tasks: 1) checking every command's grammar and semantic, 2) checking user's identity and working database, 3) composing a request which will be sent to execute engine by analyzing the command and then sending the request to the execute engine, 4) obtaining the result and presenting it to the client. In order to finish the first three tasks, the parse engine needs to know the user’s information and the metadata of all users' databases and tables.

The execute engine listens to the request from the parse engine and carries out corresponding work. In order to fulfill its responsibilities, it connects to the resource delegate and passes the sub-task to the resource delegate. And then when it gets the result from all the delegates, it will do integration work. When it completes executing, it will return the result to the parse engine. The resources may be distributed worldwide and belong to different organizations. If the resource owner wants to share their resources with others through VDM, the owner must provide a resource delegate so that the execute engine could send commands to the delegate and get the executing results. The resource owner could make its own resource sharing strategies and security policies by implementing the resource delegate. The VDM service could be seen from two perspectives. From users’ perspective, by using VDM, the user can integrate the data from distributed heterogeneous resources and manipulate these resources like manipulating a single resource of its own. From resource owners’ perspective, the resource owner can share its resources and provide services in a more controllable manner.

The service of the system provided to the users can be characterized into four categories: (1) user management, (2) resource management, (3) virtual database and table definition, (4) virtual database and table manipulation.

10. Related Work

The data in databases of autonomous organizations quickly and flexibly is an important problem. Traditional data integration systems are supposed to be a solution to the problem. Unfortunately, they require a global schema before they can be used to share data and they are centralized systems. They cannot support data sharing without a common schema and on-demand data sharing in applications.

Data integration is one of key issues about data exchange and information sharing in distributed computing environment. Researches on data integration, data transformation are made much of with marine data because of its characteristics of polymorphism, multi-sources and great capacity etc. Based on the analysis of the heterogeneous data environment, the system integration platform is built to resolve the heterogeneous data sharing and mutual visit in view of the status quo of the heterogeneous data and integration demand in the coal enterprise. Data is becoming more and more important in the current computing environment and enterprise application. In the grid, many systems and middleware aim to hide the heterogeneity of the data resources and to provide a unified way to access them. However, there are a few systems focusing on data integration which means combining structured and unstructured data residing at different data resources and providing the user a unified way to manage these data. Data integration becomes even more necessary given the increasing availability of data from distributed and heterogeneous sources. To address such heterogeneity, crisp ontologies have been employed in order to represent the semantics of integrated data. Our work is related to the wealth of research on data integration and peer-to-peer data management. With the help of mediator, a data integration system provides functions for users to access the data stored in heterogeneous and autonomous sources, without the need to know the physical characteristics of such sources and the precise location of the data. More recently, database community has begun to investigate data management in Peer-to-Peer environment.

11. References


