ISSN:0975-9646

Creating Materialized View Over Integrated Heterogeneous Databases

Ankush J.Paturde [#1], Anil V. Deorankar [*2], P.N.Chatur [#3]

Student, Computer Science and Engineering, Government College of Engineering, Amravati, India [1]
Associate Professor, Computer Science and Engineering, Government College of Engineering, Amravati, India [2]
Head of the Department, Computer Science and Engineering, Government College of Engineering, Amravati, India [3]

***Abstract*— With the development of computer network and database, traditional database has been increasing unable to meet the needs of data sharing and interoperability. Meanwhile, it is impossible to abandon all the existing database systems; therefore, the research of simultaneously accessing and processing data from a number of databases has become an inevitable trend. For the Health care information system its not the issue to retrieve the information from their own databases. But when we want the information other than the own databases, then its an issue to get that information to our system. And the data which we want from other health care organizations may not be in same format. To solve this problem the proposed architecture is to integrate different geographically dispersed databases that are heterogeneous with regard to their logical schemas.For the Integration of heterogeneous databases MyAccess , MySQL, SQL and Oracle databases are taken. These databases are having different characteristics of data types and semantic conflictions may occure while integrating heterogeneous databases. Using the technology of Java , SQL Language the heterogeneous database integration system is proposed and designed and key technologies are also described in detail. For the queries which are frequently fired to the databases for retrieving the information, the materialized view is created. By creating materialized view the information which requires most to the user is stored in the dataware house. Because of which response to the frequent queries is fast, no need to search the the whole database. It reduces the overhead for the frequent queries and gives fast response to the fired quieries.**

***Keywords*— Data Integration, data transformation, Heterogeneous databases, Java Technology, SQL query**

I. INTRODUCTION

In Modern information systems usually consist of a number of departments, which have different functions. These departments are usually independent from each other, utilizing a number of diverse databases to accomplish their day-to-day data management functions. When we design the relevant databases dependently, it does not create heterogeneity problems to integrate the data sources. However, when databases have been designed independently, there are heterogeneity problems such as different terminology, data types, units of measurement, domains, scopes, and so on. Effective decision making often requires access to data from multiple such databases belonging to different departments. The need to integrate these disparate systems has led to heterogeneous distributed database systems [1]. Since different departments may be geographically distributed in a wide area and are supposed to have control over their own affairs, it is worthy to develop a mechanism that can access data from multiple databases, while preserving the local autonomy of the databases, i.e., without changing the existing databases. The structures of these heterogeneous databases are different obviously, and semantic conflictions may occur. Then we developed an architecture to integrate different geographically dispersed databases that are heterogeneous with regard to their logical schemas. Data integration shields the heterogeneity of the various heterogeneous data sources, and carries out unified operation to different data sources through heterogeneous data integration system. The data forms involved in heterogeneous database are mainly structured document is a template. data, semi-structured data and unstructured data three types. Structured data widely exists in a variety of information system database, the most common relational database. Semi-structured data commonly has Web pages as the chief representative. Unstructured data has common files, email and various documents. A practical information integration system should have intelligence, openness and initiative. Intelligence is to carry out unified processing, filtering, reduction, abstraction, integration and induction works for the structured, semi-structured and unstructured data from different databases. Openness is a heterogeneous and distributed database, which must solve the mismatching problem of the information expression with the structure. Initiative is to regulate the existing Internet data representation, exchange and service mechanism to provide proactive service mechanism.

Then creating materialized view of the frequent queries. Materialized views are constructed and stored in a data warehouse with the purpose of improving response time of analytical queries. Since all possible views cannot be materialized, as it would violate the storage constraint, the aim is to construct those materialized views that maximize the profit in terms of their ability to answer future user queries.

II. LITERATURE SURVEY

Information from different sources cannot be integrated or interoperated if it has not passed the process of conflict resolution. In terms of database integration, conflicts are

differences of relevant data between component local data sources. Schema conflicts are discrepancies in the structure level such as Naming conflicts, Structural conflicts, and Identity [5; 8; 9]. A number of integration approaches have been introduced throughout the last twenty years to bring about the interoperability among heterogeneous systems. Missier, Rusinkiewicz, & Jin [10] categorise heterogeneity resolution methodologies into four main approaches: Translation, Integrated, Decentralised, and Broker based. Translation approach needs highly specialised translation for each pair of local database systems. Therefore, the number of translators grows exponentially especially when the number of local systems increases. The development of these ad hoc programs is expensive in terms of both time and money.

A. Data types matching method

Each of the database tables are made of one or more columns (fields). While creating table, a data type must be designated in each column(field), which is used to describe the varieties of values in the appointed column(field) accurately. In different types of databases, the field's types and lengths aren't the same. Therefore, there must be corresponding transformation in different types of fields in order to accord to the criterions of target database application. This is the key in the derivation of database, also is the emphasis and difficulty in the development of the software system.

B. Lack of Uniform Information Standards

Due to lack of unified planning and fragmentation, each small system has its own set of information standards, leading to confused information standards, unable to achieve information sharing, seriously restricting the application of building pace, bringing barrier to the campus information exchange and sharing, also generating a lot of redundant information. In Fully integrated or Tight-coupling approach, individual schemas from multiple data sources are merged by one (the global schema approach) or more schemas (the federated database approach). This approach requires complete pre-integration. This approach allows users to query local database systems directly by placing the integration responsibility on users. However, it requires users to have semantic understanding and to be able to resolve conflicts in creating their schemas. easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

C. Connection pool

Connection pool is a kind of entity which manages the connection as a resource, and a typical example of such resource is the database connection. The basic idea of the connection pool is to pre-establish some connections to store in the memory for use. To establish a database connection will consume considerable system resources, but once established, the query can be sent to obtain results through it[11]. The number of queries a connection in its life cycle can process is not limit, so a database connection from a certain way is a resource.

- *Query manager*

Query manager is to provide a unified interface for the client to query the data source, responsible for receiving the global query requirements from browser, and then accordlllg to the corresponding integration information to decompose them into several local query requests to be transmitted to the data packager, finally to process the results returned from it to return to the browser, while to maintain the integrated information to ensure the accuracy and consistency of the global transaction execution. From the data conversion process, the query manager integrated the existing shared data model and data[11]. The integration is mainly embodied to the integration between the existing different databases, the need to differentiate, despite the current database middleware also support data integration, but their data integration refers to the access to different databases through them, that IS, the access integration, rather than the establishment of links between the data of different databases. The integration in this paper not only includes the data access integration, but also the simple integration between the data of heterogeneous databases.

- *Data packager*

The reason for integration is that for a specific task between the heterogeneous databases their may exist a general logical link, and the data stored in them may be different stages of the same thing description[11]. Query manager is responsible for submitting the standard query the client submitted to the query parser, the query analysis unit is to realize the query parsing and use meta data management module to check the legitimacy of the query request to judge whether the query data exists, and convert the query request into standardized form, while recording the result set target format to submit it to the appropriate data packager. Choosing intermediate data files

According to the characteristics of Access, MySQL, SQL Server and Oracle databases, with the heterogeneous database transformation requirement, in C#.net environment, using the standard SQL language, the heterogeneous database transformation software is developped. It is not enough to achieve more universal data transformation about heterogeneous database only dependent on the software tools provided by database itself. It is still a main method to exchange data by using intermediate data file. There are generally three ways, one is the manner based on binary text, one is based on the database, the other is based on XML[13]. For this heterogeneous database import-and-export system, the derivation in intermediate data file must meet the following conditions: one is high efficiency to access database, the other is unified format structure. When this software runs it must connect with source database and target database, so it consumes a lot of time. If access efficiency is too low, user may can't suffer this delay. Consequently, unified data structure can improve the efficiency to access database. Based on the database. This manner requies a third database to cache when exchange data between two specific databases. This method, completely relying on the intermediate database to exchange data, needs less modules and is more extensible, but it is relatively complex to

transform. Because the method increases a layer of data exchange, so it is more difficult to achieve.

D. XML language

The emergence of XML technology makes the standardization description of all kinds of irregular information and rule information possible, and gradually becomes the standard to describe the data in Internet, and sets up an enterprise information integration platform in the XML technology, is an inevitable trend of information technology development [I] . XML as an extensible markup language, its self-description makes XML itself suitable for data exchange between heterogeneous applications, and this exchange does not make the pre-specified data structure definition as the premise, so it has strong open and broad application prospects. And almost all of the existing large-scale applications are related to the database, so the data exchange and information sharing with XML as neutral carriers must be linked to databases; at the same time, XML-based data exchange and the realization of the database XML data bidirectional access can integrate the XML data with specific applications, thereby enabling the combination with the existing business rules, and finally truly achieving XML-based distributed data exchange and information integration.

The XML-based query extended system carries out processing on the complex query of relational heterogeneous database, avoiding the defects of using general processing approach, enhancing the importance of application software reusability and lowering the cost of software development. Complex query plan should firstly be input by the user. Here, we introduce the SQL like query plan to describe complex query plan, use XML query file structure to convert into XML query plan document by the rewriter, and then complete the query task through operations of parsing, decomposition and results integration, etc. There is only one level in the reference of SQL query, that is, the reference of specific fields.

E. GIRD-Based Integration Model

In this paper we put forward a Gird-Based Integration Model (GBIM) for uniformly accessing heterogeneous database systems. In GBIM database operations are registered as grid services by different sites and a uniformly standard interface is provided to different users. In GBIM every database site is autonomous and its structure and position are transparent to users. GBIM makes use of GSI (Grid Security Infrastructure) to shield database sites from unauthorized access and transmit data safely via Internet. GBIM is suitable and feasible for the large scale distributed heterogeneous database systems. The implementation of its prototype is also described.

GBIM mainly consists of Access Entry, Database Service, Notification Manager, Query Manager, Schema Integrator and Data Assembler.

- *Access Entry*

The Access Entry deals with authorization and authentication. It prevents unauthorized access to GBIM and finds the available Database Services by referring to the Register Center to execute a query set in a proper order.

- *Database Service*

Local database system provides its data operations in the form of grid service, Database Service in figure 1. Database Service is a kind of stateful Web Service according to OGSA and local databases are treated as corresponding Data Resource.

There are two advantages to pack database operations in to grid service. First, Database Service provides standard Web Service interfaces, enabling cooperation among different database systems. Second, most Web Services use HTTP for transmitting messages. This is a major advantage if you want to build an Internet-scale integration of heterogeneous databases, since most of the Internet's proxies and firewalls won't mess with HTTP traffic.

- *Notification Manager*

This would allow users to specify what data (data set) they are interested in and will notify them the data changes in data resource. Notification Manager can be built according to WSNotifications family of specifications.

- *Schema Integrator*

The Schema Integrator is a important component of GBIM, it is responsible for collecting and managing the data resource metadata, constructing integrated data schema and conducting schema mapping. When a database registered itself to the Register Center, it also provides metadata of the data resource and its schema. The Schema Integrator will use this information to construct integrated data schema adopting methods discussed in section 1 and perform schema mapping with the mapping rules recorded.

III. CONCLUSIONS

With the reference of the existing research work, it can be concluded that integration of heterogeneous databases is important for the disperate information stored in the different databases.

For this java technology can be used. Using java technology we will get the fast response to the fired quiries. Over this integrated databases the materialized view of the frequent queries will be created. Due to which for the frequent queries we don't have to search the different databases. Firstly materialized data will be search and then after goes for the different databases to search. Because of which we get the fast response to the frequent queries.

REFERENCES

[1] BRITO, M. S. et al. An Architecture for Integrating Databases with Replication Support Based on the OGSA-DAI Middleware. 12th IEEE International Conference on Computational Science and Engineering (CSE 2009). Vancouver 2009. p. 298-305.

[2] Wu Xiaoli,Yao Yuan "XML-based Heterogeneous Database Integration System Design and Implementation". Department of Computer Science and Engineering, Henan University of Urban Construction IEEE 2010.

[3] Zhou Shudao, Zhu Guotao, Wang Yanjie,Shen Ye, Liu Zanyi" Research on Heterogeneous Database Transformation"Institute of Meteorology, PLA Univ. of Sci. & Tech., Nanjing, Jiangsu, 2011, China

[4] C. H. Goh, S. E. Madnick, & M. D. Siegal, "Context inter-change: overcoming the challenges of large-scale interoperable database systems in a dynamic environment", in Proc. Inf. and Knowledge Management, MD USA,1994.

[5] R. D. Holowczak, & W. S. Li, "A survey on attribute correspondence and heterogeneity metadata representation", nstitute of Electrical &

Electronics Engineers. Available: http://church.computer.org/conferences/meta96/li/paper.html, 1996.

[6] K. Abdulla, "A new approach to the integration of heterogeneous databases and information systems", Dissertation, University of Miami, Florida, 1998.

[7] Chaiyaporn Chirathamjaree. 'A Data Model for Heterogeneous Data Sources' EngineeringSchool of Computer & Information Science, Edith Cowan University, Australia,IEEE International Conference on e-Business 2008.

[8] W. Kim, I. Choi, I. Gala, & M. Scheevel, "On resolving schematic heterogeneity in multidatabase systems". Distributed and Parallel Database, vol.1, no.3, pp.251-279, 1993.

[9] R. J. Miller, "Using schematically heterogeneous structures", SIGMOD'98, WA, USA, pp.189-200, 1998.

[10] P. Missier, M. Rusinkiewicz, & W. Jin, "Multidatabase languages", Management of heterogeneous and autonomous database systems. CA: Morgan Kaufmann Publishers, 1999.

[11] Wu Xiaoli and Yao Yuan "XML-based Heterogeneous Database Integration System Design and Implementation" Department of Computer Science and Engineering, Henan University of Urban Construction 2011.

[12] Mingli Wu and Yebai Li "Investigations on XML-based Data Exchange between Heterogeneous Databases". Ninth Web Information Systems and Applications Conference 2012.

[13] Y.Badr,M.Sayah,F.Laforest,A.Folry. Transformation Rules from Semi-structured XML Documents to Database Model, Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Lebanon, 2001.

[14] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[15] K. Elissa, "Title of paper if known," unpublished.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[16] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].