Chapter 8

Automating the Generation of Joins in Large Databases and Web Services

Sikha Bagui
The University of West Florida, USA

Adam Loggins
Zilliant Inc., USA

ABSTRACT

In this data-centric world, as web services and service oriented architectures gain momentum and become a standard for data usage, there will be a need for tools to automate data retrieval. In this paper we propose a tool that automates the generation of joins in a transparent and integrated fashion in heterogeneous large databases as well as web services. This tool reads metadata information and automatically displays a join path and a SQL join query. This tool will be extremely useful for performing joins to help in the retrieval of information in large databases as well as web services.

INTRODUCTION AND RELATED WORKS

As we are working with more and more data, the sizes of databases are getting larger and larger. As businesses are going global, web services are becoming a standard for sharing data (Srivastava et al., 2006; Resende and Feng, 2007). Enterprises are moving towards service oriented architectures where several large databases may be layered behind web services, hence databases are having to become adaptable with loosely-coupled, heterogeneous systems (Srivastava et al., 2006) too. In such scenarios of web services and service oriented architectures, which may be dealing with several loosely coupled heterogeneous distributed large databases, it is no longer humanly possible to have handy all the information on all the tables and primary keys in all the large databases. Although considerable work is
being done on the challenges associated with web services addressing the problem of multiple web services to carry out particular tasks (Florescu et al., 2003; Ouzzani and Bouguettaya, 2004), most of this work is targeted towards work-flow of applications, rather than coordinating how data can be retrieved from multiple large databases in web services via SQL (Srivastava et al., 2006). In this paper we try to address one aspect of this problem of retrieving data from multiple heterogeneous large databases using SQL. Specifically, we present a tool that automatically formulates joins by reading the metadata of databases in the context of very large distributed databases or in the context of web services which may employ the use of several large heterogeneous distributed databases.

Let us look at an example of a query presented to a web service:

Suppose a health insurance company needs to verify the salary, health, and travel patterns of a person before determining the amount of health insurance he/she needs to pay. In a web service, this will require joining of several tables. And, of course, no one person will have knowledge of all the primary key/foreign key relationships between the tables to join in the web services.

When databases were smaller, it was possible to have knowledge of most of the tables and primary key/foreign key relationships in databases, and SQL join queries could easily be built by joining tables in databases. But, in large databases layered behind web services, it will not be possible to have knowledge of all the database schemas.

The join operation, originally defined in the relational data model (Codd 1970, 1972), is a fundamental relational database operation, facilitating the retrieval of information from two relations (tables). Writing efficient joins is simple for small databases since few relations are involved and one has knowledge of the complete database schema. But, writing efficient joins is a challenge in large database scenarios and web services where it may not be possible to have a complete picture of the database schema and it’s relations and relationships.

Since joins are one of the most time-consuming and data-intensive operations in relational query processing, joins have been studied extensively in the literature. Mishra and Eich (1992) present a very comprehensive study of works that have been done on joins. Query optimization issues in joins have been discussed by many, for example, Kim et al. (1985), Perrizo et al. (1989), Segev (1986), Swami and Gupta (1988), Yoo and Lafortune (1989), and Yu et al (1985, 1987). More recent works have also focused on devising strategies for distributed join processing, for example, works by Scheuermann and Chong (1995), Rao and March (2004), Michael, et al (2007), Ramesh et al. (2009), Frey et al. (2009), and Zhao et al (2010).

Scheuermann and Chong (1995) introduced an algorithm for join processing in distributed databases that makes use of bipartite graphs. Bipartite graphs are tuples that can be joined, hence reducing the state of the relations. Rao and March (1997) developed a genetic algorithm-based solution which considers design decisions, communications and local processing costs to determine efficient join processing plans. Michael et al. (2007) showed how to improve performance of distributed joins with the use of Bloom filter based algorithms. However, the authors mention that the full potential of bloom filters has not yet been exploited, especially in the case of multi-joins where the data is distributed among several sites. Ramesh et al. (2009) also worked algorithms for distributed joins using bloom filters. Frey et al. (2009) worked on a distributed join execution mechanism that leveraged modern networking facilities to run joins of arbitrary size in distributed main memory rather than on disk. Zao, et. al (2010) studied the distributed resource allocation problem for synchronous fork and join processing network, with the goal of achieving the maximum total utility of output streams.