CS 848: Project Proposal
W Anthony Young - 20161423
October 3rd, 2004

A database is a collection of records stored on a computer. A database is more than that: databases contain the information that keeps our society working. Businesses store employee, product, inventory, customer and invoicing information. Schools store student academic records, contact and timetable information. Health professionals store patient and surgery information. Our world is composed of vital data, and that data comes from many different sources

Traditionally, data is stored in several different tables, within several different databases, and often on several different machines. For example, a moving company with offices around Canada could potentially have a different database server for each office. In this manner, it would be up to a local database administrator to control the type of information stored, the architecture of the system used to store it, and so on. Suppose a customer arrives at one office and has questions requiring information stored at a second office. How can this information be retrieved? How does the employee know what the table and column names are?

Enter "multidatabase systems". A multidatabase (or federated database) system is a piece of software capable of bridging the gap between information at different data sites. Federated database systems have the power to retrieve information stored in many different databases on many different servers across a network. With our above example, a federated database system would be able to link the office databases from around the country for our moving company. Then, our employee would be able to find the information required by our customer quickly and efficiently.

Federated database systems pose many interesting challenges for query routing, optimization and execution. Some of the greatest challenges faced are:

- Varying capabilities: Each site may run a different database management system (DBMS) to store their data. As well, some sites might not implement all features of the query language they use to access data. How can we perform queries that require data from sites that do not have the capability to perform certain operations? For example, how can we perform ranking operations on data stored at a site that does not implement the SQL ranking specification?

- Schema translation: When a global schema is created to give a view of the data stored at local sites, we must often map table and column names in order to consolidate data into common stores at the federated level. For example, one site may store customer data in a table called "cust_data" while a second site might store data in a table called "customer_information". In the global schema these tables might be mapped to one common table called "Customers". When submitting queries to local sites through the federated system, a query must be translated to contain the names of local tables and columns. Speed and efficiency are paramount when performing these translations.
- Global query optimization without local information: A federated database consists of a global view of the data stored at multiple local sites. How can we optimize a query to be run at a subset of local sites if we do not have cost parameters for data stored at those sites? Additionally, we must take into account the overhead of local system load, disk access times, indices and network transport. Much of the information required to make these cost evaluations exists at local sites, but is not accessible by our global query optimizer at the federated site. Can we access those statistics in some way?

Currently, I am working with the iAnywhere Solutions Research and Development team investigating federated database technology. My position requires me to research state of the art technologies for federating data and globally optimizing queries. Unfortunately, I am not able to discuss my position further due to my agreement with iAnywhere Solutions. I propose to combine some research performed at iAnywhere with my course project for CS 848. As part of my iAnywhere project, I will be reading about and evaluating different federation technologies. However, I will not be discussing ways of collecting statistics for use in global query optimization. I propose a paper outlined as follows:

- Discuss and evaluate federated database technologies and techniques: Some systems to be evaluated include Mermaid, InterViso and IBM's Information Integrator. I hope to gain an insight into the way these systems work and how they can be improved.
- Discuss and evaluate statistical global query optimization methods: There are many ways to use statistics to optimize queries. Sampling and reduction-based methods are two of the strategies that I would like to investigate further to understand some of the challenges that different strategies address.

- Discuss and evaluate different methods for collecting local statistics for global query optimization: Global query optimization depends quite heavily on statistics. As such, we need methods to collect these statistics from local sites. Collection utilities and query piggybacking are two proposed statistics collection methods. I would like to investigate others and see how well they stack up against each other.

This project would require additional work beyond my iAnywhere project, as its scope does not currently include researching local statistics collection methods. It is my intention to publish my findings in a paper submitted to reputable journals and conferences.