INTRODUCTION TO DISTRIBUTED DATABASE SYSTEM

Distributed database system (DDBS) is a database in which storage devices are not all attached to a common CPU. It may be stored in a multiple computers located in the same physical location, or be dispersed over a network of interconnected computers. Take it simply, it is a database system that is logically centralized but physically distributed. It can be regarded as a combination of database system and computer network. Though this is an important issue in database architecture, the storage and query in the distributed database system is one of the most significant problems in the present database systems. Therefore, when preparing the lecture that focuses on the database storage and query, we inevitably encounter this topic. Here I'll provide a brief introduction to DDBS, especially focusing on the query operations.

1. Design Principles of DDBS

C.J.Date, the early designer of relational database besides E.F.Codd, proposed the 12 design principles of DDBS which is now widely accepted and regarded as the standard definition of DDBS. They are:

(1) local autonomy; (2) no reliance on central site; (3) continuous operation; (4) location transparency and location independence; (5) fragmentation independence;

(6) replication independence; (7) distributed query process; (8) distributed transaction management; (9) hardware independence; (10) operate system independence; (11) network independence; (12) DBMS (database management system) independence.

As for the users, they will not feel like they are using a distributed database system. To the contrary, the main goal of DDBS design is to make the using of the system like operating on a local database. In this case, it is a real and general DDBS.

2. Data Fragmentation

The using of DDBS is to distribute the data to different servers so as to improve security and enhance the ability to deal with large scale of data, thus data fragmentation is the first step in storing a distributed database. Three types of data fragmentation are involved:

Horizontal Fragmentation: divide every tuples in the global relation into mutually exclusive sets, and each set is a fragment of the global relation. To retrieve the primal database we have to UNION the data fragments.

Vertical Fragmentation: divide the attributes of the global relation into different sets. Applying the JOIN operation to the sets, we can retrieve the original relation.

Mixed Fragmentation: the combination of the previous methods of fragmentation.

With the methods above we are able to fragment the original relations and store them in different sites.

3. Independence of Data in DDBS

When using the DDBS, users do not have to have the knowledge about the distribution of global data. That is to say, the logical fragmentation of data, the physical location of the data fragments and the data model of every site is transparent to users. Hence, independence of data is also called data transparency. According to their relative hierarchy in the DDBS, there are three types of data transparency:

Fragmentation Transparency: users do not have to care about the data fragmentation. Even if the fragmentation changes, it will not influence on the use of the database.

Location Transparency: users do not have to know about the duplicates of data fragments and where these duplicates are stored.

Local Data Model Transparency: users do not have to know about the specific data model and the properties of the objects on every site.

4. Distributed Data Query Process

Query in DDBS is different from the traditional query since it involves in the communication between sites. Traditional query deal with two main costs: CPU and I/O. However, in DDBS, the time spent on data communication must be considered. It can be calculated by the following formula:

$$T(X) = C0 + C1 * X$$

In the formula, *C0* stands for the fixed cost when communicate between two sites, *C1* refers the transmission rate, with the unit "second per bit", and *X* is the amount of data been transmitted, with the unit "bit".

For the system with high transmission speed, the main bottleneck of query efficiency is the same as query on a local database. However, in a system with low data communication rate, the time spent on the data transmission must be considered.

In implementing the distributed data query, there are four major processes:

(1) Query decomposition: decompose the query to an expression of relational algebra.

(2) Data localization: Transform the global algebraic expression into the expressions on each data fragments on different sites.

(3) Global optimization: Find the best sequence of query operation, taking the cost of CPU, I/O and transmission into consideration.

(4) Local optimization: It is implemented by each site to optimize the query there. It is the same as optimize the query on a local database.

5. Introduction to Semi-Join Algorithm

In the system where data transmission costs more time than data processing, an algorithm called semi-join algorithm, is applied. The operation semi-join is the combination of projection and joining. It only joins the common attributes between the two relations. Therefore, if we first get the semi-join of relation A and B, and transmit it to relation A and implement the joining operation, and then transmit the result to B and implement another joining operation, the data transmission is much less than the method of directly transmit the relation A to B and then join them. According to the property of semi-join, if we need to join a small part in one relation to another relation, using semi-join is a desirable strategy.