Multi-Agent Environment for Multiple Database Systems *

6E－6

Chiaki Yahata, Makoto Takizawa †
Tokyo Denki University ‡
e-mail{chii,taki}@takilab.k.dendai.ac.jp

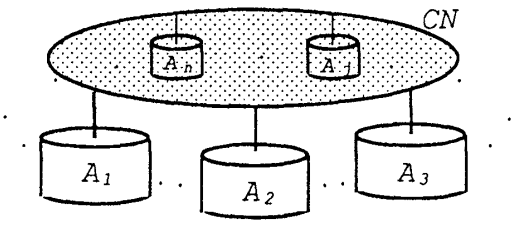
1 Introduction

A *cooperating database system* (*CDBS*) [2, 3] is composed of multiple agents interconnected by communication networks where some agents provide database systems(*DBSs*). It purposes to provide easy access to various kinds of multiple autonomous database systems [1] under a situation that the system configuration is changed dynamically. In this paper, we would like to present the architecture of the *CDBS* and a protocol for doing the negotiation among multiple agents.

In section 2, a system model is presented. In section 3, we discuss the *acquaintance* relation among the agents. In section 4, a protocol is discussed. In section 5, a learning mechanism of each agent to obtain information on the change of the system state is presented.

2 System Model

The *CDBS* is composed of multiple autonomous agents interconnected by networks [Figure 1]. An



CN: network A_j: agent ($i = 1, \ldots, n$)

Figure 1: Cooperating database system

agent is a system which provides a *DBS* and may access another agent to answer user's requests. The agent considers the database as a collection of data objects. Objects are tuples in the relational *DBS*. Each user U issues an agent A a request R which describes what data objects U would like to access. A takes the request R from U, and finds what agents have objects required by U. A accesses its own database if the database includes the objects, and asks another agent to answer the request. Even if A has the objects, A may ask another agent to answer R if A thinks of it to be more suitable to answer the requests, e.g. from the performance point of view.

Even if some strategy for accessing multiple agents is pre-decided based on the statistical information, the agents may not behave as expected in the strategy, for example, because the states or policies of the agents may be changed. Hence, the negotiation among the agents is required to make clear what and how each agent can do. The agents does the negotiation with other agents to find what agents have the objects and how they could obtain them. Thus, the users can manipulate multiple *DBS* through the agents without being conscious of the heterogeneity, distribution, and autonomy of the *DBSs*.

3 Acquaintances

Let MDB_A and DB_A be a metadatabase and a database of an agent A, respectively. Each term t in MDB_A denotes not only objects on t which A has but also another agent which A knows has t. MDB_A is T_A. If MDB_A includes t, A is referred to as *know* about t. A *directly knows* about t iff DB_A includes some object on t A is referred to as *indirectly know* about t if A knows about t but does not directly know about t. Here, although A has no object about t, A has t in MDB_A. Hence, A cannot obtain objects on t from DB_A but can ask another agent denoted by t in MDB_A which directly or indirectly knows about t. If A directly knows about t, A can obtain objects on t from DB_A.

[Definition] A is an *acquaintance* of B on t ($A \xrightarrow{t} B$) iff A knows that B knows about t. For some term t, $A \rightarrow B$ if $A \xrightarrow{t} B$. □

If A cannot answer a request R on t, A can send R to an acquaintance B of A on t.

[Definition] For some term t, if $A \xrightarrow{t} B$, $B \xrightarrow{t} C$, and not $B \xrightarrow{t} A$, A *transitively knows* C about t (or C is an *indirect acquaintance* of A) ($A \overset{t}{\hookrightarrow} C$). A *directly knows* B about t (written $A \overset{t}{\Rightarrow} B$) (or B is a *direct acquaintance* of A) iff $A \xrightarrow{t} B$ and not $A \overset{t}{\hookrightarrow} C$. □

It is clear that $A \xrightarrow{t} B$ if $A \overset{t}{\hookrightarrow} B$. We assume that A directly knows B if A can access B, i.e. A has the access right on B. If A transitively knows B about t, A cannot access B because A may have no access right on B. Hence, A can access only the direct acquaintances. There are two ways to access B. One way is that A finds the direct acquaintance C such that $C \rightarrow B$ and asks C to access B. In the other way, A has to obtain the access right on B and then A accesses B directly. In order to obtain the access right on B, A has to do the negotiation with B. There are two ways to obtain the access right on B. In the first way, A asks B directly to grant the access right to A. In the second way, A first asks the direct acquaintance C to allow A to access B. C asks B if A could access B.

*多データベースシステムに対する多エージェントシステム
†矢羽田 千哲，滝沢 誠
‡東京電機大学

4 Negotiation

In this paper, we would like to think about only retrieval operations on multiple *DBS*s because it is difficult to consider update operations on multiple *DBS*s and most users would rather retrieve data objects.

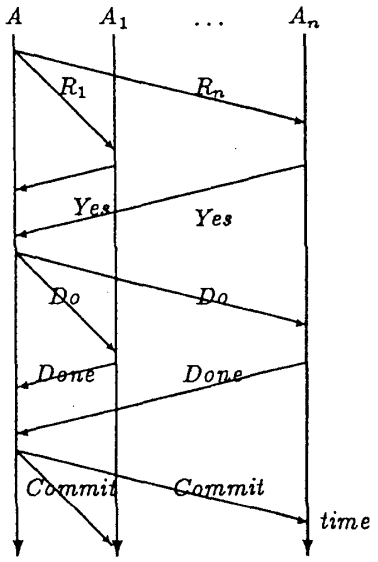


Figure 2: Negotiation protocol

A negotiation protocol is shown as follows.

1. First, an agent A takes a request R from a requester U. R is composed of a qualification Q and a *preference* P, i.e. $R = \langle Q, P \rangle$. If A can answer R, A executes R. Otherwise, A decomposes R into $R_1, \ldots, R_n$ $(n \geq 1)$, where $R_i = \langle Q_i, P_i \rangle$ $(i = 1, \ldots, n)$. If R cannot be decomposed, A sends the *Failure* back to U.

2. A finds for each R_i an acquaintance agent A_i which A thinks can answer R_i. If no agent can be found for R_i, R_i is further decomposed into smaller subrequests $R_{i1}, \ldots, R_{im_i}$. Then, this step is repeated until some agent is allocated to each subrequest. If R_i cannot be further decomposed, all the executions of the subrequests are aborted, i.e. A sends *Abort* to $A_1, \ldots, A_n$. Then, R is differently decomposed by step 1.

3. A asks each A_i whether A_i can answer R_i and how A_i can answer R_i. If A_i cannot answer R_i, A tries to find another acquaintance at step 2. If A cannot find any agent for R_i, R_i is further decomposed by returning to step 2.

4. A asks A_i to execute R_i by sending a *Do* to A_i. On receipt of the *Do*, A_i executes R_i. If A_i can not obtain the answer of R_i, A_i sends the *Failure* to A. On receipt of the *Failure* from some A_i, A returns to step 3 and tries to find another candidate of R_i. If A_i can obtain the answer RP_i of R_i, A_i sends the *Done* with RP_i to A.

5. A integrates all answers $RP_1, \ldots, RP_n$ into an answer RP for R. A sends the RP back to U.□

5 Learning

An agent A can obtain newly terms and relations among terms from another agent through the negotiation. The terms and relations among the terms are stored in MDB_A. The process is named a *learning*.

In Figure 3, an agent A knows that *London* and *Paris* are *capitals*, but does not know of *Tokyo*. Agents B and C know that *Tokyo* is a capital. B and C are acquaintances of A. Suppose that A takes a request to obtain a set of capitals. A asks B and C. B and C return the sets of capitals derived from DB_B and DB_C, i.e. $RP_B = \{Tokyo, London\}$ and $RP_C = \{Tokyo, Paris\}$, respectively. On receipt of the replies RP_B and RP_C from B and C, A newly knows that *Tokyo* is a capital. A adds *Tokyo* and *is_a* relation "*Tokyo* is a *Capital*" in MDB_A.

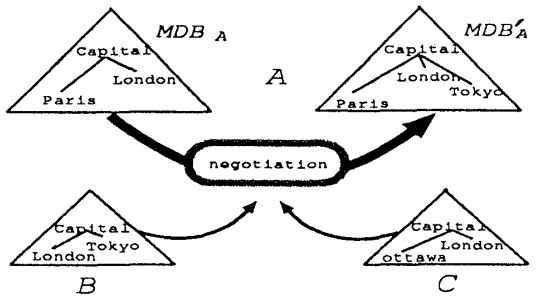


Figure 3: Learning

The metadatabases are finite. If MDB_A is too full to store new terms and relations, some terms are removed from MDB_A. It is named an *oblivion* process. Problem is what terms and relations to be removed from MDB_A. The following terms are not removed: the terms which A directly knows, the terms frequently used, and the terms of the higher levels.

6 Concluding Remarks

In this paper, we have discussed the architecture of the *CDBS* which is composed of multiple agents interconnected by the communication network. We have shown the negotiation protocol among the agents. By this protocol, agents can obtain the reply by taking advantage of another agent. We have also shown how to maintain the metadatabases, i.e. learning module.

Reference

[1] Sheth, A. P. and Larson, J. A., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol.22, No.3, 1990, pp.183-236.

[2] Takizawa, M., Hasegawa, M., and Deen, M., "Interoperability of Distributed Information System," *Proc. of the 1st International Workshop on IMS'91*, 1991, pp.239-242.

[3] Yahata, C., Takizawa, M., "Cooperation of Multiple Agents to Access Multiple Database Systems," *Proc. of ICPADS'93*, 1993, pp243-246.