Multi-Agent System for Large Distributed System *

5R−5

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

1 Introduction

A *cooperating database system*(*CDBS*) is composed of multiple agents interconnected by communication networks. Each agent provides a database system (*DBS*). In a system which includes a large number of *DBS*s, it is difficult, maybe impossible for users to obtain enough information on what kinds of *DBS*s are included and how to access then. An agent is a system which assists users with accessing multiple database systems. 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 model of the *CDBS* is presented. In section 3, we discuss *acquaintance* relations among the agents. In section 4, a protocol for doing the negotiation among multiple agents is discussed. In section 5, a learning method for each agents to obtain information on the change of the system state is presented.

2 System Model

A *cooperating database system* (*CDBS*) [2, 3] is composed of multiple autonomy agents interconnected by communication networks.

2.1 Passive and active agents

There are two kinds of agents, i.e. *passive* and *active* ones. The passive agent A takes a request R from a requester U, i.e. a user or another agent, and then answers R if A can answer R. A sends the answer of R back to U if A makes a success in answering R. If A cannot answer R, A informs U of the failure. The passive agent does not issue requests to another agents. It can take the request and reply it. Conventional *DBS*s and server system like print servers are examples of the passive agents. There are passive agents named *kind* agents. If a kind agent A knows what agent, say B can answer the request R from U, A informs U of B when A cannot answer R. On receipt of the reply from A, U may send R to B.

2.2 Behaviour of agent

On receipt of a request R from U, an agent A behaves as follows.

1. A decomposes R into subrequests $R_1, \ldots, R_n$ $(n \geq 1)$.
2. A decides what agent can answer each subrequest. Suppose that an agent A_i is selected to execute each R_i $(i = 1, \ldots, n)$.
3. A asks each A_i whether A_i can answer R_i, and then negotiates with A_i on how A_i can answer R_i if A_i can answer R_i, e.g. how long it takes to answer R_i.
4. A asks A_i to answer R_i according to the way negotiated in the step 3. A_i answers R_i and sends back the reply RP_i to A.
5. A collects the results $RP_1, \ldots, RP_n$ from $A_1, \ldots, A_n$, respectively, and generates the result RP of R from $RP_1, \ldots, RP_n$. A sends RP to U. □

2.3 Structure of agent

The *CDBS* is composed of agents interconnected by a communication network CN. Each agent A_i is composed of two parts, i.e. *head* H_i and *body* B_i. B_i includes a database system DBS_i. DBS_i is composed of a database DB_i which is a collection of *objects*. In this paper, we assume that every *DBS* is homogeneous, i.e. *relational*. B_i manipulates objects in DB_i. For each object o, $Term_o$ is a collection of *terms* $\{t_1, \ldots, t_m\}$. Each t_j corresponds to a *keyword* in the information-retrieval systems. Here, let O be a set of objects and T be a set of terms in the *CDBS*. Each agent A has a subset O_A of O and a subset T_A of T. For two agents A and B, O_A and O_B may not be disjoint, and T_A and T_B may not either. That is, each object can exist redundantly in multiple agents. For each term t in T, $Obj(t)$ denotes a set of objects on t in O. $Obj_A(t)$ denotes objects on t in A if t is in $T_A(t)$, i.e. A *knows about* t.

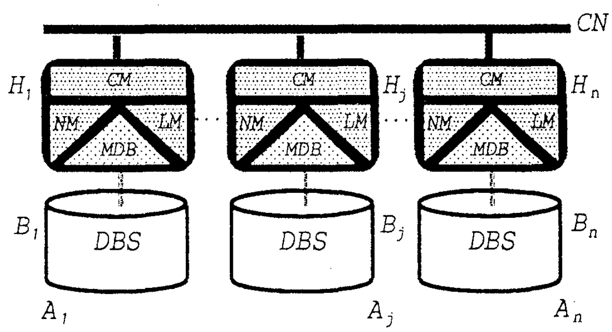

Figure 1: Agents

H_i is composed of a *metadatabase* MDB_i, *communication* module CM_i, *learning* module LM_i, and *negotiation* module NM_i [Figure 1]. Each term t in MDB_i denotes not only objects on t which A_i has but also another agents which A_i knows have t. If MDB_i includes t, A_i *knows* about t. A_i *directly knows* about t iff DB_i includes some object o on t, i.e. $t \in Term_o$. A *indirectly knows* about t if A_i knows about t but does not directly know about t. Here, although A_i has no object about t, A has t in MDB_A. Hence, A_i cannot obtain objects on t from DB_A but can ask another agent denoted by t in MDB_i which directly or indirectly knows about t. If A directly knows about t, DB_A has some objects on t. Hence, A can obtain objects on t in DB_A.

*Multi-Agent for Large Distributed System
†Chiaki Yahata, Makoto Takizawa
‡Tokyo Denki University

3 Negotiation

In this paper, we would like to think about only retrieval operations on multiple database systems. We consider a protocol for doing the negotiation among multiple agents.

First, an agent A takes a request R from a requester U. R is composed of a qualification Q and a *preference* P. Q is written as follows. Let t and *qual* denote a term and a qualification, respectively. *qual* is defined as $qual_1 \mid qual_2$, $qual_1$ & $qual_2$, $qual_1 - qual_2$, or t. For an expression *exp*, *Result*(*exp*) is the meaning of *exp* which is defined as follows. *Result*(t) is a set of objects on t, i.e. $\{o \mid o \in Obj(t)\}$. $Result(qual_1 \mid qual_2) = Result(qual_1) \cup Result(qual_2)$. $Result(qual_1 \,\&\, qual_2) = Result(qual_1) \cap Result(qual_2)$. $Result(qual_1 - qual_2) = Result(qual_1) - Result(qual_2)$. For a request $R = \langle Q, P \rangle$, $Result(Q)$ has to be obtained from the *CDBS* by the cooperation of the agents.

There may be multiple ways to obtain the result which satisfies Q, i.e. $Result(Q)$. P is used to select one way among them. P is a list $\langle P_1, \ldots, P_m \rangle$ or a set $\{P_1, \ldots, P_n\}$ ($m \geq 0$) where P_i is either a preference or a preference item. The list $\langle P_1, \ldots, P_m \rangle$ means that P_i is preferred to P_k if $i < k$. Let W_0 be a set of ways which $Result(Q)$ can be obtained.

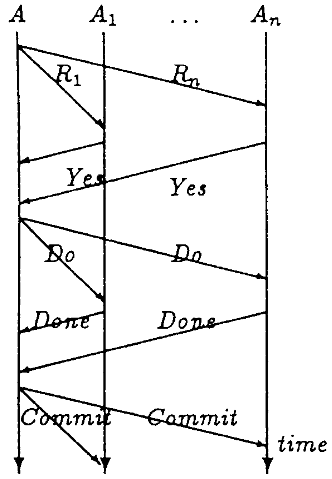

Figure 2: Negotiation procedure

A negotiation procedure of each agent A to obtain the result of request R from U is shown as follows.

[Negotiation procedure][Figure 2]

1. A takes a request $R = \langle Q, P \rangle$ from U. If A can answer R, A executes R. Otherwise, A decomposes R into subrequests $R_1, \ldots, R_n$ ($n \geq 1$), 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 agent A_i among the acquaintances of A which A thinks can answer R_i ($i = 1, \ldots, n$). If no agent can be found for R_i, R_i is further decomposed into smaller subrequests $R_{i1}, \ldots, R_{im_i}$ ($m_i \geq 2$). 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 an *Abort* message to $R_1, \ldots, R_n$. Then, R is tried to be differently decomposed by returning to the 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 can. If A_i replies A that A_i cannot answer R_i, A tries to find another agent among the acquaintances in the step 2. If A cannot find any agent for R_i, R_i is tried to be further decomposed into $R_{i1}, \ldots, R_{im_i}$ by returning to the step 2.
4. A asks A_i to execute R_i by sending a *Do* message 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* message to A. On receipt of the *Failure* from some A_i, A returns to the 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* message 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.□

If R_i changes the state of A_i, i.e. R_i is an update operation on DB_i, the update data obtained by R_i is saved into the secure storage, i.e. a log L_i of A_i at the step 4. Then, A_i sends the *Done* to A. On receipt of all the *Done* messages, A sends a *Commit* to $A_1, \ldots, A_n$. On receipt of the *Commit*, A_i changes the state by using the update data in L_i. This process is similar to the two-phase commitment .

4 Learning

An agent A can obtain newly terms and relations among terms from other agents through the negotiation. The terms and relations are stored in MDB_A. The process is named a *learning* to obtain the terms and relation among new terms which A has not had in MDB_A.

5 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 a negotiation protocol named a *three-phase negotiation* protocol among agents. By this procedure, agents can obtain the reply by taking advantage of another agents.

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 First International Workshop on Interoperability in Multidatabase Systems*, 1991, pp.239-242.

[3] Yahata, C., Hamada, S., and Takizawa, M., "Coorperating Database Systems," *IPSJ SIG Notes*, Vol.92, No.76 , pp.121-128.