PROLOG INTERFACE SYSTEM ON DISTRIBUTED NAVIGATIONAL DATABASE SYSTEMS

3N-1

Motoshi KATSUMATA and Makoto TAKIZAWA
Tokyo Denki University

1. INTRODUCTION

Current information systems have included various database systems interconnected by communication networks. Database systems provide different types of data models. In order to provide users with easy access to the distributed database system, a common interface system has to be provided. In this paper, we try to provide a common Prolog interface through which users can access more than one navigational database system. Navigational database systems are not only the conventional network database systems but also the file systems like UNIX file system. Also, we present how to execute them in parallel on the distributed navigational database systems.

2. QUERIES

Next, we present the query form. We take two databases DB1 and DB2 as an example.

DB1: s(@s, sname) p(@p, pname) b(@b, role)
DB2: sp(@sp,@p,@s) pb(@pb,@b,@p) bp(@bp,@b,@p)

Fig.2.1 Databases

A view is defined on the database like this.

```
sp(SNAM,PNAM) :- s(S,SNAM),sp(P,S),p(P,PNAM).
```

The view "sp" means that a supplier SNAM supplies parts PNAM. A query "find parts supplied by a supplier a" is written like this.

```
query(PN) :- sp("a", PN).
```

Here, PN is a target variable and "a" is a constant.

3. TRANSLATION OF SIMPLE QUERIES TO NAVIGATIONAL PROGRAMS

Now, we try to translate a simple query to a program which accesses the database.

3.1 PREVENTION OF MEANINGLESS BACKTRACKINGS

```
query(Y,PP) :- p(P,Y), pb(B,P),
               p(PP,X), b(B), bp(B,PP). --(1)
```

A navigational program for a query (1) is composed of nodes interconnected by one-way channels. Each node denotes an atom in query. Node A is composed of an ordered set D(A), state variable P(A), and two input ports, A.FS and A.NX, and two output ports, A.SC and A.FL. ST is a start node. OUT node outputs the answer. In the Prolog program, for example, if (4) fails, (4) sends a token from (4).FL to (3).NX[Fig.3.1]. However, since substitutionsobtained by (3) include no binding of the variable B in (4), the backtracking from (4) to (3) is meaningless. Navigational program by SP is shown in Fig.3.1. SP prevents the meaningless backtrackings [TAKI87a].

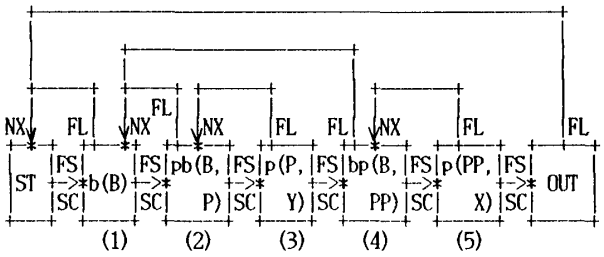

Fig.3.1 Navigational program by SP

3.2 REDUCTION OF REDUNDANT ANSWERS

Next, we try to get all the answer substitutions without redundant refutations. Here, let T be a navigational program. X and Y are some nodes in T. If there exist a edge from X.FL to Y.NX, let Y be parent(X). For a node A in T, suppose that there are n nodes $B_1,\ldots,B_n$ where A = parent(B_j) for j=1,...,n. Let T_j be a subtree of B_j. Suppose that a substitution θ_A is obtained by the resolution of A. Let Ans(B_j) be a set of refutations obtained from $T_j\theta_A$. In the Prolog program, for each $A\theta_A$, a cartesian product CA_A = Ans(B_1) x...x Ans(B_n) is obtained. Here, let T_{hk} be a subtree which includes targets in $T_1,\ldots,T_n$ (k=1,...,p, p≤n). In our method, only a projection of CA_A on $T_{h1},\ldots,T_{hp}$, i.e. AA_A = Ans(B_{h1}) x...x Ans(B_{hp}) is accessed. Since it is clear that $|CA_A| \geq |AA_A|$, we can get all the answer substitutions in less accesses to the database than the Prolog program. Here, we improved the backtracking for getting all the answer substitutions and introduced the new output ports LFL and RSC. Our navigational program is shown in Fig.3.2.

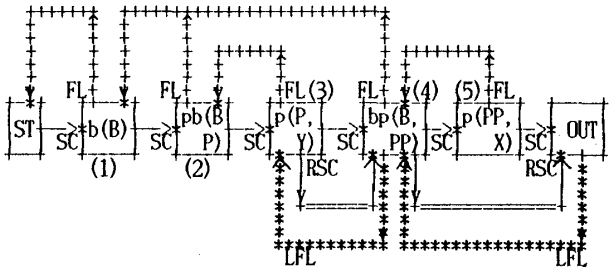

Fig.3.2 Navigational Program

[Proposition] For a given query Q, less tuples are accessed and less number of redundant answers are obtained by our navigational program than the Prolog one.■

PROLOG INTERFACE SYSTEM ON DISTRIBUTED
NAVIGATIONAL DATABASE SYSTEMS
Motoshi KATSUMATA, Makoto TAKIZAWA
Tokyo Denki University

4. OR PARALLELISM

Next, we try to execute a query in parallel. First, let us consider the following query.

```
query(T,U) :- A(X,Y), B(X,T), C(T), V(Y,U).
V(P,Q) :- D(P,Q,Z), E(Z).
V(P,Q) :- F(P,V,Z), G(V,Q), H(Z).
```

The views in the query are replaced by the right hand side of the views. For example, the following two queries are obtained.

```
query1(T,U) :- A(X,Y), B(X,T), C(T),
               D(Y,U,Z), E(Z).
query2(T,U) :- A(X,Y), B(X,T), C(T),
               F(Y,V,Z), G(V,U), H(Z).
```

One method to get the answers of query is to take the union of two results obtained by query1 and query2. However, in both queries, A, B, C are commonly evaluated. D,E in query1 and F,G,H in query2 can be executed in parallel. We try to reduce this redundant processing and execute them in parallel. So we introduce an object named OR for controlling this case to the access program. When constructing a navigational program for a given query, if a rule atom A is selected for some parent node P, an OR node is created as a child of P. Navigational program for a given query is shown in Fig.4.1. Here, node D,E and F,G,H are executed in parallel.

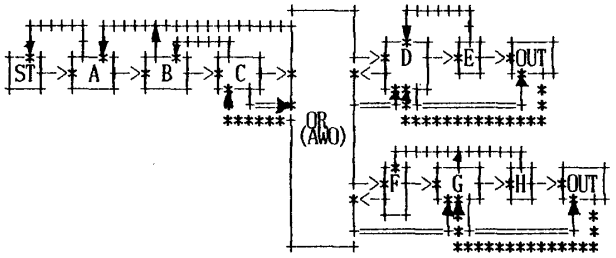

Fig.4.1 Navigational Program

There are two kinds of OR objects. i.e. all-wait(AWO) and one-wait(OWO) OR objects. Suppose that the subsequences at the right hand side of an OR node O do not include any target node. If one of the subsequence S_j is found to success, we do not have to wait for the completion of all the other subsequences at the OR node O. This type of OR node is named a OWO node. On the other hand, let us consider that the subsequences of the OR node O include some target node. It is clear that of some subsequence includes target node, all of the subsequences of O includes target node. In this case, even if all the answer substitutions are obtained from one subsequence of O, we cannot stop to wait for the other subsequences. When all the refutations for all the subsequences complete, we can backtrack to the ancestor node of O. This type of the OR node is named AWO node. OR parallelism is achieved without making intermediate files.

5. IMPLEMENTATION

Our distributed database system is composed of one mainframe m380, a super mini-computer a400, and five Sun workstations which are interconnected by Ethernet. As the database systems, m380 provides INGRES, and a400 and Suns provide UNIFY, and all of them provide UNIX file systems and indexed file system C-tree. Navigational objects are implemented for relations and files by using C language.

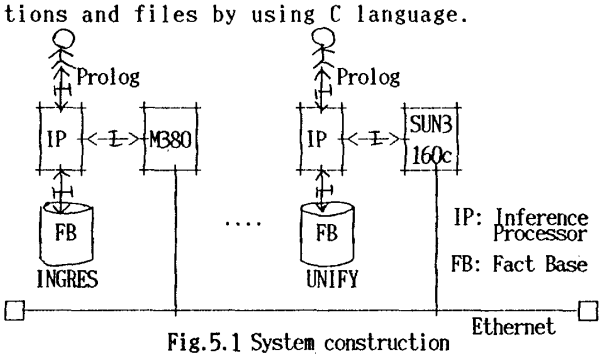

Fig.5.1 System construction

Objects communicate with another objects through the datagram socket, i.e. UDP provided by Unix. When a Prolog query Q is accepted, an access program T is created. For each node A, an object instance is created by issuing creatOBJ(A) to the object type. After all the object instances are created, addresses of objects, which correspond to the directed edges, are sent to them. By making the ST instance send a token, the parallel execution of the access program is started. When the ST instance terminates, the program terminates.

6. CONCLUDING REMARKS

In this paper, we have presented a Prolog-like query language interface on the multiple various navigational database systems like the conventional network database systems and Unix file systems. In our system, derivation of the redundant answers are prevented by accessing navigationally the database without making intermediate files. Also, we have shown the method to realize the OR parallelism so as to decrease the redundant answer substitutions.

REFERENCES

[TAKI87a] Takizawa,M., "Deductive Network Database System,"(in Japanese), Journal of JSAI, Vol.2, No.2, 1987, pp.182-191.

[TAKI87b] Takizawa, M., et al., "Logic Interface System on Navigational Database System," Lecture Notes in Computer Science, Springer-Verlag, No.264, 1987, pp.70-80.

[TAKI88] Takizawa, M., "Transaction Management by Prolog," Lecture Notes in Computer Science, Springer-Verlag, 1988.

[WARR]Warren, D. H. D., "Efficient Processing of Interactive Relational Database Queries Expressed in Logic," Proc. of the VLDB, 1981, pp.272-281.