STUDIA UNIV. "BABEŞ-BOLYAI", INFORMATICA, Volume I, Number 2, October 1996

A GRAPH MODEL OF DISTRIBUTED DATABASE SYSTEMS

ZOLTÁN KÁSA AND VIORICA VARGA

Abstract. In this paper we deal with a graph model of the distributed database systems, in which the nearest sites with some given properties can be easily found.

1. Introduction

A distributed database system consists of a collection of sites connected in a network by some kind of communication lines, in which each site is a properly database system, but a user at any site can access data at any other sites [2, 3]. By point of view of any user the system must look as a nondistributed one. There are several reasons why, for a distributed system to be successful it must be relational.

A system supports data fragmentation if a given stored relation can be divided up in pieces (so-called fragments) for physical storage purposes. Fragmentation is for performance reasons. In such a case data can be stored at the locations where they are most frequently used, operations are local, network traffic reduced.

There are two kind of fragmentations: horizontal and vertical. Let $\mathcal{R}[A_1, A_2, \cdots, A_n]$ be a relation, where $A_i, i = \overline{1,n}$ are attributes. A horizontal fragment is obtained by a restriction: $\mathcal{R}_i = \sigma_{cond}(\mathcal{R})$, where *cond* is the guard condition. The initial relation can be reconstructed by union of the horizontal fragments: $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdots \cup \mathcal{R}_m$.

A vertical fragment is obtained by a projection operation:

$$\mathcal{R}_j = \prod_{\{A_{k_1}, A_{k_2}, \cdots, A_{k_p}\}} (\mathcal{R}),$$

Received by the editors: January 27, 1997.

1991 *Mathematics Subject Classification.* 68P15, 68R10, 68M10.

1991 *CR Categories and Descriptors.* C.2.0 [Computer-Communication Networks]: General – data communications; C.2.1 [Computer-Communication Networks]: Network Architecture and Design – distributed networks; G.2.2 [Discrete Mathematics]: Graph Theory – graph algorithms, breadth-first search.

where $A_{k_l}, l = \overline{1,p}$ are attributes. Such a fragmentation must be nonloss, the initial relation would be reconstructed by join of vertical fragments: $\mathcal{R} = \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \cdots \otimes \mathcal{R}_q$.

A system with fragmentation should also support fragmentation independence, i.e. users should be able to work as in the case of nonfragmented data. It is the of the system optimizer to determine which fragments need to be physically accessed in order to satisfy any given user query.

A system supports data replication if a given stored relation or a fragment can be represented by many distinct copies or replicas, stored at many distinct sites. The main advantage of data replication resides in local operations on data. The major disadvantage is that updated replicated data must be updated at all sites where there exist copies. Replications must be transparent to the user, and the system optimizer is responsible for choose the nearest sites to satisfy a user request.

In a distributed system the query optimization is more important than in a centralized one. In a query which involve several sites, there will be many possible ways to access the corresponding data, and the strategy which is used is crucial. Strategies can be easily modeled by graphs.

2. The Graph Model

We consider a graph $\mathcal{G} = (V, E)$ associated with a distributed database system: the sites are represented by vertices (V) and direct connections between sites by corresponding edges (E). The data stored in the system are characterized by some properties, the same data by same properties anywhere they are. The same data can be found at many sites.

Problem 2.1. *For a given vertex i find the nearest sites which satisfy a given property.*

The corresponding graph problem is:

Problem 2.2. *Let given a graph, in which each vertex is characterized by some properties. For a given vertex find the nearest vertex which satisfy a given property.*

Let us denote by $\Gamma(x)$ the set of all adjacent vertices of x (vertices which are connected by an edge to x).

The following breadth-first algorithm [1] determines the nearest vertex from i with the given property P. When a vertex is visited it is marked and inserted in an initially empty queue Q. At the beginning all vertices are unmarked.

The procedure *Visit (x)* visits the unmarked vertex x, marks it, tests if P is true for it, if it is, then exits, if it is not then insert x in the queue Q:

procedure Visit (x):
  if P is true for x **then** exit

else mark x
insert x in Q

end if
end procedure

Problem 2.1 is resolved by the following algorithm:

procedure Nearest (i):
Visit (i)
while Q is nonempty **do**
let x be the removed front element of Q
for all unmarked $j \in \Gamma(x)$ **do** Visit (j)
end while
end procedure

Let us denote by $\mathcal{P} = (P_1, P_2, \cdots, P_n)$ the set of properties characterizing the data in the system, and by m the number of sites ($m = |V|$). Let us consider the matrix

$$\mathbf{d} = (d_{ij}) \quad \begin{matrix} i = \overline{1,m} \\ j = \overline{1,n} \end{matrix}$$

in which $d_{i,j} - 1$ represents the minimal distance from the site i to the site for which the property P_j is true, if $d_{ij} > 0$. If there is no sites with property P_j, reachable from the site i, then $d_{ij} = 0$.

Problem 2.3. *For a given vertex i find the minimal distances to vertices which satisfy the properties P_j.*

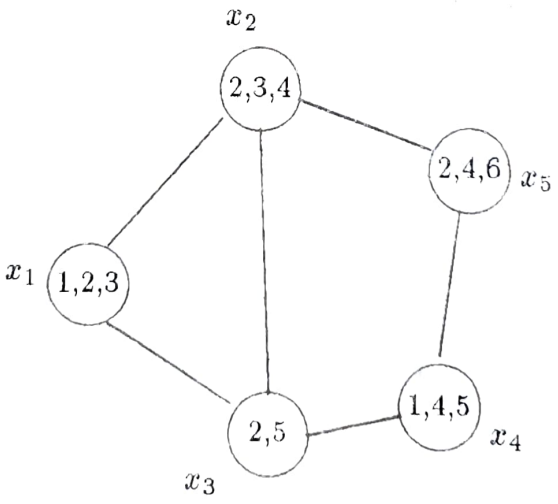
Procedure *Visit2 (i)* visits the unmarked vertex i, marks it, and assign to d_{ij} the correspondent distance k if P_j is true for i. To increase correctly the value of k we will use two queues.

procedure Visit2 (i):
for $j = 1, 2, \cdots, n$ do
if $d_{i,j} = 0$ and P_j is true for i **then** $d_{i,j} := k$ **end if**
end for
mark i
insert i in Q_1
end procedure

The algorithm which resolves the problem 2.3 is the following:

procedure Distance (i):
  for $j := 1, 2, \cdots, n$ **do** $d_{i,j} := 0$ **end for**
  Initialize the queue Q_1
  k := 1; Visit2 (i); k := 2
  Move Q_1 to Q_2
  Repeat
    Initialize Q_1
    while Q_2 is nonempty **do**
      let x be the removed front element of Q_2
      for all unmarked $y \in \Gamma(x)$ **do** Visit2 (y) **end for**
    end while
    Move Q_1 to Q_2; k:=k+1
  until Q_1 becomes empty
end procedure

Using the matrix **d** we can characterize the entire system. Let us denote by $\kappa(k)$ the number of copies of the data characterized by P_k, this is the number of 1's in the k^{th} column of **d** (see the example).



	P_1	P_2	P_3	P_4	P_5	P_6
x_1	1	1	1	2	2	3
x_2	2	1	1	1	2	2
x_3	2	1	2	2	1	3
x_4	1	2	3	1	1	2
x_5	2	1	2	1	2	1
$\kappa(k)$	2	4	2	3	2	1

$\kappa = \frac{14}{6} = 2.33\ldots$

We can define the average value of the copies:

$$\kappa = \frac{\sum_{k=1}^{n} \kappa(k)}{n}.$$

This value can vary between 0 and $m = |V|$. The case $1 \leq \kappa \leq m$ is interesting only. If $\kappa < 1$, then there exists at least one k for which $\kappa(k) = 0$, i.e. the corresponding property P_k is false for all sites. If $\kappa = 1$ each data in the system exists only at a single site, but if $\kappa = |V|$ each data exists at all sites.

If $\kappa(k) = 0$ for a given k, then the corresponding column (the k^{th} column) in the matrix **d** must be identically equal to 0.

If in the matrix **d** there exists an element equal to 0, but not the entirely column, the corresponding graph is disconnected.

Problem 2.4. *Having the distance matrix* **d**, *find a path from the site* i *to the site for which* P_j *is true.*

If $d_{i,j} = 0$ then the target site cannot be reached, the graph is disconnected. If $d_{i,j} = 1$ then the target site is identical to the source one. The following algorithm is given for $d_{i,j} > 1$, and finds a the possible path with the given property.

```
procedure Path (i, j):
    mark i
    k := d_{i,j} − 1
    while k > 0 do
        for all unmarked y ∈ Γ(i) do
            mark y
            if d(y, j) = k then i := y
                                memorize y
                                k := k − 1
                                exit the loop for

            end if
        end for
    end while
end procedure
```

If in the distributed system a direct connection between two sites is broken, we must recompute the distance matrix. This can be made easily by the above algorithms.

3. Conclusions

Breadth-first search algorithms are well-known in graph theory. We have presented a breadth-first search algorithm to find data characterized by some conditions in a distributed database system. This algorithm can be used in a dynamic manner i.e. after any modification in the structure of the system (lost connections, new connections) a simple recalculation of the distance matrix ensures the maintenance of the system.

References

[1] S. Baase, *Computer algorithms: Introduction to design and analysis*, Addison-Wesley Publishing Co., 1983.
[2] C. J. Date, *An introduction to database systems*, Addison-Wesley Publishing Co., 1995.
[3] M. T. Özsu, P. Valduriez, *Principles of distributed database systems*, Prentice Hall, 1991.

Babeş-Bolyai University, Faculty of Mathematics and Informatics,
RO 3400 Cluj-Napoca, str. Kogălniceanu 1, Romania
E-mail address: {kasa,ivarga}@cs.ubbcluj.ro