Identification of partitions in distributed deductive databases

Hussien Hassan Aly
Computers and Automatic Control Dept, Faculty of Engineering, Alexandria University

In this paper, the problem of describing and identifying partitions of a general recursive relation that is fragmented over many sites of a deductive database is considered. We assume that the partitioning depends on the transitive closure relationships between the elements of the relation and cannot be described by direct rules or obvious predicates. This problem was studied previously by other researchers, for acyclic relations using the lattice approach. In this paper, no restrictions on the type of the relation are imposed, so the relation can be cyclic relation or acyclic one. Since using lattices will fail in case of existence of cycles, the concepts of boundary nodes and landmark nodes of strongly connected components are used to properly and uniquely describe each partition of the relation. We provide a simple characterization of the necessary and sufficient nodes that must be included in the description of arbitrary fragments of the relation. We present a linear algorithm that produces an optimum description set in case of acyclic relations and a near optimum description in case of cyclic one.

في هذا البحث، نقوم بدراسة عملية التوصيف والتعرف على أشتات العلاقات المعاودة والموزعة على كثير من مراكز الحاسبات. ويفترض أن تقسيم هذه الأشتات معتمد على ارتباط العناصر من خلال العلاقة المتعدية المتعدية التامة والتى عادة ما يصعب أو يستحيل وصفها بقوانين أو نماذج رياضية مباشرة . وهذه المسألة سبق دراستها من قبل بعض الباحثين بالنسبة لكون العلاقة المعاودة غير دائرية (acyclic) باستخدم النموذج الرياضى الشبكى (Lattice). أما فى هذا البحث، فنحن لا نضع أى شروط على نوعية العلاقة بحيث يمكن أن تكون دائرية أو غير دائرية. وقد استخدمنا مفاهيم الارتباطات ونقاط الحدود وعلامات التمييز للوصول إلى توصيف تام، اساسى وكافى، لأقل عدد من النقاط التى يمكن أن يحتويها هذا التوصيف. وكذلك عرضنا خوارزم خطى للوصول على افضل توصيف فى حالة العلاقات الغير دائرية أو للحصول إلى توصيف قريب من الأفضل فى حالة العلاقات الدائرية.

Keywords: Deductive databases, Fragmentation, Cyclic recursive relations, Boundary nodes, Land mark nodes.

1. Introduction

Relations that can be part of a recursive or a transitive closure rule in a deductive database are sometimes called recursive relations. The main characteristic of such relations is that the locality of referencing the relevant tuples not only depends on the query, but also on how these tuples are related to each other. In other words, the relevant data to a query depends on the transitive closure of the query node in the corresponding relation graph.

Fragmentation of recursive relations in deductive databases is studied extensively in the literature. See for example Refs. [1-5]. The main goal of these research studies is to arrive to an optimal fragmentation of the relations and distribute these fragments through the network to achieve best performance using the parallel and distributed computing capability of the network. However, each of these studies considers a special performance metric to optimize. Many of them use the classical approach to horizontally and/or vertically decompose the relation based on well defined predicates or hash functions as in [6] regardless of the locality behavior of the relevant data. In [2], different strategies were suggested to get small and equal size partitions of a recursive relation suitable for some parallel computation of the transitive closure described in [3].

One of the problems associated with fragmenting a recursive relation, taking into consideration the locality of relevant data, is how to describe and identify the resulting fragments given that no obvious rule is used in partitioning. In other words, given a node

in the corresponding graph, there is no direct predicate or a hash function that can identify the fragment to which it belongs. This is of vital importance since these fragments will be distributed throughout the network and we need to access them to complete the query evaluation task.

In [7], a method based on lattice structures was proposed to solve this problem in the case of acyclic recursive relations, i.e. when the graph corresponding to the relation is guaranteed to have no loops or cycles. The idea is to describe each fragment by a set of lattices that collectively contain the nodes and only the nodes of a fragment. Since the graph is assumed acyclic, all elements are in a partial order relation. Each lattice in turn is described by two elements, namely: the *min* element and the *max* element. The size of the description is proportional to the number of lattices generated. Ref. [7] also, provided a proof that obtaining an optimal number of the lattices is an NP-complete problem, so a heuristic to get a sub-optimal one was given.

The condition that the graph G must be cycle-free is a very restrictive condition for any practical implementation. Many practical relationships, especially that arise in document analyses and data mining, are of cyclic nature. Testing a large graph for cycles is costly and transforming a cyclic graph into an acyclic one may lead to loosing some information. Also, the continuous maintenance of insertions and updates to guarantee that the graph remains cycle-free is very costly.

In this paper, we relax this condition and consider a general graph. Since there is no partial ordering relationship between nodes of the general graph, we cannot use the lattice structures to describe the fragments any more. In our model, we will use the connectivity principles and graph-theoretic concepts to derive the description of the fragments and identify them.

The organization of this paper is as follows. In Section II, the description of partitions is discussed and a necessary and sufficient criterion, which provides the minimum number of nodes required for unique description and identification of the partitions, is presented. In Section III, the algorithms needed to construct these descriptions together with their complexities are presented together with a comparison of our method with the lattice approach is given. In Section IV, our conclusion is given.

2. Description of fragments

As discussed in the introduction, the relevant data to a query (locality) are likely to be related to the transitive closure of the query node in the corresponding relation graph.

In this paper, it is assumed that the whole graph G = (V, E), which describes the relationship between nodes, is stored in each site. However, the table containing the detailed description of the nodes and their attributes is partitioned into several smaller tables (fragments). Each site stores only the relevant fragment of the detailed tables. This environment is the same as the one considered in [7] and can be found in systems that use deductive database tools in document analysis and in distributed data-mining operations. In such environment, the size of the fact data associated with each node is relatively too large to be stored with the relation tables. Sometimes, for security and copyright reasons the fact data are stored in specific sites. Also, there may be many kinds of relationships between the nodes that make it difficult to store the actual facts with tables representing these relationships.

A fragment description table (FDT) is used to maintain the location and description of each fragment. A copy of the FDT is stored in each site. The space cost of the method will depend on the relative size of the relation representing the graph G compared to the size of the detailed table of the nodes. The overhead of repeating G at every site is obviously reduced as the relative size of G gets smaller.

We assume that the graph G is represented as adjacency list structure. For simplicity, this structure will be a list of records of the form (FromNode, ToNode). However, the concepts and methods can be applied with little or no modifications to other

data structures.

When a site has a query corresponding to a node in the graph G, it uses G to derive the identity of the fragment that contains that node, and then consults the FDT to locate the corresponding site that has this fragment. The problem now is how to describe and identify fragments in such a situation where the original data are partitioned using no obvious predicate rules.

Example 1

Consider the two graph structures of Fig. 1 of cyclic and acyclic relationships. Each node represent a unique document and edges represent some relationship (e.g. contain related key words) between the document. The documents are distributed on three different sites F1, F2, and F3. It is impractical, if feasible at all, to store the actual documents in each site for doing search of related documents. However, we can easily store the whole graph representing this relationship at each site. Since there is no clear clue of how these documents are distributed and partitioned, we need a method to relate nodes in the graph to their site location.

In our model, we will use the connectivity principles to derive the description of the fragments. First, we need some definitions that will be used in deriving the description. In the rest of the paper, the following notations are used: $G = (V, E)$ is a general directed graph, F_i is a partition of the nodes in V such that $\bigcup_{\forall i} F_i = V$, $F_i \cap F_j = \Phi \forall i \neq j$ and. $G_{F_i} = (F_i, E_i)$ is the subgraph induced by the nodes in F_i where $E_i \subseteq E$.

Definition 1:

In the subgraph G_{F_i}, a node $v \in F_i$ is a boundary node if there is an edge $e = (v, v') \in E$ from this node to an outside node $v' \notin F_i$.

Definition 2:

In the subgraph G_{F_i}, a node $v \in F_i$ is a *terminal node* if there is no edge $e = (v, v') \in E$ from this node to any other node v'. In other words, the out-degree of a terminal node is zero.

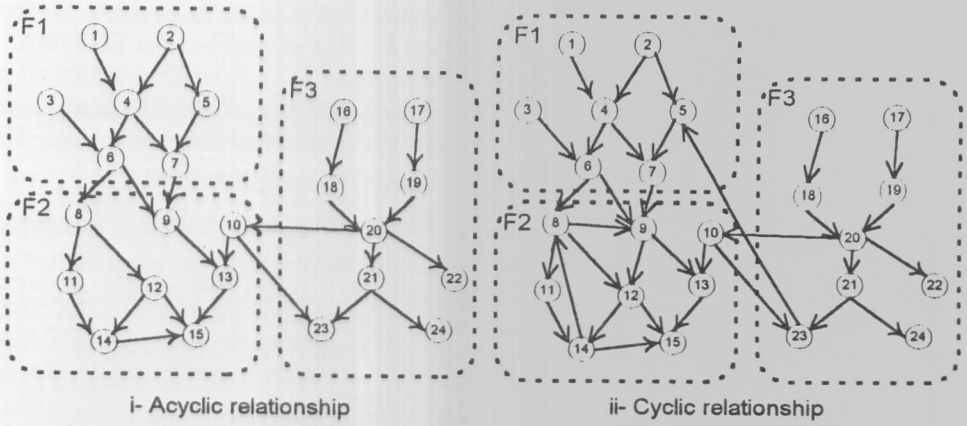

Fig. 1. Three fragments of a relationship.

Definition 3:

In the graph G_{F_i}, a node $v \in F_i$ is a *candidate* landmark node if it belongs to a strongly connected component of G_{F_i}.

Definition 4:

A set of candidate landmark nodes is called *landmarks*, Γ, if for each strongly connected component C in G_{F_i}, there is a node $v \in C \cap \Gamma$. The set of all landmarks will be denoted by $LM(F_i)$.

Definition 5:

In the subgraph G_{F_i}, a node $v \in F_i$ is a *contributing node* if it belongs to any of the set of boundary or terminal nodes otherwise it is a *non-contributing node*. The set of all contributing nodes in F_i will be denoted by $CONT(F_i)$.

Definition 6:

The reachable set from node v, "reach(v)" is the set of nodes reachable from a node $v \in F_i$ by traversing the graph G.

Example 2:

To illustrate the above definitions, consider the graph in Fig. 1-ii. The set of boundary nodes in F1, F2 and F3 are {6,7}, {10}, and {20,23}, respectively. The terminal nodes are Φ,{15},{ 22, 24}. In F2, {8,9,11,12,14} is a set of candidate landmark nodes and the set {14} is a possible landmark.

Now given a query node, we want to identify the partition to which it belongs. The idea of the procedure that is suggested here is to describe each partition by a special subset of its nodes. Then, traverse the graph G - which is available in every site - starting from the given query node until the first special node(s) is reached. Then use one of these special nodes as an index to the partition that contains the given query node.

Notice that this method requires that during traversal, a guarantee that at least one special node of the subset describing the current partition will be reached before any node in other partitions. Although this procedure is simple, the main difficulty is in identifying those special nodes that can be used in the identification of a partition.

One obvious trivial - but very costly - identification is to use a list of all the nodes in a partition to describe it, so the query node itself will immediately identify the partition.

The aim is to use the smallest set of nodes to identify each partition and guarantee the correct behavior of the above procedure. In [7], the lattice approach, a proof that obtaining an optimal number of the lattices to describe the fragments is an NP-complete problem and provides a heuristic to get a sub-optimal one.

In this paper, we do not use the lattice approach; instead we use simple graph theoretic concepts to derive a necessary and sufficient characterization for the minimal set of nodes that can be used in describing the fragments. Due to the existence of cycles, to construct an optimal description set, the problem is exponential in the number of strongly connected components in the subgraph. Although the number of strongly connected components is far smaller than the number of edges, we provide a linear algorithm to construct a near optimal description set. The following theorem provides a lower bound on such sets.

Theorem 1:

In G, the set of contributing nodes, $CONT(F_i)$, is the smallest set necessary to identify or describe the partition F_i.

Proof: Suppose that F_i is identified uniquely using some set S of nodes. This means that starting traversal of G from any node in F_i, we surely arrive to a node in *S* before any other node in other partitions. We will prove that $CONT(F_i) \subseteq S$.

Let $v \in CONT(F_i)$ and $v \notin S$ be the starting node. Two cases are there:

1. v is a terminal node $\Rightarrow$ reach(v) = {v} $\Rightarrow v \in S \Rightarrow$ contradiction.
2. v is a boundary node $\Rightarrow$ reach(v) will contain nodes from other partitions. $\Rightarrow$ cannot identify partition

uniquely ⇒ contradiction.
This imply that S ⊇ CONT(F_i).

Although theorem 1 provides a necessary set, this set is not sufficient in the general case where the graph G may contain cycles. Theorem 2 will provide a necessary and sufficient set to identify the partitions.

Theorem 2:
Let $\Gamma_k \in LM(F_i)$ such that $|CONT(F_i) \cup \Gamma_k| = \underset{j}{Min}(|CONT(F_i) \cup \Gamma_j|)$. Then the set $CONT(F_i) \cup \Gamma_k$ is the smallest necessary and sufficient set to identify F_i.

Proof: To prove that the set is sufficient to identify F_i, we must prove that if we start traversing with any node $v \in F_i$, we definitely will arrive to a node in $CONT(F_i) \cup \Gamma_k$ before leaving that partition.
If v, the starting node, is already a member of $CONT(F_i) \cup \Gamma_k$, then it is done. Now suppose that v is not a member of $CONT(F_i) \cup \Gamma_k$, and we start traversing G starting from v. There are three cases:

1. Traversal leaves F_i to another partition.
2. Traversal never leaves the partition F_i and never stop due to the existence of cycles.
3. Traversal stops inside F_i.

Case 1; implies that at least one boundary node $v_b \in CONT(F_i)$ is reached before leaving the partition to another one.
Case 2; implies that it must pass through all the nodes of at least one strongly connected component in G_{F_i}. So it must reach a node $v_c \in \Gamma_k$.

Case 3; implies that traversal reached a terminal node $v_t \in CONT(F_i)$. In other words, we must reach a node in $CONT(F_i) \cup \Gamma_k$ before any node in other partitions.

The $CONT(F_i) \cup \Gamma_k$ the minimum necessary set follows proof that directly from theorem 1 and the condition that $|CONT(F_i) \cup \Gamma_k|$ is minimum.

3 Constructing partition descriptions set

In order to construct a description for a fragment F_i, we have to find the members of $CONT(F_i)$ and the members of Γ_k as described by theorems 1 and 2.. We first present the algorithm for obtaining the description, then we discuss it and give total complexity. The correctness of the algorithm is based on the proof of previous theorems.

3.1. Procedure to compute the description set

Given G and F_i, the main steps of the procedure is given in Fig. 2. Steps are numbered for reference. The output of the procedure is a description_set that is minimal necessary and sufficient to describe and identify the fragment F_i. In the following, we explain each step in the Figure.

1. Determine the induced graph G_{F_i}. This is the graph that consists of all vertices in F_i and edges in the original E with both end points in F_i. The time complexity of this step is linear in the size of E, $O(E)$, since we can inspect each edge in turn and select it if it satisfies the condition.
2. Determine the set of leaving edges E_l. This is the set of all edges from a vertex in F_i to a vertex outside F_i. The time complexity of this step is also $O(E)$ and can be executed in parallel with the previous step.
3. The contributing set $CONT(F_i)$ is the union of two sets. The first one gives the boundary nodes, which are the starting nodes of edges in E_l. The second one gives the terminal or sink nodes which are the nodes in F_i that never appear as FromNode in any edge. The complexity of this step is at most $O(E)$.
4. Determine the strongly connected components of G_{F_i}. A well-known efficient algorithm such as Tarjan's depth-first search algorithm [8,9] can be used here. The function get_strong_comp() will return the strongly connected components, each

1. G_{F_i} = (F_i, $\{(x,y) \in E | x \in F_i \wedge y \in F_i\}$); // The induced graph
2. E_l = $\{(x,y) \in E | x \in F_i \wedge y \notin F_i\}$; // The set of leaving edges
3. CONT(F_i) = $\{x|(x,y) \in E_l\} \cup (F_i - \{x|(x,y) \in E\})$.
4. SC = get_strong_comp(G_{F_i}).
5. not_covered = ϕ
6. For each $C \in SC$ if $C \cap$ CONT(F_i) = ϕ then not_covered = not_covered $\cup$ {C};
7. rest_nodes = get_min_cover(not_covered);
8. description_set = CONT(F_i) $\cup$ rest_nodes

Fig. 2. Procedure to construct the description of a fragment.

in a form of a set of the nodes constituting the components. The complexity of this step is O(E).

5. We initialize a temporary set that will be used to hold the uncovered components by nodes in CONT(F_i). In other words, not_covered will hold the components that have no common node with the contributing set of nodes.
6. For each strongly connected component, if no common node between it and CONT(F_i) then append it to the uncovered set. So, the next step will only operate on these uncovered strongly connected components.
7. The function get_min_cover () will return a set of candidate landmark nodes for all the not_covered strongly connected components. If we insist on getting the optimal cover, then this function has to enumerate all possible covers and select the minimum. This can be seen as a *set cover* problem and is proved to be NP-complete [9]. For arbitrary graphs, the number of strongly connected component may be large and the evaluation of this function may become not practical. However, we can decide on having a near optimal solution by using the articulation points of the resulting components of step 6 or just using any node from each of them. The time complexity in this latter case is O(N_s); where N_s is the number of strongly connected components found in G_{F_i}.
8. In the last step, the description set is formed by the union of the contributing nodes with the nodes returned from get_min_cover().

Example 3:

Let us apply the above procedure for each fragment in Fig. 1-i and ii. The following table gives the description of each fragment. Notice that since only F2 in Fig. 1-ii has a strongly connected component, a landmark node "14" is included in the description.

Fragment	Description (Figure 1-i)	Description (Figure 1-ii)
F1	6,7	6,7
F2	10,15	10,14,15
F3	20,22,23,24	20,23,22,24

3.2. Complexity of the procedure

As seen from the steps above, step 7 is the most crucial one, considering the time complexity of the method, if the absolute optimum description (in terms of the number of nodes) has to be used. All other steps has O(E) in the worst case for cycic and acyclic graphs. In practice, we can resort to a near optimal description. For example, by including the articulation nodes (two strongly connected components can share only one node) or just any node from each of the uncovered components. In this case, the complexity of step 7 is very efficient and reduced to O(N_s), where $N_s << E$. So, the whole

procedure will be O(E).

3.3. Comparison with the lattice approach

The lattice approach given in [7] is valid only for acyclic recursive relations. Our approach is valid for both cyclic and acyclic without any change. Also, in our approach, we can construct optimal description set for acyclic relations in linear time in contrast to the NP-complete problem with size $|E|$ in case of lattice approach. This is *not* saying that we discover a linear solution to an NP-complete problem, but the lattice approach is too complex for the requirement of just describing arbitrary acyclic graph. However for graphs with cycles, to get an optimal description we may have to solve a set-cover problem (NP-complete) with size equal to the number of strongly connected components. This latter size is usually very small and negligible compared to $|E|$.

The heuristic in [7] to get sub-optimal solution for acyclic graphs, as given there, is of $O(|E| + (N_a + N_d) \Sigma |A_n| + |CL|^2.|F_i|)$, where N_a and N_d are the maximum number of ancestors and descendants, respectively, of a node in the subgraph, A_n is the number of ancestors of the *min* node n, CL is the number of suitable candidate lattices in the graph. Our method is linear, $O(|E|)$, and give the optimal in case of acyclic graphs. To conclude the comparison, it is instructive to compare the result of applying the lattice approach and our approach to the acyclic graph in Fig. 1-i. The following table gives an optimum lattice description of the fragments in terms of a pair (max-element, min-element) for each lattice. Comparing it with the first column in the table of Example 3, we see that our method is more simpler and the description contain fewer nodes.

Fragment	Lattice description
F1	(3,6), (1,6), (2,7)
F2	(8,15), (9,15), (10,13)
F3	(16,23), (17,24), (20,22)

4. Conclusions

We use graph theoretic concepts of connectivity and boundary nodes to describe partitions of a recursive relation in a deductive database. We provide a simple characterization of the necessary and sufficient nodes that must be included in the description of arbitrary fragments of the relation. A linear algorithm is presented that produce the optimum description with the minimum number of nodes in case of acyclic relations. In case of cyclic relations, the algorithm produces a sub-optimal description. The optimum in this latter case can be found by solving an NP-complete problem of size equal to the number of strongly connected components in the graph representing the fragment of the relation which is usually very small compared to the number of edges in the graph. A comparison between our approach and the lattice approach given in [7] reveals that our approach is superior to the lattice approach.

References

[1] W Nejdl, S Ceri, and Wiederhold, G; "Evaluating recursive queries in distributed databases"; IEEE Trans. Knowledge and Data Eng. Vol.5 (1), pp.104-120 (1993).

[2] M.A.W. Houtsma, P.M. Apers, and Schipper G.L.V.;"Data fragmentation for parallel transitive closure strategies."; proc. IEEE 9th Int'l Conf. on Data Eng, pp. 447-456 (1993).

[3] M.A.W. Houtsma, P.M. Apers, and Ceri, S.; "Distributed transitive closure computation: the disconnection set approach."; proc. 16th Int'l conf. on Very Large Databases; pp. 335-346 (1990).

[4] S. Ganguly, A. Silberschatz, and Tsur, A.;" A framework for the parallel processing of datalog queries"; proc. ACM-SIGMOD Int'l Conf. Manag. of Data; Atlantic city, pp.143-152 (1990).

[5] Y. Huang, and J.; Cheiney, "Parallel computation of direct transitive closures"; Proc. IEEE 7th Int'l Conf. on Data Eng. pp.192-199 (1991).

[6] J. Cheiney, and C. Maindreville," A parrallel strategy for transitive closure using double hash-based clustering";

Proc. Int'l Conf. on Very Large Databases (VLDB) pp.347-358 (1990).

[7] S. Pramanik, and S. Jung, " Description and identification of distributed fragments of recursive relations"; IEEE Trans. Knowledge and Data Eng. Vol.8(6), pp.1002-1016 (1996).

[8] R. Trajan, "Depth-First Search and linear graph algorithms." SIAM J.Comput., Vol.1, pp.146-160 (1972).

[9] S. Even, Graph Algorithms. Computer Science Press (1979).

Received August 10, 1999
Accepted April 15, 2000