In a distributed relational database, an operation that produces the union of fragments stored at different sites at a specified site is called a distributed union operation. In implementing the distributed union operation, operations other than the union operation, such as intersection and difference, may be used to facilitate the union. Optimization of the distributed union operation when the fragments are not disjoint requires the ability to predict the size of the result of set operations. A minimal basis of the result of multiple non-disjoint horizontal fragments of a relation is derived. This is used to develop a model for the optimization of the distributed union operation. As the problem is NP Hard, heuristic and lower bounding procedures are developed for a special case.

Even in today’s geographically dispersed national and multinational organizations, data generated at a location is commonly used at that location. This characteristic of data usage, in which the transactions at a site refer primarily to a proper subset of all the data stored in the database, is called locality of reference (Peebles and Manning, 1975). In environments in which data has this characteristic, it is advantageous to store the data at the geographic location where it is used (Sacco, 1986; Segev, 1986). In a relational database environment the relations may be horizontally fragmented and distributed based on the usage patterns.

Even in such an environment some transactions refer to data at sites other than the data origination site. These transactions are called global transactions. Optimization of global queries includes, in addition to the usual consideration of processing and input/output for local queries, the selection of fragments and operations to reduce communication costs. In this paper we focus on the minimization of cost of implementing global queries. Consider the following example to illustrate these concepts.

Example 1. A manufacturing firm has two plants, one in Dallas and the other in Seattle. Each plant has an inventory of spare parts. By and large, requests for parts are met locally. On occasions when the demand is not met locally, the stock at the other plant is checked. A relation `PARTS` is used to keep track of the spare parts. It has an attribute LOCATION with domain `{DALLAS, SEATTLE}` that describes the locality of the part. Most of the transactions originating in Dallas deal with parts in Dallas. The Seattle transactions are similar. This is locality of reference. In this case it would be advantageous, from the standpoint of reducing communication costs, to create two relations, each a subset of `PARTS`, with `PARTSD` having tuples of parts in Dallas and `PARTSS` with tuples of parts in Seattle. Transactions that deal with parts not available locally are global transactions.

By and large, Union operations arise naturally in the implementation of global queries on horizontally fragmented databases. To illustrate this, consider the previous example. Suppose that the relation `PARTS` is fragmented into `PARTSD` and `PARTSS` and stored at Dallas and Seattle, respectively. Further suppose that the headquarters at Chicago needs some data describing parts used at any of the plants. A common strategy for query execution, is to reformulate the query on `PARTS` to refer to `PARTSD` =
PARTSS and then optimize and execute it. The reformulated and simplified query includes unions. The current literature on query optimization in horizontally fragmented relational databases does not explicitly consider the optimization of the underlying union operation. Some examples from the current literature illustrate this point.

A basic optimization strategy is to use the relational operators that reduce the size of fragments, such as selection and projection, before transmitting the fragments for a union. Referring to the example again, a query about red parts, \( \pi \sigma^{\text{color}=\text{red}}(\text{PARTS}) \) would be replaced by \( (\pi \sigma^{\text{color}=\text{red}}(\text{PARTS}_1)) \cup (\pi \sigma^{\text{color}=\text{red}}(\text{PARTS}_2)) \). The selection and projection operations are carried out at the local sites. This is the idea behind Ceri and Pelagatti’s distributed query optimization strategy (1983). They consider the operator tree and recommend pushing down selection and projection below union. However, they do not specify how the union of distributed fragments is to be performed. For instance, in the example described above, the location of the union operation is not specified. More generally, they do not consider replacing \( (\cup \pi \sigma^{s}) \) with some other expression such as \( (\pi \sigma^{s_1} \cup \pi \sigma^{s_2}) \) where different unions may be performed at different nodes. Such considerations may result in savings when the fragments are non-disjoint. This is illustrated in Example (b) later in this section.

Other examples from the current literature on query optimization that ignore the union operation are works by Segev (1986) and Yu, et al (1985) where they discuss the join of horizontally fragmented relations. Segev assumes that the result of semijoins will be sent to the query node for union while Yu et al assume that the union of the results will be performed at the site with the largest fragment. Neither consider distributing the union operation itself. A common assumption (Garfinkel and Nemhauser, 1972) that is used to justify this strategy for the implementation of the unions is that the fragments are disjoint.

While restricting our attention to the union operation in this paper and ignoring queuing effects in the communication network, a number of assumptions made in the earlier literature on query optimization of global queries in a horizontally fragmented databases are relaxed. In particular, in this analysis,

(i) fragments may not be disjoint,
(ii) use of processing nodes other than ones with fragments and query is permitted and,
(iii) operations such as relational algebra difference may be used to further reduce communication costs.

The impact of the relaxation of these assumptions is illustrated by the following examples.

**Example 2.** Consider a computer network with nodes \( \{1,2,3,q\} \) where node q is the query node. The following links are available for data transmission: \( \{(1,q),(2,q),(1,2),(1,3),(2,3),(3,q)\} \). The variable data transmission costs for the specified links are 4, 5, 2, 2, and 3, respectively. The fixed transmission costs are zero. The union of two union-compatible and non-disjoint fragments, of size 100 each and stored at nodes 1 and 2, has to be transmitted to query site q. The union of the two fragments has a size of 125. The topology of the network and the sizes of fragments are shown in Figure 1. The nodes are labelled with fragment sizes and the links are labelled with variable costs. All nodes, including node 3 which does not have any stored fragment and is not a query node, have data processing and data transmission capabilities. The least cost transmission schedule, with a cost of 775, is to send the fragments at nodes 1 and 2 to node 3 where the union is performed and to send the result of the union to the query site q. Note that node 3 has no stored data fragments but is used for data transmission and union. It is instructive to compare this with ad-hoc solutions proposed by others. The fragments are sent to the query node by using the shortest paths to the query node (Sevev, 1986). In this case, the fragment at node 1 would be sent to the node q via link (1,q) and the fragment at node 2 via link (2,q). This would cost 800. The fragment is sent to the site with the largest fragment (Yu, et al., 1985). This strategy would cost 1000.

As Example (2) illustrates, having non-disjoint fragments introduces an interdependence in determining transmission paths for fragments. In such cases, the transmission paths cannot be determined on a fragment by fragment basis. Comparison of this example with union implementation suggested by others shows the promise of such a global approach.

**Example 3.** Consider a computer network with nodes \( \{1,2,q\} \) and links \( \{(1,2),(1,q),(2,q)\} \) with variable cost rates of 10, 20 and 15, respectively. Nodes 1 and 2 have fragments of sizes 25 and 20, respectively. Node q is the query node. As with example (b), Figure 2 describes the topology of the network. The union of the two fragments has a size of 26. All fragments are sent to the node with the largest fragment, the union is performed there and then the...
Related Content

Node Partitioned Data Warehouses: Experimental Evidence and Improvements
Pedro Furtado (2009). Selected Readings on Database Technologies and Applications (pp. 490-498).
www.igi-global.com/chapter/node-partitioned-data-warehouses/28568?camid=4v1

A Metadata Oriented Architecture for Building Datawarehouse
www.igi-global.com/article/metadata-oriented-architecture-building-datawarehouse/3269?camid=4v1a

Temporal Aggregation Using a Multidimensional Index
www.igi-global.com/article/temporal-aggregation-using-multidimensional-index/3371?camid=4v1a

Making Query Coding in SQL Easier by Implementing the SQL Divide Keyword: An Experimental Query Rewriter in Java
www.igi-global.com/chapter/making-query-coding-sql-easier/52306?camid=4v1a