An Architecture for a Distributed Deductive Database System

M. K. Mohania    N. L. Sarda
Dept. of Computer Science and Engineering,
Indian Institute of Technology, Bombay 400 076, INDIA
Email: \{mukesh, nls\}@cse.iitb.ernet.in

Abstract
This paper presents a three level architecture for a distributed deductive database system which extends the power of distributed database systems to include deductive capabilities. It allows both knowledge-based and data-based information to be shared by the sites of a computer network in order to process user queries. The benefits and the possible ways of segregating the rulebase into different clusters are also discussed.

1 Introduction
The need for processing large amount of data efficiently and effectively has led the researchers to focus their attention on the knowledge based systems. Consequently, the problem of defining and processing a large knowledge base has attracted a lot of interest recently. As a result of this shift in emphasis from RDBMS [9] to KBMS [3], processing of a large knowledge base has become an active area of research.

Logic can be used to represent knowledge, which is the key feature towards building more intelligent systems in general and knowledge based management systems in particular. Knowledge ranges from simple facts stored in a conventional database to complex statements about the real world. Many Logic-oriented database languages are being proposed, e.g., LDL [13], Datalog [7] etc., with declarative semantics. These languages are based on horn clauses and the bottom-up model of computation.

A Distributed Deductive Database System (DDedDBS) is a collection of deductive database systems which are physically distinct from each other but logically belong to the same application domain and communicate with each other through a communication network. In a distributed database system, data-based information is shared by the sites of a computer network, whereas a DDedDBS allows both knowledge-based and data-based information to be shared and integrated by the sites of a computer network. Thus, a DDedDBS subsumes the capability and functionality of a distributed database system. We describe a three level architecture for a DDedDBS which addresses the problems of partitioning and distributing a large rulebase and efficient query handling. In the proposed architecture, a Distribution Enabling Module (DEM) is built on top of a Distributed Database System (DDBS) to extend the power of a DDBS including the deductive capabilities. DEM only sees the global relations. A DEM consists of a User Interface Module, Query Analyzer, and Query Compiler.

A user may submit a query at any site of the DDedDBS. The query analyzer of that local site uses, called the knowledge cluster dictionary to decide whether the local site can answer the query or does it requires use of other sites. If the local site requires the use of other sites, communication must take place among the participating sites. The subqueries not answerable locally are sent to other sites, where they would be compiled and executed as if they were independent queries, and their results are sent to the initiating site. The requesting site collects the answers and does final operations like PROJECTION, JOIN as required, and passes results back to the user.

The rest of the paper is organized as follows. Section 2 describes some definitions and the impact of rulebase clustering is described in section 3. Section 4 describes the architecture for a distributed deductive database system. Section 5 describes processing of queries in this architecture. Each section includes a survey of the relevant literature. Finally, section 6 gives the concluding remarks.

2 Definitions
We use Datalog rules for knowledge representation. A horn clause or Datalog rule has the form

\[ P_0(X_0) : \neg P_1(X_1), \ldots, \neg P_n(X_n), \]

where each \( P_i \) is a predicate name and each \( X_i \) is a vector of terms where each term involves a variable or a constant. Each \( P_i(X_i) \) is called a literal. The left hand side of the rule is the head of rule. It is a derived predicate. The right hand side of the rule

-196-
3 Rulebase Clustering

The clustering problem is encountered in the DDBMS, Object Oriented Database (OODB), DKBMS with different perspectives. In DDBMS [6], the database can be fragmented into three types: horizontal, vertical, and mixed fragments. Each fragment is a cluster of data such that they are predominantly used together by applications. Many OODBMS [4] [11] have been designed using different clustering schemes which are static in nature. Objects are clustered based on their inter-relations and/or their simultaneous access potentials. A dynamic re-clustering scheme of objects has been discussed in [8]. Henceforth we define the process of fragmenting the knowledge base into various subsets (i.e., clusters) as clustering.

A cluster contains a set of rules that are meaningfully related and defined for an application domain or a set of related application domains. The derived predicates are visible only within the cluster in which they are defined. A cluster can export any of its local predicates and import any predicates that have been exported by other clusters. To handle such cases, a unique identifier is assigned to each cluster. If any literal of a rule uses the rules of other clusters, then it should be preceded by the identifier of the calling cluster. We do not allow two predicates, which are defined in different clusters to be mutually recursive.

Clustering of the rulebase has a great impact on the overall system performance. For example, rules can not be fired in parallel if they are assigned to the same cluster. On the other hand, if two rules are mutually dependent, then assigning them to different clusters may result in increase in communication overheads. Therefore, it is desirable to minimize the communication cost between the clusters.

There are many more motivating reasons for introducing clustering and distributing a rulebase. Clustering of very large rulebase exploits parallelism over a computer network, reduces query response time, and increases knowledge availability. Hence, the clusters serve as logical unit of allocation suitable for efficient computations on a network of computer.

3.1 Horizontal Clustering

Horizontal clustering partitions the rules of a rulebase into disjoint clusters. This is useful in distributed databases. These clusters can be stored at their associated locations. Horizontal clustering is needed because of the following reasons.

1. An efficient search of rules can be achieved because of the smaller size of the cluster and therefore it reduces compilation time.
2. The rules of the cluster can capture the properties of an application domain or a set of related application domains.

Let \( r_1, \ldots, r_n \) be the rules of RB and \( p_1, \ldots, p_n \) be their heads. Let \( k \) be the number of clusters of RB. The function \( h(r_i) \) gives head \( p_i \) of a rule \( r_i \). The following criteria are required for horizontal clustering.

Criterion 1: Disjointness
If \( \{p_1, \ldots, p_l\} \) is the set of heads belonging to \( RB_1 \) and \( \{p_{l+1}, \ldots, p_m\} \) is the set of heads belonging to \( RB_2 \), then \( \{p_1, \ldots, p_l\} \cap \{p_{l+1}, \ldots, p_m\} = \emptyset \). That is, horizontal clustering partitions the rules of rulebase into disjoint cluster and also rules with the same head would result in the same cluster.

Criterion 2: Completeness

\[ RB = \bigcup_{i=1}^{m} RB_i. \]

That is, the knowledge base can be constructed by the union of the clusters.

3.2 Vertical Clustering

The vertical clustering of RB is meaningless due to the unique definitions of the rules, but partitioning the body of a rule can be beneficial. The reason for partitioning a rule may be that the part of the body may be a meaningful unit of knowledge and can be queried directly. It also provides a means to factor out common literals from the body of rules. It may happen that base predicates and/or derived predicates used by the part of the body of a rule might be stored at other sites. Then it is beneficial to partition a rule so that parallel computation and processing can be achieved. For example:

\[ P_0(X_0) : - P_1(X_1), P_2(X_2), \ldots, P_k(X_k). \]
The above rule can be partitioned, say, into two rules.

\[ R(X_1) : - P_i(X_1), \ldots, P_i(X_k). \]
\[ P_0(X_2) : - R(X_1), P_{i+1}(X_{i+1}), \ldots, P_i(X_k). \]

It means that \( R \) and \( S \) are the meaningful units of knowledge and can be queried directly by the user.

The disadvantages of vertical clustering are that it increases the level of expansion during compilation and may lead to communication overheads during execution.

### 3.3 Database Fragmentation

In a distributed database system, data are often partitioned and distributed at different sites. Data can be partitioned horizontally, vertically, or by a combination thereof. The partitions are so designed that the data required most often by the users are available locally to them. The problem of fragmentation of a relation is well studied.

### 4 Architecture for Distributed Deductive Database Systems

Many architectural aspects of distributed knowledge based management system have been discussed in the literature. Li [12], has proposed a distributed knowledge representation framework for multiagent systems in which he has classified the knowledge into local knowledge, group knowledge and global knowledge. All the knowledge is not accessible to all users. A DKBMS architecture has been discussed in [4]. Here, a group of related knowledge based systems are integrated and managed by a separate KBMS node. The authors have described here a two level architecture, where the user can submit queries either at local site or to the separate KBMS node, called DKBS. A KBS cannot communicate with another KBS. They can communicate with each other only through the DKBS node. This may increase the communication cost. We assume that rulebase has already been segmented into clusters and allocated to different sites of a computer network according to their geographical properties.

We describe a three-level architecture for a distributed deductive database system which provides functionality similar to that of [6]. A user can submit the query at any one of the following levels:

1. **Global level:** When a user is not aware of the identifier of the rulebase clusters, but not aware of the allocation of the clusters. For example, \( :-C_0: P(a,X) \). It means that this rule falls under that cluster whose identifier is \( C_0 \).

2. **Allocation level:** When a user is aware of both the clustering and allocation of the rulebase clusters. For example, \( :-C_0: P(a,X) \) at site 1.

In the proposed architecture, a Distribution Enabling Module (DEM) is built on top of a DDDBS. DEM has the following components:

- **User Interface Module:** allows user to submit the Datalog query.
- **Query Compiler:** converts a Datalog query into a base query (compiled query) which is defined on the global relations.
- **Knowledge Base Manager:** manages the knowledge cluster dictionary, which contains the information about the clusters of the rulebase i.e., identifier of the cluster and the site(s) of its allocation, to decide whether the local site can answer the query or it requires the use of other sites. If the local site requires the use of other sites, communication must take place among the participating sites. The subqueries not answerable locally are sent to other sites, where they would be compiled and executed as if they are independent queries and their results are sent to the initiating site. The requesting site, which collects the answers, does final operations like PROJECTION, JOIN as required and passes results back to the user interface module.

### 5 Query Processing

The efficient bottom-up evaluation of queries in distributed database systems has been discussed in [14]. It is assumed that the underlying medium for such a system is a relational database which is used to store data i.e. facts. The rules are stored in the system in a "flat" form and called upon, if needed, to further expand a user query. Once the rule has been substituted with its definitions, the resulting query is processed by the DBMS.

Our approach for query processing uses a compiled method which compiles a query wholly in advance in the rulebase and decomposes the compiled query into a set of subqueries which can be evaluated only in the database. Compilation in the rulebase means that all
literals of a query should be base predicates i.e. global relations.

5.1 Analysis of Query

When a user submits the query, which may be a conjunction of derived predicates, the user interface module will process the query and then transmit it to the query analyzer which uses knowledge cluster dictionary for determining the level of the query and the site(s) of its compilation.

When the query is at global level, the query analyzer produces subqueries equal to the number of clusters of the rulebase. Each subquery is a query predicate preceded by the cluster identifier. At this level, the query analyzer adds the site of allocation of the cluster with the subquery for compilation using the knowledge cluster dictionary. Now the subquery will be treated at the allocation level. Finally, the query analyzer sends the query (subquery) to the assigned site.

If a cluster is stored at more than one site, the query analyzer will decide the site for sending the query. This problem is called materialization and has been studied [5]. If the user poses the query as a combination of these levels, the query analyzer will decompose the query into subqueries. Each subquery will be treated as an independent query and will be analyzed as described above.

5.2 Query Compiler

The query compiler compiles the query (subquery) and produces an equivalent query (subquery), called the compiled query (subquery), which is defined on base predicates.

The process of compilation is expressed as follows.

\[ Q'_n = c [ Q'^n, (RB'^n, \bigvee_{j=1}^{m} RB^i, j \neq i)] \]

where \( c [ Q'^n, (RB'^n, \bigvee_{j=1}^{m} RB^i, j \neq i)] \) denotes that the query \( Q'^n \) compiles in \( RB^i \) and/or may use the RB of other sites also and \( Q'_n \) denotes that compiled subquery is defined on global relations.

The assigned site may compile the query either fully or partially or not at all. During compilation, if the literal of a rule uses the rules of any other cluster, which are not on the same site, then the query compiler of a site will send the subgoal to the query analyzer which treats it as a query at fragmentation level. This query is analyzed and compiled as described above. Thus, parallelism can be obtained by such an inferencing process during compilation.
When a query predicate matches with the head of more than one rule, at least one rule must be nonrecursive. If a nonrecursive rule exists, the compilation should proceed. If a point of recursion is reached in compilation, the recursive literal should be ignored and the compilation continued. After the compilation is over, the recursive literal should be inserted at the appropriate places.

Before the query is sent to the distributed database module, the query compiler also translates the query into standard relational query language.

5.3 Query Evaluation

The evaluation of distributed query is a well studied problem [6]. Each site executes the query (subquery) and sends the answer to DBMS of the query site, which collects the answers and does PROJECTION or JOIN or both, if required. Semi-naive algorithm [14] may be used for the execution of the recursive query.

Finally, DBMS of query site passes the result back to the user interface module which gives the appropriate message to the user.

6 Conclusion

An architecture for a Distributed Deductive Database System is proposed. It extends the study of deductive databases into a distributed environment, where several sites of a DDedDBS are interconnected through a communication network. We find that the DDedDBS can handle large rulebase applications systematically by cooperative execution of multiple deductive database systems, each handling only a part of the rulebase. That is, the architecture also permits a user to access multiple domains of knowledge in the system. We have also shown the possible ways of clustering of the rulebase and their benefits in the distributed environment. We are now looking at various possible heuristics for clustering the rulebase so as to minimize the communication overhead.

References