DO-GIS - a distributed and object oriented GIS

B.V.S.S.D. Kiran Kumar , Prasun Sinha and P.C.P. Bhatt

Department of C.S. and E., I.I.T. , New Delhi, India.
email : pcp@cse.iitd.ernet.in

Abstract

This paper reports "first cut" of the design of a distributed and object oriented geographic information system, **DO-GIS**, under development at I.I.T. Delhi. The paper deals with data model, architecture, query processing and data management in a GIS.

Considering that geographical data includes information about spatial reference frame, topological connections and requires maintenance of physically distributed spatial data, we argue that a distributed and object oriented GIS is the most natural choice for both data processing and management. The distribution of data (over a set of sites) has implications on the query processing, i.e. it requires query partitioning and distributed processing. **DO-GIS** architecture employs meta database and order preserving protocols to make both the distributed query processing and subsequent collation of partial results totally transparent to the user. Overall the operation is in the spirit of co-operative computing paradigm.

1 Introduction

In diverse applications, information systems are being used increasingly as a means to manage, retrieve and store large quantities of data which are tedious to handle manually. Spatial data handling is no exception. Systems for such application are known as Geographic Information Systems (GIS) with spatial data acquisition, preprocessing , management, manipulation and analysis, and output generation. Since GIS deals with spatial data which is graphical in nature each of the above components need to have a well designed Human Computer Interface (HCI).

Current computer based information systems offer very little in terms of semantic control of information processing. This is because the application semantics are difficult to map onto the descriptive information available in the input data. Object orientation promises to remedy this situation. Besides, currently GISs are rarely designed to operate with inherently distributed geographic data. Distributed GIS provide many incentives compared to monolithic systems basically taking advantage of distributedness in both management and processing of spatial data.

2 Data Model

The usual paradigm for general purpose information management is the relational model. Majority of GISs implementing a relational model typically adopt a hybrid architecture where attribute data and their spatial references are stored, and managed, in independent structures. Such systems maintain the attribute data in conventional DBMS but organize and manipulate spatial data using conventional file handling techniques with some special purpose software for both the management of data and query processing.

The hybrid approach lacks rigorous management, data security and integrity control. Further it is not amenable to multiple user access and concurrency management. Also, and perhaps most importantly, it does not exploit natural distributedness of data. As Worboys [Worboys 94] argues, "If benefits are to be derived from distributed database technology, then GISs must be developed that deal with the particular complexities of distributed database management without being hindered by an approach that stores some data outside the DBMS, inaccessible to any distributed functions provided therein". In such hybrid systems query optimization is a very complicated issue as the query has to be first partitioned into spatial and aspatial components before any optimization and evaluation can be attempted [Ooi 89]. It has been shown that special characteristics of spatial data prevents development of unified extended relational systems with spatial and aspatial data under the same architecture [Healy 91].

Object oriented approach is being promoted for such areas for which the application of pure relational or hybrid technology leaves much to be desired.

2.1 Object Oriented GIS

There are two broad based and opposing classes of models of geographic information [Worboys 94]. The class of *field-based* models treats such information as collections of spatial distributions, where each distribution may be formalized as a mathematical function from a spatial framework (e.g. a regular grid placed upon an idealized model of an earth surface) to an attribute domain. Patterns of topographic altitudes, rainfall and temperature fit into this view. On the other hand the class of *entity-based* models treats the information space as populated by discrete, identifiable entities (objects), each with a geo-reference.

Some of the conventional data models used in GIS data representation are *Raster Model, Spaghetti Model, Topological Model, Tesselation Models* etc. [Kasturi 89].

DO-GIS is broadly in the category of systems that describe a collection of geographic objects over a two dimensional map. Clearly each geographic object can be classified as belonging to a particular entity class such as city, lake, state, river etc. These objects are described by their associated aspatial (alphanumeric) attributes (e.g. population, name, usage etc.). On a given map geographic objects may *intersect* , may *contain* other objects, and may be *components* of other objects. Queries in a GIS are typically concerned with these spatial relationships. As an example the spatial relationship *intersect* must be used to answer the question:

"Which are the states in INDIA through which the river GANGA flows ?".

All geographic objects can be grouped into three generic spatial object classes namely, *Point* , *Line* and *Polygon*. The class hierarchy currently supported by **DO-GIS** is shown in Fig. 1. Hence it is clear that entity based geographical information models fit naturally with object orientation concepts.

2.2 Distributed GIS

Distributed databases have been shown to have several advantages over a centralized database though it means increased data management complexity. We look at why these advantages are much more relevant when we consider a GIS.

- Geographic information can be said to be inherently distributed. The cartographic organizations which deal with maintenance of geographical data are distributed all over a country. Large geographical data corresponding to unit of administration like states/districts/blocks etc. is located at the administrative headquarters. In addition various organizations collect basic data sets at their own location and exchange them with others.
- The queries in a GIS are usually based on data collected locally. The fraction of queries which access data from other locations are not large enough to justify storing all the data at each location.

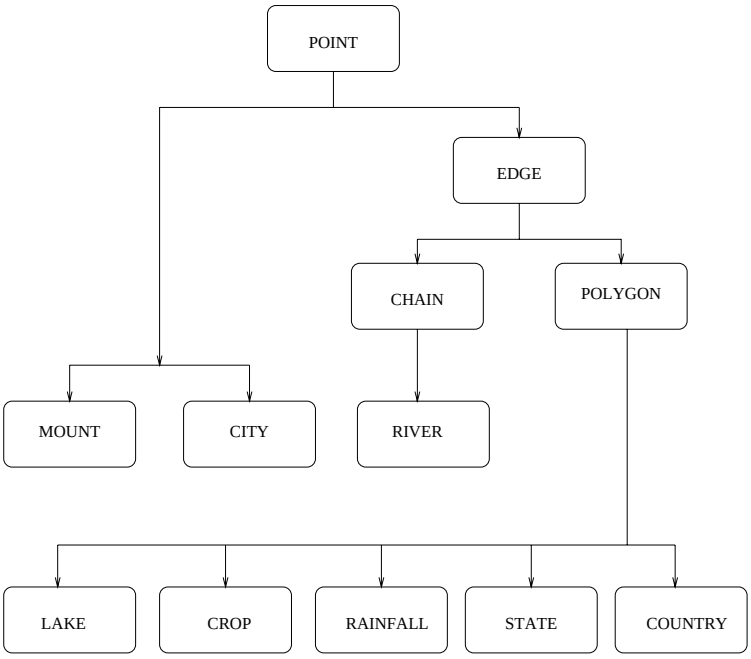


Figure 1: Class hierarchy in DO-GIS

- Addition of new sites as more and more regions are mapped and surveyed is easier with decentralized data management system.
- Geographic information potentially includes maps, aerial photographs, satellite images, site photographs, survey records, engineering and architectural plans, document images and tabular data [Newton 92]. When monolithic databases are maintained, all data has to be transported to all sites so that queries can be evaluated at any other site with a hierarchically more important role in organizational structure. In terms of data transmission this involves phenomenal amounts of data transfer and enormous storage requirements at all sites.
- GISs which serve emergency systems such as Fire, Storms, Floods, Tornadoes etc. need the greater reliability and speed offered by a distributed system.
- Cost of updating geographic data dominates all other costs of the GIS [Newton 92]. Periodic updates to all sites is practically infeasible. An update in a monolithic system has to be propagated to all the sites with copies; giving rise to various concurrency issues. An update in a distributed object based GIS is much easier as it is localized and is performed with lesser communication overheads.
- Sharing information among heterogeneous databases is easier with decentralized system since all data need not be transformed to a single format. Only the queries and the query results need to be transformed. We do not address the issues realted to integration of heterogeneous GISs into a federated system but they will become very important in the near future.
- GISs require interactive graphics and for real time response of queries based on local geography necessiates a decentralized GIS which can serve queries on local geography faster. Users querying remote data are usually willing to wait for a longer time.
- With the phenomenal growth of the World Wide Web (and increasing number of spatial data servers on the web such as Xerox Map Server, the Alexandria project of the Libraray of Congress etc.) it is becoming more and more evident that spatial data servers need to be distributed to effectively serve the large number of queries and also to keep pace with the huge amount of new information being added each day to the web.

Overall we can see that the disadvantages of maintaining monolithic databases at each site far overweigh the delay in serving a query based on remote data in a distributed system [Ceri 85, Newton 92].

2.3 DO-GIS System Architecture

A distributed GIS needs a *global data manager* to manage the distributed database as a whole. The *global data manager* can be in a central site with all queries routed through it or can be replicated at each site [Newton 92]. **DO-GIS** uses an intermediate approach with meta knowledge servers. A Meta Knowledge Base server maps logical object names to physical sites thus providing somplified consistency, reliability, and data migration.

DO-GIS [Kiran 95] has been built over the multiprocessing environment on UNIX hosts. The message passing subsystem we have used is the ISIS [Birman 88] like environment built by Verma and Goyal [Goyal 94]. The modules of **DO-GIS** are depicted in Fig. 2. We describe each module and its basic operational role briefly.

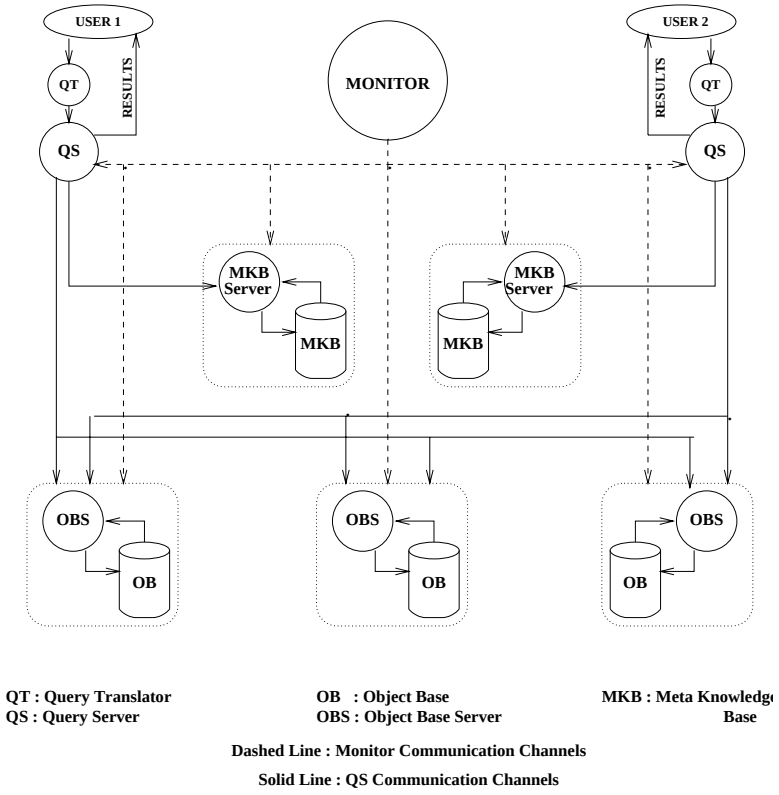


Figure 2: Sample view of DO-GIS

2.3.1 OB - Object Base:

OB is a counterpart of a typical database and consists of the files and data structures which make up the persistent store for the objects. The data in the whole system is present in non-replicated parts at various sites. It is natural and convenient to use a non-replicated database for geographic data since it is continuosly extended and updated.

2.3.2 OBS - Object Base Server:

OBS resides at each site where an **OB** is present. Its main function is to serve requests for data objects. Whenever an object is requested the **OBS** checks whether the object is in memory. If it is not it loads the object into local volatile memory. It uses various hash tables to maintain information about the objects and their location on disk in the local **OB**. Whenever an object is added to an **OB** an Object Identifier[1] is created for that object by combining information about the creator site, the class of the object and the instance count of that class at that site.

2.3.3 MKB - Meta-Knowledge Base:

MKB basically serves as a repository (or a directory) which has the information about the actual location of the objects. There can be several **MKB**s in the system and these can be active at the same time. Multiple **MKB**s are used to split the query processing load among them. The update to these **MKB**s is done by exploiting the virtual synchrony provided by the underlying platform [Goyal 94]. It is also used for targeting subqueries to relevant **OBS**s in a co-operative mode of operation. Fig. 3 captures the details of the processes and the nature of communication protocol used for all the updates. The abcast messages between Query Servers (#1, #2) and Object Base Servers (#1, #2) ensure the consistent ordering of data fetched from these object base servers.

2.3.4 QS - Query Server:

QS is at the user end and can be started up on any site irrespective of the presence of **OBS**s or **MKB**s at that site. It is used for evaluating end user queries. The evaluation of the query input by the user is done by collecting the required objects from the relevant object bases and then using the data from these objects in evaluating unit queries like equality of strings and numbers such as the name of a country, population of a state etc.

23.5 QT - Query Translator:

This is the front end to the **GEOSQL** based query. It analyzes a query and builds up the query tree. This query tree is used by the QS for evaluating the queries.

2.3.6 MON - Monitor:

Monitor provides a minimal level of fault tolerance and the load distribution in the system. It also serves as the system information database. Whenever a new module of the system comes up Monitor gives it the initialization data and also the information about various active modules in the system. When the Monitor detects the failure of any OBS or MKB module it sends update messages to the required modules. It also answers queries about the status of the system. In future versions of the system **DO-GIS** will provide fault tolerance by using the Monitor to detect failures of modules and starting them up automatically.

---

[1] For example, A:STATE:25 is the identifier which is assigned to an object which is 25th instance created at site A, of entity class STATE.

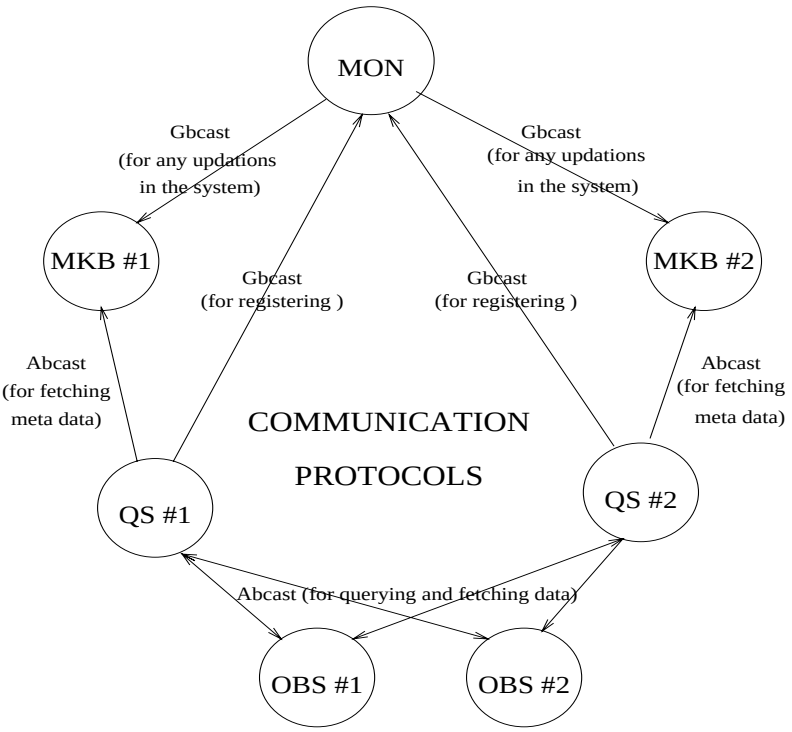


Figure 3: Communication Protocols

3 GEOSQL and Query Processing

In a GIS, the information describes entities that have a physical location and extent in some spatial region of interest. Queries in a typical GIS involve the identification of these entities based on their aspatial and spatial attributes, and spatial relationships between entities. The interface language should be powerful enough to express a query involving both spatial and aspatial components.

Structured Query Language (SQL) is by far the most popular query language but lacks constructs needed to express queries based on spatial attributes [Egenhofer 92]. We have retained the template structure of SQL and introduced the operators that reflect the GIS nature of a query. For example, consider a typical query like "*Which are the states in INDIA through which the river GANGA flows* ?". We need to specify entity classes *state*, *country* and *river*, spatial attributes like the *boundary* of a state and the spatial relationships *contains* and *intersects* in the query. We have designed a query language **GEOSQL** by adding spatial constructs, operators and relations to the basic constructs in SQL viz. "SELECT", "FROM" and "WHERE". The template of a typical query in **DO-GIS GEOSQL**[2] is:

```
FROM    <ENTITY_CLASS>  <VAR_NAME1> , ...
SELECT  [QUALIFIER] <VAR_NAME1>.<ATTRIBUTE>
WHERE
        <VAR_NAME1>.<ASPATIAL_ATTRIBUTE>  =  |  <  |  >   <CONSTANT>
        <LOGICAL_OPER> <VAR_NAME1> <SPATIAL_OPER> <VAR_NAME2>
```

For the convenience of the end user there is heavy overloading of operators. As a result there can be queries that contain expressions such as *river* INTERSECT *state* or *crop* INTERSECT *rainfall*. One more important feature to be noted is that the end user need not be aware of the

[2]Hereafter GEOSQL shall refer to **DO-GIS** GEOSQL

distribution of data over various sites. The query language interface makes this distribution of data transparent.

Now expressing our example query above in **GEOSQL**:

```
# Select the names of the states in INDIA through
# which river GANGA flows

FROM STATE S ,RIVER R , COUNTRY C
SELECT S.NAME
WHERE
        C.NAME = "INDIA"
    AND C CONTAINS S
    AND R.NAME = "GANGA"
    AND EXISTS ( R INTERSECT S )
```

A special feature of **GEOSQL** is the CONSTRUCT-AVOID clause. Suppose one wants to explore the feasibility of constructing a facility like a tramway and wants to find the number of bridges that he may have to construct, he can express it using the CONSTRUCT clause in **GEOSQL**. The AVOID clause is used when the user desires to study an alternate path with the condition that it avoids one or more of geographic objects such as a lake, a mountain or crossing a state boundary.

A part of a GEOSQL query will clarify the use of these constructs:

```
# A tramway is to be constructed between two cities

# avoiding the lake in between

FROM REAL TRACK_LEN ,LINE L , CITY A ,CITY B,
        COUNTRY C, LAKE THELAKE
SELECT TRACK_LEN
CONSTRUCT
            L = ((A),(B))
            AVOID THELAKE
WHERE
        ......
```

A special class that is supported by **GEOSQL** is AMORPHOUS. A variable of this class fits in place of object of any class. This class is very useful for queries in which the classes of the objects to be selected are not known beforehand. Those classes can then be declared as AMORPHOUS and at runtime (i.e. while evaluating the query) , the actual class would be assigned to it.

3.1 Query Processing

For an object oriented and distributed system the strategies used for processing a query are entirely different from those used in conventional database systems [Egenhofer 94]. There are two ways in which the queries may be processed. The naive approach is by gathering all the required objects from relevant sites and then proceeding to solve the query. A better approach is by sending some extra information along with the data request to the remote site so that the remote data server can do some pruning on its own and send only the selected objects. This is also the basis of co-operative computing wherein different segments of a solution are generated in a distributed manner. The co-operative framework may result in parallel evaluation of a query for which the underlying message ordering protocols can be exploited.

The major steps in query processing are described in Fig. 4. Basically the steps followed in the figure are:

1. Parse and construct the query tree.
2. Preprocess the query tree and check for its validity, i.e. ensure that scope rules are followed, variables are declared etc.
3. Extract information regarding every variable declared in the FROM clause. This information includes type declaration, all attributes referenced, subtrees which can be used to restrict the distributed search space etc.
4. Send a request to MKB for information regarding sites where objects satisfying the class declaration in the FROM clause can be found.
5. Send data requests using the appropriate protocols to sites to gather the relevant data.
6. Evaluate the query using recursive calls for the expression in WHERE clause.
7. Display the text and graphical data corresponding to the results.

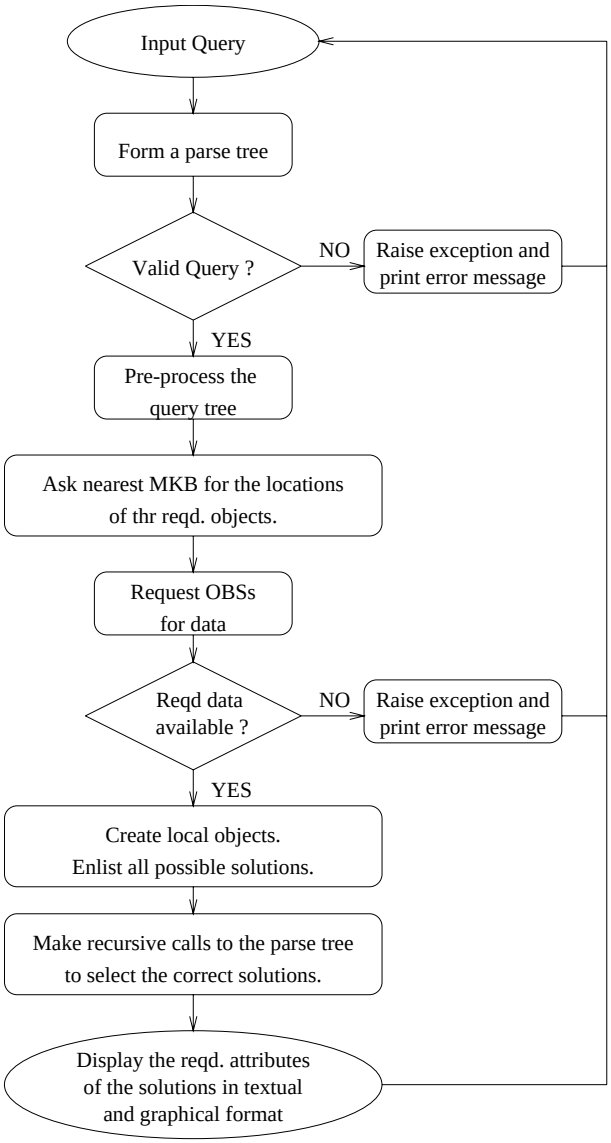


Figure 4: Query processing in **DO-GIS**

Query processing in general involves the use of some heuristic methods for fast evaluation. In a distributed system, use of heuristics would reduce communication overheads and hence improve the performance substantially. Various heuristics which we have identified are :-

- Object Name : Use the name of the object while requesting for the object if it is specified by a clause like O.NAME = "XYZ" where O is the object and "XYZ" is the name of the object.
- Required Attributes : Select all the attributes that are referenced in the query and fetch only those attributes of the required objects rather than fetching all the attributes of the objects from remote sites.
- Remote pruning of the data : If the query has a clause like C.POPULATION > 10000 then this information can be sent to the remote site which can then select only those objects satisfying the clause and send them.
- Overlapping computation with communication : When data required for some subtree of the query tree is being fetched computation required in other subtree can be done. Threads such as Light Weight Process library supported by SunOS operating system can be used for such a purpose.
- Parallel evaluation of query tree : Parts of tree can be sent to remote sites and the results sent back to the original site. This involves development of algorithms which collate the results from various sites to get the final result.

In **DO-GIS** we have all the above mechanisms supported by the order preserving protocols. Note that the overlapping of communication and computation and parallel evaluation of query tree are in the co-operative framework of computation.

4 An Example

To explain the basic operational steps and subquery management we portray a hypothetical scenario of a GIS dealing with the geographical data of India. A logical organization of the data would be at the four metropolitan cities viz., New Delhi, Calcutta, Bombay and Madras. Each city stores maintains the data pertaining to a specified geographic region around it. Since the actual objects are not distributed over various sites, data about objects falling in two regions is arbitrarily stored at any one of the two.

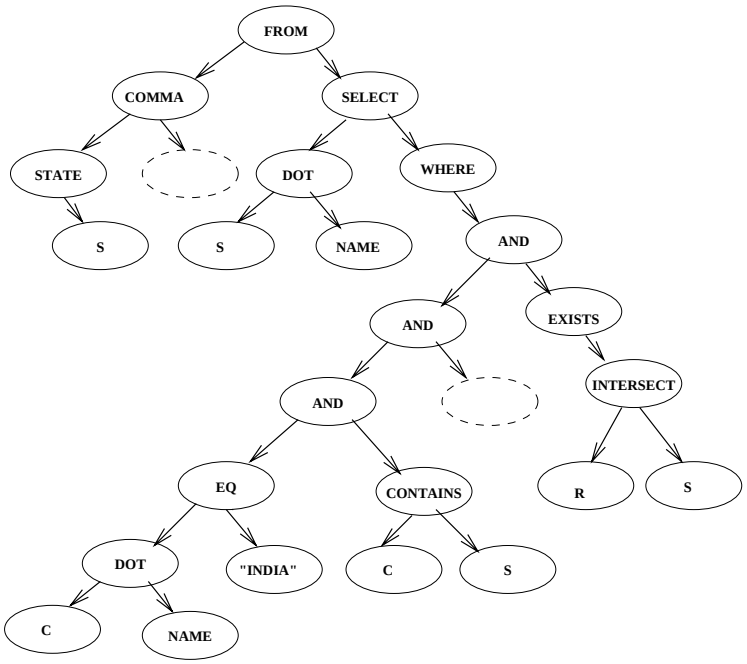
Now consider the query discussed in section 3 i.e.

```
# Select the names of the states in INDIA through
# which river GANGA flows

FROM STATE S ,RIVER R , COUNTRY C
SELECT S.NAME
WHERE
        C.NAME = "INDIA"
    AND C CONTAINS S
    AND R.NAME = "GANGA"
    AND EXISTS ( R INTERSECT S )
```

First this query is parsed and processed to check the validity. The query tree constructed by the parser is shown in Fig. 5 . Now the three objects classes that are used in the query are State , River and Country. The query has two clauses which specify the names of objects Country and River. The location of INDIA and GANGA is obtained from a Meta Knowledge Base and both the objects are accessed from their locations, which in this case happens to be New Delhi. Now there is no clause in the query which provides any information about the states so a query is sent to the Meta Knowledge Base to get location of all objects of class State. So states from all four sites are accessed as shown in Fig. 6. Now

the clauses “C CONTAINS S” and “R INTERSECT S” are evaluated and then the list of states which satisfy both the clauses is selected as the solution.



PARSE TREE FOR A SAMPLE QUERY

(The empty ellipses represent subtrees)

Figure 5: Query Tree for Example Query.

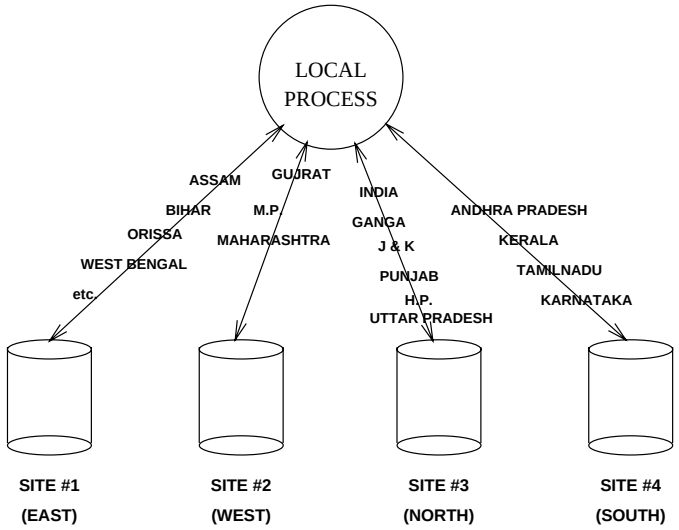


Figure 6: Data access from different sites.

5 Conclusion

We have vindicated Worboys [Worboys 94] statement on taking advantage of natural distributedness of geographical data in designing an effective distributed GIS. The distributed character of data can be taken advantage of, if the database architecture supports distributed management and query processing. **DO-GIS** employs object oriented framework to encourage reuse of methods and captures natural data hierarchy besides its distribution. The distributed query processing is attempted in co-operative mode with adequate support from order preserving protocols.

References

[Birman 88] BIRMAN K.P., JOSEPH T.A., KANE K.P., SCHMUCK FRANK, 1988, ISIS - A distributed Programming environment User Guide and reference manual.

[Ceri 85] STEFANO CERI, GIUSEPPE PELAGATTI, 1985, Distributed Databases - Principles and Systems, McGraw Hill.

[Egenhofer 92] EGENHOFER M.J., 1992, Why not SQL ?, International Journal of Geographic Information Systems , 6(2), 71-85.

[Egenhofer 94] ELISEO CLEMENTINI, JAYANT SHARMA, MAX EGENHOFER, Modeling Topological spatial Relations : Strategies for Query Processing, To appear in Computers and Graphics , Special Issue on Modeling of Spatial Data.

[Goyal 94] ARUN VERMA, DEEPAK GOYAL, 1994, Design of a Shared Visual Space System for Co-operative Problem Solving. B.Tech. Thesis, DCSE, I.I.T. Delhi.

[Healy 91] HEALEY, R.G., 1991, Database Management Systems. In Geographical Information Systems : Principles and Applications , edited by D.J.Maguire, M.F.Goodchild and D.W Rhind , Longman, 251-267.

[Kasturi 89] RANGACHAR KASTURI et al., 1989, Map Data Processing in Geographic Information Systems, IEEE Computer , December 1989, 10-21.

[Kemper 94] ALFONS KEMPER, GUIDO MOERKOTTE, 1994, Object Oriented Database Management - Applications in Engineering and Computer Science, Prentice Hall.

[Kiran 95] KIRAN KUMAR, PRASUN SINHA, 1995, DO-GIS : Distributed and Object Oriented Geographic Information System. B.Tech. Thesis, DCSE, I.I.T. Delhi.

[Nagy 79] GEORGE NAGY, SHARAD WAGLE, 1979, Geographic Data Processing. Computing Surveys , 11(2), 139-181.

[Newton 92] NEWTON,P.W., ZWART,P.R., CAVILL,M.E., 1992, Networking Spatial Information Systems, Belhaven Press.

[Ooi 89] BENG CHIN OOI, RON SACKS-DAVIS, KEN J.MCDONELL, 1989, Extending a DBMS for Geographic Applications. Proc. of 5th Intl. Conf. on Data Engineering,, Los Angeles, California, 5, 590-596.

[Worboys 94] WORBOYS,M.F., 1994 , Object-oriented approaches to geo-referenced information. International Journal of Geographical Information Systems , 8(4), 385-399.