Knowledge-Based Approach for an Object-Oriented Spatial Database System

Yang Hee Kim
Division of Liberal Art, Korea National Sport University
(yangh-kim@knupe.ac.kr)

In this paper, we present a knowledge-based object-oriented spatial database system called KOBOS. A knowledge-based approach is introduced to the object-oriented spatial database system for data modeling and approximate query answering. For handling the structure of spatial objects and the approximate spatial operators, we propose three levels of object-oriented data model: (1) a spatial shape model; (2) a spatial object model; (3) an internal description model. We use spatial type abstraction hierarchies(STAHs) to provide the range of the approximate spatial operators. We then propose SOQL, a spatial object-oriented query language. SOQL provides an integrated mechanism for the graphical display of spatial objects and the retrieval of spatial and aspatial objects. To support an efficient hybrid query evaluation, we use the top-down spatial query processing method.

Key Words : object-oriented spatial databases, knowledge-based approach, object-oriented data model, spatial type abstraction hierarchy, spatial object-oriented query language

논문접수일 : 2003년 6월 게재확정일 : 2003년 11월 교신저자 : 김양희

1. Introduction

We consider a spatial database system to be a full-fledged DBMS with additional capabilities for the representation and manipulation of geometric data. As such, it provides the database technology to support applications such as geographic information systems (GIS), image databases, VLSI, and CAD/CAM.

There are several attempts at supporting spatial applications using existing database systems. And common data models such as the relational model are not adequate for handling spatial data. While it is usually easy to develop a schema that will capture all the data and relationships in an application, it is difficult to specify even the simplest spatial operations using query language. Furthermore, the access paths

* 이 논문은 2003년 한국체육대학교 교내 학술연구비 지원사업에 의하여 연구되었음.

supported in existing database systems are unlikely to offer good performance since they were not designed for spatial data. Even if there are some limitations, several spatial database management system prototypes have been built by extending an RDBMS. Among examples of such prototypes are Gral (Guting, 1989) and PROBE (Orenstein and Manola, 1988). Gral offers extended relational model and uses many sorted algebra for its extensible system architecture. But query language based on this algebra is complex and does not display spatial objects. Also it cannot represent spatial and aspatial objects in a uniform way. PROBE offers extended functional data model and uses POINT-SET data type for spatial data types. System provides approximate geometry process, but it lacks a formal framework for data modeling. Also, query language using function is complex and does not display spatial objects.

To overcome these limitations the object-oriented DBMSs are being used. Object-oriented database systems provide the incorporation of object-classes that provide a collection of specialized operations. But, it is still necessary to extend OODBMS for handling spatial objects. Especially, spatial object types must be added to its basic type hierarchy and spatial operators should be defined over such types. An extension of query language is also required. Toward this end several prototypes have been designed: examples are GeO_2 (David et al., 1993) and GODOT (Gunther and Riekert, 1993). However they lack a formal framework for data modeling, an integrated mechanism for the display of spatial objects. And if the exact answer is not available, they provide limited answers and options, or even no information at all.

To remedy such restrictions, it is necessary to use knowledge-based approach for an object-oriented spatial database system. The advent of knowledge-based systems provides many new ways to enhance spatial database system. A knowledge-based system provides neighborhood or generalized information relevant to the original query and within a certain semantic distance of the exact answer.

In this paper, we present a knowledge-based prototype system called KOBOS(Knowledge-Based Object-Oriented Spatial Database System) for the GIS-application. However, it can be applied for other applications of spatial database system. Most geographic information systems use multiple map layers to organize geographic objects. We propose three different types of map layers: a *structure layer*, a *thematic layer* and a *knowledge layer*. A structure layer is a set of logically related features which are all based on the same topological data structure and a thematic layer is a set of features that belong to the same theme. And a knowledge layer contains the logic for interpreting semantics and similarity of spatial object. In order to support three different types of map layers, we propose the three levels of object-oriented data models: a spatial shape model for the knowledge layer, a spatial object model called spatial signature (SAS) for the thematic layer and an internal description model for the structure layer.

We use the commercially available OODB ObjectStore (Lamb et al., 1991) as an implementation platform and we add several mechanisms for handling spatial objects to such platform. The spatial shape model captures the structured abstracts and knowledge needed for spatial object retrieval. We use spatial type abstraction hierarchies (STAHs) to represent the knowledge of the selected spatial object features and spatial relationships. The spatial object model gives a formal framework to support spatial object types and their polymorphic operations which make used of a spatial query language suitable for manipulation of thematic maps. The notion of generalization is supported in the spatial object model. The internal description model supports efficient and appropriate framework to store geographic coordinates and topology between two geographic objects. The internal description model is based on the formal data structure for single-valued vector map and multi-valued vector map (Falcidieno, Pienovi, and Spagnuolo, 1992). Each structure layer consists of modules and maintains topological integrity. The layer corresponds to one of three topological description levels: geometry level, network topology level (corresponding to Non Planar Graph and Planar Graph), and full topology level, that may match the user's need. Each module consists of internal primitive objects which are the lowest internal object. The generic type of the type hierarchy for the internal primitive objects is vector_type. And vector_type is partitional into four subtypes: node, edge, face, and ring. The independence between spatial object model and internal description model gives us correctly separate the user view from the map description view.

For providing an integrated mechanism for the graphical display of spatial objects and the retrieval of objects, we design a spatial object-oriented query language (SOQL). SOQL is an OQL-based query language and for OQL (Object Query Language), we refer to (Cattell and Barry, 1997). To manipulate the presentation of spatial objects, variation of graphical presentations by colors, symbols, and filling are added in SOQL with pull down menus. A top-down query processing method is used to evaluate hybrid query efficiently. That consists of four stages: parsing, unnesting, decomposition, and subquery sequencing. Finally, we propose the architecture of KOBOS which uses ObjectStore as an implementation platform.

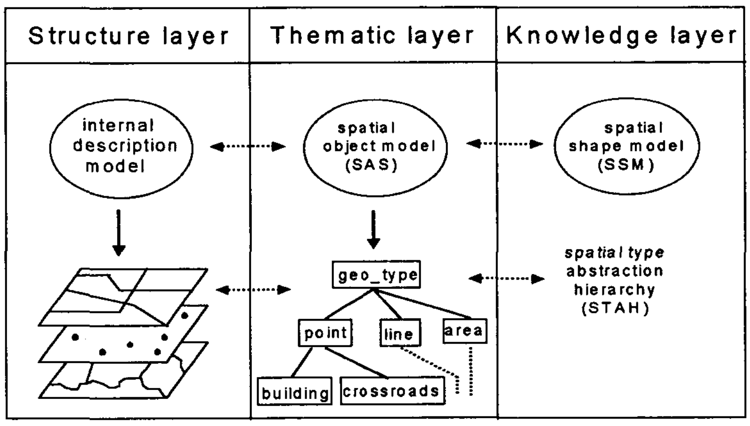
This paper is organized as follows. In Section 2 we introduce the three levels of spatial data models, a spatial shape model, a spatial object model called SAS and an internal description model. In Section 3 we present the syntax and semantics of the spatial object-oriented query language, SOQL. Section 4 describes the architecture of KOBOS and a top-down query processing method. The conclusions are presented in Section 5.

2. Three Levels of Spatial Data Modeling

The main difference between spatial data and aspatial data is their complexity. This complexity arises from the mix of alphanumeric and geometric information necessary to describe a spatial data. Spatial data can be discrete or continuous. When they are discrete (e.g., points in a multidimensional space), then they can be modeled using traditional techniques from relational DBMSs. In particular, the coordinate values of the point can be treated as additional attributes in a tuple. In contrast, lines and areas are continuous, that is the data span an area in space. Hence, the attribute value holds at more than just one point. Since the application domain of a spatial data is so wide, we consider modeling in GIS application.

Geographic information systems are spatial database systems that allow the manipulation, storage, retrieval, and analysis of geographic object as well as the display of data in the form of maps (Ooi, 1990). In such a system, the database describes a collection of geographic objects over a two dimensional map. Each geographic object can be classified as belonging to a particular class such as city, road, lake, etc. Most geographical database systems use multiple map layers to organize geographic objects. There exist two different types of map layers: a structure layer and a thematic layer. A structure layer is a set of logically related features which are all based on the same topological data structure and a thematic layer is a set of features that belong to the same theme.

In this section, we propose the three levels of spatial data models: a *spatial shape model* (SSM) for the knowledge layer, a *spatial object model* called spatial signature (SAS) for the thematic layer and an *internal description model* for the structure layer. <Figure 1> shows the structure of the three levels of object-oriented data models.


Structure layer
Thematic layer
Knowledge layer
internal description model
spatial object model (SAS)
spatial shape model (SSM)
geo_type
point
line
area
building
crossroads
spatial type abstraction hierarchy (STAH)

<Figure 1> Three levels of data models

2.1 Spatial Object Model

In this subsection, we give the spatial object model called SAS. SAS describe a formal framework for modeling the structure of spatial objects in space as well as their relationships, properties, and operations. SAS data model is a collection of spatial object types (spatial sorts) and spatial polymorphic operators. It is based on a signature and an extended signature. A signature is a pair (S, Σ) with S is a set (whose elements are called *sorts*) and $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$, is a family of sets (whose elements are called *operators*) and an extended signature is to be introduced for list sorts, product sorts, and function sorts (Guting, 1993).

2.1.1 Spatial Object Type

In SAS, we extend ObjectStore's metatype hierarchy to support spatial object types.

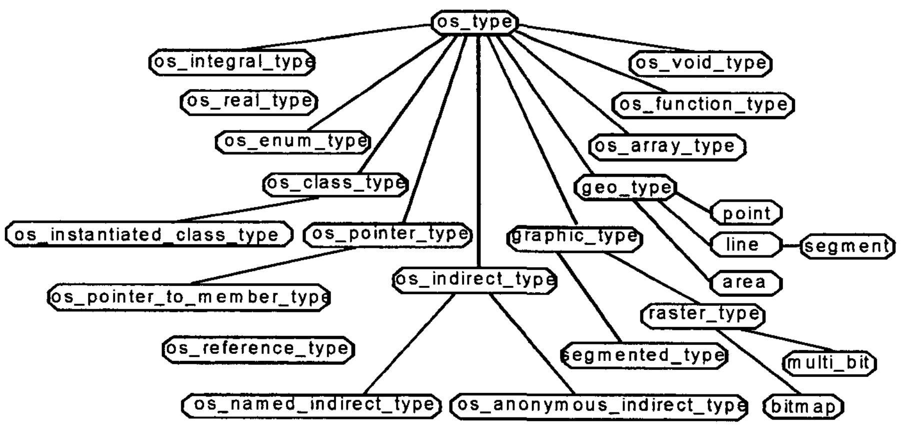
<Figure 2> depicts the extended type hierarchy for aspatial and spatial objects. The leaf nodes are directly instantiable.

Every object representing a type is an instance of a subtype of the metatype os_type in ObjectStore. To extend metatype hierarchy, we add two new types, geo_type and graphic_type. Graphic_type represents things for graphic display. Geo_type represents various vectorized objects such as points, lines, and various type of areas. A point is the values of n coordinates in n-dimensional vector space. A line is a finite sequence of straight line segments. An area is a polygon and it may have holes.

2.1.2 Spatial Operator

We give spatial operators on spatial sorts.

- *Location operators* are the most basic operators for the spatial object processing. They search the location value of the spatial object and retrieve the coordinate of the object, or the boundary of the object.



<Figure 2> Spatial object type hierarchy

- *Arithmetic operators* perform arithmetical measurements for spatial objects and have os_integral_type as their return type. An example would be the calculation of the distance between two geo_types.
- *Topological relationships* are the most fundamental operators on geo_type. Topological relationships of two spatial objects are DISJOINT, MEETS, EQUALS, INTERSECTS, COVERS, COVEREDBY, INSIDE, and OUTSIDE.
- *Semantic spatial relationships* are semantic relationships among spatial objects. Semantic spatial relationships of two spatial objects are NEARBY and FAR_AWAY.
- *Mouse operators* provide the function of selecting an object from the previous query result using the mouse. Examples of the mouse operators are pick(), window(), point(), line(), and circle().
- *Object construction operators* are defined to construct new spatial objects which satisfy certain topological relationships. If the relationships do not hold, the operator returns an empty spatial object. Examples of the object construction operators are split, merge, difference, and intersection.

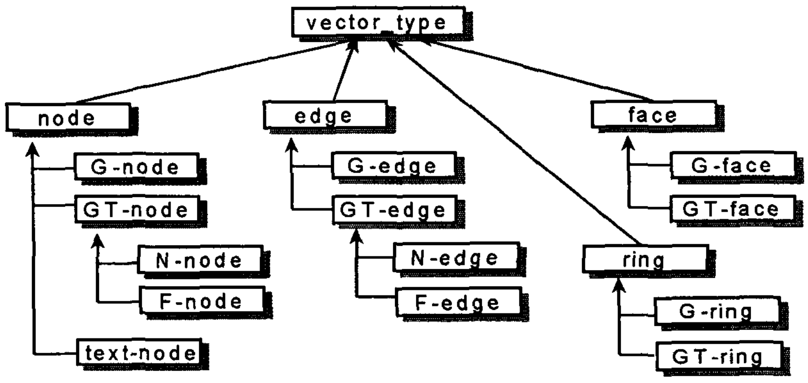
2.2 Internal Description Model

The internal model together with internal data structure is important in GIS. Different geometrical and topological data structures have been formalized by the results of recent work on exchange format standardization, such as SDTS (SDTS, 1992) and DIGEST-VPF (DGIWG, 1994). In this subsection, we present the internal description model which supports internal vector map structure with various levels of topology. Maps are the basic internal objects in geographical database system. The internal description model is on a basis of the multiple map layers whose type is a multiple-valued vector map structure layer (Falcidieno et al., 1992). Using multiple map layers technique for internal structure is nature to organize data and is efficient in terms of object manipulation and storage.

2.2.1 Structure Layer

Multiple map layers with a multiple-valued vector map structure represent several thematic objects which are captured by spatial signature. Each layer consists of the set of modules and maintains the topological integrity. An efficient and appropriate data model to handle geographical locations and spatial relationship between objects are required to manage spatial object. That is very important since a poor internal structure would strongly affect response time to queries. For this reason, we have considered three topological levels (geometry, network topology, and full topology level) that may match the user's need.

Each module consists of the set of internal primitive objects which is the lowest internal object. <Figure 3> depicts the type hierarchy for the internal primitive objects. The end nodes of the hierarchy are instantiated with objects.



<Figure 3> Internal primitive object type hierarchy

All of the internal primitive objects shown in <Figure 3> are discussed in details. Node type represents a zero dimensional object that specifies geometric location, or geometric location and the topological connectivities of one or more edges. G-node type represents an internal or extreme point of a line, or not a topological junction of two or more lines. GT-node type represents a point used for identifying the location of point features (or areal features collapsed to a point), such as towers, buildings, places, buoys, etc., and a zero-dimensional object that is a topological junction of two or more links or chains, or an end point of a link or chain. F-node type represents a node which is contained in a single face only. N-node type represents a zero-dimensional object that is a topological junction of two or more lines. Text node type represents a reference point used for displaying map and chart text (e.g., feature names) to assist in feature identification. G-node type represents an internal or extreme point of a line, or not a topological junction of two or more lines.

Edge type represents a connected non-branching sequence of line segments specified as the ordered sequence of points between those line segments. G-edge type represents a sequence of G-nodes. GT-edge type represents a sequence of one or more N-nodes and zero or more G-nodes. N-edge type is an edge which has N-node as start node and end node. F-edge type is a N-edge which has information about left and right faces.

Ring type is a sequence of nonintersecting edges with closure. A ring represents a closed boundary, but not the interior area inside the closed boundary (for full topology level only). G-ring is a sequence of closed G-nodes. GT-ring is a sequence of closed F-edges.

Face type represents the area which is constituted closed boundaries (for full topology level only). Face has one outer boundary and zero or more inner boundaries. G-face type represents a face which has G-ring boundaries. GT-face type represents a face whose boundaries are made up GT-rings.

2.2.2 Three Topological Levels

Using the internal description model, we are able to manage concurrently three different levels of topology. So user can match topological levels depend on his necessity. The definition of primitive objects depends on topological levels.

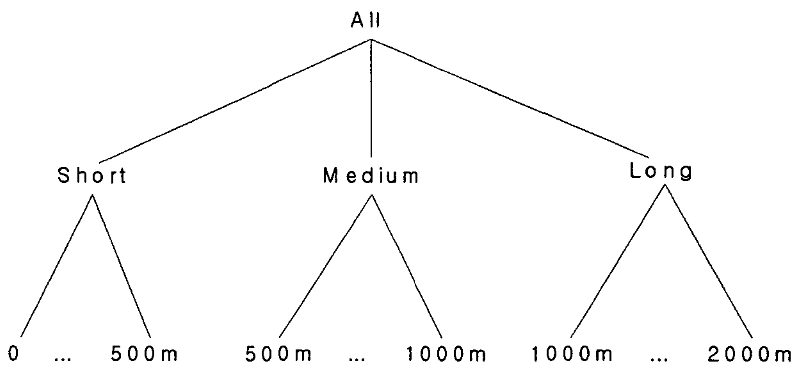
- *Geometry level* describes only geographical location information of each spatial object, but no topological information is explicitly present. Area information may be captured in geometry level by the use of closed lines (G-ring) which circumscribe an area and G-face. Primitive objects are text node, G-node, G-edge, G-ring, and G-face.
- *Network topology level* describes locational information as well as edge to node topological relationships, where every edge must begin and end on a node; however, edges may cross. This level of topology is sufficient to describe connectivity. Planar graph introduces the additional mathematical constraint that edges may not cross except at a node. This permits adjacency to be calculated although it is not directly stored in the structure. Primitive objects of network topology level are G-node, text node, N-node, and N-edge.
- *Full topology level* introduces the concept of a face and describes face to edge as well as node to face topological relationships and locational information. Primitive objects of full topology level are N-node, F-node, F-edge, GT-ring, and GT-face.

2.3 Spatial Shape Model

In this subsection, we give the spatial shape model which contains the logic for interpreting semantics and similarity of spatial object.

Type abstraction hierarchies(TAHs) (Chu et al., 1996) can provide approximate answers when no exact answers are available. TAHs represent the approximation knowledge at different levels of abstraction. For example, the Short-Range (i.e., from 0 to 500m) in the TAH of running distance(in Figure 4) is a more abstract representation than a specific running distance in the same TAH, e.g, 400m. We use this method to create spatial type abstraction hierarchies (STAHs). A STAH differs from a regular TAH in that it is based on multiple attributes rather than a single attribute. For example, the spatial distance between objects can be more precisely represented by (ρ, θ), where ρ is the distance and θ is the angular distance. STAHs represent general spatial object concepts in the higher levels and specific concept in the lower levels, are used to represent the knowledge of the selected spatial object and spatial operators on spatial sorts. STAHs provide a way to represent the semantics and similarity of spatial object.

The features of spatial objects in a spatial database are extracted according to the spatial shape model. These features are then classified by



<Figure 4> Type abstraction hierarchy of running distance

a conceptual clustering algorithm (Hsu et al., 1996) and the feature classification hierarchy is represented in STAHs which provide a multi-level knowledge representation of the object based on analyzed features. Such STAHs are used to process queries with semantic operators (e.g., "Find a large building NEARBY the Seoul Arts Center") and queries with similar-to operator (e.g., "Find parks with similar shape to namsan park"). The conceptual terms (e.g., NEARBY) can be translated to value ranges of relevant features via STAHs.

3. Spatial Object-Oriented Query Language

Spatial object-oriented query language (SOQL) is an OQL-based query language. OQL (object query language) (Catell and Barry, 1997) is a SQL-like object oriented query language, and its syntax looks as follows:

```
query ::= SELECT[DISTINCT] query
          FROM identifier IN query{, identifier
          IN query}
          [WHERE query]
```

The most obvious difference between conventional and spatial systems is the ability to spatial data graphically (Egenhofer, 1994). Interactive-graphic presentation is powerful for mapping systems, because the content of maps can be quickly modified. Objects can be added to, removed from, or modified on an existing map without the need to start with a new drawing from scratch again. This implies that the user must have tools to manipulate a map. Unlike conventional systems which treat each result as an entirely new representation and do not refer to earlier results, several related graphic drawings are often overlayed over each other or on 'thematic layer' is removed from another. These processes are the particular power of spatial database systems. Furthermore, individual objects on a map may be highlighted to facilitate their identification in complex situations. Hence, query language should support various presentation types for maps.

For providing an integrated mechanism for the graphical display of spatial objects and the retrieval of objects are executed, we add DISPLAY-ON-STORE clause in OQL syntax. The SOQL has the following structure:

```
[DISPLAY <display-form-comma-spec-list>]
[ON <layer-name-comma-spec-list> : <display
    condition>]
[STORE <temporary-storage-name>]
OQL statement
```

The DISPLAY clause allows users to tailor the graphical display of query results to their specific needs. To fulfill this, SOQL supports various display forms (COLOR, REMOVE, BLINK, FILL, and SYMBOL) in <display-form-comma-spec-list>. To specify display forms on spatial objects, users select the current settings using the full-down menus for the setup. The set of display forms is extensible to apply different applications.

The ON clause is used in order to display query results on a layer of the base map. The <layer-name-comma-spec-list> represents the base layer where spatial objects are displayed, and <display condition> takes one of the following two display modes: (1) NEW clears the viewpoint before drawing the result maps; (2) OVERLAY overlaps the current query result on an existing maps. When the ON clause is omitted, the query results are displayed on the layer where the spatial objects are stored.

The STORE clause is used to store the query results in the specific storage name <temporary-storage-name> in order to use previous query results in the following query. The <temporary-storage-name> represents the name of the temporary storage named 'temp_sys'.

The following query is an example of SOQL.

Query : Display blinkly on road and building maps the hospitals *nearby* within 500m from the Sungbuk-ku Office and the roads more than two lanes. And store the result in temp_1.

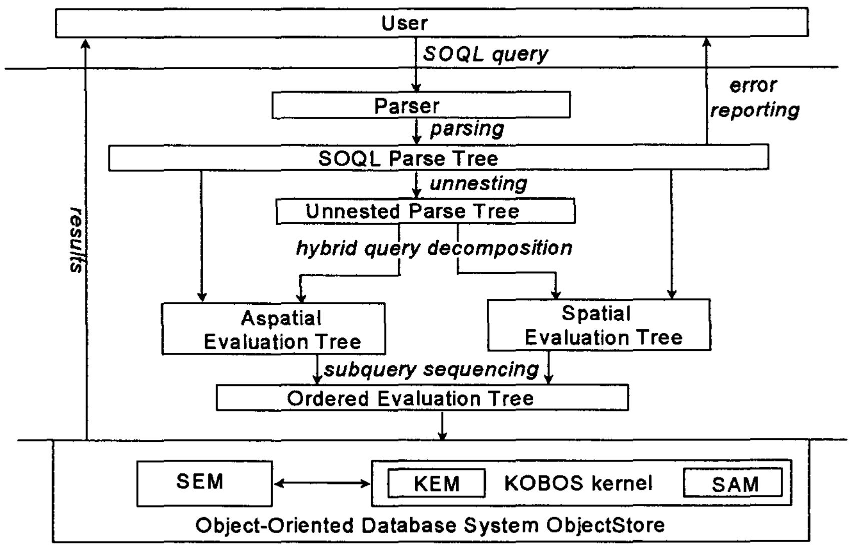
```
DISPLAY BLINK(b2.location)
ON road, building:NEW
STORE temp_1
SELECT b2.location
FROM b1 IN building, b2 IN building, r IN road
WHERE b1.name = "Sungbuk-ku Office"
AND b2.purpose = "hospital"
AND b1.location NEARBY b2.location)
AND r.lane >= 2
```

4. The Architecture

It is highly desirable that a spatial database system provides a storage and query processing structure that integrates both the aspatial and spatial components of the database. In such an integrated system (Guting, 1994), there is no difference in principle between spatial and aspatial data type and operators, and a user would be presented with a single query language that expresses spatial and aspatial properties. Such a system also support efficient execution of these hybrid queries. An integrated architecture can be obtained by full implementation or by extensions to a given DBMS. In our prototype system KOBOS, we choose the second approach to reduce our times, resources and implementation efforts.

The KOBOS is an integrated spatial database prototype which uses a commercial OODB ObjectStore (Lamb et al., 1991) as an implementation platform. That covers the basic object-oriented database functionalities, including transaction management and indexing. Nevertheless, an OODBMS does not directly support spatial object types and spatial operations. That is, ObjectStore does not support efficient spatial object access structure such as R-tree (Guttman, 1984), and collection type provided by ObjectStore does not provide spatial query processing capability. Also it does not contain relationship between spatial objects which is necessary for spatial query processing. Therefore, in KOBOS, aspatial subsystem called KOBOS kernel is added in OODB ObjectStore to support these functions. The architecture of the KOBOS is illustrated in <Figure 5>.

Spatial query processing of SOQL is done by first, compilation of the query, and secondly, excution. A top-down spatial query processing consists of the following four stages: parsing, unnesting, decomposition, and subquery sequencing. Parse tree is created corresponding to the SOQL statement inputted in the SOQL statement by the user. The parser checks the



<Figure 5> Architecture of the KOBOS

syntactic and semantic correctness of the SOQL query, and then if there is a syntactic error, no parse tree is created and corresponding error message is reported. One of the powerful features of SOQL is the ability of nested queries. But, the processing of nested queries costs too much. If the parse tree is nested, it transforms unnested parse tree which is a partially ordered set of unnested queries using the method proposed in (Kim, 1982). The unnested hybrid parse tree object for SELECT-FROM-WHERE clause is decomposed into subtrees which are either totally spatial or aspatial evaluation tree objects as in (Ooi, 1990).

Using heuristic query optimization techniques, spatial and aspatial evaluation tree objects choose the cheapest ordering among all evaluation tree objects. The ordered evaluation tree object is created by this ordering. We adopted the following subquery sequencing rules which is proposed in (Ooi, 1990).

Heuristic Rule 1. For an expression and $(Q_1(S_1),\dots,Q_i(S_i),\dots,Q_n(S_n))$, If $S_i \not\cap (S_1 \cup \dots \cup S_{i-1} \cup S_{i+1} \cup \dots S_n)$ then $Q_i(S_i)$ is executed just before the result is required.

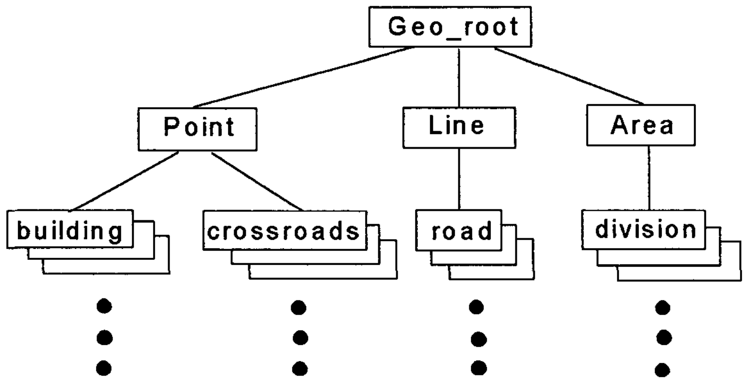
Heuristic Rule 2. For an expression $or(Q_1(S_1),\dots,Q_n(S_n))$, the order of execution of subqueries is immaterial to cost saving. So the queries may be executed in any order.

Heuristic Rule 3. For an expression and $(Q_1(S_1),\dots,Q_i^a(S_i),\dots,Q_j^a(S_j),\dots,Q_k^s(S_k),\dots,Q_n(S_n))$, suppose $|S_i| = 1$, and $|S_j| = 1$, and S_i, $S_j \subset S_k$, then at least one of Q_i^a or Q_j^a must be executed before Q_k^s, where Q_i^a and Q_j^a are aspatial predicates, and Q_k^s is spatial predicate.

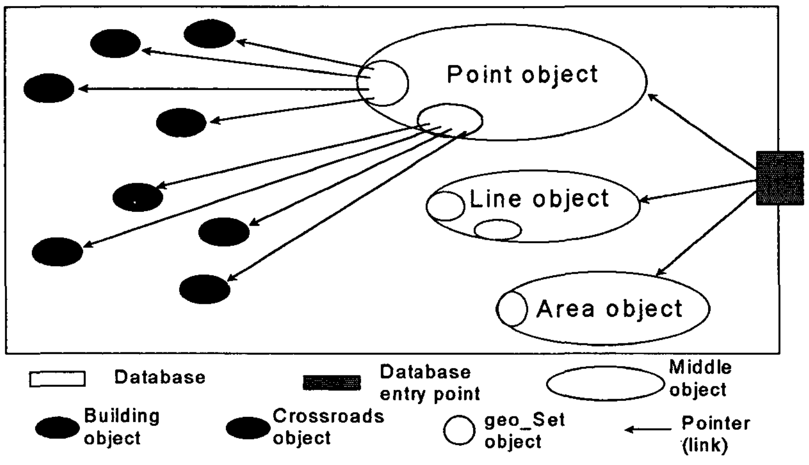
Geo_Collection is the collection class which groups spatial objects together in KOBOS. And it has the following subtypes: geo_Set, geo_Array, geo_List, and geo_Bag. KOBOS kernel contains the SAM(spatial access manager) and KEM(knowledge processing manager) module. SAM stores geo_Collection in the form of os_Collection (Lamb et al. 1991) in ObjectStore. The spatial query processing module takes a spatial query as an input and returns a geo_Collection as a result of query processing. And KEM processes the semantic queries using the STAHs.

A *schema* is a set of type definitions including their definitions of the structure and the behavior. And it is not fixed once written but evolves over time in order to capture the ever-changing requirements. New types are introduced, old types deleted, operations modified, errors eliminated, and so on. It is also true for the schema of spatial objects. The SEM(Schema Manager) component is tightly linked to a KOBOS kernel and build schema hierarchy, import schemata, rename schema components, and check the validity of a schema. KOBOS is to be a GIS server for a diverse and distributed collection of applications. In order to implement a query examples in Section 3, we use geo_root which provides a way of recording and accessing persistent data. This root's value serves as an *entry point* into a database. Most of the persistent data in that database is then retrieved automatically when referenced, that is, via navigation from an entry point object.

<Figure 6> shows the data management hierarchy for the example and a persistent variable 'geo_root' serves as the entry point object of an example database. The process of reading persistent data is entirely invisible to the user. Therefore when spatial query processing, user can search data with entry point and public functions in database root. A storage structure of an example illustrates in <Figure 7>.



<Figure 6> Data management hierarchy for an example



<Figure 7> Storage structure of an example

5. Conclusion

Most of spatial database systems are built above a RDBMS. But there are several limitations for modeling complex spatial objects. To overcome these limitations, using object-oriented paradigm, we have developed a knowledge-based prototype system called KOBOS on top of the commercially available OODB ObjectStore.

In this paper, we have presented the three levels of object-oriented data models: a spatial shape mode, a spatial object model called SAS and an internal description model. The spatial shape model captures the structured abstracts and knowledge needed for spatial object retrieval. We use spatial type abstraction hierarchies (STAHs) to represent the knowledge of the selected spatial object features and spatial relationships. SAS supports spatial object types and their polymorphic operations. To make the spatial sorts of the SAS, two new object types, geo_type and graphic_type, have been added in the ObjectStore's metatype hierarchy. Also several polymorphic operations have been defined on spatial sorts such as location operations, arithmetic operations, topological relationships, mouse operations, and object construction operations. Using this model frame, it is possible to define additional types and operators depending on its applications, which makes the system extensible. The internal description model suppports efficient and appropriate framework to store spatial coordinates and topology between two spatial objects. In the internal description model, we considered map as a set of layers. The two levels of an object-oriented data models give the independence between the thematic division of data and cartagraphic presentation.

A spatial query language should provide graphical display of spatial objects in query results interactively, and spatial object type values to be used as "constant" in queries using pointing device such as mouse. For providing an integrated mechanism for the graphical display of spatial objects and the retrieval of objects, SOQL was designed. This gives the unified view for spatial and aspatial objects.

Finally, we proposed the architecture of KOBOS which uses ObjectStore as an implementation platform. Since an OODBMS does not directly support spatial object types, spatial operations, and spatial indexing, we added a spatial subsystem called KOBOS kernel. The KOBOS kernel contains the definition of classes and methods that are important for the representation and management of spatial objects. And it contains the manager for processing the semantic queries using the STAHs. In particular, SAS data model and internal description model is implemented here and we used R-tree spatial indexing structure. And an overview of spatial query processing was presented. The current trend is oriented towards a top-down approach and we used a top-down query processing method to evaluate hybrid queries.

The prototype system KOBOS has been partially implemented. We will add a new spatial indexing structure and the cost model for ordering the subqueries. Also, we will extend our model to support cartographic time later on.

References

R. Cattell, D. Barry, *The Object Database Standard : ODMG-93*, Morgan Kaufmann, 1997.

W. W. Chu and Q. Chen, "A structured approach for cooperative query answering", *IEEE Transaction on Knowledge and Data Engineering*, 6(5), Oct(1994), 738-749.

W. W. Chu, H. Yang, K. Chiang, et al., "A scalable and extensible cooperative information system", *Journal of Intelligent Information Systems*, 6(6), (1996).

B. David, L. Raynal, G. Schorter and V. Mansart, *GeO2: Why objects in a geographical DBMS ?*, Proc. of the 3rd Internatinal Symposium SSD, June(1993), 264-276.

DGIWG, *DIGEST(digital geographic information exchange standard)*, edition 1.2, Defence Mapping Agency, USA, Digital Geographic Information Working Group, January 1994.

L. Fegaras, C. Srinivasan, A. Rajendran, D. Maier, *Lambda-DB: An ODMG-Based Object-Oriented DBMS*, ACM SIGMOD International Conference on Management of Data, (2000), Dallas, Texas, 583.

M. Egenhofer and R. Franzosa, "Point-set Topological Spatial Relations," *Int. J. Geographical Information Systems*, (1991), Vol. 5, No. 2, 161-174.

M. Egenhofer, "Spatial SQL: A Query and Presentation Language," *IEEE Transaction On Knowledge and Data Engineering*, (1994), Vol.6, No.1, 86-95.

B. Falcidieno, C. Pienovi, and M. Spagnuolo, *Descriptive modeling and prescriptive modeling in spatial data handling*, Proceedings of the International Conference GIS-From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, (1992), Pisa, Italy, Sep. 21-23, LNCS Vol.367, New York: Springer-Verlag, 122-135.

Spatial Data Transfer Standard (SDTS), Federal Information Processing Standards Publication, August 1992.

O. Gunther and W. Riekert, *The Design of GODOT: An Object-Oriented Geographic Information System*, Bulletin of the Technical Committee on Data Engineering, Sept., (1993), Vol.16, No.3, 4-9.

R. Guting, *Gral: An extensible relational database system for geometric applications*, Proceedings of the Fifteenth International Conference on VLDB, (1989).

R. Guting, *Second-order signature: A tool for specifying data models, query processing, and optimization*, Proceedings of the ACM SIGMOD Conference, (1993), 277-286.

R. Guting, "An Introduction to Spatial Database Systems," *VLDB Journal*, (1994), 3, 357-399.

R. Guttman, *R-trees: A dynamic index structure for spatial searching*, Proc. ACM SIGMOD International Conf. on the Management of Data, (1984), 47-57.

C. C. Hsu, W. W. Chu & R. K. Taira, "A Knowledge-Based Approach For Retrieving Images By Content", *IEEE Transactions on Knowledge and Data Engineering*, (1996), 8(4), 522-532.

W. Kim, "On optimizing an SQL-like nested-query," *ACM Trans. on Database Sys.*, (1982), Vol.7, No.3, 443-469.

C. Lamb, et al., "The ObjectStore database system," *Communications of the ACM*, (1991), Vol.34, No.10, 50-63.

B. Ooi, *Efficient Query Processing in Geographic*

Information Systems, Lecture Notes in Computer Science 471, Springer-Verlag 1990.

J. A. Orenstein and F. Manola, "PROBE spatial data handling and query processing in an image database application," *IEEE Transactions on Software Engineering*, (1988), Vol. 14, 611-629.

C. G. Pires and J. C. Machado, *DORS: Database Query Optimizer with Rule Based Search Engine*, SugarLoafPLOP 2002, August 5-7(2002).

요약

지식기반 객체지향 공간 데이터베이스 시스템

김양희*

본 논문은 지식기반 객체지향 공간 데이터베이스 시스템 KOBOS를 제안한다. 객체지향 공간 데이터베이스 시스템의 데이터 모델링과 근접 질의답변에 지식기반 접근법을 도입한다. 공간 객체와 근접 공간 연산자를 다루기 위해 다음과 같은 세 단계 객체지향 데이터 모델을 제안 하고 있다: (1) 공간 형상 모델; (2) 공간 객체 모델; (3) 내부 기술 모델. 근접 공간 연산자의 범위는 공간 타입 추상 계층으로 알 수 있다. 또한 객체지향 공간 질의어인 SOQL을 제안한다. SOQL은 공간 객체의 다양한 출력과 공간 및 비 공간 객체의 검색을 수행할 수 있는 통합 기능을 제공해 준다. 효율적인 혼합 질의 처리를 위하여, 하향 공간 질의 처리 방법을 이용하여 처리해 준다.

* 한국체육대학교 교양 과정부