Using Object Deputy Database to Realize Multi-Representation Geographic Information System

Zhiyong Peng
Computer School
Wuhan University
Hubei, China, 430072

Yuwei Peng
State Key Lab of Soft. Eng.
Wuhan University
Hubei, China, 430072
peng@whu.edu.cn

Boxuan Zhai
Computer School
Wuhan University
Hubei, China, 430072

ABSTRACT
Multi-representation geographic information system (GIS) is used to store and manage geographic objects with different scales and semantics. It is difficult for relational and object-oriented databases to support multi-representation GIS. This paper proposed a new approach to realize multi-representation GIS using our object deputy database, where an object can have multiple deputy objects which inherit attributes and methods of the source object and can have their own attributes and methods. Multi-representations of a geographic object are defined by its deputy objects. So deputy objects can be used to represent viewpoint-multiplicity and role-multiplicity of geographic objects. Dynamic classification and automatic consistency maintenance of geographic objects can be supported through the update propagation mechanism of our object deputy database. In addition, the bilateral links between a geographic object and its deputy objects make cross-representation query easier. We have implemented a multi-representation GIS based on our object deputy database. Experiments show our method is more efficient than the traditional ones.

Categories and Subject Descriptors
H.2.1 [Database Management]: Logical Design—data models

General Terms
Management, Design, Experimentation

Keywords
Multi-Representation, GIS, Object Deputy Database

1. INTRODUCTION

In this paper, we proposed realization of multi-representation GIS using a new kind of database which is based on the object deputy model. Our object deputy database can provide unified realization of object views, role multiplicity and dynamic classification. It has both flexible view mechanisms of relational databases and complex data management capabilities of object-oriented databases. In our object deputy database, geographic objects are allowed to have one or more deputy objects. Deputy objects inherit attributes and methods of geographic objects in different ways so that geographic objects can be displayed at different scales. Deputy objects can also be added with some application related attributes and methods so that they can have different semantics. Thus, deputy objects can be defined as multiple representations of geographic objects. There are bilateral links between geographic objects and their deputy objects, which are used to realize object update propagation so that geographic objects can be classified dynamically and the consistency can be maintained automatically. In addition, bilateral links allow cross-representation queries so that we can provide richer queries based on multi-representation of geographic objects. We have implemented a multi-representation GIS based on our object deputy database. The experiment shows our system is more efficient than the multi-representation GIS based on the object-relational database PostgreSQL.

The remainder of this paper is organized as follows. Section 2 elaborates how to realize multi-representation GIS based on the object deputy database. Section 3 describes our system implementation and evaluates its efficiency. The paper is concluded in Section 4.

2. MULTI-REPRESENTATION GIS

2.1 Multi-Representation Modeling
Modeling multi-representation by the relational data model needs multiple tables each of which is used to define a single representation. The multiple representations can be generated by joining these tables. The join operation is usually time-consuming. In addition, we need to develop complex mechanisms to maintain the consistency of these tables [3].

The object-oriented data model can be used to define complex geographic objects. Most of object-oriented databases restrict that each object is a direct instance of only one class and indirectly belongs to all super classes of this class. That is, an object cannot reside simultaneously in two classes which are not related by a sequence of IS-A relationships.
One representation of a geographic object is usually defined by one class. If there are not IS-A relationships between the multiple representations of a geographic object, they are hard to be stored in object-oriented databases [1][7].

There is another approach [8] to realize multi-representation GIS by integrating multiple GIS with different scales and semantics. However, how to identify geographic objects in different GIS which represent the same real world entity [2] is a very difficult problem faced by this approach. In addition, data redundancy and consistency maintenance [5] are also great handicaps.

In order to solve the problems of the traditional data models, we proposed the object deputy model [6]. That is, every entity of real world is represented by an object. Its deputy objects are used to represent its views, roles and support dynamic classification. The schema of objects (or deputy objects) which have same attributes and methods is defined by a class (or deputy class). The object deputy model has the following features:

- A deputy object has its own persistent identifier, and may have additional attributes and methods that are not derived from its source object(s).
- A deputy objects can inherit all or part of attributes and methods from one object or several objects.
- The inheritance is realized through switching operations that can change the names and types of the inherited attributes and methods.
- There is a bilateral link between an object and one of its deputy objects, which allows not only inheritance but also update propagation between them.
- A deputy object can have its own deputy objects as well. Many objects can be combined through a single deputy object.

Based on the object deputy model, we have developed a new kind of database system named as Totem which enables users to define flexible object views, support the object dynamic classification and realize cross class queries. In our realization based on Totem, each representation of a geographic object can be defined as one of its deputy objects which inherit properties of the geographic object by switching operations and can be added with some application-related attributes and methods. Thus, we can avoid data redundancy as much as possible. One geographic object is allowed to have multiple deputy objects as its multiple representations. Because there are bilateral links between a geographic object and each of its deputy objects, cross-representation queries can be supported without many time-consuming join operations. Furthermore, the consistency among multiple representations can be maintained automatically by the update propagation.

In our multi-representation GIS, a class named as BGO (Basic Geographic Objects) is used to define basic geographic objects which have basic attributes, such as name, position (encoding of longitudes and latitudes of natural shapes) and basic methods, such as drawing shapes of basic geographic objects. In order to display the same geographic object at different scales, multiple deputy classes can be defined from the class BGO. If the geographic shapes at some scale can be derived from ones at the other scale by various cartographic generalization algorithms, we can define switching operations for the deputy class to generate geographic shapes at that scale. Since the same geographic object can be used by different applications, multiple deputy classes can be defined with some additional attributes and methods from the class BGO to represent multiple different semantics.

As shown in Figure 1, the class BGO contains 5 geographic objects which have multi-scales and multi-semantics. The deputy class Most_detailed is the most precise one. The attribute ‘name’ is inherited and the attribute ‘geo_data’ is extended, which stores the most detailed geometry data for each geographic object.

Suppose the geographic objects at the Less_detailed can be derived from ones at the Most_detailed. The value of ‘geo_data’ at the Less_detailed is the result of invoking function ‘Generalize’ with parameter of Most_detailed.geo_data. The function ‘Generalize’ is a proper cartographic generalization operation [4].

In this example, geographic objects are dynamically classified according to their semantics which are defined by deputy classes such as dock, navigation fix and sea route. After the creation of the deputy class, the deputy objects which are representations of corresponding geographic objects will be generated automatically. The deputy class Dock can be furthermore classified by defining its deputy classes Commercial_dock and Fishing_dock.

2.2 Cross Representation Query

Since an object and its deputy objects are linked bilaterally, the query on one class or deputy class can obtain the related information from the other classes or deputy classes by following bilateral links. We call this kind of queries as cross class queries. In order to support the cross class query, we defined a symbol → to represent the navigation among classes.

In our model, there may be a path formed by bilateral links between any two objects, no matter how long the path is. For example, we can define the following cross class query:

```
SELECT Fishing_dock.name, Fishing_dock.geo_data FROM Fishing_dock
WHERE Fishing_dock → Commercial_dock
```

The system will firstly scan the deputy class Fishing_dock, and then check its connected deputy object in the deputy class Commercial_dock, and see whether its thruput is greater than 200. If the condition is satisfied, then its inherited attribute name will be returned, along with the value of its local attribute coastline. So the result will be <Dock_b, 2000>.

Actually, we can use the arbitrary class as the beginning class of the path expression, and the final result could be a combination of several simple queries like the above. Thus,
the complex queries involving various semantics and scales can be defined through a few of simple selection statements. This allows the user to emphasize the viewpoints of the objects they are interested in.

### 2.3 Consistency Maintenance

When the attribute values of a geographic object are modified, its multiple representations will be updated automatically without other extra operations. This change can be reflected by the switching operation.

When geographic objects are modified, their representations should be updated accordingly. Our object deputy database can provide the update propagation mechanism to maintain their consistency. That is, when a geographic object is added, its representations may be created automatically according to semantic constraints; when a geographic object is deleted, its representations will also be deleted; when a geographic object is updated, some of its representations may be deleted and some other new representations may be added. Thus, our multi-representation GIS can classify geographic objects dynamically.

### 3. IMPLEMENTATION AND EVALUATION

Using our object deputy database, we implemented a multi-representation GIS, which provides users a drawing function of selecting the geographic attributes from querying the non-geographic attributes of the entities.

As we can see in Figure 2, the area on the top left corner is 'map area', which displays the chart. The area on the right is 'classification tree', through which users can select various deputy classes which are used to represent different scales and semantics. While the lower left corner area is 'condition window', in which querying conditions of the selected deputy class are specified. When users select a deputy class in the 'classification tree', the corresponding query input box will appear in 'condition window'. Then users can input their query conditions and add them into the final condition set. After all conditions on different representations are input and confirmed, the chart satisfying the conditions will be drawn in the 'map area'.

In order to illustrate the efficiency of our multi-representation GIS, we developed the same functions based on an object-relational database PostgreSQL. The experiments were performed using Totem and PostgreSQL as databases respectively under such a platform: Intel(R) Celeron(R) CPU 2.00GHz CPU with 256 MB memory and RedHat Linux 9.0 with Mapserver 2.6.0.
In the experiments, we will compare the storage space of the actual geographic data and the response time needed when querying the same amount of geographic objects between two systems based on two different databases. Then the response time can be gotten by computing the time consumed by Mapserver in querying the entities required. We have done 5 tests using 5 data sets extracted from the chart data of the Tianjin port. The sizes of these 5 data sets are 190, 4750, 9500, 14250, 19000. The testing results are shown in Figure 3 and Figure 4.

From the results of the above experiments, we can see that: (1) when the data size is considerable, the response time of the multi-representation system based on Totem is obviously excelled that of the system based on PostgreSQL. (2) the consumed storage spaces are almost the same between these two systems.

4. CONCLUSION
So far the realization of a multi-representation GIS has been considered to be very difficult due to lack of advanced database functions. Our object deputy database can provide a unified realization of object views, role multiplicity and dynamic classification. We implemented a multi-representation GIS based on it, which can store geographic objects together with their multiple representations and support cross representation queries. The experiment shows our multi-representation GIS is more efficient than the traditional ones.

Only the consistency maintenance among multiple representations of a single geographic object is considered in this paper. The consistency maintenance of topology among different representation objects will be discussed in future.

5. ACKNOWLEDGMENTS
This research is supported by National Natural Science Foundation of China (60573095), the Program for New Century Excellent Talents at University of China (NCET-04-0675), Research Fund for the Doctoral Program of Higher Education(20050486024), Hi-Tech Research and Development Program of China (2006AA122210), the Key Project of Chinese Ministry of Education (107072), National Basic Research Program of China (2007CB310806) and State Key Lab of Software Engineering under grant: SKLS05-01.

6. REFERENCES