OBJECT-ORIENTED DATA MODELS IN GIS

Gong Jianya Li Deren
Department of Photogrammetry and Remote Sensing
Wuhan Technical University of Surveying and Mapping
Wuhan, P. R. China

Abstract

Currently, experts in Geographic Information System are promoting an object-oriented approach for GISs. This advanced technique, based upon the definition of object types in combination with the corresponding operations in a modular fashion, is characterized by clear coding which can be easily maintained. This paper discusses object-oriented models for geometric and attribute data in GIS. The investigation of a GIS with this concept has been carried out and some approaches and applications are also described.

Key words: Object-oriented data model, Classification, Generalization, Assoiation, Aggregation, Inheritance, Propagation.

1. INTRODUCTION

There are three traditional data models: hierarchical, network and relational models, which all try to deal with the same problems with tabular data. The relational data model is the most popular DBMS model for GIS. For example, ARC/INFO uses INFO, System 9 uses EMPRESS, several GISs use Oracle, some PC-based GISs use Dbase III. However, the relational model exhibits some problems such as efficiency, data semantics, model extension, object identity, and program interface. It should be pointed out that the three traditional data models including the relational model do support these vital features in various degrees.

2. THE CONCEPTS OF OBJECT-ORIENTED DATA MODEL

Object-oriented approach is a new technology in software engineering and database. A definition of object-orientation is that an entity of whatever complexity and structure can be represented by exactly one object [Egenhofer and Frank 1989a], and every object is a single programming entity that combines data and procedures or functions that operate on that data [Shafer 1988]. The object-oriented data model is built upon the four basic concepts of abstraction: classification, generalization, association, and aggregation. The semantic model is enriched by the concepts of inheritance and propagation which describe the derivation of properties in generalization hierarchies and values in aggregation hierarchies, respectively.

2.1 Object

In an object-oriented approach, all conceptual entities model objects, for example, a node, an arc, a river or a province on a map are objects. The object contains a set of data to describe its behavior and a set of methods to operate itself. That is:

$$Object=(ID, S, M) \qquad (1)$$

Where: ID is an identifier of the object;
M is a set of methods;
S is the behavior of the object, it is represented by either attributes or identifier of other objects. That means that an object can include another object in an object-oriented approach.

2.2 Classification

Classification is the mapping of several objects (or instances) within a common class (or type) [Egenhofer and Frank 1989a]. In the object-oriented approach, every object is an instance of a class. Each class characterizes the behavior of its instances by describing the operators that can manipulate these objects. These operations are the only means of manipulating objects. Classification is referred to as the instance of relationship because the individuals are instances of the corresponding class.

Thus, we establish an abstraction class by :

$$Class=(CID,CS,CM) \qquad (2)$$

Where: CID is an identifier of the class, i.e. class name;
CS is the behavior of the class;
CM is a set method of the class.
Then

$S \in CS$ and $M = CM$ when Object Class

For example, a city GIS includes the classes building, street, park, phone-line, etc. A single instance, such as the building with the address 'King street 51', is an object of the class building. Operations and properties are assigned to object types, so for instance the class building may have the properties address, holder and building date which are specific for all buildings. The operations like data retrieval and update are also the same for all buildings.

2.2 Generalization and Inheritance

2.2.1 Generalization Generalization organizes several classes of the objects with common properties and operations to form a more general superclass. There are two classes:

$$Class_1=(CID_1,CS_A,CS_B,CM_A,CM_B) \qquad (3)$$
$$Class_2=(CID_2,CS_A,CS_C,CM_A,CM_C)$$

$Class_1$ and $Class_2$ have the same subset CS_A of behavior and a subset CM_A of method.

$CS_A \subset CS_1$ and $CS_A \subset CS_2$ as well as $CM_A \subset CM_1$ and $CM_A \subset CM_2$

So, they are abstracted to form a superclass:

$$Superclass=(SID_1,CS_A,CM_A) \quad (4)$$

Here SID is the identifier of the superclass.

After defining the Superclass, the $Class_1$ and $Class_2$ are written:

$$Class_1=(CID_1,CS_B,CM_B) \quad (5)$$
$$Class_2=(CID_2,CS_C,CM_C)$$

$Class_1$ and $Class_2$ are called subclasses of the Superclass.

Subclass and superclass are related by an is_a relation. For example, the class building is the superclass of the class hotel because a hotel is a building. A subclass may also have a further classification. The building/hotel generalization can be extended with the classes inn, guest house and motel etc. While hotel is a subclass of building, it can be at the same time a superclass of guest house.

2.2.2 Inheritance In generalization hierarchies, some of the properties and methods of the subclasses depend upon the structures and properties of the superclass(es). Inheritance is a tool to define a class in terms of one or more, other more general classes. Properties and operations which are common for superclass and the subclasses are defined only once, and the properties and operations of the superclass are inherited by all objects of the subclasses. Inheritance is the transitive transmission of the properties from one superclass to all related subclasses, and to their subclasses, etc. But subclasses can have additional, specific properties and operations which are not shared by the superclass. This concept is very powerful, because it reduces information redundancy and maintains integrity.

The inheritance can be single or multiple. Single inheritance requires that each class has at most a single immediate superclass. Figure 1 has shown an instance of single inheritance.

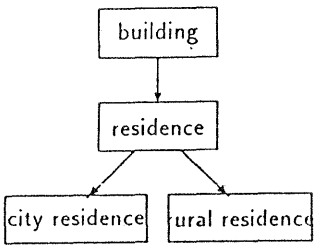

Fig. 1 An instance of single inheritance

In multiple inheritance, one subclass has more than a single direct superclass, such as the subclass hotel may have superclass building and superclass business.

Suppose:

$$Class_3=(CID_3,CS_A,CS_B,CS_C,CM_A,CM_B,CM_C) \quad (6)$$

Here CS_A, CS_B, CM_A, CM_B, are in common with two other classes. Therefore two superclasses can be constructed:

$$Superclass_1=(SID_1,CS_A,CM_A) \quad (7)$$

$$Superclass_2=(SID_2,CS_B,CM_B)$$

In order to acquire all information and methods, $Class_3$ must inherit the data and operations from $Superclass_1$ and $Superclass_2$.

An example from geography shows how multiple inheritance often combines two distinct hierachies. One hierarchy is determined by the separation of artificial and natural transportation links, whereas the other hierarchy distinguishes water bodies. Classes with properties from both hierarchies are channels, that are artificial transportation links and water bodies, and navigable rivers, whereas ponds belong only to one hierachy(Fig. 2)

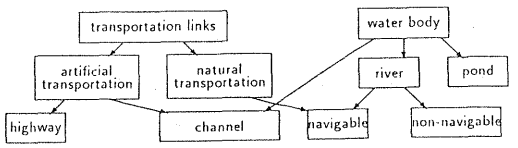

Fig. 2 An example of multiple inheritance

2.3 Association,Aggregation and Propagation

2.3.1 Association An association relates two or more independent objects by considering the relation among objects as a higher level set object. For example, parcel 1 and parcel 2 all belong to Mr. Neilbsen, they can be grouped as Neilbsen's parcel. Parcel 1 and parcel 2 are called members of Neilbsen's parcel. Suppose:

$$Object_1=(ID_1,S_A,S_B,M) \quad (8)$$
$$Object_2=(ID_2,S_A,S_C,M)$$

A new object contains $Object_1$ and $Object_2$:

$$Object_3=(ID_3,S_A,Object_1,Object_2,M) \quad (9)$$

In this case, the $Object_1$ and $Object_2$ are writen:

$$Object_1=(ID_1,S_B,M) \quad (10)$$
$$Object_2=(ID_2,S_C,M)$$

Here $Object_1$ and $Object_2$ are termed component objects. Hence, this abstraction is referred to as the member_of relation.

2.3.2 Aggregation A similar abstraction mechanism to association is aggregation which models composed objects. i.e., objects which consist of several other objects. Several objects can be combined to form a semantically higher level of object where each part has its own functionality. The operations and properties of an aggregate are not compatible with those of parts. For example, the class building is an aggregate of all walls, doors, windows and roofs which are parts_of it.

There are two different component objects:

$Object_1=(ID_1,S_1,M_1)$ (11)
$Object_2=(ID_2,S_2,M_2)$

They are composed to form a new object:
$Object_3=(ID_3,S_3,Object_1(S_u),Object_2(S_v),M_3)$ (12)

Here $S_u \subset S_1, S_v \subset S_2$.

2.3.3 Propagation The mechanism to describe such dependencies and ways to derive values is called propagation. It supports complex objects which do not own independent data and is based upon the concept that values are stored only once, i.e., for the properties of the components, and then propagated to the properties of the composite objects. For example, the number of the beds in a hotel is the sum of the beds of all bedrooms. The propagation model guarantees consistency, because the dependent values of the aggregate are derived and need not be updated every time after the components have been changed.

The classification, generalization, association and aggregation enrich the semantic models so that the object-oriented approach supports multiple semantic functions, They are written as the following table of relations of the abstractions:

classification	instance_of
generalization	is-a
association	member_of
aggregation	parts_of

Comparisons between inheritance and propagation are made by the following:

- Inheritance is defined in generalization (is_a) hierachies, while propogation acts in aggregation (parts_of) or association (member_of) hierachies.
- Inheritance is a top-down approach, inheriting from the more general to the more detailed class; propogation, on the other hand, acts bottom-up.
- Inheritance describes properties and operations; while propagation derives values of properties.
- Inheritance is implicit, all superclass properties are inherited from its subclass; while propagation is explict, i.e. only if specified. In addition; it must be stated for which relationship the propagation holds; which component properties are used; which composite property is derived and how the result is obtained.

3 OBJECT-ORIENTED DATA MODEL IN GEOMETRY

3.1 Introduction

In geometry, there are four types of features: point-, line-, surface-, and complex features, which are considered as four superclasses of all features. Figure 4.3.1 shows the relationships between the four geometrical superclasses and various feature classes according to geographical properties. Therefore, in geometry, we can abstractly deal with point-, line-, surface-, complex features, when we define a feature class, besides defining geographical class, we must declare its geometric type. For example, when a building is being defined as a class of building, it is simultaneously defined as a surface feature, and it is automatically linked to the data structure of surface features so that it inherits geometric information and the procedures to act on the data.

The geographical objects will be identified and described by their geometric characteristics and their thematic attribute (non geometric). That means that two kinds of data need to be stored in GIS,and an identifier must be required to link them (see Fig. 3). Here we discuss the object-oriented model of the geometric data.

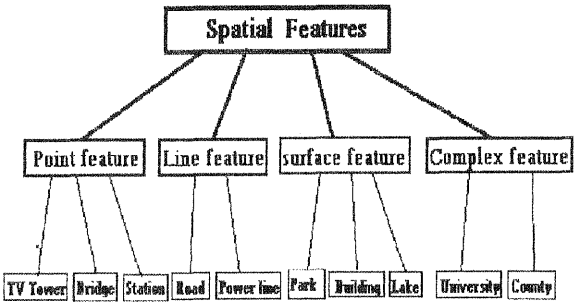

Fig. 3 relation between four geometric superclasses and geographic feature classes

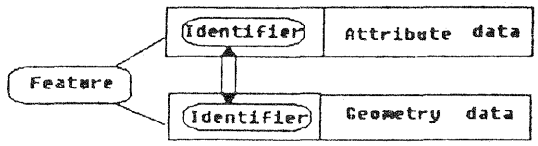

Fig. 4 Link between geometric and attribute data

3.2 Geometrical Topology, Data Sharing and Object-oriented Model

Nodes and arcs play a central role in unified data structure, since they are the only geometric primitives which have position information directly associated with them. The terminal points of an arc share common "coordinate" information by pointers to their topological nodes (see figure 5). Line features are not defined in terms of geographical coordinates by identifiers to the arcs composing the line, and surface features are defined by identifiers to the arcs surrounding the surface. Actually, this topological data structure is implicitly represented by sharing the common nodes and arcs. On the other hand, that is like the aggregation implement of an object-oriented data model.

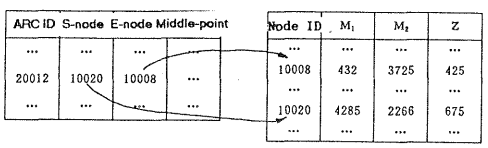

Figure 5 Propagation of a component object

The concepts of object-orientation can be employed to build the geometric data models. In geometry, there are only four types of basic features and several primitive elements such as node, arc and "leaf node" in 2DRE (see Li & Gong, 1992). A feature consists of one or more primitive elements, more than one feature may

compose a more complex object. Some underlying data are propagated from the components to the composite object in terms of the identifiers or the pointers. The data models including the primitive elements, the simple and the complex features are shown in the following diagram.

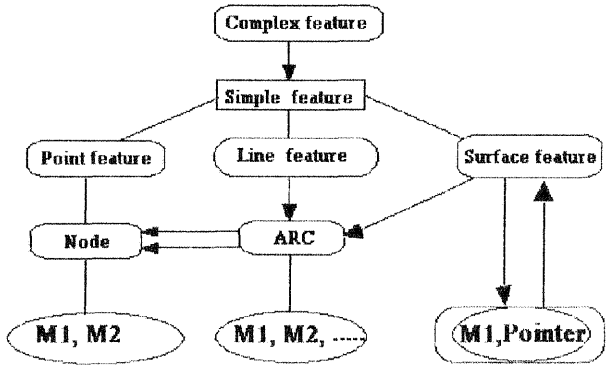

Fig.6 Data model based on unified data structure

The arrows in the diagram express the identifier linking or the pointers in the data files, the ellipses contain the most primitive data in GIS. The diagram shows that the vector structures and the raster structures do not have any obvious distinctions because they use the same primitive data format--Morton key (see Li & Gong, 1992), which can guarantee that the unified data structure serves the two kinds of data and support object-oriented data model.

3.3 Encapsulation Approach

An outstanding characteristic is that the data and operation of an object are encapsulated in an entity, which is helpful for large software design and database protection. It is also required in GIS. Our system is designed with an object-oriented program language C++, which support encapsulation of data and functions, besides the operations for creating, deleting, modifying, displaying, it may contain various spatial queries and analysis operations. If all operations are encapsulated with data, it is not efficient. A new strategy presented here is that the operations acting on geometric data are divided into basic ones such as creating,deleting, modifying, displaying objects, which are encapsulated in a class, and appendant ones including query and analysis, which are set outside the class. Program construction is written by Turbo C++ in figure 7

```
Class: name                 Appendant operations
{                           main()
values of type              {
public:                     common values
basic operations            call basic operations
}                           call appendant operation
                            }
```

(a) Definition of a class construction

(b) main program

Fig.7 program construction and definition of a class

4 OBJECT-ORIENTED MODELS FOR ATTRIBUTE DATA MANAGEMENT

4.1 Relational Approach and Object-oriented Data Model

Relational database management system are often used to manage attribute data in GIS. There are many similarities between the relational approach and the object-oriented model. The relational model has to a certain extent semantic mechanisms of classification, generalization, and aggregation. In a loose sense, a table such as Relation 1 represents a class of parcels. The tuples in this table such as that for parcel I, are objects of this class. Generalization can be used when there is a need to store properties common to several classes. For example, the classes of parcels and highways can be generalized by the superclass of spatial features in Relation 6. An aggregation can model composite objects from their components, Relation 2 implies an aggregation relationship between a parcel and its boundaries. A map composed of different kinds of spatial features is also an aggregation.

However, the fact that they share common behavior can not be deduced from the relational data model, which is lack of inheritance implement. A strategy proposed here is to integrate a object-oriented language such as HyperTalk with a RDBMS like Oracle, which manages attribute data in GIS.

4.2 Object-oriented Programming in HyperTalk

Object-Oriented Programming (OOP) is different from the traditional approach like Pascal, Fortran. In OOP, data and procedures that operate on the data are together, packaged in something called an object. There are five central ideas in OOP: object, message, method, class, and inheritance.

HyperTalk is a programming language for Apple Computers. In HyperTalk, some key ideas are very close to OOP. HyperTalk uses some of the same terminology, adopts some of the same methods and adapts others, and in many ways looks and "feels" like OOP. But it is not a real object-oriented programming language, so it is called Object-Like Programming language.

4.2.1 Object As mentioned above an object is an entity that includes data and procedures that operate on that data. Viewed from a programming standpoint, objects are the elements of an OOP system that send and receive messages. In HyperTalk, there are five types of objects: stacks, backgrounds, cards, buttons and fields, and they can be considered as the operations "icon" or data "container". Like OOP objects, each of these can send and receive messages. Each type of object can also be associated with a script that contains handlers, which correspond to methods (as will be seen in a moment).

4.2.2 Message A message in an OOP world corresponds to a procedure or function called in a procedural language like Pascal. Everything in OOP is accomplished in only one way: one object sends a message to another object and the receiving object reacts. In HyperTalk, the messages can be held by five types of objects. Each type of message can be addressed to one or more of the types of objects encompassed by HyperTalk. The objects In HyperTalk can also receive the messages that are sent by using other language like SQL.

4.2.3 Method A method corresponds to a function or a procedure. A method is the code in an object that tells the object how to react when it receives a message with the same name as the method. In HyperTalk, the handlers correspond closely to OOP method. A Handler is associated with each type of message the object can receive. But a message is different from a method, besides the handler descriptions, the message also includes other scripts, such as card name, card fields, etc.

4.2.4 Class The concept of class is the same as classification in object-oriented data models. The most important thing objects in class have in common is the way they react to one or more messages. When we define a class, each object we create as a member of that class is contained. That means that it is not necessary to send the messages to each single object. In HyperTalk, there is no strong analogue to OOP's concept of classes. For example, there is no class called a button class to which all buttons belong and which has individual instances of buttons. The concept of card backgrounds however, comes close to emulating an OOP class. All cards in a stack with the same backgrounds have many common characteristics.

4.2.5 Inheritance In a true object-oriented world, objects inherit behavior from their ancestors in an ever-expanding and descending chain of heredity. That implies that an object of a subclass reacts to the messages not only from the same class but also from its ancestors. There is also no true inheritance in HyperTalk. An object does not inherit behavior from its ancestors. But like the class in HyperTalk, we can copy new cards with the same backgrounds which can react to the same operations or have the same properties. When we define some backgrounds for a class in a stack, all objects in this class and its subclasses have the same backgrounds. However, this is not inheritance because the new cards of the same backgrounds are not the descendants of the original; both are on the same level of the hierarchy. According to inheritance in object-oriented programming, the operations defined specifically for a subclass are not applicable for objects of the superclasses. But in HyperTalk, if a background is modified or added in the card of the subclass, the cards of its superclasses in the same stack will be affected.

4.3 Object-oriented Models in attribute database

Geographical information systems are employed to store and describe the complex information and phenomena in the real world. The object-oriented data model can serve as a conceptual model. The classification, generalization, association and aggregation are used to define the classes of objects, to describe the relationship of the classes and to compose the complex objects. The object-oriented language such as HyperTalk provides an environment to encapsulate data and operation in a modular, and the mechanism of inheritance and propagation. In technique, it has provided a seamless combination between Oracle and HyperTalk.

4.3.1 Classification in GIS The features in GIS can be classified according to the classification criteria of GIS. Some ideas of the classification in object-oriented models are proposed here. A class corresponds to a table in the relational database and uses a card in HyperTalk. The card includes some operation 'icons' like select, insert, delete, etc., some property 'containers' and some classification 'icons'. A column in a table corresponds to a container in a card. A row in a table represents an object (an instance).The features in GIS can be classified as road, building, park, water-system, vegetation, phone-line and so on. Fig. 8 shows the classifications of the features. Every subclass in the card GIS can be classified further. For example, the class building can be classified into residence, hotel, supermarket, factory, bank and so on (see Fig.9). Sequentially, for instance, the hotel can be decomposed into the classes of employer and room&bed are not the subclasses of the class hotel, and the relationship between the employees or room_bed and the hotel is the part_of relation. So the properties and the operations of the class hotel and class building do not inherit them. However, the class hotel can derive the values from the tables of employees and room&bed. Fig. 10 shows the classifications of the features and their relationship.

4.3.2 Generalization in GIS Several classes of objects with common properties and operations are organized to form a more general superclass. For example, factories, farms, supermarkets, hotels and restaurants have the common properties income and tax. They are formed into a higher level class business, and use a table which has the columns business name, income and tax.

4.3.3 Inheritance and Propagation in GIS The mechanisms of inheritance and propagation are important in GIS. Some simple tables are established in an attribute database. But the user sometimes needs to query the detailed information of an object such as a hotel. A table of the hotel only includes the fields identifier, hotel name, manager name and phone number. However, the user needs its more detailed information like its address, income, employees, beds and so on. Therefore, the user must manipulate several tables. The Oracle system has the function of querying multiple tables. In card hotel, the properties of its superclass building and business are inherited to it, and several fields that represent the properties in the hotel table are built singly. Sequentially, a field employees and two fields rooms and beds are used for propagation which aggregate the values in the table employee and room&bed. The button select holds the handlers of querying multiple tables and adding sum. When the user uses the command select, the select "icon" receives the message and sends it into the handler, and then manipulates the tables building, business, hotel, employee and room&bed. The information is put into the corresponding fields (see Fig. 11).Actually, the query is divided into two steps. The first, querying multiple tables hotel, building and business gets the information of hotel and its superclass building and business. The second, according to the name of the hotel shown in the field, querying the tables employee and room&bed of this hotel, gets the sums of the corresponding columns. If HyperTalk were not used we could not do the second step. Here HyperTalk send a message name, which is obtained from the first step, to the field of hotel name, then query the tables corresponding to this hotel name because each hotel use an employee table and a room_bed table. This is an advantage of HyperTalk, because an object or "icon" can hold several handlers.

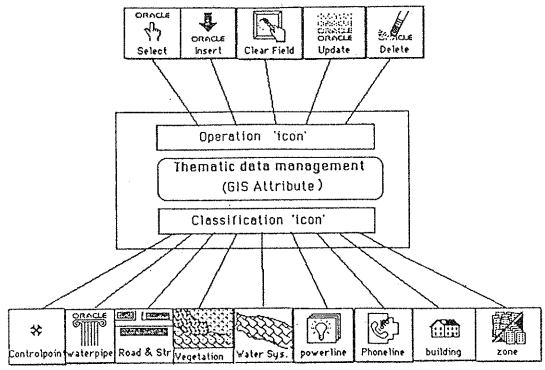

Fig. 8 The classification of features in GIS

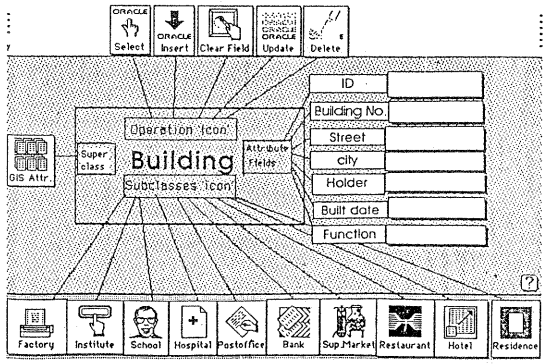

Fig. 9 Building Card

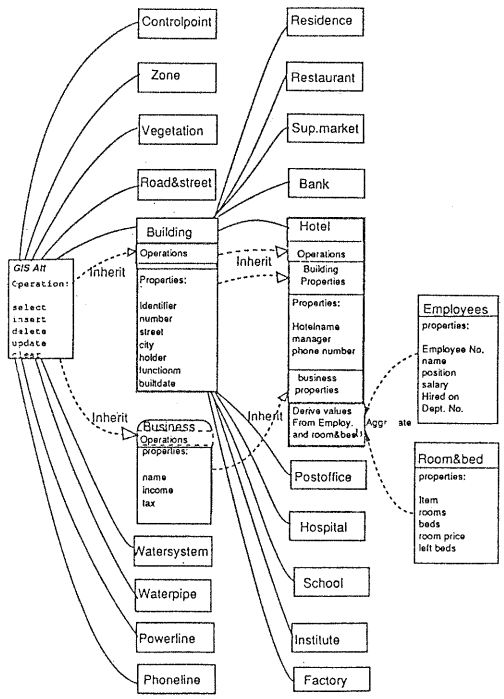

Fig.10 Relationship of various feature classes in GIS

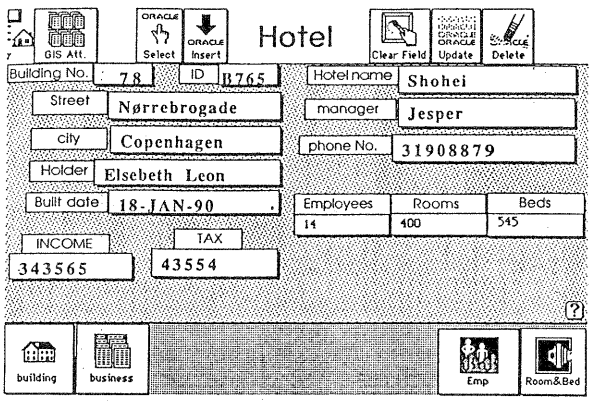

Fig. 11 Hotel Card

4.3.4 Comment The object-oriented model emphasizes the entity concept. A card in HyperTalk is used to represent a complete object including its attributes, operations and the relationship between it and other objects. From user view. the semantics of model is explicit and clear. Here it has been investigated how the Oracle system and HyperTalk programming can serve as suitable tools for object-oriented models in GIS. Though Oracle and HyperTalk are not real object-oriented DBMS the integration of them can realize the object-oriented database management in GIS. Of course, some problems, such as the lack of real class and inheritance in HyperTalk, need to be overcome. Recently, research in non-standard database environments promoted an object-oriented model which looks promising to overcome some problems that make conventional database management systems unsuitable [Frank 1989b]. However, the realization of the real object-oriented database management depends on the powerful database management system and object-oriented programming language. At present, only a few languages support object-orientation sufficiently [Frank 1989], and the OODBMS is being studied.

REFERENCE

[1] Fritsch, D., 1989: "Acquisition, topology and structuring of spatial data" 42 Photogrammetric Week at Stutgart University.

[2] Egenhofer,Max J., Andrew U. Frank, 1989a: "Object-oriented software engineering considerations for future GIS". Second International Geographic Informatio Systems (IGIS) Symposium, Baltimore, MD.

[3] Egenhofer,Max J., Andrew U. Frank, 1989b: "Object-orient modeling in GIS: Inheritance and Propagation". AUTO-CARTO 9, Ninth International Symposium on Computer-Assisted Cartographic, Baltimore, MD.

[4] Egenhofer,Max J., Andrew U. Frand, 1989c: "Inheritance vs. propagation". University of Maine, USA.

[5] Frank,Andrew U., 1989: "Multiple inheritance and genericity for the integration of a database management system in an object-oriented approach". The second International Symposium on Object-oriented database, Bad Stein am Ebernberg, Germany. September 27-29, 1988.

[6] Ehlors,Manfred, Geoffrey Edwards, Yuan Bedard, 1989: "Integration of remote sensing with geographic information system: a necessary evolution". PE&RS No.11.
[7] Dan, Shafer, 1988 "HyperTalk Programming". Indianapolis, Indiana 46268, USA.
[8] Gong Jianya, Li Deren, 1991. "An Integrated Data Structure and object-oriented data model in GIS." Proceedingsi of fig PC'91 Meeting and international Symposium , Beijing, 1991.
.[9] Lee,Y.C., 1990, "A comparison of Relational and Object-oriented Models for Spatial Data." Manuscripts for Tutorial, Comm. III, ISPRS, Wuhan, China.
[10] Sa Saxen, 1989, "The Database Technique faced new applications," computer science, 1989. No.2.
[11] Li Deren & Gong Jianya, 1992, "A Unified Data Structure Based on Linear Quadtree", << Archives of 17th ISPRS Congress, Comm. III>>, Washington.