

# DATABASE MANAGEMENT SYSTEMS FOR SPATIAL DATA STRUCTURES – AN OVERVIEW

AL. M. Imbroane<sup>1</sup>, L. Bucur<sup>2</sup>

## ABSTRACT

*Database Management Systems for Spatial Data Structures - An Overview:* The database is the foundation of Geographical Information System (GIS). One obvious drawback of the standard Database Management System (DBMS) is that they cannot manipulate efficiently geographical data. Different GIS architectures try to solve these problems in their own manners, each with its own advantages and disadvantages. Some of them, use pure relational database, others object-oriented database (OO-DB). For a better perception of spatial data models in relational DBMS (RDBMS) some authors propose some extensions to the relational model to make it more suitable for geographical operations. The extension is a transition between RDBMS and OO-DB. The OO approach is an alternative to the relational model for information system and especially for GIS software. The main objective of this paper is to identify a number of keys issues that need to be addressed in the near term before we can expect to see support of spatial data management in commercial database systems. Another objective is to clarify the definitions used by GIS designer and users, like entity, attribute and object which is used for different meaning.

\*

## 1. GENERAL PROBLEMS.

A GIS is more than just a database model in which different types of data structures must be incorporate. There seems to be a consensus in the literature about the architecture of a GIS. Most of the authors describe the following five main components: data input, storage, analysis, output and the user-interface. The ways these five components are implemented dustings a GIS from other information systems. The fact that GIS have to deal with spatial or locational data, in primary and applied geographic research (Kolejka 2006), separates them from other types of information systems. Because spatial data are the main component of a database they are also called spatial information systems. The CAD/CAM systems deal also with spatial data, but there is a big difference between them. In GIS spatial and alphanumeric (aspatial) data are linked so that a processing in one component is reflected in the other. GIS also support digital images, which can be integrated together with other data.

Developing of any spatial information system, begin with *analysis*, when the issue of what the system is required to do is clarified. It follows *design*, when the problem of how the system will satisfy its requirements is tackled. The translation of the design into something that relay works is the *implementation* stage, which may be the customisation of a proprietary system. At the end are usage and *maintenance*. This sequencing of activities in system development is not meant to suggest a strict temporal dependency between stages. The analysis phase can be decomposed into two stages. First, phenomena in the application domain are represent as processes in an appropriate abstract application domain model; then this model is transformed into the conceptual computational model that is computationally tractable but independent of any specific computational paradigm. The design phase moves from the specification provided within the conceptual computational

---

<sup>1</sup> Babeş-Bolyai, University, Faculty of Geographie, 400006 Cluj-Napoca, România.

<sup>2</sup> Universitatea Oradea, Facultatea de Istorie-Geografie, 3700 Oradea, România

model to the design of the logical computational model, embedded in a particular computational paradigm, either relational database, object-oriented, or others. The physical computational model corresponds to implementation and is the representation of the real world phenomena as computational process.

## 2. DATA IN GIS.

Geographic databases contain collections of spatial data representing the variety of views for the real world at a specific time. The term *spatial* refers to the location of objects positioned in geographic space. *Spatial objects* are representations of the elements of the real world such as rivers, countries, railways, and buildings. The diversity of data that are collected over the same area, often from different sources, imposes a question of how to integrate and to keep them consistent in order to provide correct answers for spatial queries.

The geographic entities or objects in GIS are based on two different types of data: *spatial* and *attribute*. Spatial data have two formats: *raster* and *vector*. The vector format can be represented in two components: *geometric (graphic)* data and *topological* data. Geometric data have a quantitative nature and are used to represent coordinates, lines, areas etc. The two-dimensional vector format has three subtypes: *point*, *line* and *polygon*, named *features*. These features are called also 'graphics primitives' (definition borrowed from CAD/CAM software). The geometric or non-topological data model is those in which any positional information is recorded. The recognition of individual spatial units as separate unconnected elements characterised the early data organisation for digital cartography. Topological data describe the relationship between the geometric data. Topological relationship is invariant under topological transformation like translation, scaling and rotation. These are several types of topological relationship: *connectivity*, *adjacency*, and *inclusion*. Examples of queries concerning topological relationship are: which area is neighbor of each other (adjacency), which line are connected and form a network or a road (connectivity), and which lakes lay in certain country (inclusion). Topological data are not always stored explicitly, because in principle they can be derived from geometric data. Note that this definition of topological data is slightly different from the stricter one used in mathematics.

Attributes or descriptive, or aspatial data are alphanumeric data related to the graphic entities. They are also called thematic data because they contain themselves a layer of the graphic features. A unique identifier links them. This operation is called *geocoding* or *address matching*. The fact is that attributes data can be linked with other tabular data (external) and so the new data can be processed together with the others. This is one of the powerful specific operations in GIS. Spatial data and attribute data can be stored separately or together depending on the database design performed by the GIS proprietary. A geographical database is a set of geometric entities (spatial features) and attributes. The problem arises: how is better to organise these kinds of data?

The GIS database can be thought of as a representation or model of real-world geographical systems. To this end it is useful to make a distinction between *geographical entities* and *spatial entity*. A geographical entity is a term used with respect to an element of real-world system; that is entities are contained in geographical space. A *spatial entity* is a GIS representation of geographical entity. For example, geographical entities such as town, highway and county are represented in GIS database in form of point, line, and area (polygon). Some authors (Malczewski, 1999) use the *object* instead *spatial entity (spatial features)*. The problem is that the *object* is a consecrated thing used in computer

programming (the object-oriented programming) which has different meanings. It is better, to avoid confusions, the word object to be use just when we are talking in programming language terms.

The data stored may be in the form of one or more files, or in the form of database. The difference between a files and a database is semantic and varies somewhat from one discipline to another. For GIS purpose, a file is regarded as a single collection of information that can be stored, whilst a collection of files is regarded as a library. A database comprises one or more files structured in a particular way by a DBMS and accessed through it. Today almost all GIS data is stored in databases. The various methods of data storage differ primarily according to the facilities or operation supported by the DBMS. GIS Databases are currently investigated in Romania by many authors (Haidu,C. 2006).

### 3. DATABASE CONSTRUCTION.

Data may be stored in separate unrelated tables, in a single table, in connected tables or as list, sets or collections. A *data set* is a single collection of information, without any particular requirement as to form of organisation. A database is a collection of non-redundant data sharable among different applicants. Within database, data are organised in files, which contain particular records of data, the rows of data, and the record has data for a particular place, event or entity (Laurini and Thomson 1992). The database approach to storing geographical data offers a number of advantages over traditional file-based datasets (Date, 1995): collecting all data at a single location reduces redundancy and duplication; maintenance costs decrease because of better organization and decreased data duplication; applications become data independent so that multiple applications are used the same data and they can evolve separately over the time; user knowledge can be transferred between applications more easy because the database remains constant; data sharing is facilitated and a corporate view of data can be provided to all managers and users; security and standards for data access can be established and enforced.

Large and complex databases require specialised database management systems (DBMS) software to ensure database integrity and longevity. A DBMS is a software application designed to organize the efficient and effective storage and access of data. To carry on these functions DBMS provide a number of important capabilities (Longley et. al., 2001): data model, data load, index, query language, security, controlled update, backup and recovery, database administration, applications and programmable API.

A DBMS must be customised to meet the needs of particular applications. In order to do this we must have a precise idea of the way that information is structured in the system and the kinds of algorithms that will act upon the data. There are three main stages in the process of customising a database system for a particular application.

1. **Construction of the conceptual model (database analysis).**
2. **Translation of the conceptual model into the logical model (database design).**
3. **Translation of the design into a system that works on a particular DBMS and platform or internal model (database implementation).**

The conceptual model corresponds to a synthesis of all external models. It is called this for two reasons (Laurini and Thomson 1992): first because it is made of very sound concepts, secondly because it is the basis for the conception process. Although an abstraction of the real world, the result of the conceptual modelling is supposedly quite concrete in nature, consisting of schematic representations phenomena and how they are

related. The organisation scheme created at this stage generally deals with only the information content of the database, not the physical storage, so that the same conceptual model may be appropriate for diverse physical implementations. Many difficulties arise in designing the conceptual model as a synthesis of several external models, especially in dealing with the geometric and topological data.

The next step is called database design, in which the conceptual model is converted in the logical models. In fact it is the first step in computing. The expression logical model has two meanings. First it constitutes a mathematical bases or a set of mathematical concepts. Secondly, it corresponds to the transformation (mapping) of the conceptual model with the tools offered by the logical modelling. In other words, we have a transfer between the conceptual models and a new modelling level, which is more computing, oriented.

The realisation of conceptual model is achieved by mapping the conditions of the semantic data model into the definitions, constraints and procedures for one of these models, most especially today, the relational or other new types, extension of relational or object oriented. In this way the permanent properties of the database are clearly specified, whatever the circumstances pertaining to a particular set of instances of phenomena. This procedure is associated, in practical sense, in the creation of data dictionary, a set of statement about important properties of the data items, such name, type of data, range of values or missing values.

#### **4. FIRST GENERATION OF DATABASE SYSTEM: HIERARCHICAL AND NETWORK DATABASE SYSTEM.**

A hierarchical database system support a hierarchical structure of record organise in files at various logical levels with connections between the levels. Each record contains a field defined as the key field, which organise the hierarchy. There are no connections between records of the same level. Hierarchical database systems are easily expanded and updated. However, they require large index files, must be frequently maintained and are susceptible to multiple entities. Searches are rapid, but search routines are fixed and constrained by the structures. Thus, records at the same level cannot be searched, nor can new links or new search routines be defined. The elements or the structure are related only through one-to-many connections. This constraint imposes the presupposition that all quarries are known in advance and accounted for structuring and entering data. This constraint is not always natural or suitable for GIS applications. As a result, hierarchical database structures are usually restricted to storing digital map data in GIS (Bernhardsen, 1992).

On the other hand, raster data may be stored in compressed form using run-length encoding. Raster data may be even more efficiently stored in quad tree hierarchical structure. Quad tree is a generic name for several kinds of index that are building by recursive division of space into quadrants. The quad tree paradigm divides the area into square cells of size varying from relatively large down to that of smallest cell of the raster. Usually, the squares are successively quartered into four smaller squares. The quartering may be continued until a square is found to be homogeneous, so that it no longer needs to be divided, and the data on it can be stored as a unit. The quad trees can use also to store lines and polygons. Each object is rasterised and enclosed by a square. The square is then progressively subdivided into blocks of four squares of equal size until the squares are completely inside or outside the object or the maximum depth of the quad tree is reached.

This structure requires two tables: one for attribute data for homogeneous areas and one for pointers that locate attributes.

In a network database each element, or a collection of like records, has connections to several different-level elements. The term essentially describes the logical and physical view of the data as a group of records with links between records. Interconnections are made in hierarchical organisation, and a characteristic may be associated with two main objects. The network structure is more closely represent the complex interrelationship, which often exists between real-world geographical objects. It improves the flexibility and reduces the multiple entities of hierarchical structure. The elements of network may be related through one-to-many, many-to-one, and many-to-many connections (Bernhardsen, 1992). The network structures better suited to geographical data than the hierarchical structure, it is frequently used in GIS applications. It is more suitable for utility projects (AM/FM).

## 5. THE RELATIONAL DATA MODEL

RDBMS is a database system made up of files with data elements in two-dimensional array (rows and columns). This DBMS has the capability to recombine data elements from different relations resulting in a great flexibility of data usage. The relational model contains the following components: collection of objects or relations set of operations to act on the relations and data integrity for accuracy and consistency.

As their name implies, relations play a major impact in relational database. A formal definition of relation is (Codd, 1970): Given a collection of sets  $D_1, D_2, \dots, D_n$ ,  $R$  is a relation on these sets if it is a set of ordered  $n$ -tuples  $\{d_1, d_2, \dots, d_n\}$  with  $d_i \in D_i$ . The sets  $D_i$  ( $1 < i < n$ ) are called domains of  $R$  and  $n$  is the degree of  $R$ . The element  $d_i$  are the attributes of the relation.

A relation consists of a number of tuples or records that can be described by two-dimensional table. Each row of this table contains a tuple. The columns each describe a certain attribute. Within a relation at least one attribute must have unique value for each tuple. This attribute is called the *primary key*. It identifies the tuple and other attribute that is dependent on it and may not have an empty value. The normal forms of relations are important during the design of relational database applications. They help to remove redundancy, so that it is easier to keep the database consistent if changes are made. For the attribute data relational database is an ideal environment. The problems appear at spatial features and their relationship.

### 5.1. A pure relational solution. Definition of data

We examine the possibilities to storage geographical data in traditional RDBMS. We consider only static geographical data, a simplified version of the data structure described by Smith (1985). This data deals with point, node, chains and polygons. The following relations are defined (bold denote a primary key and *italic* denote foreign key):

- Point (**pntid**,  $x$ ,  $y$ )
- Node (**nodeid**, *pntid*)
- Chainpoints (**chainid**, **pntseq**, *pntid*)
- Chaintopol (**chainid**, *strnode*, *endnode*, *lftpol*, *rgtpol*)
- Polygon (**polid**, *chnseq*, *chainid*)

An isolate point is determined by an unique, called point identifier, and two numbers, which denote the bidimensional co-ordinate ( $x, y$ ). A node also contains a node

identifier (primary key) and a co-ordinate pair. For this reason we have two possibilities: to storage the co-ordinate like points or to make a relationship to the point, using foreign key *ptid*. The second one is much better for the integrity of database. *Pntid* is a primary key for point feature and foreign key for node feature. The chain concept in topological data model is captured in two relations: chainpoints and chaintopol. The first of these describes the shape of chain. The second relation captures the topological aspect of a chain. Chainpoints denotes an arc feature, which is a chain of linked segments. It also has a chain identifier (chainid) and a sequence of points, both primary keys. Because point contain an identifier *ptid* must be include as foreign key. The next entry, chaintopol, contain itself the topology. These kinds of chains delineate polygons. Beside its own identifier (chainid), it contains start node, and end node, left polygon and right polygon. All are foreign keys. The polygon feature is defined by it own identifier, a sequence of chains, as primary keys, which enclose polygon and subsequently chain identifiers as foreign keys. We make a notice here. Chainpoints is better to contain start node and arc node for facilities in network operations. Also a polygon is determined by an isolate point, which lies inside, called centroid, as primary key.

## 5.2. The entity-relationship model.

One of most compelling and widely uses approach to forming a conceptual model is known as the entity-relationship (ER) model. An entity type is an abstraction that represents a class of similar objects, about which the system is going to contain information. A relationship type connects one or more entity types.

The entity type can have attributes that serve to describe them. As we mentioned, the spatial database and attribute database can be storage separate or together. The first case is just presented. For the other alternative, the structure can be

- Point (**ptid**, x, y, pattr)
- Node (**nodeid**, *ptid*, nattr)
- Chainpoints (**chainid**, **ptseq**, *ptid*, chainattr)
- Chaintopol (**chainid**, *strnode*, *endnode*, *lftpol*, *rgtpol*, chainattr)
- Polygon (**polid**, **chnseq**, *chainid*, polattr)

Where pattr is a field that contain the description of that point (tower, well and so on). This field can be used for matching large RDB stored separately. Otherwise it must be used the first primary key to that link. Same for the others attributes.

The relationship may have their own attribute, which is independent of any of the attribute of the participant entities. The ER model; allow the expression of a limited range of integrity constraints. Relationship type is subdivided into many-to-many and one-to-many relationship.

The advantage of RDBMS: rigorous design methodology; all other database structures can reduced to a set of relational tables; easy to use compared to other databases systems; modifiable (new tables and rows can be added easily); very flexible and powerful; fast processing.

A reason for adopting RDBMS in geographical data is the simplicity of the structure. Beside it is well known by large group of user. In every geographical database then also has to be a way to store the thematic (alphanumeric) data. This is another reason for designers of commercial GIS. Because of the problems with the spatial data, their database is often split into two parts: a relational part for attributes and a special part for the

geometric data. Then two parts are linked by corresponding object-ids of the attribute and geometric parts: the dual architecture.

## 6. EXTENDED RELATIONAL DATABASE.

The extended data model approach is also described as spatial database management system approach, with the GIS serving as the query processor sitting on top of the database itself. Most implementation to date are of the vector-topological kind, with relational tables holding map co-ordinate data for point/nodes and line segments, together with other table containing topological information. Attributes may be stored in the same tables as the map feature database or in separate accessible via relational joins.

### 6.1. The extend entity-relationship model (EER).

The *extended* or *enhanced* entity-relationship (EER) model has features additional to than provided by ER model. These include the construct of subclass, superclass and category, which are closely related to *generalisation* and *specialisation* and the mechanism of *attribute inheritance*. This allows the expression of more meaning in the description of the database and leads toward object-oriented modelling.

An entity type E1 is a subtype of entity type E2 if every occurrence of E1 is also an occurrence of type E2. The operation of forming subtypes from types is called *specialisation*, while the converse operation of forming supertypes from types is called *generalisation*. Specialisation and generalisation are inverse to each other.

Specialisation may summarised as the creation of new types such that: each occurrence of the subtype is an occurrence of supertype; the subtype has the same identifying attributes as the supertype; the subtype has all the attribute of the supertype and possible some more; the subtype enters into all relationship in which the supertype is involved and possible some more.

The entire collection of subtype-supertype relationship in a particular EER model is called its *hierarchy*.

Generalisation is the reverse modelling process to specialisation. A technical problem with generalisation occurs when we wish to generalise from heterogeneous entity type. The difficulty arises in finding a set of identifying attributes for the generalised type; because the subtypes may all have different identifiers. Consideration of this leads to the concept of *categorisation*, where several heterogeneous entities (with their own attributes and relationship) combine into a common category (Elmasri, Narathe, 1994). Categorisation actually adds considerable modelling power, because it provides the ability to represent the inner complexity of entities no longer be treated as indecomposable.

## 7. OBJECT-ORIENTED APPROACH.

The Object-Oriented (OO) approach, have been promoted especially in areas for which the application of pure relational technology has been found problematic. Examples of such area are CAD and GIS. Each area is characterised by one or more non-classical features, such as structurally complex information, specialised graphical requirements and non-standard transactions. OO data models and systems embody reach data structures and semantics in the construction of complex databases, such as complex data objects, class/subclass hierarchies, class composition hierarchies, property inheritance, methods of active data etc. This not only brings the power and flexibility to the system but also adds complexity to implementations, including the development of knowledge discovery mechanism (Mango M., Kodratoff Y., 1991). The main advantage of the OO paradigm is it's easy of understanding; it enables natural representation of real world objects, their

mutual relationship and behavior and is therefore close to end-user. An OO application consists of a set of objects with their own private state, interacting between themselves. The OO approach can be useful during several phases of the development of a GIS: information and requirement analysis, system design and implementation. The interest in OO GISs appears growing (Clementini et. al, 1990; Egenhofer, Frank 1989b).

OO design is a method design encompassing the process of OO decomposition and a notation for depicting both logical (class and object structure) and physical (module and process architecture) as well as static and dynamic models of the system under design (Booch, 1994). Object-Oriented database system (OODBS), has developed capabilities such as persistence, long transactions and versioning, unlike most traditional RDBMS. Through combining database functionality with OO programming, OODBS has become an expressive device for multimedia applications, client-server systems as well GIS and CAD (Wachowicz, 1999).

The OO database on the market have in common basic characteristics such as methods associated with objects, inheritance of attributes and procedures from supertype (superclass), and the ability to define the type (class) of objects, their attribute type and relationships. However they differ substantially in query language. The difference result from the fact that OODBS have been elaborated based on programming languages for their data models. The lack of a standard or a format background for OO query has caused differences in query language syntax, completeness, SQL compatibility and treatment of encapsulation (Cattell, 1991).

### 7.1 Foundation of OO approach.

As is state in ANSI Object Oriented Database Task Group Final Technical Report (1991), an object is something 'which plays role with respect to a request for an operation. The request invokes the operation that defines some service to be performed'. A *request* is a communication from an object to another and, at the system level; it is implemented as a message between objects. Exactly how object will respond to a message in any particular case is dependent on its *state*. Its state is constituted by the totality of attribute values for given object at any time. In ER model the static aspect of an object is expressed by a collection of attributes. For instance a city object might have *name*, *centre*, *population*, *area* among attributes. Objects are similar to the concept of entity.

The totally responses to messages is its *behavior*. In other words behavior consists of a collection of operations that can be performed on objects in the object type. Operations define the behavior of the objects. Objects with similar behaviors are organised into *types*. At the design and implementation stages we often talk about *object classes*, which are groupings of objects with corresponding *data structures* and *methods*. Each operation will be implement as a method, which acts on members of implemented object classes and returns a number of an object class. A class therefore represents a generalization of a set of objects with common properties and behavior. A class can be thought of as a template for objects. When creating an object data model the data model designer specifies classes and the relationship between classes. Objects are *instantiated* (generated) from this description. Thus, an object class is an implementation construct while an object type is a semantic notion.

There are three key facets of object data that make them especially good for modelling geographic systems: *encapsulation*, *inheritance*, *polymorphism* and *behavior*.

## 7.2. OO system modelling

One of the great advantages of object modelling over entity modelling is that the objects in the modelling process can be the natural objects that we observe in the real world. In a geographical object data model, the real world is modelled as a collection of objects and relationships between objects. Each entity in the real world to be included in the GIS is an object.

An object type is declared by given a name and an optional list of parameters. The optionally follows details of inheritance, state, behaviour and integrity constraints. The status of an object type consists of a collection of attributes and describes the static aspects of an entity in ER model. However, each occurrence of an attribute is itself an object. The details of the implementation of each operation are not the concern of conceptual data model. Some general groups of operations are: *constructors-destructors*, *accessors*, *transformer*, *object composition*, *aggregation and association*.

Topologic relationships are generally built into class definition. For example, modelling real world entities as a network class will cause network topology to be built for the nodes and lines participating in the network. Similarly, real-world entities model, as topologic polygon classes will be structured using the nod-line model.

Arc/Info 8.x provides us an integrated data storage plan. It uses an Object-Oriented data model to store an entity's shape data and attribute data together. All the concerned data of an entity form a record in the geodatabase, while entities that have the same characteristics will form a table in geodatabase, which is called a feature class. Under such a model, the relationship between geometric data and attribute data of an entity has been greatly enhanced; they provide us much more complete information. But unfortunately, this object-oriented data model has not taken temporal information into account. It remains a snap-shot data model and thereafter it lacks the ability to track the evolution of an entity or a phenomenon. It is just an object-oriented spatial data model but not a spatiotemporal one. However, it is still a good example which we can learn from while trying to develop a spatiotemporal data model. At least, it gives us the following principles in developing a spatiotemporal data model: Object-oriented ideas and technologies should be introduced.

## 8. OBJECT-RELATIONAL DBMS.

In spite of technical elegance of OODBMS, they have not proven to be as commercially successful as some predicted. This is because the fact that RDBMS vendors have added many of the OODBMS capabilities to their standard RDBMS software to create hybrid object-relational DBMS (ORDBMS). An ORDBMS can be thought of as an RDBMS engine adapted to handle objects (Longley et. al., 2001). That is both the data describing what an object is and the behavior that determines what an object does are stored together as an integrated whole. Although there are differences in the technology, scope and performance of these systems, they all provide basic capabilities to store manage and query geographical objects. It is important to realise that none of this is a complete GIS software system itself. They have no real capabilities for spatial editing, mapping and analysis. Spatial analysis is in many ways the core of GIS, because it includes all the transformations, manipulations and methods that can be applied to geographical data to add

value to them. Effective spatial analysis requires an intelligent user, not just a powerful computer.

An ideal geographic ORDBMS is one that has been extended to support geographic object types and functions in several ways (Longley et. al., 2001): query parser, query optimiser, query language, indexing services, storage management, transaction services and replication.

## 9. DEDUCTIVE DATABASE.

Beside the relational and OO database, there are other paradigms that offer some advantage. One of them is the *deductive database*. A deductive database (DDB) provides persistence for computation using logic-based language such as Prolog. Logic programming systems consists of two parts: a logic program represents declarative knowledge and an inference mechanism, used to answer queries on basis of this knowledge. The relational model can be viewed as allowing the expression of a set of facts. DDB allows the storage of not just facts but more general propositions. Storage of propositions may avoid the storage of much data. DDB also support deduction with their data. The use of logic provides a natural, intuitive method of generating precise definitions of parametric shapes and high-level spatial relations (Worboys, 1995).

The use of logic in design is not new. Some examples include its use in shape grammar and reasoning systems (Chase, 1989; Damski, Gero, 1996; Heisserman, Krishnamurti, 1992a; 1992b; Krishnamurti, Giraud, 1986). The weaknesses of the shape grammar implementations are in their limitations dues to computational problems; those of other logic-based systems are in their representations of design objects, which—with few exceptions—cannot support emergent features. The shape algebra formalism is extended by using logic to make more precise, generalized, parametric definitions of shape and spatial relations than has been previously possible. The value of such a model and the advantages of the representations used over more traditional ‘kit-of-parts’ models can be demonstrated by the use of these generalized spatial relations for solving typical problems involving spatial reasoning.

DDB can be through of a generalising the concept of a relational database. They comprise an *extension* (similar to a classical relational database) and an *intension*. Intension consists of virtual relations that are defined in term of other relations using logic. The important point is that we have a new relation that is defined, not by specifying tuples, but as a logical combination of other relations: hence the name ‘virtual relation’. Thus DDB allow the manipulation of data that are implied by explicitly stored facts, allowing the measuring of the data to be considered in ways that cannot be accommodated by conventional database. This ability is potentially relevant to GIS, in which much information is stored in implicit form. The use of logic provides a natural, intuitive method of generating precise definitions of parametric shapes and high-level spatial relations. Its use as a specification and programming tool has become widespread over the past two decades, providing advantages over traditional procedural programming methods, among those the ability to specify the knowledge to be encapsulated in a model (description) without the need to specify data manipulation procedures (prescription) (Kowalski, 1979). The use of logic can facilitate a top-down method of development, from the abstract to the specific. The symbolic abstractions of logic formulations enable one to denote entire classes of data structures and procedures while ignoring their details. This can be a more natural method of development than having to deal with often no intuitive formulations.

Using deduction carries out computing a query in a DDB. A conventional database consist of a body of data in some highly structured format (tabular in the case of relational database) defined and accessed by means of a database language. Most GIS store their data in this way, either in tabular form in a relational database or in some more specialised database structures. In contrast, a DDB aims to use logic as the basis for database definition, query and manipulation. It handles a body of expressions in a logic-based language that allow the expression of propositions, description of their inter-relationship and management of their storage. It both defines the structure of the data and acts as query language and control mechanism. The database operates by applying deductions to the logical expressions to evaluate their truth or to find values that satisfy them.

In designing the DDB must choose which of the wide variety of logics to implement. The system of logic must be sufficiently expressive, yet not slow the database down with its computational demands, and it should allow a wide variety of possible models. Data should be stored in a format that both flexible and compact. Geographical information has spatial component and so appropriately extended logic may be applied. For GIS it is necessarily not just deductive, rule-based tabular data but rule-based complex object, a system that takes the best of object orientation, databases and deduction adds support for graphical interactive user interface. DDB technology holds out hopes of being able to address and solve these complex problems. Object orientation and logic are not mutually exclusive, and these have been some attempts (Abdelmoty et al., 1993b) to unify them in the context of GIS.

## REFERENCES

1. Abdelmoty, A.I., Williams, M.H., Paton, N.W. (1993a), *Deduction and Deductive database for geographical information handling*, in Ale D., and B.C. Oob (eds), *Advances in Spatial Database*, Proc of SSD '93. Singapore. Lecture Notes in Computing Science No. 692, Berlin: Springer-Verlag, 443-464.
2. Abdelmoty, A.I., Williams, M.H., Quinn J.M.P. (1993b), *A rule-based approach to computerised map modelling*, *Information and Software Technology*, 35 (10), 587-602.
3. Bernhardsen, T. (1992), *Geographic Information Systems*, Viak IT, Arendal Norway.
4. Booch, J.R. (1994), *Object Oriented Design with Applications*, Benjamin & Cumings, Santa Clara, CA.
5. Cattell, A.M. (1991), *Object Data Management. Object-Oriented and External Relational Database Systems*, Addison-Wesley, Reading, MA, USA.
6. Clementini E., Felice P.D. (1990), *An Extensible Class Library for Geographical Applications*, in TOOLS'90, Edition Angkor, Paris, 619-623.
7. Chase S C. (1989), *Shapes and Shape Grammars: From Mathematical Model to Computer Implementation*, *Environment and Planning B: Planning and Design* **16:2**, 215-242.
8. Clementini E., D'Atri, A., Felice P.D. (1990), *Browsing in Geographic Database. An Object Oriented Approach*, in IEEE Workshop on Visual Language, Skoie, Illinois (Los Alamitos, CA, Computer Society Press), 125-131.
9. Codd, E.F. (1970), *A Relational Model of Data for Large Shared Data Banks*, *Communication of ACM*, **13** (6), 377-387.
10. Damski J C, Gero J S. (1996), *A logic-based framework for shape representation*, *Computer-Aided Design* **28:3**, 169-181.
11. Date C.J. (1995), *Introduction to Database Systems* (6<sup>th</sup> ed) Reading, Mass: Addison-Wesley