Requirements for a Database Management System for a GIS

Andrew U. Frank
Department of Surveying Engineering, University of Maine, Orono, ME 04469

ABSTRACT: In geographic information systems (GIS) large amounts of data are stored and must be made available to multiple users. Database management systems (DBMS) were designed to facilitate storage and retrieval of large data collections. They include facilities to protect and secure data, enforce consistency of the data stored, and make data available to multiple users at the same time. These services are necessary for GIS, and GIS should therefore be built using database management systems. However, geographic information systems demand high performance and pose some very special requirements for database management. DBMS designed for commercial usage are not well suited for GIS because they cannot accommodate spatial data and cope with retrieval of map graphics. An overview of the architecture of a DBMS especially suited for spatial data handling is presented. For each layer, specific techniques, e.g., for buffer management, clustering of data, and spatial access, that are useful for GIS DBMS are indicated. Efforts to implement the PANDA DBMS are described.

INTRODUCTION

GEOGRAPHIC INFORMATION SYSTEMS (GIS) must store large amounts of data and make them available on demand. Users have learned from their personal computer experience to demand nearly instantaneous responses even for relatively complex requests. Traditional solutions in which data are stored on disk or on magnetic tape and must be searched sequentially cannot respond fast enough to user queries and are no longer sufficient to accommodate frequent changes in the users needs.

A modern GIS is expected to be able to integrate data for different topics and from different sources. The integration of multiple data sets, often visualized as multiple data layers, is expected to produce a synergistic effect and yield better information for decision making. Traditional file oriented storage cannot easily respond to this requirement either.

Geographic information systems are comprised of a complex of several parts that interact. In order to build computerized GIS, we have to deal with organizational, software, and hardware problems.

It must be noted that **organizing** the cooperation of different groups to collect data and to share the results is an especially difficult task, for which few guidelines and rules are available. Many projects fail not for technical reasons, but for lack of organizational arrangements or because of a poor understanding of social or economic implications.

Hardware problems are more easily resolved—the components for storage and processing of very large amounts of data are available from various manufacturers. Prices are increasingly reasonable and the general trend is toward "zero cost hardware" (Dangermond and Morehouse, 1987). GIS **software**, on the other hand, is much more difficult to build than many had previously thought. The software system to manage GIS data must contain a module that provides database management system functionality. This paper deals primarily with this software component and the requirements placed on it by GIS applications.

Database management systems (DBMS) are appropriate tools for GIS. Fast access to spatial data out of a large data collection is difficult to achieve. Many current GIS store data as a collection of map sheets (or similar spatial partitions) which are then handled as units. This requires all users to understand their structure and hinders access by postal addresses or other logical concepts, for example. To achieve the desired "seamless" database where objects (i.e., map features) are not arbitrarily divided by map boundaries and where users can freely move or zoom over the map, special methods and optimizations are necessary.

DBMS software provides the services needed to integrate and protect the data. But, the conventional DBMS does not deliver the performance and cannot retrieve map data quickly enough for interactive work. Not all GIS software packages currently on the market contain a DBMS or include all the services necessary for data protection.

In this paper, we detail these necessary DBMS services and show in an architectural overview how they interact. We use modern software engineering concepts to organize the discussion. Particular attention is given to the integration of database management systems with other software specifically written for spatial data processing. Emphasis is placed on data storage and retrieval functions, including the protection of the data in a GIS. Equally important problems of adequate modeling of reality and the data model support necessary for GIS are excluded and treated elsewhere in order to conserve space (Egenhofer and Frank, 1988a). The discussion of access methods and, especially, query languages is therefore intentionally limited.

Many of the ideas reported here are based on experience with the PANDA database management system (Frank, 1982a, 1984b, 1986a; Egenhofer and Frank, 1987a). We identify methods successfully implemented, and include a critique of methods which have not worked as well and will be replaced in the future.

SPATIAL INFORMATION SYSTEMS

The use of computers for "batch" processing, where all the input data are collected and an output with the result is delivered later, has been largely replaced by interactive information systems, where the system maintains a collection of data which is then interrogated by users as they need the information.

In general terms, an information system contains an image or **model** of reality, which we can use to make decisions and need not re-investigate the facts each time. This is extremely important in all situations where data collection is expensive, cumbersome, or slow, and is one of the major forces behind GIS: substantial savings by sharing the cost of data collection and at the same time improved usage of the data and higher quality information output is expected (National Academy of Science, 1980).

Geographic information systems deal with data related to location in real world space—here referred to as **spatial data**. Many operations of government at all levels, as well as planning and research, exploit data which have a spatial component.

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 54, No. 11, November 1988, pp. 1557–1564.

0099-1112/88/5411–1557$02.25/0

Such systems are referred to by various other names (e.g., land information system, AM/FM, multi-purpose cadastre). We will concentrate on general aspects of systems dealing with spatial data referred to as "**spatial information systems**", without consideration of differences between systems designed for specific tasks.

We will concentrate on systems which store data with an exact reference to location and which describe geometry using points and vectors. This is not to exclude systems of other types—there are obvious advantages in the use of raster operations for certain tasks, but they seem to have substantially different requirements for data storage and warrant a separate discussion.

A GIS is a **model** of reality and not just a repository of cartographic data necessary to draw maps (Frank, 1984a). Methods to represent complex aspects of reality in a computer system therefore become important. Only if the structure of reality is appropriately modeled in the data stored can we expect that the combination of multiple data sources and the extraction of complex information will produce results that are meaningful. In such situations we encounter relations between the data elements, e.g., a building is at the same time related to a lot on which it is built, to a street it is on, and to persons who are living in it. A method to store and retrieve the data, using and preserving these multiple relations, is necessary.

DATABASE MANAGEMENT SYSTEMS

Data collected in a database are valuable because much effort is necessary to collect and enter the data into the system and to keep the data up-to-date. Data stored must be available for a long period of time to justify expenses of data entry. New, unforeseen changes will likely occur in applications during the lifetime of the data. File oriented programs have a tendency to require changes in all programs that access a file if a change in this file becomes necessary. Database management systems separate the processing of the data from their storage, and confine changes to the directly affected programs.

Making the same data available for many applications and integrating data from different sources is difficult in a file oriented system because it creates more dependencies between the programs and the file and thus makes adapting programs to the changing requirements more expensive. Under these circumstances, the traditional simple file structure designed to facilitate a special application program is no longer adequate.

A database management system should provide the following functionality:

- Storage and retrieval of data; selection of data based on a multitude of access keys (e.g., name of a person, street address of a building);
- Standardized access to data and separation of data storage and retrieval functions from the programs using the data (this makes database and application programs independent, so that changes in one do not necessarily lead to changes in the other);
- Interface between database and application programs based on a logical description of the data (details of the physical storage structure should be transparent to the applications);
- Make access functions in applications independent of the physical storage structure, so adaptations to expanding storage needs do not influence the application programs;
- Allow for access to the data by several users at the same time; and
- Provide for the definition of consistency constraints for the data which will then be automatically enforced. Consistency constraints are rules which must hold for all data stored, and are an excellent technique to reduce the number of errors in a large data collection.

Access to data should be possible both from a high level language and from a user-friendly query language. The level of integration of the database manipulation language with the programming language used influences the effort necessary to develop and change application programs. A free-standing query language is helpful for casual users to retrieve data from the database to answer *ad-hoc* questions without any formal programming. This will make the database usable for one-of-a-kind questions, which are often posed in dealing with abnormal situations or in planning applications.

A database management system is thus a method of encapsulating the valuable data to make it available to a multitude of users while simultaneously protecting the data (Figure 1).

Early in the history of data processing, programmers became aware of the similarity in needs of different applications to store and retrieve data. Instead of rewriting procedures for these functions for each application, an attempt was made to write a generally applicable program to provide these services. The idea of a (generalized) database management system was born (CODASYL, 1962, 1971).

SPATIAL DATABASE MANAGEMENT SYSTEMS

Standard commercial database management systems, as used for keeping personnel or client data, etc., are designed for different usage patterns than found in engineering and scientific applications. Commercial users require the telephone number or the address of a person or transfer some amount of money from one to another account. In a spatial information system users ask for a map-like sketch on the screen showing, e.g., a building with its boundaries, the neighboring buildings, and possibly the utility lines to which it is connected. The entities in a spatial information system are often logically connected to many more other entities than in a commercial system. Additionally entities are related by spatial relations like "neighbors" or "near by" which are not found in commercial applications.

The crucial task in a spatial information system is the retrieval of a set of entities necessary to draw a small map on the screen. After counting entities in such drawings for different applications, we estimate that 2000 to 5000 entities (points, lines, symbols, etc.) must be retrieved from the data collections to produce such a drawing. Screens with substantially less data seem empty and do not convey enough information about an area, whereas screens with more data are too crowded and are difficult to read. In an interactive operation response must be faster than half a minute, otherwise operators start working on other tasks, their concentration is lost, and productivity suffers.

Commercially oriented database management systems are not designed for fast retrieval of so many spatially related entities. Current GIS software is either based on specific file structures, at least for the spatial data, and lacks many of the other benefits of DBMS or it partitions the data in smaller data sets which are then stored as separate databases. This is acceptable in systems for maintenance of maps where updated paper maps are pro-

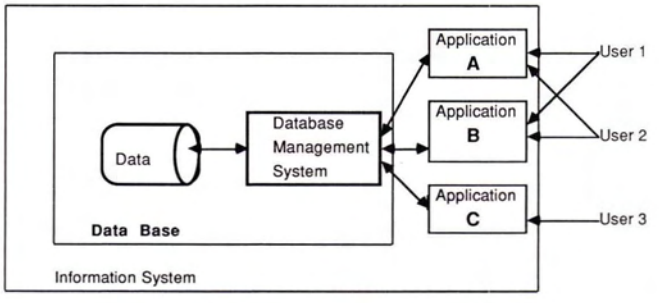

FIG. 1. Database management system.

duced, but is not useful when end users ask questions like "how is building 23 Mill Street connected to the water main." Such users must not be bothered with the limits of map sheets, as this distracts from their primary task, and they must be able to select areas by logical criteria as well as by zooming and roaming graphically in a seamless database.

Database research found that a number of engineering and scientific applications (e.g., Computer Aided Design, medical data processing) all need the basic functionality of DBMS as found in commercial systems, but have some similarity in requirements that make standard DBMS unsuitable (Plouffe *et al.*, 1984). Research in these so called "non-standard" databases started in the early '80s (Frank, 1981; Haerder and Reuter, 1982; Schek and Lum, 1983) and continues to the present (Manola *et al.*, 1987). The next sections will give first a framework for such non-standard DBMS and then mention some specific techniques which have been found useful.

SPATIAL DBMS

We propose a layered architecture for a spatial DBMS. Its architecture is composed of a hierarchy of modules, each providing certain types of services or functions to the next layer above. The lowest layer* is directly related to the services provided by the operating system, whereas the top layer provides services to the GIS user (Figure 2).

Layer 1 **stores data,** using the operating system to access the file system. This layer is mainly concerned with improving the performance of data access. Services offered are "store" and "retrieve" operations for data elements (records) using internal record identifiers.

The next layer provides essentially the same operations, but makes them **secure**. Changes in the database are guaranteed against loss or interference by other users.

The third layer adds different types of **access methods,** e.g., access to data based on a value (e.g., street address) or spatial location.

The fourth layer offers a **logical structuring tool for the data** and manipulations based on this logical schema. These services are then offered as an extension to a high level programming language or an independent query language.

REQUIREMENTS FOR THE DATA STORAGE LAYER

The main purpose of layer 1 is to interface with the operating system and to improve speed of storage and retrieval. Performance of a database management system is primarily perceived as response time to queries or generally time to retrieve data stored in the system. Experience shows that retrieval time of data is determined by the number of physical disk accesses, and that processing time of data once transferred to working memory is of minor influence.

Spatial information systems typically require large databases stored in a permanent manner on mass storage devices (disk). Only small parts of these collections are usually accessed for the execution of a simple operation. Access to data on a disk takes about 30 milliseconds per access and is nearly independent of the amount of data read. Disk technology has improved considerably over the last decade, resulting in much larger storage capacity and lower prices. Access time has, however, remained nearly constant. Disk access time must be compared with the time to process data in central memory (less than 0.1 microsecond – thus 300,000 times faster).

A major requirement for this layer results from the maximum delay we can allow for the drawing of a map on a screen. If each of 2000 to 5000 data records necessary for "one screen full of data" were fetched from the disk with an independent access (taking at least 30 msec), the user would have to wait 1 to 3 minutes for the map to be drawn, completely unacceptable in an interactive environment. Therefore, layer 1 must reduce the number of physical accesses necessary to retrieve the records for the map. Two basic techniques are known: clustering of data and buffer management. We use both. It must be noted that operating systems sometimes use similar methods to improve performance of disk operations and these optimizations techniques can interfere with the methods a database uses. It may be advantageous, therefore, to use low level disk access operations and by-pass operating system services for buffering.

CLUSTERING

The primary method to reduce the number of disk accesses is to form clusters of logically or spatially related objects on disk pages (Figure 3). A physical access to the disk brings in a larger chunk of data, usually called a disk page (512 bytes to a few thousand bytes large). If we can arrange our records on disk pages in such a way that each page contains several data records necessary for the map drawing, the number of time-consuming disk accesses to transfer the data to working memory will be greatly reduced. In 20 seconds, we might read about 200 pages. If each contained 25 of these records needed (and possibly some others), all the necessary data will have been read.

This is feasible in cases where a reasonable prediction of what data will be used together is possible. These data are then stored together and form a **physical cluster** on the disk (Salton and Wong, 1978). Fortunately for spatial retrieval for map drawings, such predictions are easy, based on the **neighborhood** relation. We retrieve data for a map situated within a certain area. If we retrieve a data element of an object, chances are good that data from other objects in the vicinity are needed next. It would also be advantageous if data of different types (e.g., houses, roads and rivers) could be stored on the same "page" and not necessarily require different physical accesses. If such data are clus-

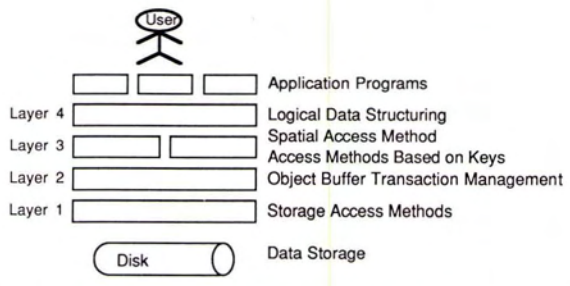

FIG. 2. The layers of a spatial database management system.

*Layer in this context describes a collection of software modules (programs) and must not be confused with the use of the term to describe a cartographic layer in an overlay system.

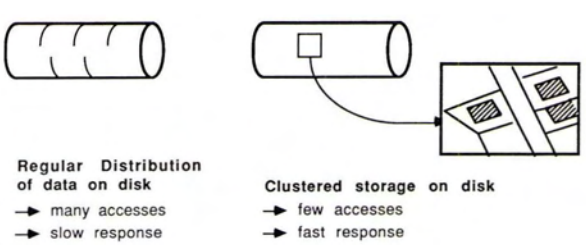

FIG. 3. Spatial clustering.

tered together and retrieved with one physical access, we can achieve the goal of retrieving a map within a short time span, permitting interactive work.

A database system for spatial data must at least provide for physical clustering of data (typically a command like STORE NEAR X" where x is an already stored record must be available to the programmer of low-level data storage routines). This requirement excludes many of the simpler relational DBMS, where physical storage of data is directed by the primary key and cannot be influenced by the user. It is neither necessary nor desirable that physical clustering be visible on the level of the user interface. It should rather be used internally for fast spatial access and be of no concern to the user.

BUFFERING

Many programs show a locality of access (i.e., the same data elements are used repeatedly over a short period of time). If these data elements can be kept in a buffer, the number of physical accesses to a slower mass storage device can be reduced. For each data element, only the first access uses the slow device, and all following requests are satisfied using the buffered data. This strategy is generally employed in computers (e.g., virtual memory, cache). A DBMS usually contains a single set of buffers for pages brought in from the disk in order to exploit a physical clustering of data on the storage device.

Programs dealing with spatial data typically show much locality of access to data but use a large working set (e.g., all the records for a map drawing). Users tend to work in a geographical area for several interactions before they request another base map. Very often they ask for an overview map first, then zoom in on a detail of interest and start to work on this. In our experimental database management system, PANDA, we included a second level of buffers for simple records. We currently set the size of the buffer to 5000 records, which allows us to keep a complete map drawing in buffer. Redrawing a map or zooming in does not require any additional physical accesses and is consequently very fast.

PROTECTION OF DATA

Large collections of geographic or administrative data are, as with other data collections, valuable assets and must be protected. Even if the information is maintained in parallel in other, traditional forms (registers, maps), the cost of transferring this information into a machine readable form is considerable. Furthermore, information that may be deduced from the data may be of enormous economic value for someone who knows how to take advantage of it (information is power). Even if this aspect is less pronounced for a spatial database than for databases in commercial usage, data collected must be kept from unauthorized use. For example, most national statistical bureaus collect data which must only be disclosed as statistical aggregates that do not disclose values for individuals.

It is customary to differentiate the following four classes of threats:

- loss due to errors in operating or malfunctioning of hardware or software, or erroneous manipulation by authorized users;
- destruction or access by unauthorized users;
- introduction of false data by authorized personnel using correct procedures; and
- corruption of data by multiple users at the same time (concurrent updates).

Digital data are quite complex to protect and organizations typically have little experience in protecting data bases. Human beings cannot sense the presence (or absence) of data directly. One needs computer equipment and programs to assess a data collection. Also, data quality cannot easily be evaluated, as the presence of data does not guarantee that the data are useful, correct, updated, or complete.

Although layer 2 does not add substantial new functionality, it will increase security of operations. Most of the functions in this layer will, however, reduce performance. That is the price we pay for the security gained.

In order to determine appropriate security measures, we must assess possible damages resulting from loss of data (e.g., cost of reentering data, cost associated with interruption of operations) and balance them against the cost operations to prevent such losses. Most commercial operations place great importance on uninterrupted operations and confidentiality. In consequence, very involved but secure schemes have been devised. In GIS applications it may be acceptable that data are not available for a few hours or even a day if such interruptions occur very seldom. Lower levels of security may thus be acceptable.

PROTECTION AGAINST LOSS

Protection of data against loss by malfunctioning hardware or software is absolutely necessary. In all situations in which more than just infrequent changes are made, the protection mechanism should include the updates of the database. A change the user has affected and the confirmation received from the database must not be lost. It is customary to distinguish two types of problems:

- Interruption of the database management program due to operating errors, problems in the operating system, or failure of the hardware. Such interruptions occur in most installations quite frequently (one per day to one per week). During such events, all contents of main memory are lost, and it is therefore necessary to write changed data to permanent mass storage before confirmation of an update.
- Loss of the storage media, again due to errors in operations or hardware defects (the so-called "head crashes"). Such problems are usually rare (once a year) and slower recovery procedures are acceptable.

Services offered in commercial database management systems are generally sufficient. However, many of the systems for use on micro- and personal computers, as well as the systems specialized in geometric and geogrpahic data, very often do not adequately protect the data.

SECURE DATA FROM UNAUTHORIZED USAGE

Some databases will contain confidential information. It is necessary to protect it against unauthorized use. Access must not be given to people prohibited from using its contents. Measures to exclude certain otherwise authorized users from access to certain classes of data must be provided. For example, personnel in telephone companies need not know the amount of the mortgages on a property. Some users will only be authorized to read data, but not to change them. Clear regulation of responsibility for data quality is required, and only the groups responsible should be allowed to finalize changes. A special problem in statistical databases (and many spatial databases will be used as such) is that of restricting authorized users to generalized information only and preventing direct access to individual data. Most of the known systems have proved insufficient to protect against a determined attack, and the protection measures proposed are extremely involved (Denning and Schlorer, 1980).

In general, we may assume that the operating system contains methods to screen out unauthorized users as well as to identify authorized users. However, only the database management system can restrict access to certain parts of the database accessible for certain users.

TRANSACTION MANAGEMENT

Transaction management deals with safeguards to secure data against accidental loss by authorized users. We must deal with

user input errors, but must also consider all sorts of hardware malfunctioning. We must also contend with power-loss, errors in programs, and multiple simultaneous users. The goal is to prevent loss of data by authorized users, and/or the introduction of new, false data. Control of authorized users is primarily concerned with changes inserted into the database, assuming that read-only accesses do not change the database and thus cannot introduce errors.

It is customary to group together a number of logically connected changes to the database to form a transaction. Transaction management in a database management system is concerned with

- atomicity of the transactions,
- concurrency of transactions by multiple users,
- integrity of the database after the transaction, and
- durability of the transactions,

yielding the mnemonic ACID (Haerder, 1985).

Atomicity of Transactions

From the point of view of the database, all the changes included in a transaction are logically connected (that is the idea expressed in bundling them together as a transaction) and, thus, either all of them should be executed or none. A database should never contain partial results of a transaction, even if a transaction is stopped by a sudden hardware failure (or loss of power). Atomicity allows programs to be restarted and continue their work after a sudden stop, as the database is always in a defined state (namely, at the end of the last executed transaction) and the program never needs to deal with the partial execution of a transaction.

Concurrency

Several users may access and change the same piece of data at the same time. Such concurrent actions may conflict and must be coordinated. In a database management system one generally requires that concurrent users see only the effects of completed transactions and cannot observe any partial results of transactions run by other users. Further, the database management system must check that the actions of one user do not invalidate changes another user has made. Commercially available systems for spatial data management typically do not contain automatic prevention of conflicts between users, but rather rely on organizational mechanisms to insure that only one user is working on a file at a time.

Integrity

A collection of data can easily be destroyed by adding "wrong" data. Users will not accept a database from which they often get erroneous or contradictory information. Unfortunately, there are no easy ways to have a program check that entered data (or changes) are correct (i.e., that the data describe reality correctly). The best we can achieve is including rules in the DBMS which test that new (or changed) data do not contradict other facts already stored in the database. We say that a database is consistent if it is free of such contradictions.

Durability

A transaction which is once confirmed should never be lost because of malfunctioning hardware or other problems. A database management system should include mechanisms for maintenance of a journal of all changes posted such that an archival copy, made regularly, can be updated to the newest version if the current data are lost. This is similar to, but more elaborate than, the customary "two-generation" method used to secure files in a batch processing system.

Methods for transaction management are well-known, but unfortunately reduce the performance. They add considerably to the processing time of transactions and can decrease performance during updates by roughly half. This is tne reason they are often not included in commercially available spatial information systems. In comparison of systems (in benchmarking for example), it must always be specified what type of transaction management is used and what levels of protection of data against inconsistencies and loss are provided.

In the experimental PANDA database management system, we have included a large object buffer, which can hold all the data objects a user accesses during a transaction. With this buffer, we can prepare all changes to the database in the buffer. Data files are not affected before the transaction is completed. Then all the changed objects are written to disk at once. This is a relatively simple scheme that can be expanded to include transaction logging, as a journal of all changes can be (must be) written before any updates of database files are accomplished.

The use of transaction management to achieve several competing goals restricts the flexibility of what can be defined as a transaction. If transactions serve as units for durability, they must be short because it is not acceptable to lose much work. Similarly, transactions used to control concurrency must be fast because barring other users from changing data for long periods of time cannot be tolerated. On the other hand, certain operations in a GIS are much too complex to be accomplished in a short transaction. In some applications a user level transaction may be very complex and take a long time to complete (e.g., the subdivision of a lot). It must be possible to record components and partial results, even if overall consistency can only be checked at the end. It is proposed to compose such "long transactions" from short (ordinary) transactions and have the database contain special rules that control completion of the long transaction. This is an area of current database research and no implementations are available today.

ACCESS METHODS BASED ON SPATIAL LOCATION

The layers discussed so far provide for storage and retrieval of data using an internal identifier (database key) which is assigned to a data record when it is first stored. The record itself is considered to be a fixed number of bytes which are stored and later reproduced. For non-standard applications, it is especially important that the storage system does not depend on interpretation of the data stored because this would limit the flexibility of arranging information in a record and produce undesirable dependencies between the different layers of the system and the application's definition of the record types. The database subsystem of a larger application must handle arbitrarily structured data records without dependency on their internal structure. Record data structures constructed from the basic data types (i.e., integers, reals, and strings) is not even sufficient for commercial applications, as exemplified by numerous extensions offered (e.g., data types for money, date).

Standard DBMS include methods to access data elements based on key values or other constructions in the data model:

- Users often need to access data records based on the value of one or several data fields (key fields). This requires, in the simplest case, that the stored values for these fields be unique, so that given values unequivocally identify the desired record (this will be a consistency constraint of the database). It is generally necessary to be able to define more than one key for a record, so the record can be found using either value (e.g., a person's record must be found by social security number or name).
- The data model permits the grouping of objects to units of a higher order. Such combinations create access paths, as users can go from an object to the groups it is contained in or to the objects it consists of. For example, all the buildings on a street must be accessible once the street is found.

A great deal of data in a spatial information system is related to objects in space. Location and extent are known for these

objects. Many applications need access to data based on location (Burton, 1978). This is obvious for all retrievals associated with map production (all objects within the map frame must be retrieved and then graphically rendered). Similar accesses are necessary for internal operations to check incoming geometric data and for geometric manipulations of stored data. Spatial access to data is based on a query of the form: *Retrieve all <object-type> within <area>*, where <area> is the description of the area of interest.

In order to provide a general efficient function, it seems appropriate to reduce <area> to a rectangular box parallel to the coordinate system (Figure 4). Similarly, for each object the location and extension is recorded as a rectangular box (minimal bounding rectangle) (Figure 5). This seems to be a general and computationally simple solution.

More specific requests are then treated in two steps. In a first step, all data within the rectangular envelope of the area are retrieved (based on a comparison of the query rectangle and the object box). The more expensive, exact selection process is performed in a second step only on the data passing the first test. For example, to retrieve all buildings within a town, we retrieve first all the buildings within the minimal bounding rectangle of the town and then, in the second step, check only these against the exact town boundary.

The access method should be divided into (1) a method of physical clustering to achieve a minimal number of physical disk accesses and (2) a logical data structure which permits spatial access and guarantees its correctness, preferably even in the presence of some shortcomings in the physical clustering. The physical clustering will speed up the retrieval of map output and it benefits many other forms of geometric data processing, such as area computation. We can thus exploit the specific geometric locality of access observed in most algorithms of computational geometry (Dutton, 1978).

The traditional approach found in many of the GIS on the market is to divide the world space into map sheets (sometimes called "facettes"), and to store the data for these map sheets as individual files. The amount of data within such a file is then small enough to permit the use of linear search methods. This is, in our opinion, insufficient for the following reasons:

- The granularity of access is fixed and provides rapid access only if the query area is comparable to the sheet size. For queries about much larger areas, response is slow because many different files must be opened and read in.
- Access to more than one sheet requires recombination of objects at the sheet borders. This is a complicated and time consuming operation and only possible with an understanding of the internal structure of the representation of the objects. Thus, the types of geometric objects treated are defined in the storage layer, and it is difficult for a user to add new geometric object definitions.
- Most systems do not hide the sheets from the user, nor do they provide a query language working automatically across sheet boundaries. It seems acceptable to ask draftsmen to know which map sheet they update; it is, however, clearly inappropriate to require this knowledge of a casual user needing a map of water mains during an emergency.

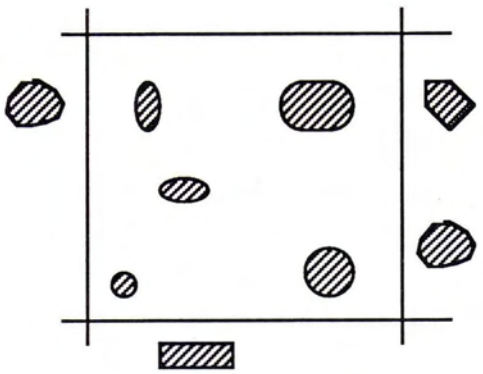

FIG. 4. Query window.

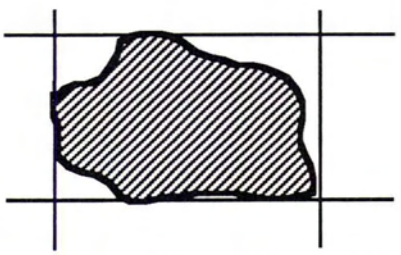

FIG. 5. Object with box (minimal bounding rectangle).

Only a few logical data structures are known to permit fast response to this type of spatial access, technically called a two-dimensional range query. They are all based on self-adjusting partition of space and clustering data of one partition in a disk page, so that access to one page brings in many entities useful for producing the requested map (Nievergelt *et al.*, 1984; Tamminen, 1982; Guttman, 1984; Samet, 1984). The method described in Nievergelt *et al.* (1984) is an adaptation of a more general hash based structure for storage of multidimensional objects. It turns out to be very similar to the *FIELD TREE* method we have developed in *PANDA* (Frank, 1981, 1983).

In the *FIELD TREE* method, clustering is not only guided by location and extension of objects, but includes also a component of level of importance of an object. This speeds up responses to "overview" requests where only "important" objects must be retrieved for a large area (e.g., all towns in a state) or for detail maps where all objects for a small area are retreived (e.g, a building with all public utility connections). Response time for queries is linearly dependent on the number of objects retrieved; thus, a screen full of map data always takes about the same amount of time. Other influences are the size of the area of the query and the number of objects stored for this area. The total number of objects stored in the system has virtually no influence on response time.

QUERY LANGUAGE

Query languages are very important for the retrieval of data to satisfy certain immediate needs which were not planned for and for which no programmed access procedures are available. Interactive languages for *ad hoc* queries are usually included in commercial database management system. It appears today that the SQL language, originally developed by IBM (Astrahan *et al.*, 1976), is becoming the accepted standard, and a respective proposal is currently being dealt with in the American National Standards Institute. A number of groups are presently working on extensions of SQL to make it useful for spatial data bases (Egenhofer and Frank, 1987b). Spatial applications need facilities to retrieve data based on spatial relations, and the query language must be extended to include these. Specifications for the graphical rendering of the data must, for example, be included in the query (Frank, 1982b; Egenhofer and Frank, 1988b).

EXPERIENCE AND FUTURE WORK

During the last seven years, we have built an experimental DBMS suitable for use in a GIS. It is based on an extended network data model and uses the FIELD TREE organization for fast spatial access. PANDA consists of about 20,000 lines of highly modular Pascal code and has been installed on IBM (under VM/CMS) and Digital Equipment Corp. (under VMS and TOPS-10)

computers. A commercial company is currently revising PANDA to build a complete GIS. We have used PANDA to implement methods to represent spatial data using cell complexes (Frank and Kuhn, 1986). In order to test this method, some simplified map data were loaded and maps drawn on the CRT. We found that a powerful object-oriented programming language would allow us to write generally applicable programs which can be integrated to form specific applications later. This would simplify the building of applications. We are currently changing the programs interface of PANDA to become completely object-oriented. We found this to be a necessary step to allow use of object-oriented software engineering methods to build GIS from basic building blocks. We also work on a query language for GIS. It is not clear if extensions to SQL are sufficient or if a more object centered design would result in a language easier to use for GIS applications.

There is need for research to deal with "long transactions." We plan to add support to PANDA to be able to maintain multiple versions of the same object (e.g., lot or public utility line) at the same time where each long transaction is related to a specific version. It will be necessary to build tools to simplify the merging of different versions when they are confirmed. This closely related to the necessity to deal with events and other time related data in many types of GIS applications. For example, for planning purposes not only the current land use but also previous uses are important and often there are considerable interest in the rate of change as well. Preliminary studies make us believe that the incorporation of temporal data will not be an easy change but affect at least storage and retrieval methods, transaction management, and the user interface. The query language will need temporal extensions, too.

CONCLUSIONS

We conclude with a number of recommendations:

- Spatial data collections including GIS should be built using the database management system concept. This concept is crucial for an interactive information system which can serve multiple users and multiple requirements.
- Spatial data collections pose some special requirements, which render the commercially available database management systems unsuitable. They generally lack the special provision to achieve physical clustering necessary for fast access to spatial data, and they are optimized for data with quite different characteristics. Their performance is generally observed to be insufficient for GIS applications.
- Spatial database management systems must include many of the standard features found in commercial systems, especially data protection and transaction management to preserve loss of data due to malfunctioning hardware and to permit concurrent users.

We recommend a layered architecture of the spatial database management system, with a database kernel providing generally required services, and additional modules to adapt the database management system to the special needs of spatial applications (Haerder, 1986).

REFERENCES

Astrahan, M. M., D. D. Chamberlin, *et al.*, 1976. Sequel 2: A Unified Approach to Data Definitions, Manipulations and Control, *IBM Journal Research and Development*, Vol. 20, p. 560.

Burton, W., 1978. Efficient Retrieval of Geographical Information on the Basis of Location. *First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems* (G. Dutton (Ed)), Harvard Papers on Geographic Information Systems, Harvard University, Cambridge, Massachusetts.

CODASYL, 1962. An Information Algebra, Phase I Report, *Commun. ACM*, Vol. 5, p. 190–204.

———, 1971. *Data Base Task Group (DBTG) Report, 1971.*

Codd, E. F., 1982. Relational Database: A Practical Foundation for Productivity, *Commun. ACM*, Vol. 25, p. 109.

Dangermond, J., and S. Morehouse, 1987. Trend in Hardware for Geographic Information Systems, *Proceedings Eighth International Symposium on Computer-Assisted Cartography* (N. Chrisman (Ed.)), Baltimore.

Denning, D. E., and U. Schloerer, 1980. A Fast Procedure for Finding A Tracker in a Statistical Database, *ACM Transactions on Database Systems*, Vol. 5.

Dutton, G., 1978. Navigating ODYSSEY, *First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems* (G. Dutton (Ed)), Harvard Papers on Geographic Information Systems, Harvard University, Cambridge, Massachusetts.

Egenhofer, M., and A. Frank, 1987a. *PANDA: An Object-Oriented Database Based on User-Defined Abstract Data Types*, Report No. 62, Surveying Engineering, University of Maine, Orono, Maine.

———, 1987b. An Extended SQL Syntax to Treat Spatial Objects. *Proceedings of the Second International Seminar on Trends and Concerns of Spatial Sciences*, Fredericton, New Brunswick.

———, 1988a. "Project Oriented Modeling a Powerful Tool for GIS", submitted for publication.

———, 1988b. Towards a Spatial Query Language: User Interface Considerations. *14th International Conference on Very Large Data Bases*, Los Angeles, California.

Frank, A., 1981. Applications of DBMS to Land Information Systems, *Proceedings VII International Conference on Very Large Data Bases*, Cannes (France), p. 448.

———, 1982a. *PANDA: Pascal Network Database Management System* (in German), Report 62. Institute for Geodesy and Photogrammetry, Swiss Federal Institute of Technology, Zurich, Switzerland.

———, 1982b. MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and Their Graphical Representation, *SIGGRAPH '82 Conference Proceedings, Computer Graphics*, Vol. 16, p. 199.

———, 1983. "Storage Methods for Space Related Data: The FIELD TREE", *Spatial Algorithms for Processing Land Data with a Microcomputer* (M. Barr (Ed.)), Boston, Lincoln Institute of Land Policy.

———, 1984a. Computer Assisted Cartography: Graphics or Geometry? *Journal of Surveying Engineering*, Vol. 110, No. 2, pp. 159–168.

———, 1984b. Extending a Network Database with Prolog, *Expert Database Systems, Proceedings of the First International Workshop on Expert Database Systems*, Kiawah Island, South Carolina.

———, 1986a. *PANDA: An Object Oriented Pascal Network Database Management System*, Report 57, Surveying Engineering, University of Maine, Orono, Maine.

Frank, A., and W. Kuhn, 1986. Cell Graphs: A Provable Correct Method for the Storage of Geometry, *Proceedings Second International Symposium on Spatial Data Handling*, Seattle, Washington.

Frank, A., and M. Tamminen, 1982. Management of Spatially Referenced Data, *Proceedings International Symposium Land Information at the Local Level*, (A. Leick (Ed.)), University of Maine, Orono, Maine, p. 330.

Guttman, A., 1984. *New Features for a Relational Database System to Support Computer Aided Design*, Memorandum No. UCB/ERL M84/52, Electronic Research Laboratory, College of Engineering, University of California, Berkeley.

Haerder, Th., 1986. New Approaches to Object Processing in Engineering Database, *Proceedings 1986 International Workshop on Object-Oriented Database Systems*, Pacific Grove, California.

Haerder, Th., and A. Reuter, 1982. *Database Systems for Non-Standard Applications*, Report 54/82, Fachbereich Informatik, Universitaet Kaiserslautern, (FRG).

———, 1985. Architecture of Database Systems for Non-Standard Applications (in German). *Database Systems in Office, Engineering, and Scientific Environment, (A. Blaser and P. Pistor, eds.), Springer Verlag, New York.*

Manola, F., J. Orenstein, and U. Dayal, 1987. Geographic Information Processing in the Probe Database System, *Proceedings Eighth International Symposium on Computer-Assisted Cartography*, Baltimore.

National Academy of Science, 1980. *Need for a multipurpose Cadastre* by

Panel on a Multipurpose Cadastre Committee on Geodesy. National Academy Press, Washington, D.C.

Nievergelt, J., *et al.*, 1984. The Grid File: An Adaptable Symmetric Multikey File Structure, *ACM Transactions on Database Systems*, Vol. 9.

Plouffe, W., *et al*, 1984. A Database System for Engineering Design, *A Quarterly Bulletin of the IEEE Computer Society Technnical Committee on Database Engineering*, Vol. 7, No. 2.

Salton, G., and A. Wong, 1978. Generation and Search of Clustered Files, *ACM Transactions on Database Systems*, Vol. 3.

Samet, H., 1984. The Quadtree and Related Hierarchical Data Structures, *ACM Computing Surveys*, Vol. 16, No. 2.

Schek, H.J., and V. Lum, 1983. Complex Data Objects: Text, Voice, Images: Can DBMS Manage them? *Proceedings 9th International Conference on Very Large Databases*, Florence (Italy).

Tamminen, M., 1982. Efficient Spatial Access to a Database, *Proceedings SIGMOD Conference*.