AN INTEGRATED DBMS APPROACH TO GEOGRAPHICAL INFORMATION SYSTEMS.

M S Bundock
Cambridge Interactive Systems Ltd.,
Harston Mill, Harston
Cambridge CB2 5NH, U.K.

ABSTRACT

The key features of data continuity, topology, and very large data volumes are discussed in relation to Geographical Information Systems. A database management system approach is suggested, and the problem of response time degradation addressed. Data analysis techniques, based on Information Engineering methodologies, have been used to produce a datamodel representing the entities and relationships within the geometric data. The datamodel, which progresses beyond the standard point-line-polygon structure, is presented and described. Its applicability to a wide range of GIS applications is demonstrated, and an implementation using relational DBMS, high level language, and data dictionary technology mentioned. The implementation uses a relational DBMS to manage both geometric and attribute data in a single spatially continuous database.

INTRODUCTION

The explosive growth in interest in Geographical Information Systems (GIS), as opposed to conventional mapping systems, has evolved predominantly by the growing awareness of information, of which only part may be displayed on a map. In addition, the following issues have been raised:

- that the information displayed on maps/charts etc. is often a subset of the information that is held or required elsewhere within an organisation,

- that the same information may be displayed on a number of cartographic products, each with different purpose, scale and symbology,

- that segmenting the area into discrete partitions, imposes extreme limitations, especially as we are attempting to model a continuous phenomenon - the land, and

- that to maximise the usefulness of the organisation's data, flexible tools that link both cartographic (graphic) and geographic (non-graphic) data should be available.

Historically, the CAD/CADCAM and the general data-processing industries have in many ways have followed separate but parallel development paths. Each industry responded to the perceived requirements of their users, while very few users recognised the need for a combined integrated approach. As a consequence the products that were available to users of geographical data who required cartographic output, tended to become little more than slightly adapted computer aided drafting systems.

By utilising the developments within both the CAD and general DP industries simultaneously, we should be able to better manage an organisations most precious asset - its data, and present it in the most suitable manner to convey information. In particular, the use of DBMS technology provides us with a method of managing the data while CAD techniques provide us with a means of efficiently presenting the data graphically.

Developments associated with database management systems, have provided us with methods for analysing data and its structure to create a "logical" database design. These methods can be applied to cartographic data in just the same way that they might be applied to any other business data. The cartographic entities and their inter-relationships, may be represented in an easily understood manner using the relational model (Date 1986).

Implementation of the model, using a relational DBMS, requires that we must address the problems of performance by considering "physical" database design. The volume of cartographic data that is displayed and interrogated, in a typical interactive GIS environment, is significantly greater than that associated with more traditional DBMS activities. Techniques that minimise disk I/O can have a significant effect on overall performance.

Throughout this paper, it is assumed that the cartographic data is being represented using vector, rather than raster techniques, unless stated otherwise. The author wishes to emphasise that this is not meant to be a statement that vector techniques are always more appropriate than raster, rather that vector representations can be used in a wide variety of applications. Clearly, raster overlays from for instance, satellite imagery and aerial photography, will often be useful in helping to convey information to a GIS user.

HISTORY

Vendors of mapping and Geographical Information Systems, create/release products in response to perceived user requirements. Until recently, the perceived user requirements have been dominated by those of the drafting/surveying/engineering departments, whose main concern was to produce their products (drawings/maps) more efficiently. As cartographic applications were seen to fall outside the realm of the DP department, the drafting departments were responsible for their own cost-benefit justification and expansion program. Throughout the 60s and early 70s the demands placed on CAD vendors were principally graphics-oriented. In the mid 70s we saw an increasing demand to allow the association of non-graphic attributes to graphic entities.

During the same period, DP departments had been developing/installing business applications (e.g. payroll, MIS, inventory etc.) which involved the management of large volumes of data, often with complex inter-relationships between data entities. This led during the 60s and 70s to the development of database management systems, with facilities for managing very large volumes of data, having complex data relationships, in a multi-user environment. Also, data processing vendors started supplying more generalised and efficient capabilities for forms/screen management, query languages and high level languages for applications development.

Once non-graphic attributes could be associated with graphic entities, many CAD equipped drafting departments began duplicating the data and data capture effort already undertaken by the users of facilities supplied by the data processing department. Most of us have seen examples of the same data being duplicated within an organisation. Where there is no coordinated effort to capture, validate and update that data, the likelyhood of inconsistency and error grows immensely.

During the late 70s and early 80s we have seen acceptance of the concept of "corporate data", i.e. the data captured by an organisation should be accessible throughout the organisation, be owned by a section of the organisation, and preferrably not duplicated. The data can then be used to convey information to users in a consistent manner, and reduce the possibility of inconsistency and error. The data may be presented to each user in a different form, but the source should be the same. In this way the user extracts the required information from the data. The representation of a pipe as a line on a graphics screen, is merely one way of conveying information about that pipe to a user. The data describing this representation of the pipe is perhaps its x,y,z coordinates. The information conveyed to the user is the whereabouts of the pipe and its relationship to surrounding objects.

PERCEIVED PROBLEM AREAS

Many existing systems available today exhibit a number of significant deficiencies from the users point of view, including:-

- Inability to manage and associate large volumes of attribute data to graphics entities, without partitioning or replicating the data,
- Inability to create and maintain complex relationships between data entities, be they graphics or non-graphical attributes,
- Inability to give a performance level that is basically independent of data volume,
- Inability to provide general multi-user access to data,
- Inability to provide adequate access security,
- Inability to provide adequate data integrity validation,
- Inability to provide a continuous mapping capability, and hence network analysis, derivative mapping etc. become very difficult,
- Inability to create products for a number of uses at a wide range of scales from a single database representation,
- Inability to model the topology often associated with graphical elements.

Many of the above problems arise from the use of special display file structures, designed specifically to allow user interaction with a logical map sheet. The map sheet (design file, coverage etc.) concept followed logically from the perceived requirement to computerise the map production cycle, i.e. produce a computer version of a piece of paper. Also it conveniently broke the total data volume down into bite-sized pieces that could be more readily handled and give adequate performance. However, our world is not segmented into discrete rectangular sections corresponding to map sheets, and there is really no reason why we should not model the world as it really is, continuous.

DBMS SOLUTION

Many of the above problems can be solved by using any one of a number of commercially available database management systems. Hence it is proposed that a DBMS be utilised to manage not only the geographic attribute data, but also the cartographic data. The data defining the cartographic entities to be displayed can be managed within a DBMS in much the same way as any other data.

Clearly the volume of data associated with providing a continuous cartographic coverage of a country/state/county is large, but is often not significantly greater than that associated with geographic attribute data for the same area. Preliminary estimates of data volumes, for a range of GIS applications (including public utilities, cadastral, topographic and DTM) providing coverage at a national or county level, indicate that total data volumes would typically vary between 1 and 100 Gigabytes per organisation.

A relational DBMS provides a natural way of storing data and modelling the complex inter-relationships between data entities that occur in the real world. Relationships between data entities may be naturally expressed by each relation (table) sharing some common attribute (field), or attributes sharing a common domain. Hence a relational DBMS is able to do away with pointers (at least at the user perception level), and consequently access paths (navigation) need not be pre-defined and restricted, as with network/hierarchic systems.

Multi-user concurrency is a standard feature implemented in virtually all commercially available DBMS. Locking mechanisms are used to ensure that each user obtains a consistent view of the database, even while another user may be performing updates. Once concurrency control is implemented in the GIS environment, multiple users can view the same geographical area, each having a consistent view of that area, and each able to update it. However, deadlocks can occur when different users lock records in a different order. Most DBMS's check for deadlocks and resolve them by backing out one of the transactions.

Security mechanisms are used to provide data privacy, at either the relation (table) level, or the attribute (field) level. Hence particular types of data can be protected from inadvertent user update or viewing. Furthermore, recovery from a failed transaction (or possibly from a "what if" scenario) is possible, since DBMS's provide a facility for reversing out the incomplete changes. This is called rollback or backout. Recovery from a corrupted database is usually provided via a roll forward mechanism, which involves applying the logged changes to a copy of the database.

The capabilities of such systems, provide us with a framework within which we can manage GIS data, both cartographic and geographic, allowing multiple users to access a single continuous model of the area of interest.

DATA ANALYSIS OF CARTOGRAPHIC FEATURES

How then to model the geometry of cartographic features? Fortunately, paralleling the development of database management systems, methodologies have been developed to analyse data, its structure and its integrity constraints. We have used one such methodology to analyse cartographic data, in much the same way we would analyse any other type of data. The result is a datamodel (Figure 1) that represents the data entities forming cartographic images, and the relationships between them.

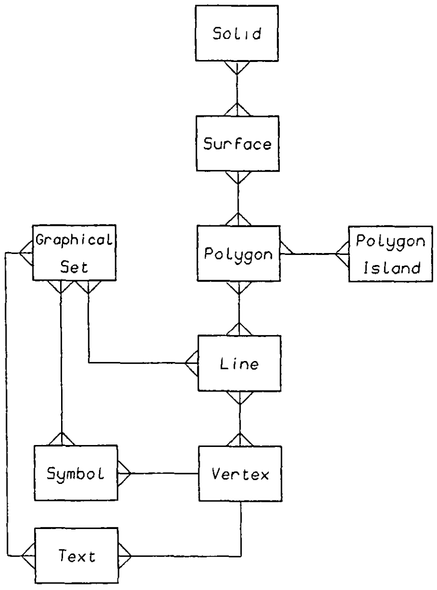

Figure 1.
Datamodel of
cartographic entities
(not normalised)

The following is a brief description of each entity and the relationships it has with other entities.

- The SOLID entity represents a solid region, formed by a closed set of surfaces. Any number of surfaces may be used to form a solid, and conversely a single surface may be common to more than one solid. hence there is a many-to-many relationship between solids and surfaces.

- The SURFACE entity represents a single continuous surface and is formed by a set of planar facets (planar polygons). Any number of facets may be used to form a single surface, and a single facet may be shared by multiple surfaces. Consequently there is a many-to-many relationship between surfaces and polygons.

- The POLYGON entity represents a closed area formed by a set of lines. Any number of lines may be used to form a closed polygon, and conversely, a single line may form part of the boundary of more than one polygon. Hence there is a many-to-many relationship between polygons and lines.

- The LINE entity represents any type of line segment, be it a straight line, an arc, a smooth curve or a line string with many vertices, that is unbroken by any other line possessing the same associated attributes. A line is formed by a set of vertices defining its position in space, and conversely a single vertex may be shared by more than one line.

- A VERTEX identifies a unique spatial location and possesses attributes that define that location, ie. its coordinates.

- The SYMBOL entity represents a point feature such as a trig station, a man-hole, a transformer etc. Each instance of a symbol is located at a single spatial location, but more than one symbol could exist at that location. Hence there is a many-to-one relationship between symbols and vertices.

- The TEXT entity represents free standing text. Normally in the GIS environment, text is stored as an attribute of a geographic entity. However, this feature has been included for users of data obtained from other systems (e.g. Ordnance Survey), that support free standing unassociated text.

- A polygon may contain other polygons (islands or holes), and they may be nested to any depth. This is a recursive relationship and can be represented in a number of ways. Here we define a one-to-many relationship between a polygon and the polygons it contains.

- The GRAPHICAL-SET entity is used to relate a set of cartographic entities that need to be referenced as a single entity. Examples include formed road boundaries, all the elements required to form a river or an airport etc. A graphical set may itself contain other graphical sets, lines and symbols.

It is important to note that the familiar "node-link-polygon" structure (by whatever name we choose to give it!) appears in a similar manner here. A significant difference to most other formulations however is that the vertex is the only entity to directly contain coordinate information.

Using this data structure, data is not duplicated, since each vertex, and line is only stored once. Consequently, moving a vertex moves the endpoints of all lines referencing the vertex. Similarly, when dealing with polygons, no duplicate lines are required since each line is shared by all polygons referencing it.

Before implementing this model in a relational DBMS we must first apply the procedure of "normalization". The objective of normalization is primarily to remove redundancies and produce a "clean" structure. The result is a new datamodel where the many-to-many relationships between entities are resolved by the introduction of "junction entities". This model may then be implemented directly, as each entity maps to a database table (relation).

A geographic feature will have a set of geographic attributes and an associated cartographic representation. In fact sometimes a single feature may have more than one cartographic representation, for example a city may be represented as a polygon at one scale and as a point symbol at another. This association is achieved by the geographic feature table containing a reference to each type of cartographic entity used to depict it.

DBMS GRAPHICS PERFORMANCE

An integrated DBMS approach where both geographic and cartographic data are managed within the same database could create severe performance problems. Normal interactive database applications typically require the DBMS to return to the user a screenful of information (alphanumeric) per transaction. Such requests require the display of very few database records/fields. Transactions which scan large numbers of database records are not usually considered as interactive tasks.

When a GIS application requests the cartographic display of a geographical area, there may be several thousand cartographic entities to display, e.g. lines, symbols etc. Consequently many thousands of database accesses may be involved.

Database management systems tend to be very disk intensive, requiring on average between 1 and 4 disk I/Os per random retrieval of a specified record, depending on the indexing method used.* Disk is a relatively slow device, with normal access rates in the range 30 to 100 seeks per second. Once on cylinder, data is transferred to the CPU at a high rate, but rotational latency and head positioning affect overall performance considerably. Competition for the device from other users could further reduce the effective seek rate for each user. Consequently, the time required to retrieve all the cartographic data for an area containing thousands of cartographic entities is potentially unacceptable unless some method can be found to reduce the number of disk accesses.

ONE APPROACH TO SOLVING THE DBMS GRAPHICS PERFORMANCE PROBLEM

Until now we have discussed "logical" database design, and possible performance problems associated with transactions on that database. The following is a discussion about "physical" database design options, which aim to solve the performance bottleneck.

There are several methods commonly used with hierarchic and network DBMS's to improve performance that involve grouping together database records of a particular type. When a request is made to read a particular record from the database, the address of the record is determined, and a request is sent to the disk subsystem to retrieve the block or sector within which the required record resides. Many records may reside in the block of data returned, and the DBMS is responsible for extracting only the required piece. The entire block may then be held in an area of memory called the "cache". If at a later time the user requests the same record, or a record within the same block, it may be retrieved directly from cache, rather than from disk. The system simply checks whether the required block address matches any of those in cache and acts accordingly. Consequently, if records can be grouped in some way, then the number of I/Os could be reduced, improving response.

DBMS use a variety of techniques to provide fast access to a particular record within the database. Different techniques possess different attributes, with respect to how they distribute the data throughout the physical database. The hash technique is often used to randomise the data throughout the physical file, although it can also be used to group similar data together. In particular, a hybrid hashing technique has been suggested (Davis 1986) that can be applied

* For B-Tree indices, in the general case of retrieval of random records, from a table of R records, we would expect the number of disk I/Os to be $O[\log_n(R)]$, where n is the number of nodes held in a node block of the index.

to the organisation and indexing of spatial data. Alternatively, indexing techniques such as the B-Tree index, order the nodes of the index tree in ascending key order. If the data records are held in the leaf nodes of the tree, then they also will be ordered in ascending key value (certainly within each block and possibly partially within the file). Throughout the remainder of this paper I have assumed that B-Tree indexing is being used.

Normal GIS activities result in the user viewing a geographical area and performing transactions relevant to that area. If that area, represents a small fraction of the possible universe contained within the continuous model, then we can say that all the cartographic entities visible in that area must have "similar" coordinate values. Consider a function

$$s = f(R, R_u)$$

and $R = (r_x, r_y)$ is the range of the cartographic object

and $R_u = (r_{ux}, r_{uy})$ is the domain of x and y

where $r_x = (x_{min}, x_{max})$ is the range of x

and $r_y = (y_{min}, y_{max})$ is the range of y

such that $f(R,R\ )$ returns a scalar, and that similar coordinate values yield similar function values. If this function is used to create values for the primary key of a cartographic entity in a B-Tree indexed database, then the probability of entities having similar coordinate ranges being in the same block on disk is increased.

Effectively, what we are attempting to do, is impose a one-dimensional ordering on a 2-dimensional space. Some of the techniques that may be used have been described (Mark et al 1986), and in particular the techniques of bit-interleaving of coordinates (Abel 1986), and quadtrees (Mark 1986) have been used to create primary keys for vector cartographic databases.

A modification of the quadtree technique has been used in a prototype GIS developed by Cambridge Interactive Systems Ltd. The quadtree technique can be thought of as a recursive subdivision of the universe until, in the vector case, further subdivision requires that the entity concerned be broken. In the raster case, subdivision continues until the cell is homogeneous. In each case the address of the cell, possesses both locational information and cell size information. The technique fails (for the vector case) when small entities cross large cell boundaries, preventing further subdivision of the cell. Overlapping cells may be used to overcome this problem.

In summary, the combined use of:

- cache memory
- an indexing technique that orders data, and
- a primary key value that is a function of geographical location,

can provide a mechanism whereby cartographic data may be retrieved from very large databases, with a substantial decrease in the number of physical disk I/Os required. Experiments (using these methods and a cache size of 32K words) have indicated that, typical GIS queries yield cache hit ratios of between 30:1 and 150:1. The ordering mechanism makes these results virtually independent of total data volume. Furthermore, the same approach can be extended and generalised for managing n-dimensional data, and non-cartesian coordinate systems, simply by selecting a suitable function to define the primary key.

OVERVIEW OF DATABASE DESIGN

A prototype GIS has been developed using the techniques described above. It utilises a relational DBMS to manage both geographic and cartographic data in a spatially continuous manner. The database is logically broken into four sections, cartographic, geographic, product dictionary and data dictionary.

The data structures required to model the geographic attributes of real world objects, will vary enormously from one organisation to another, and hence must be user defined. The relationship between a geographic entity and its cartographic representation must also be determined.

The Product dictionary manages data that describes the products that a GIS user may wish to create. It should be noted that, the same real world object, or more correctly its cartographic representation, may appear in a variety of GIS products, each with perhaps a different scale and symbology. For example, an airport may be represented by an aircraft symbol on a small scale map, with a generalised perimeter and different aircraft symbol on a medium scale map, and with detailed perimeter, runway and building information on a large scale map. The database should contain only a single representation, i.e. the most detailed level of data, and the scale and usage of the product should define the symbology. Hence in general, the appearance of the object must not be predetermined and stored with the cartographic entities defined previously. Instead, the symbology and generalisation requirements, should be associated with a definition of the product. In the prototype GIS, these requirements have been implemented as a set of symbolisation rules that may be associated with a set of cartographic products. The same technique has been used to define rules for the display of geographic attributes as cartographic text.

The data dictionary has been implemented as a set of tables within the database. It manages data describing the structure of the entire database, access rights, database users, and data to automate the creation of screen forms. A query language is being developed that makes extensive use of the data dictionary. The aim is to allow the GIS user to create, modify, and query the entities within the database in a manner that is natural to him/her, using the words that are normally used in that particular application area. In particular, we cannot expect a user to know the structure, and names of the cartographic entities in the database. Hence the query language via use of the data dictionary must support:

- aliases for both field and table names

- automatic recognition of the relationship between a geographic entity and its cartographic representation

- cartographic operators.

One further note concerning the query language is that it separates the retrieval of data from the display of data, contrary to the more normal SQL SELECT statement.

EXAMPLES OF APPLICABILITY

The prototype GIS has been developed using BaCIS II, a high level, polymorphic, procedural language developed by Cambridge Interactive Systems Ltd. The GIS environment integrates pure database access, interactive graphics, the query language, menu subsystems, and BaCIS II.

Tests have been performed in a number of GIS application areas, including:

- cadastral, electoral, administrative
- facilities management (water and electricity)
- small scale topographic
- mining and modelling of mine waste deposition
- local authority applications.

The topological structure of the database, has enabled users to perform complex network and polygon interrogation with ease, while the data dictionary provides the ability to interact with the model in the users own jargon. Graphic interaction is currently via 2D images, although 3D solids may be viewed and shaded images created.

CONCLUSION

Further development is continuing, but the feasibility of this approach has been proven. The ability to model the world in a continuous form, eliminates many of the problems associated with traditional mapping systems, and the integration of DBMS technology takes us beyond computer aided cartography into the realms of an Information System.

REFERENCES

Abel, D. J., (1986), Bit-interleaved keys as the basis for spatial access in a front-end spatial database management system: Proceedings, Volume 1, Auto Carto London, pp. 163-177

Date, C. J., (1986), Relational Database (Selected Writings), Addison-Wesley Publishing Company Inc.

Davis, W. A., (1986), Hybrid Use Of Hashing Techniques For Spatial Data: Proceedings, Volume 1, Auto Carto London, pp. 127-135

Mark, D. M., (1986), The Use of Quadtrees in Geographic Information Systems and Spatial Data Handling: Proceedings, Volume 1, Auto Carto London, pp. 517-526

Mark, D. M., and Goodchild, M. F. (1986), On the ordering of two-dimensional space: Spatial Data Processing using Tesseral Methods, (collected papers from Tesseral Workshops 1 and 2), Natural Environment Research Council, pp. 179-192