SPATIAL ACCESS TO THEMATIC AND TOPOLOGICAL STRUCTURES IN GEO-INFORMATION SYSTEMS

by *Norbert Bartelme*, Institute of
Image Processing and Computer Graphics,
Wastiangasse 6, A-8010 Graz, Austria,

and *Bruno Spaeni*, KERN & Co. AG,
Schachenallee, CH-5001 Aarau, Switzerland.

1. Introduction

Information systems that support the concept of digital maps and atlases have gained special attention in the past few years. Due to the rapid development in computer hardware technology, the huge amounts of data do no longer present an obstacle to the implementation of such a system, if a purely quantitative viewpoint is taken. However, if the quality of such a system is assessed in terms of data security and consistency, of performance aspects, of an efficient user interface, and of powerful and exhaustive query methods, a carefully designed database is of prime interest.

Although the state of the art in database design is high, and many general purpose database systems are commercially available, they are hardly suited to serve the needs of applications where practically all data relate to space. Such data must satisfy structural, thematic and topological constraints, and they must be grouped in a way that favours spatial queries. It is clear that in many cases, not all of these requirements may be completely satisfied.

In the paper, a practicable approach to this problem is presented. For some years now, the authors have been engaged in designing and implementing such a database for the geo-information system KERN - INFOCAM; see Bartelme, Kolb, Seifter-Bartsch, Spaeni (1986); this system aims at applications in the municipal planning domain, in cadastral mapping, and in utility networking. The resulting solution presents an equilibrium between user requirements, market considerations and theoretically rewarding elegancy. It is described and evaluated against the background of its competing alternatives.

Paper to be presented at the 16th ISPRS Congress,
Kyoto, July 1988.

2. *Topological and Thematic Modelling*

Digital mapping may be seen as the process of creating a model of geo-related informations. The topography, the natural resources and environmental conditions, but also the administrative and technical infrastructure may be described by the model in a formal way that can be implemented by database designers. The resulting database feeds application programs which provide ways for decision makers to understand the relationships among geo-related phenomena. Due to the irregularity of nature, there is a limit to the accuracy that can be obtained for the model; accuracy considerations also depend on the scope of applications that are envisaged. Still, there are very large amounts of data that must be handled by such a space-related database. There is a great diversity of data views that must be made available to a scope of users.

This makes it necessary to break down the model into elementary units in long-term storage that can be easily managed and checked for consistency, serving as building blocks for short-term applications (see Frank 1983). At the lowest level of any *vector-oriented* data model, *points* and *edges* must be handled. In our model, a point conforms to the terminology used by surveyors and geodesists. It has coordinates, and it may have a point number, a point type, quality informations etc. attached to it, as well as geodetic observations. We exclude *raster models*; they are important in satellite imagery and photogrammetry, but their accuracy is too low for the scope of applications that we are aiming at.

As an example for higher levels of modelling, let us consider how a river may be mapped into a digital database: it may be a simple polygon or a smooth curve that is given by a number of interpolation points and a recipe on how these points are to be connected, and it may also be seen as an area surrounded by a border and containing islands (see fig. 1). In some exceptional cases, one may even be interested in the amount of water transported per time unit, making it necessary to devise a three-dimensional model.

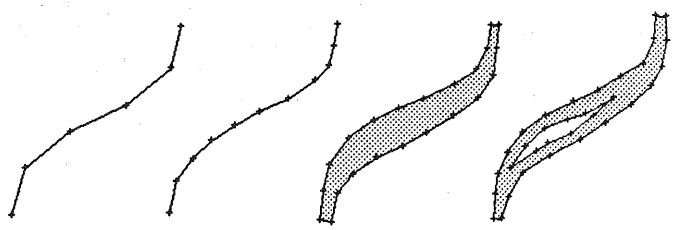

Fig. 1: Different degrees of accuracy when modelling the topological aspects of a river

So far, we have only mentioned the *geometry* and the *topological* aspects. Geometry describes positions, distances, angles, areas, etc. while topology describes connections, islands, proximity etc. But there are also *thematic* aspects that must

be handled: the fact that the entity is a river (and not a road, a boundary etc.) is a thematic statement; and usually, there are many more descriptive *attributes* being assigned, like the name of the river, the average depth, the degree of pollution, the navigability etc. Sometimes, the classification can be argued about: if the river temporarily disappears underground, this may be considered to be a topological feature, since the river flow is being interrupted; but we may also assign different thematic tags to different river portions: "above ground" and "underground".

It remains beyond doubt that applications are always - at least partly - driven by thematic considerations. During data capture, the user does not just digitize connections without paying any attention to their thematic relevance; he knows that he is digitizing contour lines, boundaries, roads etc. When data are being displayed, their thematic significance shines through by the selection of colors, fonts, line types and patterns used. Likewise, any evaluation, any data retrieval is (to some extent) triggered by thematic questions that must be answered. Topological aspects play a very important part, but without thematic backing they are meaningless. We cannot freely modify the topology without observing limitations imposed by thematic considerations. Examples for such interdependencies are numerous: houses must be areas surrounded by a closed boundary, rivers cannot contain loops, roads may not intersect buildings, urban development areas may have islands (excluded inner zones) while school districts may not, etc.

It is therefore obvious that (apart from points) thematic entities are the basic units that the user needs to have access to. According to the dimension of their topology, we classify them into *symbols*, *lines* and *regions*.

A symbol is given by one point (sometimes by several points), each point being marked by coordinates. A region is given by an exterior boundary; interior boundaries may also be present, defining "island" zones. Each line - and also each exterior and interior boundary - is defined by an ordered sequence of points, accompanied by an instruction on the way that these points are to be connected (by straight lines, arcs, smooth curves, or any combinations thereof). Boundaries must satisfy additional constraints (they must be closed, and they cannot have multiple intersection points).

For simplicity reasons, we omit three-dimensional entities. (This does not imply that we cannot store and manipulate point heights; however, they are considered to be secondary informations, just like point numbers, point types etc.)

What we have done so far amounts to defining the user's view of the data, in particular, the view of *one user* that is interested in *one thematic layer* of the data. It is needless to say that in general, a user will be interested in more than

one of these layers, and that there will also be several users working on data domains that may partially overlap. We must therefore separate the aspects that are common to all layers from those aspects that are specific to a restricted clientele. Undoubtedly, the geometry and the topology of such a space-oriented information model serves as a common base for all user needs, since data are always (either directly or indirectly) related to geometric and topological properties, and most applications and data queries comprise spatial considerations. On the other hand, thematics are more specific to a particular thematic context. Different users may superimpose different thematics on the same topology.

At this point, we may recall the two terms *conceptual database model* and *external database model* that are used in database design. There is only one conceptual model defining the basic structures and relations between data, while there are several external models, each one customized to a certain range of users and applications. The elements that we have introduced earlier (symbols, lines, regions) correspond to an external model. The conceptional model must be able to ensure consistency and non-redundancy. Let us consider a map of administrative boundaries. Each administrative unit (a municipal area, a school district, a county) is a region in our terminology. Two adjacent regions share a common boundary. On the conceptional level, we would like to avoid duplications in storage, while on the external level, the two boundary lines should be accessible independently from each other. If we want to highlight just one region, the boundary should not be mutilated by temporarily leaving out its neighbours. Similar considerations hold for lines: the bus routes in a city will most likely have some common courses.

On the conceptual level, we therefore introduce a new topological entity called *edge*. An edge is a connection between two points just as graph theory postulates it. Each edge obeys to a form parameter that states the geometry of the connection: a straight line between the starting point and the end point, an arc with a specified radius, a smooth curve that runs through a number of auxiliary points etc. Several edges may meet in a point. Edges are the topological counterparts of thematic lines and regions: each line consists of one (or several) edge(s) that are unique in the database. Likewise, the exterior and interior boundaries of regions consist of chains of edges. In the example of our map of administrative units, the common boundary between two adjacent regions consists of a chain of edges that are uniquely stored. Updating these edges will entail a consistent modification of both regions (see fig. 2).

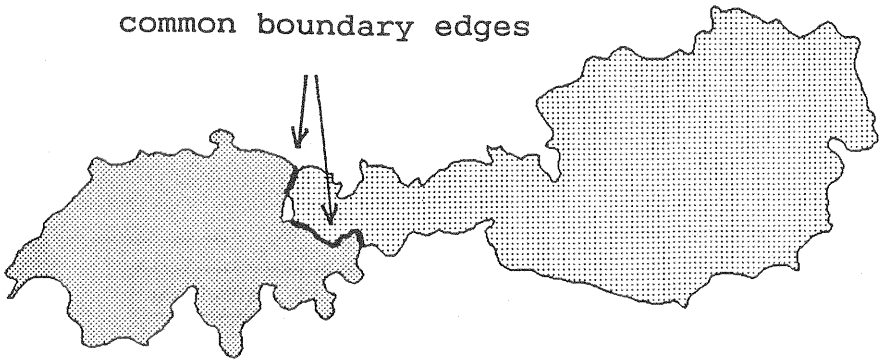

Fig. 2: Common edges ensure topological consistency of adjacent regions

For data belonging to different thematic layers, this updating cannot be done without the user's approval. If a river flows along the boundary of the two regions, the resulting edges may support both boundaries and the river. Furthermore, the river may mark the outline of an industrial zone. Should the course of the river be corrected, it is likely that we want the topology of the industrial zone to be modified accordingly, while the administrative boundaries are likely to remain unchanged. It remains up to the user to decide which strategy should be adhered to.

The idea might come up that our model only cares about the consistency of topology, while it encourages redundancies in thematics; this is not so: for a specific application, two or more different thematic layers can always be combined by evaluating all intersections and forming a new layer. As long as there is no user requirement to compare the topology of two different utility networks (e.g. water and electricity), it is better to store them separately, since consistency of long-term storage data can be guaranteed more easily if the entanglement of data by internal cross references is kept as low as possible.

The above example of the river sharing edges with a number of administrative regions raises another problem: if the course of the river is straightened and the boundaries do not follow the new course, fragments are created which may be disconnected. Since regions must be coherent in our model, there is a need to introduce higher levels of thematic structures which we call *objects*. An object may consist of several elements sharing the same thematics that we may want to manipulate as an entity. The disconnected fragments of a formerly coherent region are only one example for an object. Such an object may also consist of all houses in a city block, or we may assemble all water supply lines of a particular city street to an object.

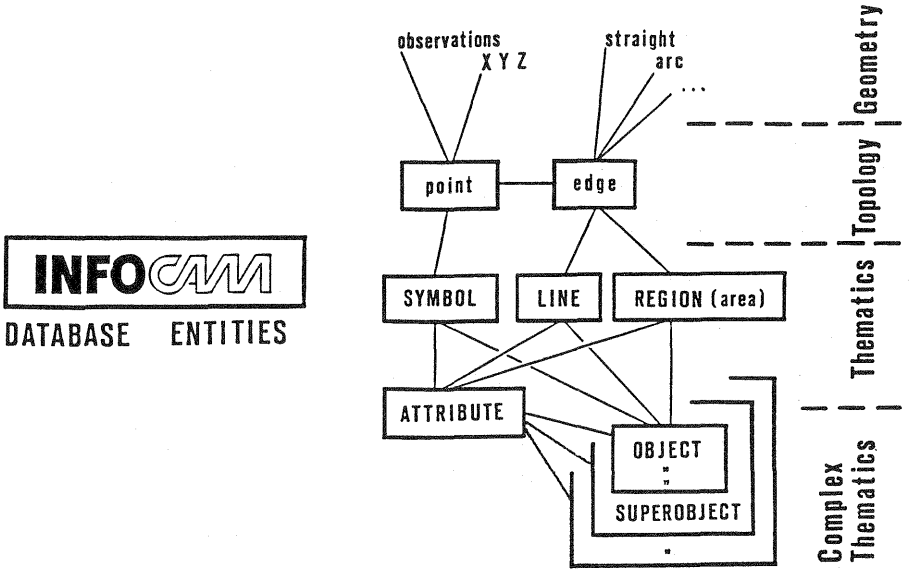

Fig. 3: DATABASE ENTITIES

There are no limits set to phantasy with regard to the assembly process, but in reality, we must observe physical storage constraints, and we must also keep an eye on the clarity of the user data model. Imposing restrictions on the size and also on the type of elements that may be combined to an object may help the user to organize his data and to keep them from getting accidentally contaminated; of course, he should be able to state the restrictions (the *thematic hierarchy*) on his own behalf. Just like the consistency restrictions, they are part of the *project description* that he has to set up before he can enter any data into a new project.

We have mentioned before that thematic aspects can be modelled in great detail by allowing attributes to be assigned to thematic structures (elements and objects). The statement that a line describes a river already constitutes such an assignment. It serves as a *frame* for other attributes (*slots*) which are more detailed. For the frame "river", such a slot may be the degree of pollution. A particular line belonging to the category "river" may have values (*instances*) for some or all of the slots. When drawing a map, such a value may be displayed (e.g. the name of the river appears on the map). Again, it is up to the user to set up frames and slots and to state rules that must be obeyed by the instances.

3. Database Mapping of Topological and Thematic Structures

In the preceding section, we have defined the topological entities *point* and *edge* and the thematic entities *symbol*, *line*, *region*, and *object*. From a naive point of view, we could maintain that this, together with the rules that we stated (e.g. "a symbol consists of one or several points") is essentially all that a database manager needs to know when adapting a general purpose database to our requirements. Such a

database would free us from any considerations with regard to long-term storage, data security and data consistency, multi-user problems, different user views, workstation clustering, networking etc. (See Grill 1982.) There is an ever-growing supply of commercially available databases (most of them of the relational type) that - justifiably - claim to fulfill these requirements.

Unfortunately, there is a number of drawbacks to this solution:

- traditional DBMS (database management systems) originate from applications which concentrate on explicit and well-defined relations between entities. The queries that we use in space-related applications commonly address implicit relations; as an example, we want to identify a point by specifying approximate coordinates; we might also be interested in all entities from a certain neighbourhood, e.g. all residential areas that are "near" a proposed factory site; sometimes, this concept of vicinity is vague; it cannot be expressed in terms of m or km. Conventional DB cannot model such topological relations in a way that is generally satisfactory.

- traditional DB model discrete relationships, while spatial relationships are continuous: a point in a plane has infinitely many neighbours - although computer precision limits this to a finite number.

- one of the greatest assets of DB, the preserving of consistency in long-term storage, proves to be a handicap in a highly interactive system when consistency checks slow down response time. Also, the user may want to perform *scratch-pad operations* (e.g. when adding last-minute "touches" to a plan for the final plotter output), that may needlessly burden the DBMS.

- geoprocessing transactions tend to be lengthy (requiring minutes and hours of computer time); in this case, DB management techniques would not work effectively.

- we must cope with fields whose lengths vary to a great extent. As an example, an edge that supports a utility line might consist of just one starting point and one end point, while another edge belonging to a contour line may contain a large bulk of data points. Traditional DB in general do not allow for variable-length fields, and if they do, they mostly treat the contents of such a field as an un-differentiated bulk, i.e. the contents of the bulk cannot be interpreted by the DBMS, but only by the application program; this means that the benefits of DB security must be given up when modifiying these bulk data.

These considerations lead to the conclusion that we cannot use one overall database that satisfies long-term storage *and* interactive performance requirements. We therefore leave long-term storage to the responsibility of a DBMS, while for

interactive sessions we advocate a local data structure that is custom-tailored to the needs of interactive graphics. In the next paragraph we shall outline how these two data structures interlock.

4. Interlocking long-term database storage and local data structures; the KERN-INFOCAM approach

We are maintaining two data structures: for long-term storage, we use a relational DB (RDB); currently, ORACLE 5.x on a DEC MicroVAX II is in use. For local storage, we use the *gridfile* (see Hinrichs and Nievergelt 1983). This data structure is ideally suited to meet the requirements of access to spatial data and interactive graphics. The domain of interest is dynamically divided into rectangles that correspond to physical storage units (*pages*), thus minimizing disk I/O operations. The size of the rectangles depends on the information density. Small rectangles correspond to densely populated areas, large rectangles appear in sparsely settled areas. The dynamical subdivision process corresponds to a tree structure. From the numerous possibilities, we chose a KD-tree that alternatively splits pages along the x- and the y-direction (see Matsuyama et al). The *data domain* and its subdivision into pages is complemented by the *address domain* that is mapped to the *gridfile directory*, i.e. to a pair of address vectors holding the addresses of partitioning lines and a matrix containing page addresses.

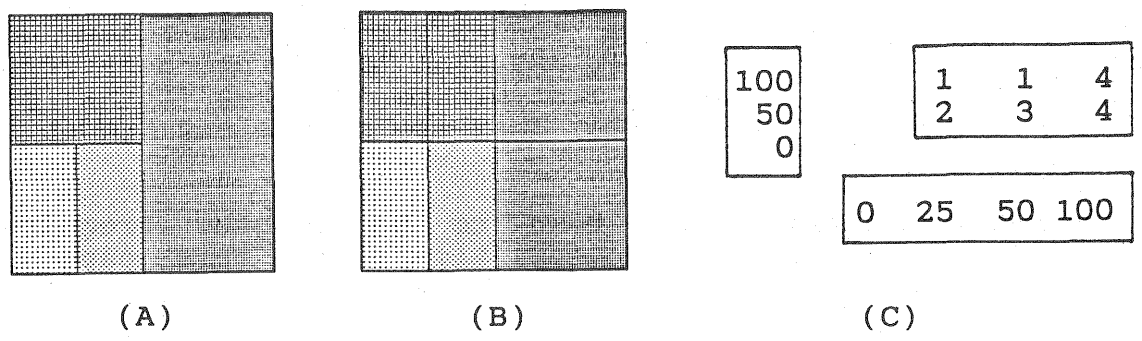

Fig. 4: Data domain (A), address domain (B), and gridfile directory (C)
(example for a range 0-100 in x and y, four pages)

The position x & y serves as a key to the data. This implies that each data entity (point, edge, element, object etc.) is referenced by such an x & y - identifier. Spatial searches in the data structure can easily be performed by comparing the search criterion with the address vectors. The corresponding pages can be retrieved, and local search procedures quickly yield the resulting identifiers.

The gridfile can be used as a storage medium, but just as well as a supplementary index for data stored elsewhere. The dimensions of the address space can also be extended to include

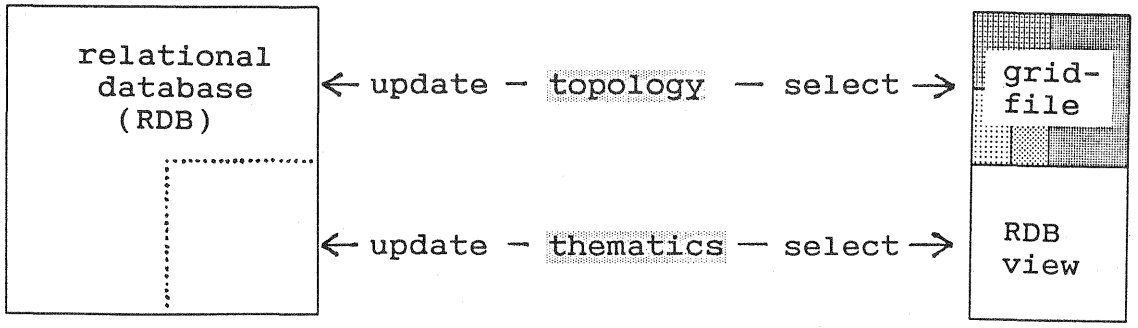

Fig. 5: RDB, local gridfile and local view;
Selection and update

We could only convey a rough idea of the workings of the INFOCAM data management. Everybody who has worked in this field will know that, regardless of the concept that is advocated, there are quite a few detail problems that must be solved during implementation; there is not enough room here (and also no need) to linger on these details. Comparing our solution to several alternatives, we argue that this transaction concept (i.e. updating from time to time and working off-line in-between) not only yields an equilibrium between the two opposite aspects of long-term data consistency and interactive performance, but it also conforms to the ways that people have always followed when facing the problem of splitting a large task into small manageable units and assigning the responsibility for local handling to local authorities, with occasional feedbacks and updates of the overall concept.

References

N. Bartelme, W. Kolb, K. Seifter-Bartsch, B. Spaeni: IMAGE - Interaktive Manipulation von Geo-Elementen. Zeitschrift fuer Vermessungswesen, 111. Jahrgang, Heft 6, Juni 1986, pp. 247-257.

A. Frank: Data Structures for Land Information Systems - Semantical, Topological and Spatial Relations in Data of Geo-Sciences. Ph. Thesis, ETH Zurich, 1983.

E. Grill: Relationale Datenbanken. CW-Edition, Munich, 1982.

K. Hinrichs, J. Nievergelt: The Grid File: A Data Structure Designed to Support Proximity Queries on Spatial Objects. Institut fuer Informatik, ETH Zurich, 1983.

INFOCAM product information: Kern & Co. AG, Aarau, 1988.

T. Matsuyama, L. Hao, M. Nagao: A File Organization for Geographic Information Systems Based on Spatial Proximity. Dept. of Electrical Engineering, Kyoto.

search criteria other than position. We use a third dimension to handle the spatial extent of lines, regions, objects etc. The values along the third axis of address space thus correspond to the radii of circumscribing disks for our elements. This makes it possible to find quickly all data entities that intersect with a specified area of interest.

The connection between the RDB and the gridfile mainly runs along two paths: when the user starts an interactive project session, a *selection process* is initiated. A spatial window must be specified, as well as the thematic layers that are required. (Access privileges for different users can easily be taken care of by the RDB.) The corresponding data are retrieved from the RDB, and a local gridfile is created. We want to stress the fact that not all data are *physically* copied to the gridfile, since its main purpose is the facilitation of spatial queries. Instead, we can take advantage of the RDB facility of creating *views*. A view is a (logical rather than physical) collection of data that is made available to a certain application. For the user, it creates the impression as if he was working on the original data. It responds to all facets of relational algebra, and all the amenities of database security and multi-user environment are preserved.

In our approach, the geometry and the topology of the data that satisfy the selection criteria are copied to the gridfile, as well as part of thematics (i.e. the hierarchy and the "main" thematic attribute that stands for the layer which contains the thematic entity), while all the other - more detailed - attribute informations are contained in the RDB view. (see fig. 5). For simplicity reasons, we refrain from going into any more details.

After the selection process has been terminated, the user goes "off-line" as far as the geometry and topology of "his" data are concerned. Many graphic applications can be performed without having to consult thematic attributes. When it comes to querying these attributes, he still has all the amenities of relational algebra, but these data are "off-line", too: if a user B changes the original data, this is not reported to user A. The entire interactive project session can be seen as a database *transaction*. All local gridfile update operations are kept in a logfile. When terminating the interactive session, an *update procedure* is initiated that processes all logfile entries. In case that any inconsistencies come up, the updating is rejected and the user is informed on the cause of the rejection. The updating of the local database view runs exactly along the same lines, the RDB taking care of this. Let us add two remarks: first, there is not always a need of updating; in the long run, "read-only" operations will constitute the overwhelming part of transactions in a spatial information system. And second, we have not mentioned yet that, apart from the selection and update with regard to our gridfile, INFOCAM also allows (read-only) queries that work directly on the original database.