GEOGRAPHIC DATA STRUCTURES

Hakan SARBANOGLU

1. INTRODUCTION

The information stored in a conventional database are the attribute values belonging to certain entity occurences. Entity types and required attributes for each of these entity types are carefuly determined by following a systems analysis and design methodology. During the design study, data definition specifications (data type, lenght, validity constraints etc.) are decided as well.

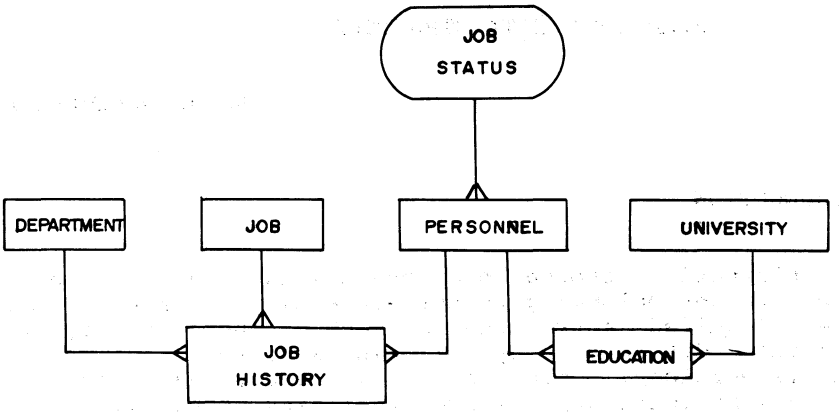
For instance:

Entity : Employee

Attributes:

Attribute name	Data type	Length	Validity constraint
employee no.	character	6	
name	character	10	
surname	character	14	
sex	character	1	M or F
address	character	50	
city	character	20	
postal-code	character	5	
date-of-birth	character	11	

There are various relationships between the real world entities; likewise these relationships should be represented in the database. Which relations between which entities are determined also during the analysis and design study by taking the users' access path requirements. Entity-relationship (E-R) Diagrams or Logical Data Structures (LDS) are the ways of illustrating these relationships. As an example, the general data structure of a personnel database is shown by the LDS on figure 1.



: one to one relationship

: one to many relationship

: many to many relationship

Figure-1 : Logical data structure of a personnel database

Each box in this chart represents an entity about which information are stored. Connecting lines show relationships between entities. The relationships are classified into three types according to whether one or many occurences (records) of an entity is/are related to one or many occurences (records) of the other entity. For example, "there are many employees working in a department but each employee works only for a department" statement indicates a one to many relationship between department and personnel entities.

Logical structures of databases are modelled in several types. A data model consists of two components :

- Dynamic structures : processes to be applied to data

- Static structures : underlying logical data structure of the environment.

Therefore designing these two components often called as data modelling. A data model dominates many features of a database such as physical storage, access paths etc. Consequently each database management software supports a certain database model.

Most common database models are :

- Hierarchical model,
- Network model (CODASYL),
- Interved list model and
- Relational model.

Databases technology, as described very briefly above, has reached a satisfactory level today and constitutes the fundementals of information systems.

The most important difference of a geographic database when compared to a conventional one is that the positional (or spatial) information are stored in addition to the other attributes. For this reason the information stored in a geographic database are groupped into spatial and non-spatial (or graphical and non-graphical) data.

Spatial (or graphical) attributes of an entity define :

- the position of the entity with respect to a reference system.

and

- the shape of the entity.

Such kind of entities are called as "geographic entities". Perhaps because of the recent developments on object oriented systems approach, the term "object" has become to be used more frequently instead of the term "entity".

2. CHANGING CONCEPTS, NEW DEFINITIONS

Computer Aided Digitizing and Drawing (CADD) systems and then the Geographic Information Systems (GIS) have brought many new concepts with them and new definitions have emerged in mapping terminology. The following is a brief list of those new concepts related to geographic data structures :

2.1 Entities or Objects

In database terminology the words "entity" or "object" are used corresponding to the term "feature" in mapping terminology

2.2. Three Dimensional Database

Each feature in the real world are 3 dimensional objects besides few exceptions. 3 dimensional database concept can be interpreted in different ways. From a viewpoint, 3 dimensional database is a database in which the shapes of features are represented three dimensionally. From another viewpoint it is a database in which the positions of the features are represented three dimensionally with the x,y,z or $\lambda \varphi h$ coordinates. Because of the limitations on hardware and especially software systems today, it is impractical to store the shapes of the features in 3 dimensions. On the other hand, it may seem feasible to store the 3 dimensional positions. This is realised either to store horizontal coordinates only but enterpolate the 3 rd coordinate value from a digital elevation model.

2.3. Scale

Since the shapes of the features can not be captured easily, features (geographic objects) are projected onto a reference surface and the frame of the projected shape is digitized. This digitisation process has got a precision limit in its nature. Let's take a spherical petrollium tank for example. Although this tank has a 3 dimensional shape and even if the projection of this tank would be a 2 dimensional shape (circle) it is generally digitized as a single point. Furthermore a number of very close tanks could be generalised to a single point.

One can claim that there exists infinite number of features to be digitized on the earths surface.Therefore an "abstraction" or "generalisation" is to be considered. Since the availabe generalisation algorithms are not so efficient, it is not possible to implement the "Scale free databases" today. In other words, the existing features are abstracted or selected considering a defined requirement and then these selected features are converted to digital form. For this reason a level of detail is considered both for the representation and the density of the features. Accordingly we talk about the scales of the geographic databases althoug it doesn't have a graphical meaning. This "scale" concept is very different from the classical meaning and is an expression of the feature density and the level of abstraction in the database.

2.4. Raster and Vector Techniques

The human eye is highly capable of recognizing the shapes. Whereas many different approaches are necessary to represent these shapes in computer memory and storage devices. There are generally two different but complementary methods to represent the shapes in computers: raster and vector representation.

2.4.1. Raster Technique

For the purpose of raster representation, a grid net is thought to be overlaid on the shape. Each square shaped cell of this grid net is called as "a pixel". What is included in each cell is stored using certain codes (letters or numbers). In other words the grid net is considered as a matrix such that each element of this matrix corresponds a cell and carries a code indicating "what is in that cell". Let's take a building on a soil parcel for example, when soil pixels are coded with the letters "S" and the building pixels are coded with the letters "B", this information (shapes of building and soil) are represented with a matrix whose elements are either B or S in the computer. (Figure-2)

The number of different codes required should be at least as many as the number of distinguishable map features or objects. Consequently, to store these different codes, sufficient number of bits should be allocated for each pixel. If, for example 8 bits are dedicated for each of the pixels, it would be possible to store 2^8=256 different features with raster coding.

Raster representation of the shapes does not apply only to main memory and tape/disk files. Raster technique is also a method of graphic data capture and graphic data output. When capturing the graphic data by raster technique, each point on a map, corresponding to a pixel, is coded with a value due to the color or grey tone of this point as sensed by a CCD (Charged Coupled Device) type sensor. These type of devices using raster technique are called as raster scanners. In fact the satellite sensors works with the same principle too.

Although such intelligent raster scanners recognizing what a feature on a map is, in real time mode, are not available today; the developments in artifical intelligence and pattern recognition are supposed to realise this in near future.

For the graphic output by raster technique, an arrangement is followed stating which colors or symbols would be used to visualize coded pixels on screens or plotters. The examples of such raster output devices include colored raster refresh screens, matrix printers, ink jet plotters and electrostatic plotters.

It is assumed that each pixel (or cell) is wholly covered with the coded feature in raster representation. Accordingly the precision of representation is inverse proportional to the pixel size. When the pixel size gets bigger, more problems may occur in representing small features and curved boundaries. (Figure-3)

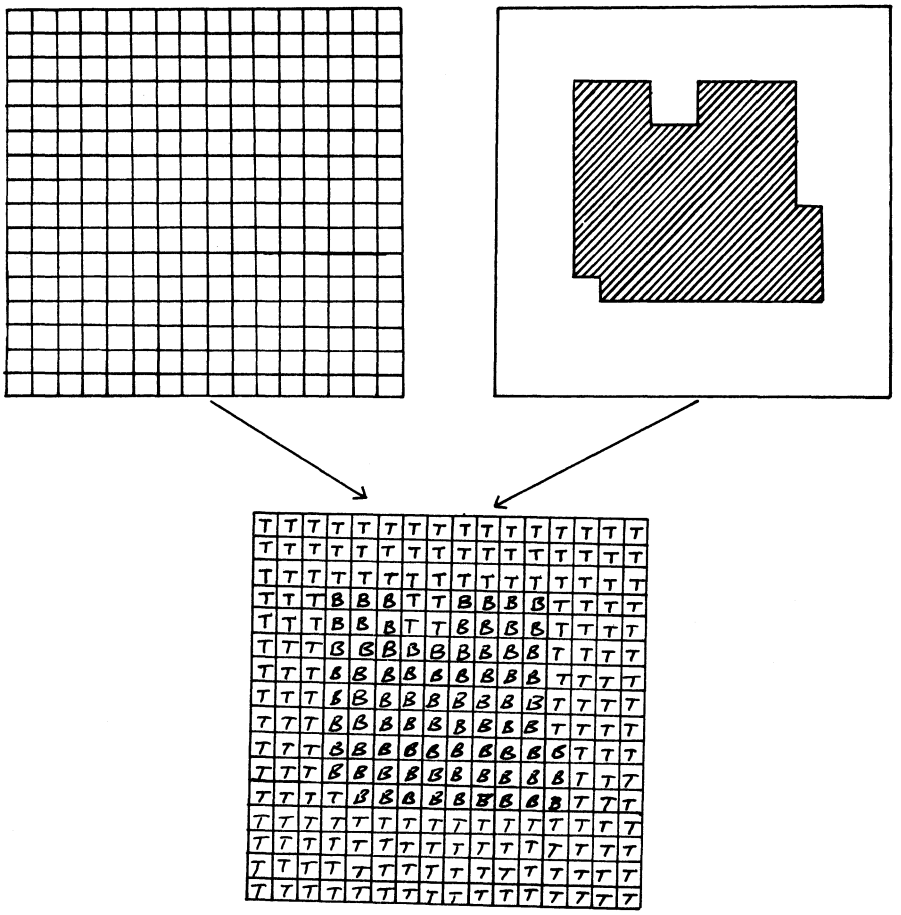


Figure-2 : Raster representation

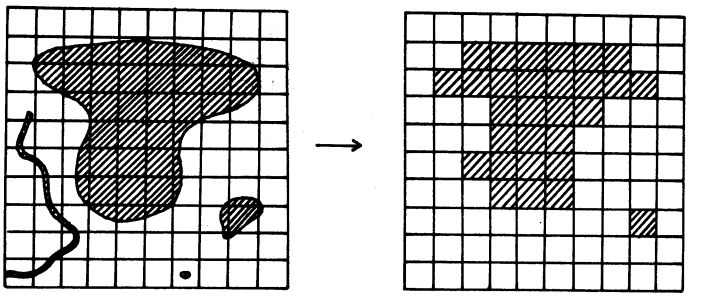


Figure-3: The precision depending on the pixel size in the raster representation

For this reason, to scan line maps accurately, it is essential to take a pixel size smaller than the minimum line width.

2.4.2. Vector Technique

Vector technique classifies the map features into three categories: point features, line features and area features. Some features are relatively small due the scale and these are represented only with a point. These features are called as "point features." Even the same type of features are so close to each other (like pylon groups) that they must be generalized and represented with a single point.

Again because of the scale, the geometric shapes of some features may require themselves to be represented with lines. The examples of "line features" include roads, rivers, long bridges etc.

Wheras the stored point corresponds to the geometrical center for a point feature, the stored line corresponds to the central line (road center line, river center line, etc.) for a line feature.

All the other features which are not so small to be represented as point features or not so thin (of course due to the scale) to be represented as line features are taken as area features.

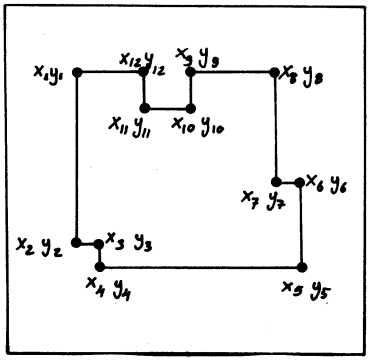


Figure-4 : Vector representation

Another way of obtaining a vectoral structured graphical file is to convert a raster file to vector. After scanning which is usually done with run length encoding, the raster to vector conversion process consists of two phases. The first phase is called as thinning in which the pixel bands are reduced to pixel strings (Figure a,b,c)

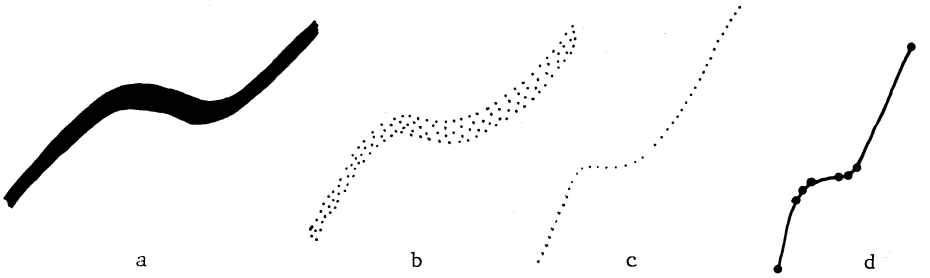


Figure-5 : Raster to Vector conversion

In the second phase the unnecessary pixel points on this string which lies on the same line in a tolerance are removed, so only the best representing points are kept. This phase is called as weeding.

The most common implementations of data output with vector technique are pen plotters. A drawing head including pens can be moved along an axis on a bar and this axis bar can also be moved along the other axis. There exists mechanical capabilities such as to pen-up, to pen-down, to move the head to a point defined with (x,y) table coordinates. An alternative to these flat-bed pen plotters is drum plotters which are able to move the paper platform on a drum instead of moving the axis bar. Another type of vectoral graphical data presentation devices is the monocolor vector screen. These devices have phosphoric screens as the displaying platform and uses a beam instead of the pen.

When a configuration is to be designed to produce and use digital geographic information after an analysis study, it should be carefully decided that which of the raster and vector techniques is to be used. But it is not necessary to use only one of these techniques throughout the whole configuration. A suitable combination can serve the purpose best. For example, whereas the graphical files are stored in vectoral structure on disks, they can be plotted with an electrostatic plotter which is a raster device. The necessary conversion is accomplished by the software. For this reason the raster-to-vector and vector-to-raster conversion software packages are included in such systems.

2.5. Attributes

In addition to the graphical information in raster or vector form representing the positions and shapes of the objects, related non-graphic informations are also stored in geographic databases. This non-graphic data consists of only height value for the contours and spot height points in a 2-D file. "Which attributes are to be stored for each of the features" is a matter of database design. For example the attributes for a forest area feature could be the tree type, tree density, average age and annual productivity. Or for a line type feature, say a road, the surface type, width and number of lanes can be selected as attributes.

We should notice an important principle at this point. The geometrical specifications of a feature which can not been represented with its graphical data must be recorded as non-graphic attributes. For example if a building is digitized as a point feature, because of the scale, then its lenght, width, height and orientation must be recorded as attributes. On the other hand, if a building is digitized as an area feature then only the height is required to be recorded with the other non-graphic attributes. Similary if a road or a long bridge is represented as a line feature then the width is taken as an attribute.

2.6 Geographic Cell

A graphical file includes the contours and features within a specified frame, so it can be thought as a "digital sheet". But it is possible to obtain a continuous database by removing the sheet boundaries after an edgematching process. But yet, it is required to divide this continuous database into parts in order to ease the data management and access functions. These divisions which are generally defined with geographic latitude and longitude lines are called as "geographic cells" or "geographic rectangles" or "tiles".

2.7. Feature type, feature code, feature number

In a vector database, the feature type indicates whether the feature is a point or line or area (polygon) type feature.

A feature code shows that "what" the feature is. Therefore a feature coding catalog is required for each database.

On the other hand, a feature number shows that "which" feature it is. The feature number uniquely identifies that feature within the other features of the same or different coded features in a geographic cell.

2.8. Feature Component

The attribute values must be valid throughout the all feature which can be an area or line type. But let's take a road for example; if the number of lanes varies along the road and changes from 2 to 3 then to 2 again, each part should be considered seperately. (Figure-6)

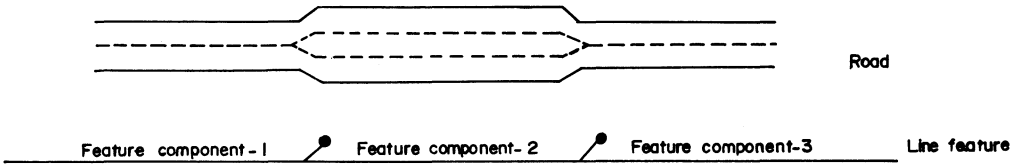


Figure-6 : Feature components

In the light of this thought, the previous definitions should be developped some more. In this case the feature can be thought of as being a thematic concept and can be considered as being consisted of geometrical feature components. The components of a feature are shown on Figure-7. On this figure a river has got three components the second of which is an area feature.

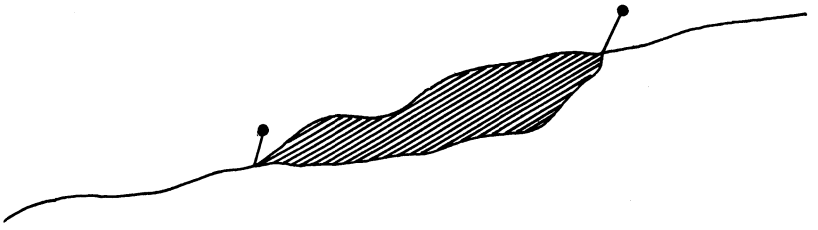


Figure-7 : Feature components in different types

There are of course many other new concepts and definitions which have been brought by the GIS and AM technologies. In this paper only some of those new concepts have been tried to be defined in order to ease the explanations in the below 3rd and 4th items.

3. RASTER DATA STRUCTURES

The simplest data structure for the raster technique whose basic principles were explained briefly in the paragraph 2.4 is the matrix schema built up with rows and columns. Although the matrix structure is very suitable for FORTRAN programs, it results some limitations and problems. The followings are valid for matrix structures :

a. The geographical information lay down on a two dimensional countable (discrete, not continuous) surface. So, due to the pixel size some depictions may occur when we calculate the lengths and surface areas. For example, the hypotenuse of the triangle on Figure-8 is 7 if we count pixel sides, but 4 if we count the pixels. Similary the area of the same triangle is 7 square pixel whereas the correct value is 6 square pixel. The only way to eliminate this problem either on calculations and on displays, is to take the pixel size as small as possible. Today the screen and plotter technology has reached a satisfactory level for this purpose.

b. Each matrix element (pixel or grid square) can be assigned to only one value. For this reason it is not possible to store more than one geographic data belonging to the same point such as soil type, underground water level and vegetation values at the same point.

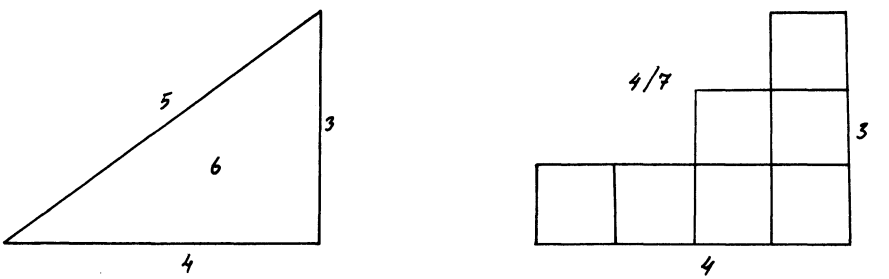


Figure-8 : The erroneous results in surface and lenght calculations due to pixel counting in raster coding.

In order to eliminate this problem, the map layer idea has been developped. In this approach a map sheet is represented with a stack of matrices. Each matrix includes features within a certain class. Thus for example topography, hydrography, transportation, settlement, energy, communication, soil type etc. could be layers of a map. (Figure-9)

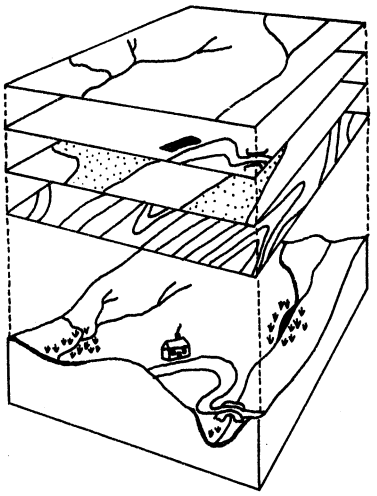


Figure-9 : Map layer concept

When each layer is represented with a matrix in computer memory or storage, the volume of storage requirement may become a new problem. Assuming for example that we represent a 50X70 cm. map sheet with 0.1 X 0.1 mm. pixels, there will be 35 million elements. If we use 1 bit (1 or 0) for a pixel, we will require 4.5 MBytes; if we use 8 bit (256 different feature codes) for a pixel, then we will require 35 MBytes storage capacity for only one layer. When we multiply these figures with the number of layers, the storage requirements would reach an unmanagable volume for only a single map sheet.

In order to solve this problem, raster logical data structures and raster data storage methods have been developped.

3.1. Logical Data Structures for Raster Representation

In order to optimize the access path to data in a raster database, the geographic data can be organized in various ways.

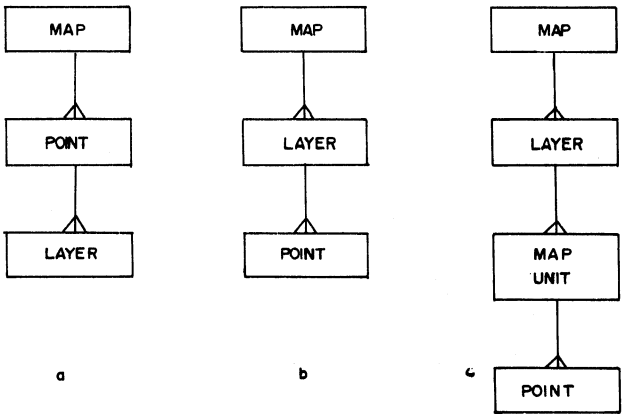


Figure-10 : Logical data structures for raster representation

In the first solution shown on Figure 10-a, each map file is thought to be consisted of map points and at each map point there is an array whose each element is dedicated to a layer. There are three different types of records in the database so, for each point including a geographical data a record with fields (x,y, layer no. ,z) is stored.

In the second solution a map file is divided into layers first. Then on each layer the points having a geographical data are recorded figure 10-b. Thus three different types of records are stored in the database. The point records includes the (map no, layer no, x,y and z) fields.

Another solution is based on the "map unit" concept. A map unit is a set of points having the same geographical data values (Figure 10-c). Accordingly four types of records are stored in the database. Map records include information specific to that map sheet such as digitizing date etc. The layer records include information on which map sheet this layer belongs to, layer name, scale etc. The map units are described in map unit records. Finally the points holding the same map units are stored with an indication to that map unit record.

In other words, the first two solutions tell "what is included at the points" but the third solution tells "where each map unit is included". (for ex. which points have a feature code indicating oak forest)

3.2. Raster Data Storing Methods

In order to encode a set of pixels (a map unit) on a layer, some compact methods have been developped to decrease storage requirements. Major techniques are briefed below:

3.2.1. Chain Codes Method

The boundary of the region to be encoded can be described by starting from and origin and giving a sequence of numbers of unit vectors in the cardinal direction. The cardinal directions are numbered as (0=east, 1=north, 2=west, 3=south) and the definition follows the clockwise direction. For example, if we start at cell row=8, column=2, the shaded region on Figure-11 can be coded by:

$$0,1,0,1^{2},0,3,0^{2},3^{3},0,3,0^{3},3,0^{2},1^{1},0^{2},3^{6},2^{7},1,2,3,\underline{2^{2},1,2^{1},1},2,1^{5}$$

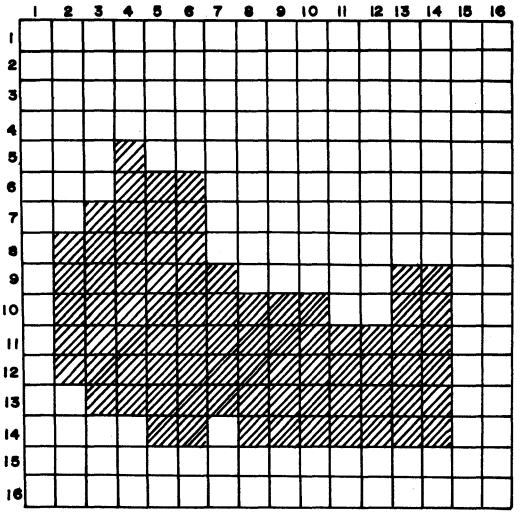


Figure-11 : Chain Codes

Considering that we would use 2 bits for each cardinal direction code 3 bits for each exponentiation, then 26x5 =130 bits =17 bytes are required to represent this shape. Whereas, if we used ordinary matrix schema with we must have used 16x16/18 = 32bytes.

Chain coding provides an effective way of saving storage and makes the perimeter and area calculations easy as well. But overlay operations such as union and intersection require to return to matrix schema presentation. Another disadvantage is to code the common boundaries of the adjacent regions twice.

3.2.2. Run-lenght Coding Method

Run-lenght encoding method takes a map unit row by row (or column by column). The pixels having the same values on a row is coded with the starting and ending column numbers.

So the same shape on Figure-11 can be run-lenght coded as :

Row 5	4,4	
Row 6	4,6	
Row 7	3,6	
Row 8	2,6	
Row 9	2,7	13,14
Row 10	2,10	13,14
Row 11	2,14	
Row 12	2,14	
Row 13	3,14	
Row 14	5,6	8,14

3.2.3. Block Codes Method

If we extend the run-lenght coding idea to 2-dimensional space we can obtain a method in which the square block having the same pixel values are coded. Each square block is coded with 3 numbers (the row and column numbers of the square origin that is the upper-left corner and the side of the square in pixel count). The square block should be selected as big as possible in order to reduce the storage requirement.

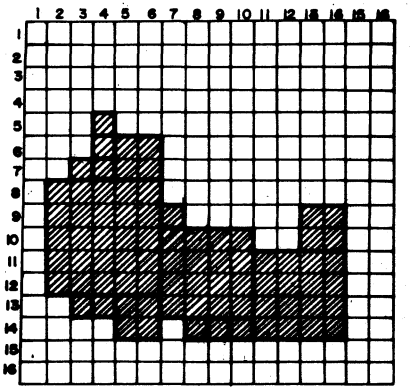


Figure-12 : Block Codes

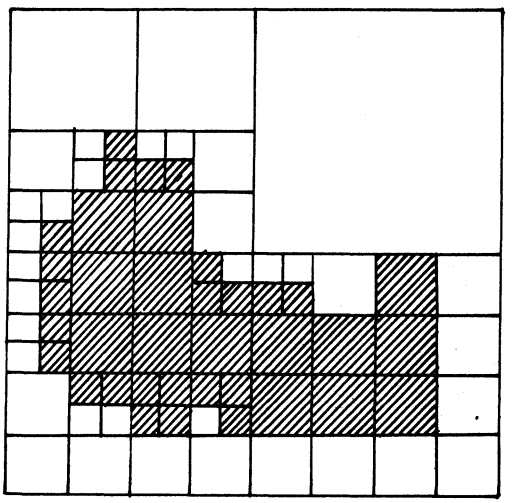
The same shape can be represented by 10 unit square, 3 2-square, 2 4-squares and 1 5-square. When 4 bits are used for each number a total of 16 X 3 X 4 /8= 24 bytes would be required to store this shape.

Block codes provide storage saving especially for coding large shapes generally found on coropleth type maps. It is also advantageous on union and intersection operations. Though it is still necessary to return to the original matrix schema for most of the other operations.

3.2.4. Quad Tree Structure

In this compact method a 2x2 dimensional array is succesively divided into quadrants. This step by step subdivision process is carried out till each subdivision is either emty at all or wholly covered with pixels. So a reverse tree structure whose every nodes have 4 branches is obtained. Notice that this is not a balanced tree. Each node is represented with 2 bits (11: terminator in, 00: terminator out, 10: node in, 01: node out).

This method requires 23 x 4/8 = 12 bytes to store the same shape. Since it is suitable for many algorithms, this method is widely used recently. Regionally variable resolution capability offers one more advantage.



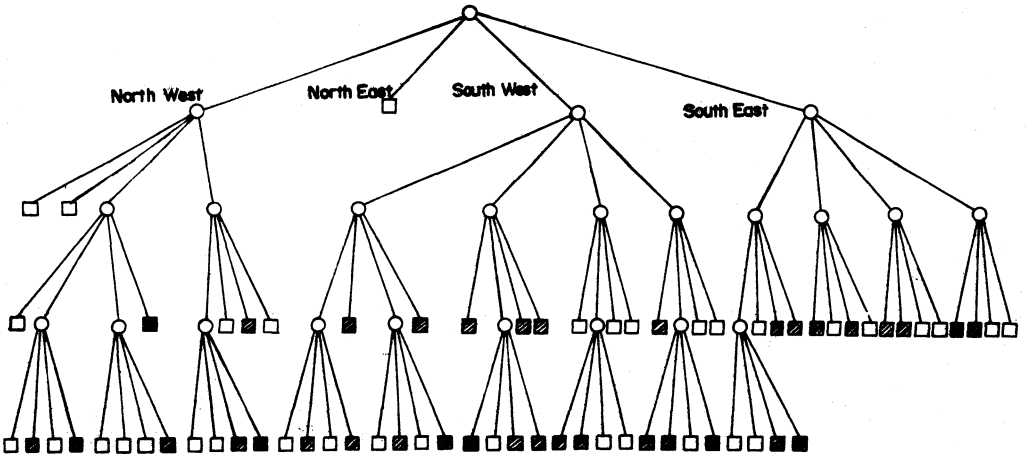


Figure-13 : Quad tree structure

4. VECTOR DATA STRUCTURES

The vectoral geographic structures will be explained in five stages ranging from primitive to more advanced types. Almost all of the data structure models used in today's software systems can be categorized into one of these five structure models, maybe with some little differences. The same symbolic map shown on figure-14 will be used throughout this section.

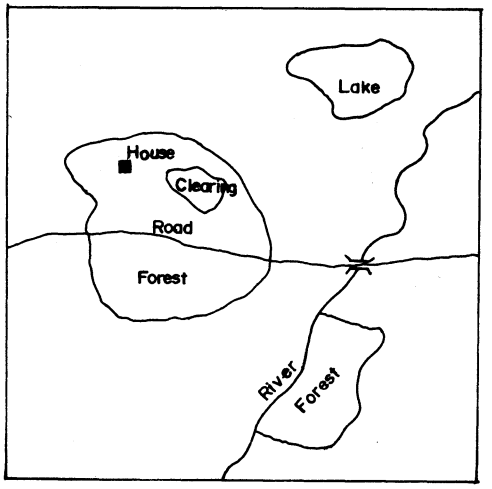


Figure: 14- Example line map

4.1. Unstructured Vector Data

Lets think that we have got a manual digitizing table. But our digitizing software doesn't have any structuring capability. We can tell the program only that we are beginning to digitize a new feature, but the software doesn't accept any feature code for this feature.

After we run the program, the output file is immediately opened and when we start to digitize, the program puts 1 for the feature number of the first feature; then it increases the feature number one by one at each time we tell the program that we are to start a new feature. If we digitize our sample map point by point (shown on figure-15) with this program, we would obtain a file like shown on Figure-16.

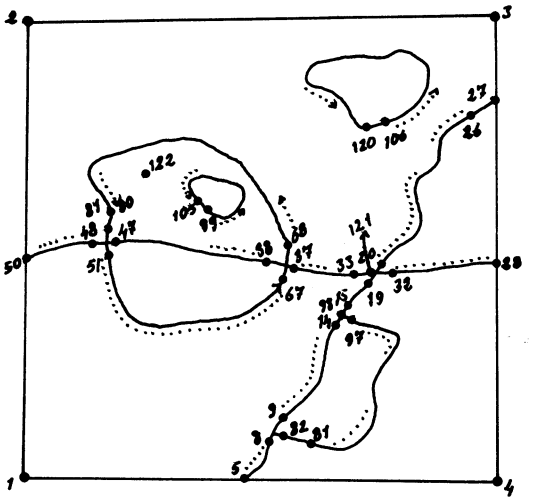


Figure-15 : Manual digitizing

1	1	
2	x_1	y_1
3	x_2	y_2
4	x_3	y_3
5	x_4	y_4
6	2	
7	x_5	y_5
.	.	.
.	.	.
.	.	.
29	x_{27}	y_{27}
30	3	
31	x_{28}	y_{28}
.	.	.
.	.	.
.	.	.
53	x_{50}	y_{50}
54	4	
	.	.
	.	.
126	x_{119}	y_{119}
127	x_{120}	y_{120}
128	8	
129	x_{121}	y_{121}
130	9	
131	x_{122}	y_{122}

Figure-16 : Unstructured geographic data file

We can claim that this file with this primitive structure is comletely useless except drawing as the line strings. A program that could be used for this drawing can draw each feature from the first to the last point. Since there are only coordinate pairs and feature numbers in the file and because the feature codes are not known; we can neither use different colors or symbols for different features nor we can determine "which feature is what?". Furthermore we cannot diffenentiate line and area feature types. For these reasons, the resulting monocolor display could be like the following (Figure-17).

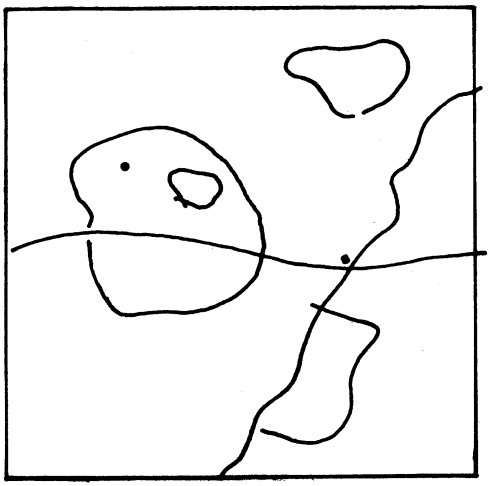


Figure-17: The drawing obtained from the unstructured data.

Problems

a. Area features are not closed (forest, clearing, lake),

b. There are overshoots or undershoots at junctions (road-frame, river-frame),

c. Points are not on the lines or at the intersections exactly (bridge at river-road intersection),

d. The features can not be distinguished.

The unstructured model, as far as we know, is not used at all today. But it is explained here just to define the problems. These problems which are often called as geometrical problems including closure errors, intersection errors and adjacency errors can not be removed with this unstructured data.

4.2. SPAGHETTI DATA STRUCTURE

The most important advancement in spaghetti data structure with respect to unstructured geographic data is the inclusion of the feature concept. While the unstructured graphical data is a collection of points and lines only, in the spaghetti structure the graphical data is considered as a collection of point, line and polygon features. Each feature is a whole and independent from each other. Asdjacent features or the features sharing one or more geometrical elements are digitized seperatelly so redundantly. In other words for the all area to area, area to line, line to line, line to point and point to point adjacencies, the shared geometrical element(s) are digitized more than once. For example, the river and the forest area on figure-14 are adjacent. They are sharing a line part. part. That line part is digitized once for the unstructured digitizing as shown on figure 18-a, but twice for spagetti structured digitizing as shown on figure 18-b.

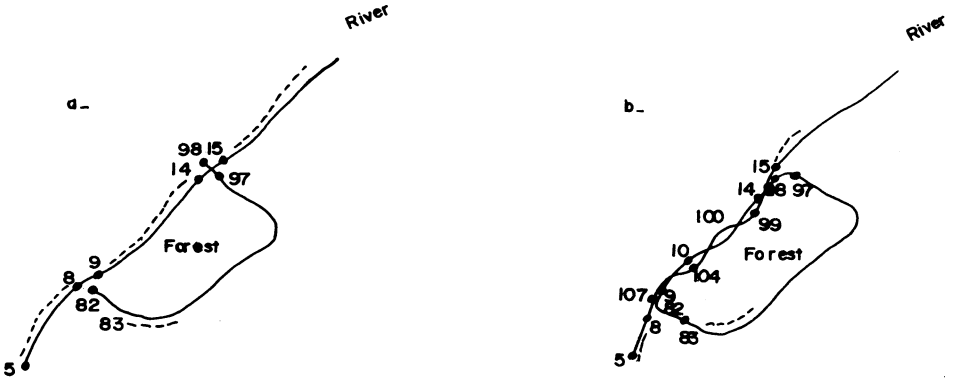


Figure-18 : Area to line adjacency in unstructured data and in spaghetti structure.

When our map is digitized with spaghetti structure with the points shown on Figure-19, the resulting file would be as shown on Figure-20

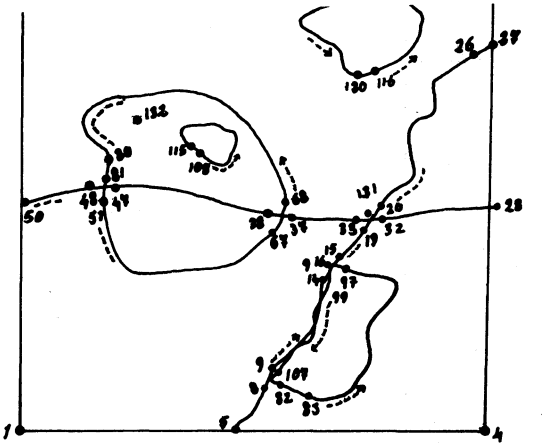


Figure-19 : Digitizing in sphagetti data structure

As can be seen on Figure-21, there are two record types in spaghetti structure. A feature header record is stored for each of the features. This record contains fields like feature number, feature code, feature type (area, line, point) and other attributes. These fields can be entered either during digitizing interactively or afterwards.

Solutions

Some problems occured in unstructured data have been solved in this structure :

a. We can know what each feature is, from their feature codes. Accordingly it is possible to use different colors or symbols on displays.

b. We explicitly know the feature types.

I	ÇRÇ	A
x_1	y_1	
x_2	y_2	
x_3	y_3	
x_4	y_4	
-999	-999	
2	AKR	Ç
x_5	y_5	
. .	. .	
x_{27}	y_{27}	
-999	-999	
3	Yol	Ç
x_{28}	y_{28}	
. .	. .	
x_{50}	y_{50}	
-999	-999	

x_{116}	y_{116}
. .	. .
x_{130}	y_{130}
-999	-999
8	KPR N
x_{131}	y_{131}
9	EV N
x_{132}	y_{132}

Figure-20 : Graphical file with spaghetti structure

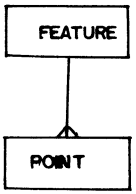


Figure-21 :Entity-Relationship diagram for the spaghetti data structure

So a plot derived from spaghetti data could be looking like shown on Figure-22.

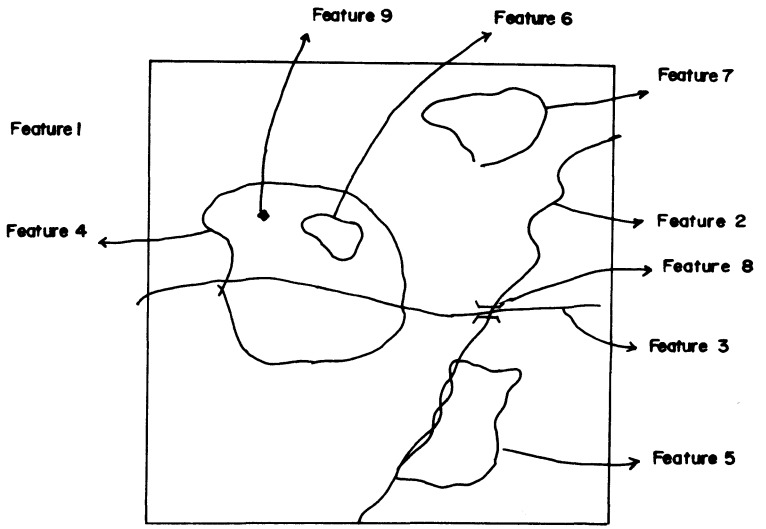


Figure-22 : A plot derived from spaghetti data

Problems :

a. Geometric problems :

1. Misclosure of area features (Figure-23)

Figure-23 : Misclosure problem on area features.

2. Since each feature has been digitized seperately, shared geometrical elements are stored in more than one record with different coordinates. Accordingly there occurs gaps and slivers on area to area and area to line adjacencies. Figure-24

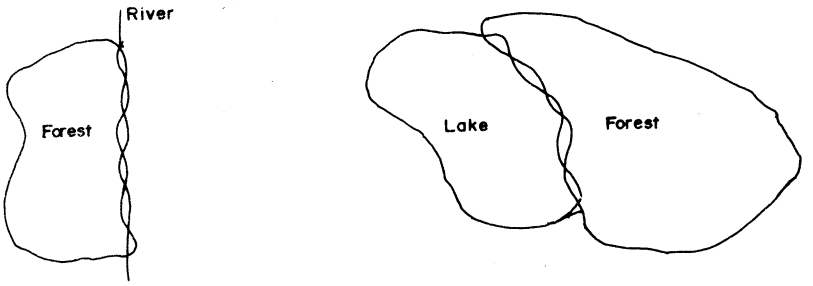


Figure-24 : Gaps and slivers

3. There exists overshoots or undershoots at juntion points. (Figure-25).

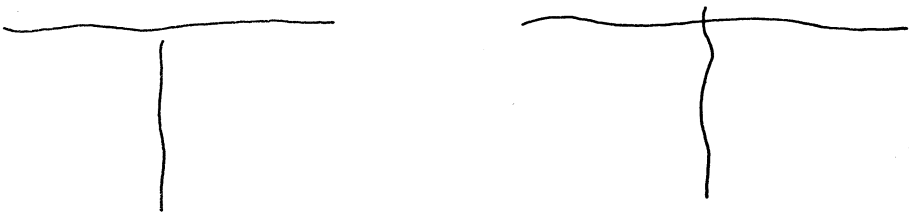


Figure-25: Overshoots and undershoots

4. The line and point features at an intersection don't join at the same point. In our example, since the bridge which must be at the river and road intersection exactly, has been digitized seperately, it may be at the intersection point only within the cartographic tolerance. (Figure-26)

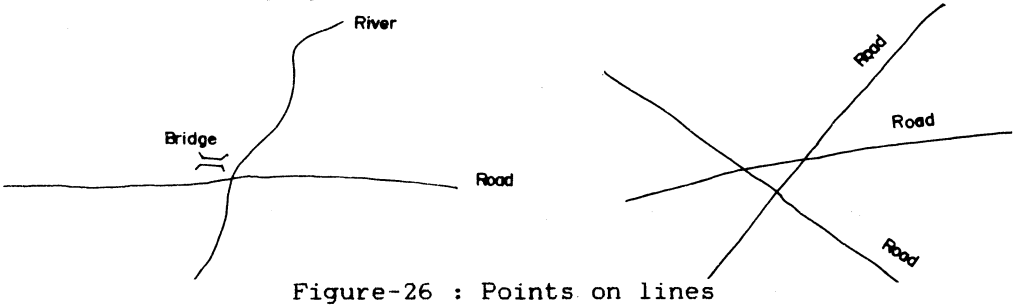


Figure-26 : Points on lines

Other problems

In addition to the above geometrical ones, there are more problems because that the structure is not "intelligent" enough:

1. It is not possible to define the junctions of branches of a river, for example, because connections haven't been explicity defined.

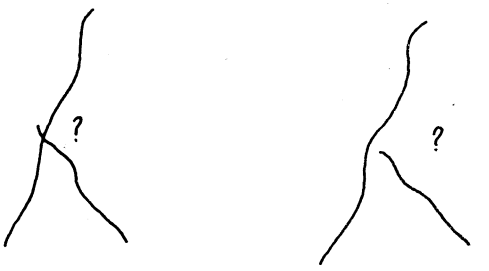


Figure-27 : Undefined junction and intersection

2. It is not possible to define the spatial relationships. For example, at an intersection of a highway and a railroad, which one is passing over is not defined. Figure-28

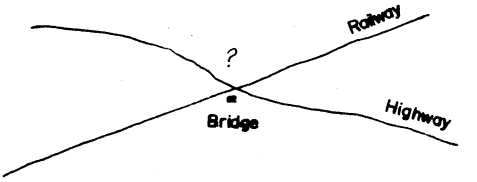


Figure-28 : Undefined spatial relationships

3. Coverage and containment problem

Area features have been digitized with their boundaries as a linear polygon. Then same questions like "Is the feature on the left or on the right with respect to the digitizing direction? ", "Are there any other features within this polygon?" arise.

In order to fix the first question, the boundaries of the area features are digitized anti clockwise direction. In other words, while digitizing, the area feature is left on the left.

But the second problem is more serious and causes to loose the contained features on the screen display. For example lets imagine that there is a clearing in a wooden area and that a road is passing through this forest and clearing (Figure-29). This road is also passing through a bridge in the woodland and there is also a house in the woods.

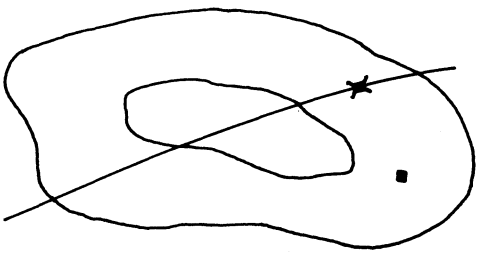


Figure-29 : Features contained in an area feature.

Let's think that we have digitized in the sequence of bridge, house, road, clearing and woodland. Then the record sequence would be the same in the spagetti structured file. Accordingly a plotting or displaying program would read and immediately displays or plots the records in the same sequence The bridge and the house would be displayed first, then when road is diplayed the pixels showing the bridge turns to the color of road, that is the house is wholly or partly disappears. Then when the clearing is displayed with relevant color (soil), the road would partly dissappears. Finally as soon as the woodland area boundary is drawn and filled up with green color, everything contained in the woodland area would disappears. (Figure-30)

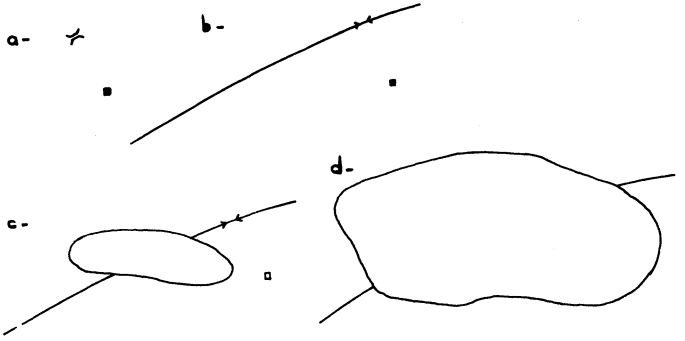


Figure-30 : Coverage problem on plotting or on displaying

In order to solve this coverage problem, a hierarchy is established between the features. Following this hierarchy the features are numbered and digitized in the order of area features (from outer to inner), line features and point features. Nevertheless this sequence which must be followed during digitizing adds up the operator's workload.

The spaghetti data structure together with all those problems has been most commonly used and still being used today, by software systems. The IGDS from Intergraph, IFF from Laser-Scan and many others are the examples of this structure.

4.3. LINK-NODE DATA STRUCTURE

This geographic data structure which is also called with different names such as edge-node or arc-node, has been designed to eliminate the problems seen in spaghetti structure.

Node is a point and represented with a coordinate pair. In this context, a point feature, the beginning and end points of a line feature, an intersection point or a junction point is defined as a node.

Link, on the other hand, is a string of points (coordinate pairs) lying between two nodes. The both end points are not included in the link.

Using these definitions of link-node structure, various geometric elements can be expressed as the followings :

point feature = node

line feature = chain = node + link + node + link +.....
.....+ node + link + node

area feature = chain
or
area feature = outer chain + inner chains

There are two basic ways of creating a link-node structured graphical file. In the first way the digitizing is done with spaghetti structure and then this spagetti structured file is converted (upgraded) to a link-node file by a batch program. In the second way the link-node structured file is created as the digitizing goes on, by calculating the nodes in real-time mode. For the both ways, the feature coding is carried out afterwards by interactive editing and tagging.

There are generally three different record types in a link-node structured graphical file. Those are fixed-lenght node records, variable lenght link records and finally the feature records. The feature records which are also variable in lenght, contains identifiers of link and nodes constituting that feature.

Figure-31 illustrates the link-node form of our example map.

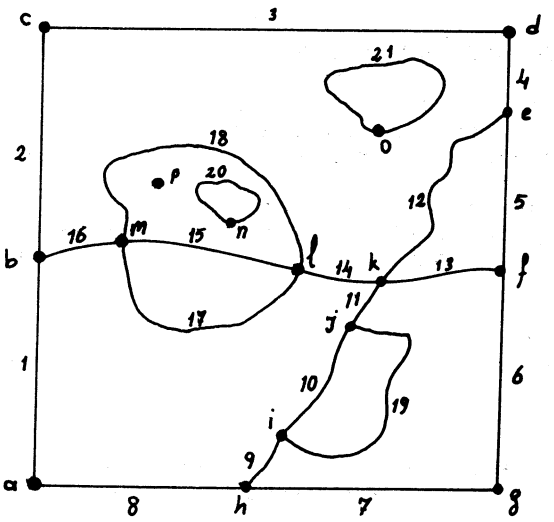


Figure-31 : Link-node data structure

The graphical file representing this map would look like the one shown on Figure-32. The logical data structure showing the entities and relationships in this structure is on Figure-33.

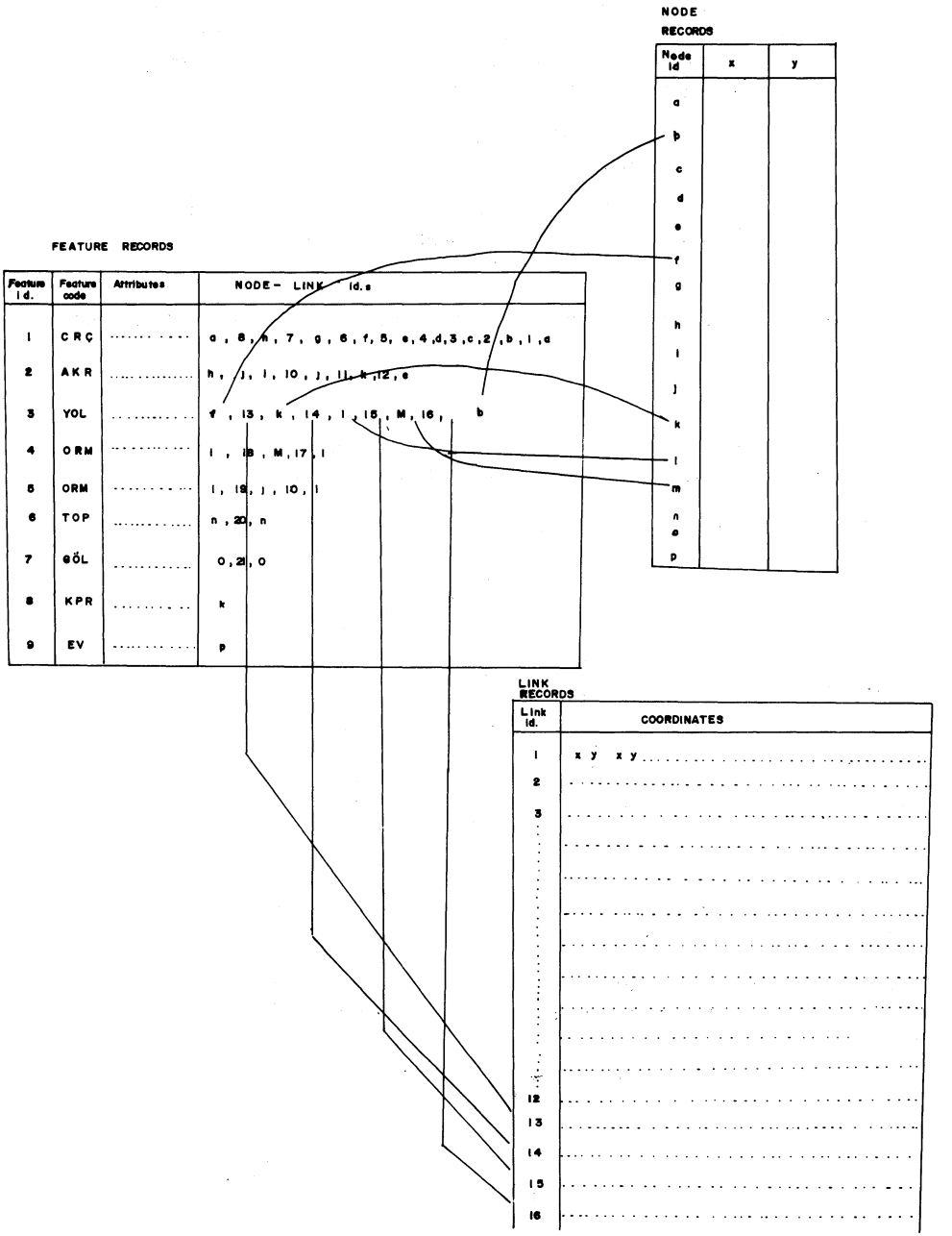


Figure-32 : Link-node structured graphical file

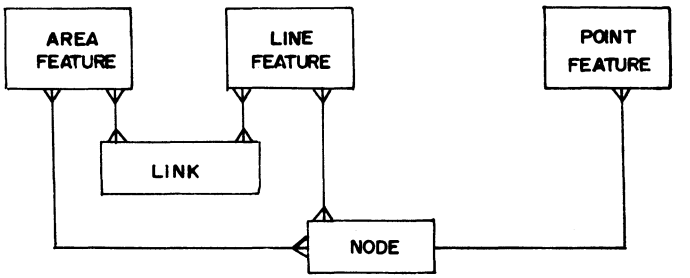


Figure-33 : Link-node logical data structure diagram

Solutions :

Link-node structure solves many of the problems, especially the geometrical ones, occured in spaghetti data structure.

a. No duplicate digitizing. So the gaps and slivers don't exist anymore. (Figure-34)

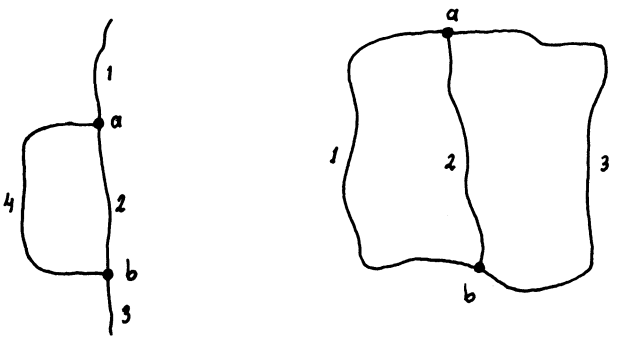


Figure-34 : Area to area and area to line adjacencies in the link-node data structure.

b. The overshoots and undershoots are eleminated at the junction and intersection points. (Figure-35)

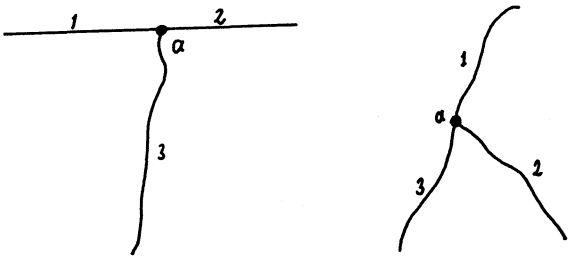


Figure-35 : Junction and intersection points in the link-node structure.

c. Area features are closed without any problem (Figure-36)

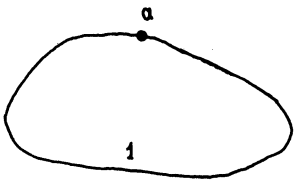


Figure-36 : Area features in link-node data structure

d. The points which must be on a line or at an intersection point are at their exact locations (Figure-37)

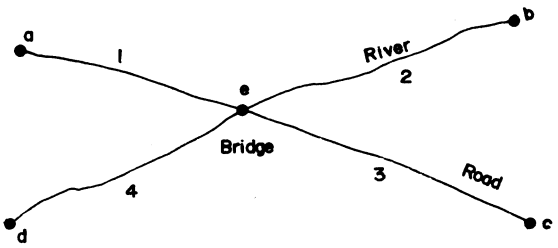


Figure-37 : Point features at the intersection points in link-node data structure

Problems

Besides all these solutions there are still some outstanding problems, first of all, the data structure is not "intelligent-enough."

a. Directions are unknown (to which direction is the river running?)

b. When we reach an intersection point, the further connection alternatives are not clear, it is difficult to navigate throughout a network. (Given that the terminating and final destination points, which path would be the shortest ?)

c. Adjacency relationships are not clear. When following a certain direction what are there on the left and on the right? (While driving from Istanbul to Ankara along the highway which gas-stations are on the right ?)

d. Other features contained in an area feature can not be determined without long computational searches such as point in polygon search. (List all the buildings and bridges in the Beynam Forest.)

These problems which haven't been mentioned before, should be solved as to answer comlicated queries, some of which were exampled in the paranthesis above, for a full GIS functionality.

Although the link-node data structure is called as "structured data" by some producing agencies, this is not a correct title. This naming is caused from a relative intuition that the spaghetti data is qualifed as unstructured when compared to the link-node structure.

The link-node structure requires more powerfull processors if it is to be created in real time mode during the digitizing (automatic node generation capability).

The link-node data structure has been increasingly used in today's GIS software systems such as SİCAD from SIEMENS and GINIS from SYSCAN.

4.4. TOPOLOGICAL LINK-NODE STRUCTURE

Some contributions from the topology branch are necessary to meet the requirements listed as problems at the end of section 4.3. For this reason, the subject of the topology branch will be briefed very shortly.

4.4.1. Topology

Topology, a brach of mathematics, studies properties that remain unchanged when the figures are transformed. Let's take the example shown on Figure-38. The figure 38-a has transformed (deformed) into the figure 38-b.

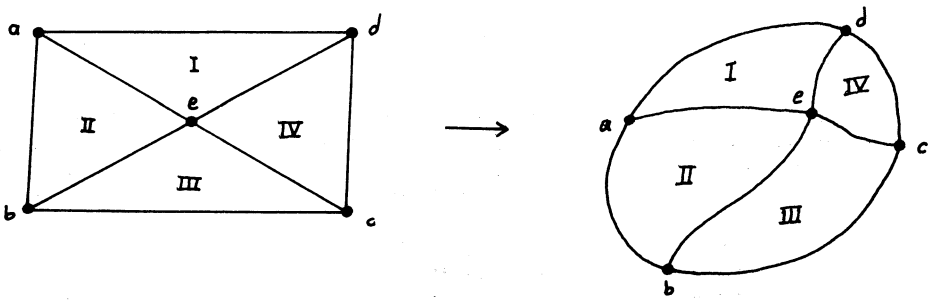


Figure-38 : Deformation on figures

After this transform, the

- actual coordinates,
- lenghts
- areas

have been changed.

But :

- the number of lines meeting at each point,
- connectivity (i.e. how connected together and relation to one other)
- area features on either side of each edge have remained unchanged.

If this is the case, we can say that two figures are topologic equivalent. So, topology deals with the properties of figures that don't change after a deformation, not with the properties like shape and size.

Although the deformations are not acceptable in cartography with few exceptional transformations, these topological properties can make significant contributions to geographic data structuring. These contributions can be listed shortly as follows :

a. By defining certain relationships such as adjacency and intersection explicitly, makes it faster to select features meeting given topological conditions.

b. When the shared geometrical elements have been defined once, stores the common elements once, so reduces the redundancy.

c. Supports the navigation throughout the geometrical data.

d. Assures that the geometrical data is being kept consistent.

When these topological relationship ideas are added into the link-node structure, a more advanced structured would be obtained.

4.4.2. Highlights of the topological link-node structure

Like the basic link-node structure, there are again node records (node file/table), link records (link file/table) and feature records (feature file/table) stored in this structure. But the records are explicitly associated to other records of the same or different type file to reflect the topological relationships (intersection, adjacency, connectivity, direction etc.). The logical data structure diagram for this structure is shown on Figure-37. As shown on the figure, a relationship is established between the link and node records. Each node record contains the identifiers of lines (links) connected to this node; each link record on the other hand contains the identifiers of begining and ending nodes. In addition, the identifiers of the area features which are on the left and right and identifiers of the line features passing through are contained in link records.

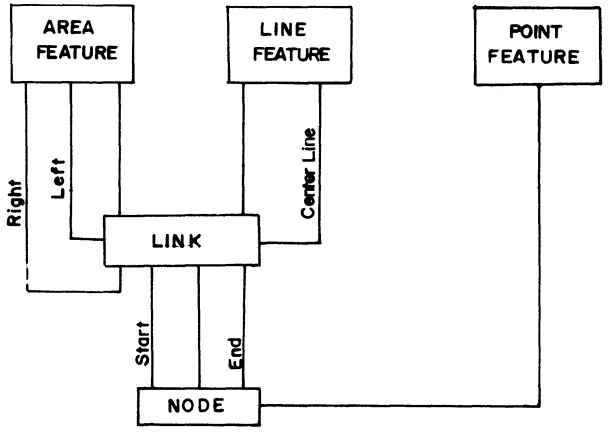


Figure-39 : The logical data structure diagram of the link-node structure

The direction concept has been imported, because the beginning and ending nodes are stored explicitly. The digitizing direction is taken positive as a rule and shown with an arrow-head.
(Figure-40).

Figure 40 : Directed edge

Additionally, again as a rule, all the area features are defined as to be lying on the left.

The topological link-node structure meets almost all of the today's GIS requirements and gives a high performance.

ARC/INFO from ESRI, SLF from DMA and GVC-GIS from Geovision are examples of sofware systems using this structure.

4.4.3. SLF (Standart Linear Format)

SLF is one of the specific implementations of topological link-node structure with some minor changes. SLF is designed and implemented by US-DMA (Defense Mapping Agency) as an intermediate format to be used from 1983 to 1990 before starting to adopt the ultimate MC&G (Mapping Charting & Geodesy) format.

The most important difference of SLF is that there is no node records. The link records do include the starting and ending points in contrary to the link record definitions made up to now. This type of link or edge is called as "segment" Accordingly if the same beginning or ending point exists in more than one segment record, this point should be considered as a node.

Another change on the SLF with respect to basic topological link-node structure is that all three types of features (area, line, point features) are stored in the same type of record. The logical data structure of SLF is shown on Figure-41.

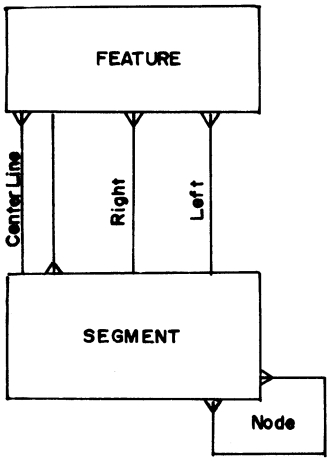


Figure-41 : SLF logical data structure diagram.

Here the feature records contains the following information.

- feature id.
- feature type (P=point, L=line, A=area)
- attributes
- segment list : (direction, segment id.)

The segment records contains such information :

- segment id.
- list of coordinates
- list of features (feature id., orientation: C=center, L=left, R=right)

So the segments obtained from the digitizing of our example map is shown on Figure-42.

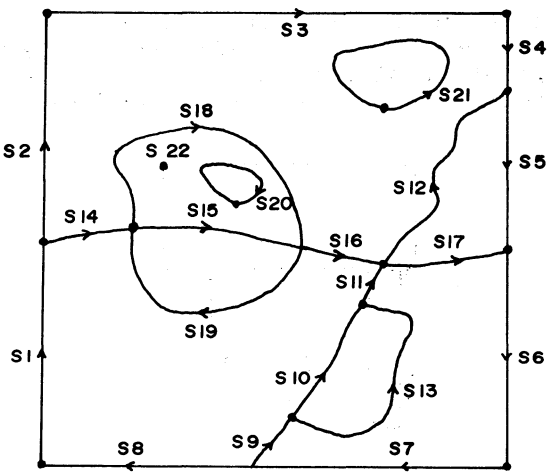


Figure-42 : Segments

The same example map would be stored in such an SLF structured file(s) like shown on Figure-43 .

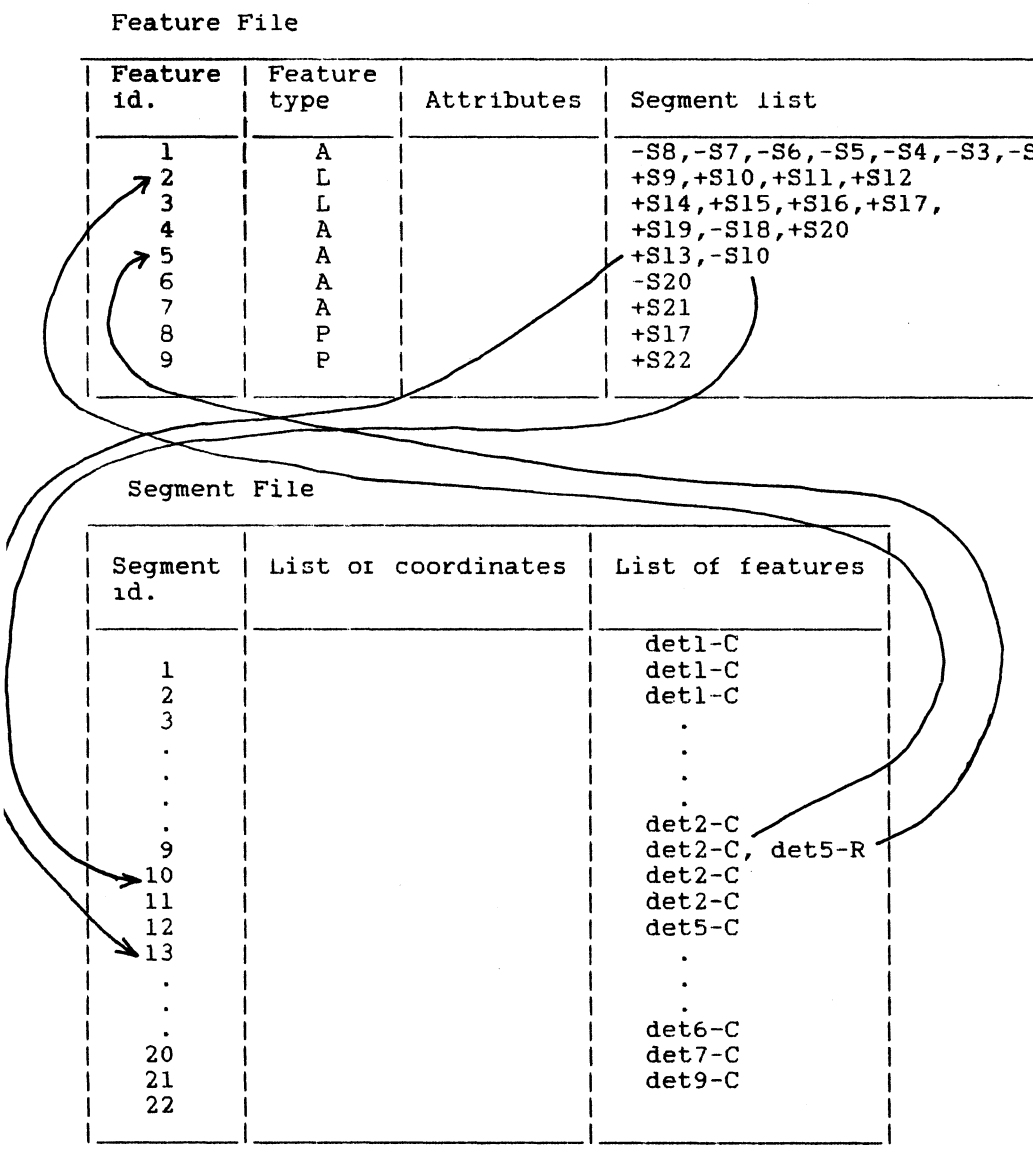


Figure-43 : SLF file

Solutions :

a. "Which features are on the left (or right)?" type questions can be answered.

b. The adjacency relationship can be defined by detecting the features sharing the same segment.

c. Navigation (on arrival at a junction, deciding on the way to be followed) is partially solved. It is easier with basic topological link-node structure because there exist node records.

d. Positional relationships between features are solved by means of sharing topology.

- point features at the same node,
- line features on the same segment/edge,
- area features sharing the same polygon (inner chain : figure-41, feature 4, segment 20)

can be defined, at least with little processing.

Problems :

There are also some problems of SLF.

a. Whether there is a connection between two points or not, or whether two line features intersect or not can be done by searching common end point(s) of the segment records. In other words, the c and d solutions above could be accomplished only by processing.

b. It is very difficult to solve the containment problem. To do this, it is necessary to run "point in polygon search" routines.

4.5. FULL TOPOLOGICAL DATA STRUCTURE

Full topological structures are being developped to eliminate the problems listed above. In addition to links and nodes a new geometrical element, face, has been defined to solve some problems.

Face is the largest 2 dimensional area, bounded by edges, that is not further divided by an edge. (Figure-44).

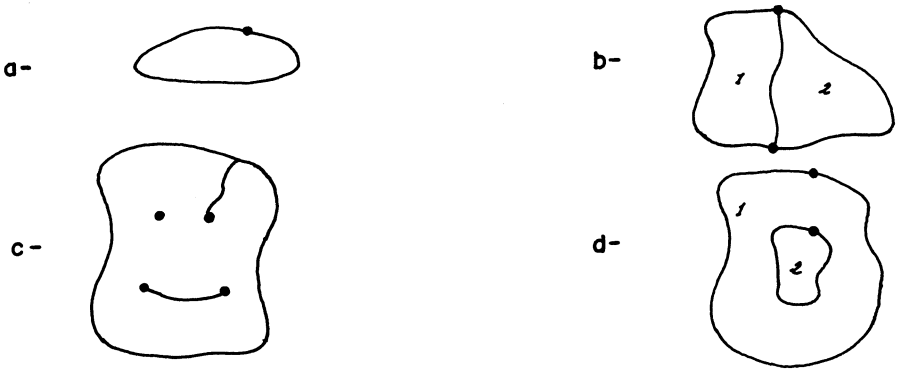


Figure-44 : Faces

There are two faces each, on Figure 42 b and d and one face each on Figure 42 a and c. Two different ways can be followed to define the faces. The first way is to convert a link-node structured file into a full topological structure with a batch program. In the second way, the faces are defined in real time mode, that is as soon as the lines are digitized. This process is called as "face division". On Figure 45, the division of a face is illustraded in a step by step fashion as the digitizing goes on.

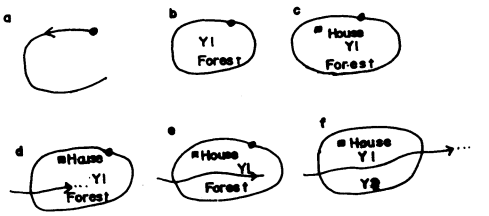


Figure-45 : Division of a face

Today there are only few implementations supporting this full topological structure. MSDS (Military Survey Data Structure) from UK-DMS and MC&G (Mapping Charting & Geodesy) from US-DMA are the names of this type of design. Intergraph Corporation has announced the first implementation of this type at the end of 1988.

If we look our bycoming example from the MC&G's point of view, we would obtain such a figure. (Figure-46)

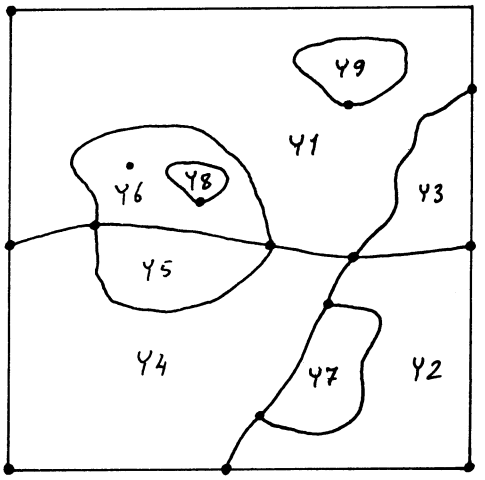


Figure-46 : MC&G data structure

This structure views a map as a collection of nonoverlapping faces, not as a collection of lines. The logical data structure diagram of this structure is shown on Figure-47.

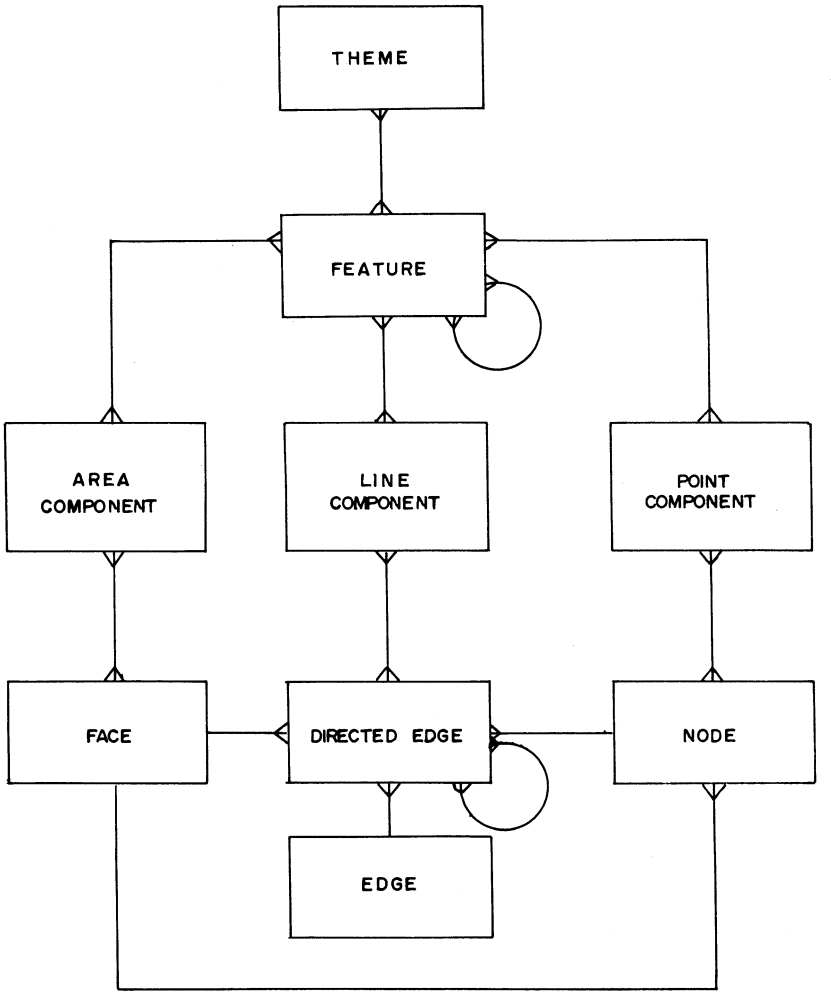


Figure-47 : TIGRIS Logical Data Structure Diagram

Most importantly, data about features (ie. attribute etc.) and data about geometrical elements (area, line, point) are seperated in this structure. Each future is composed of one or more feature components of area and/or line and/or point types. Area, line and point type feature components are represented with face, directed edge and node respectively. Each line has two directed edges (+ and -). Each face is defined with one or more directed edges and one or more nodes. The features can be associated with other features. For example a river system consists of many rivers and streams. Features can be groupped under certain themes.

The structure is highly detailed to meet complex queries. Each point is stored only once, so the consistency has been guaranteed. The feature hierarchy in the theme-feature-component sequence offers enriched query capabilities and the adjacency, connectivity, intersection and containment problems have been solved.

The followings are listed as the advantages of the TIGRIS data structure.

a. Clean centerline data
b. Limited coordinate access
c. Every point belongs to one and only one topological entity
d. Interactive spatial analysis
e. Maintained automatically
f. Hierarchy of features

However, TIGRIS brings some drawbacks too :

a. Increased data storage requirements
b. Greater software complexity
c. More powerful workstations

5. CONCLUSION

Geographic databases differs from a conventional database as it includes graphical data as well as the non-graphical data about entities which have a position with respect to a given coordinate system. A Geographic Information System of hardware and software components to capture the graphical and non-graphical data about geographical entities, to store these data as an integrated geographic database, to process (transforms and analysis), to retrieve (query) and to represent in different forms such as screen or hard copies, reports etc. The capabilities offered by a GIS is highly dependent upon the data structure of this geographic database An improved level of geographic structure extends the positional query, transform and analysis capabilities beyond processing limits and removes the geometrical problems from the displays.

As the graphical data has been represented in the computer readable form, many of the conventional concepts have been changed eventually and many brand-new concepts have come into apperance. Some of them have been briefed in this paper.

Raster and vector techniques have extremely different characteristics with their advantages and trade-offs. Reflectances of this difference can be seen on many different types of graphic data capture and output devices such namely digitizers, plotters, graphical screens etc.

In raster data structures, the first priority has been given to shorten the access paths and to reduce the storage requirements. But in vector data structures, to improve the geometrical quality and ability to positional queries have been taken as the most important goals.

The raster and vector data structures have been explained under staged categorizes in the paper. But there are of course small differences in the commercial implementations of these structures. The data structures used to represent the topographic surface (such as vector, matrix and TIN approachs) have been excluded in the paper.

Another conclusion could be that, the more capabilities a data structure offers, generally the more processing power would be required.

The developments on the geographic data structures is still going on. It is expected that, in 1990 s, the 2 dimensional data structures will reach a satisfactory level and the 3 dimensional structures seems to be the subject of discussions.

REFERENCES

/1/ Armstrong M.A. : Basic Topology. Springer. 1983

/2/ Blais J.A.R. : Theoretical Considerations for Land Information Systems. Universty of Calgary 1986.

/3/ Bruegger B.,Frank A. : Hierarchies over Topological Data Structures. Proc. ACSM/ASPRS Vol. 4 GIS, pp.137-145, Baltimore. 1989

/4/ Burrough P.A. : Principles of Geographical information Systems for Land Resources Assessment. Clarendon Press, Oxfort. 1986

/5/ Claire R.W., Guptill S.C. : Spatial Operators for Selected Data Structures. Proc. Auto Carto 5, pp. 189-200ASP/ACSM, Virginia. 1982

/6/ Date C.J. : An Introduction to Data Base Systems Addison-Wesley, Reading Mass. 1986

/7/ ESRI : ARC/INFO Geographic Information System Software. 1986

/8/ Greasley I. : Data Structures to Organize Spatial Subdivisions. Proc. ACSM/ASPRS Vol. 5. GIS, pp.139-148, St.Lois. 1988

/9/ Horowitz E., Sahni S. : Fundementals of Data Structures Pitman, Maryland. 1980

/I0/ Intergraph : TIGRIS An Object Oriented GIS Environment. 1989

/11/ Kleiner A., Brassel K.E. : Hierarchical Grid Structures for Static Geographic Data Bases. Proc. Auto Carto-8, pp.485-496, London. 1986

/12/ Mark D.M. : The use of Quadtrees in Geographic Information Systems and Spatial Data Handling. Proc. Auto Carto-8, pp.485-496, London. 1986

/13/ Martin J. : Computer Data Base Organisation Prentice-Hall, New Jersy. 1977

/14/ Monmonier M.S. : Computer Assisted Cartography Principles and Prospects. Prentice-Hall, N.J.. 1982

/15/ Sarbanoğlu H. : Coğrafi Bilgi Sistemleri Harita Dergisi, Sayı, 103, 1989

/16/ Tamminen M. : Some Aspects of Defining Spatially Referenced Data. Proc. FIG XVI-Congress Comm. 3, pp. LII.1. 1981

/17/ U.K. Directorate of Military Survey : Reports on CREST7 Project and MSDB. Feltham. 1987

/18/ Van Lamsweerede A. : Spatial Datastructures for Land Information Systems. Proc. FIG XVI Congress Comm. 3., pp. 301.5 . 1981

Y A Z A R A D R E S İ

Yük.Müh.Yzb.Hakan SARBANOĞLU
Harita Genel Komutanlığı
Cebeci
06100 ANKARA