AEGIS - A PRACTICAL EXERCISE IN DATA MANAGEMENT

David Harris and Karen Pettigrew

CACI MARKET ANALYSIS
59-62 High Holborn
London
WC1V 6DX

CACI Market Analysis has for a number of years been faced with spatial data management problems for which no commercially available system has been found to provide a satisfactory solution. The company has now designed and partly implemented support for a powerful and flexible data model to satisfy the following criteria:

1) Ability to perform editing, digitising, analysis, manipulation and mapping functions directly upon a single data structure. A prime requirement was to eliminate any need for continual data conversion between different storage formats.

2) Compatibility with existing CACI software.

3) Subroutine library support for basic operations.

4) Sufficient flexibility to cope with any cartographic information we could envisage handling in the medium term future.

This paper describes the software implementation aspects of the model, its applications, and an outline of envisaged future developments.

Introduction

CACI Market Analysis was the first commercial organisation to become an agency for resupply of census statistics in 1977, and has been a pioneer in the use of spatially referenced data for market analysis and planning. Since that time the volume and range of applications have undergone a process of continual and rapid expansion. The company now provides consultancy, analysis and data services to clients in both the private and public sectors.

Mapping as a means of presentation of data and results has always been a requirement. During the last few years, however, the need for sophisticated mapping and GIS capabilities in our application areas has grown steadily. After a review of the available software in this area CACI has decided that the most effective course of action is to independently develop new software. This paper describes the approach adopted to the management of structured vector cartographic data.

It should be stated at the outset that our requirement is for usable software rather than ideas or academic theories. The primary concern is to put ideas to work rather than to unearth their origins.

Our review of commercially available software began towards the end of 1984. CACI uses a range of software for GIS-related purposes, in particular a digitising package has already been developed in-house. GIMMS has been used for some mapping purposes, and a friendly CACI mapping package is used for maps not requiring the flexibility of GIMMS. This package has undergone gradual enhancement, and is expected soon to be used for all mapping purposes within CACI Market Analysis.

The criteria used in our search for GIS capabilities are described in the abstract. Of these, the requirement for a single vector data structure was considered paramount - the requirement to convert boundaries to and from GIMMS format has caused many problems, and these problems will inevitably multiply as spatial data becomes more abundant and more complex. Criteria (2) and (3) are to some extent equivalent, in the sense that any package satisfying criterion 3, is likely to be compatible with our existing range of data management and analysis software. Not to require criterion (4) would be short-sighted.

Design of a suitable data model and subroutine library was started in summer 1985 with the help of Phillip Wade of Hull University (funding from a SERC-CASE studentship), and was completed by the authors and Duncan Campbell of CACI towards the end of 1985. At the time of writing implementation is in progress and is expected to be complete before this paper is presented.

The Software Philosophy

Much interest in this field at present revolves around the implementation of cartographic data management within a relational (or other) database management system. We have taken a different approach in this exercise. The objectives have been to develop a data model, to identify the necessary fundamental operations on the data model, and to develop a subroutine library (named AEGIS) to support these operations. The result is a software tool which provides low-level cartographic data handling support for any application program, particulary for digitising (and editing), mapping and analysis applications.

This approach was chosen primarily because it provides support for all operations which will be required on the data, whereas a DBMS implementation is unlikely to do so in an effective manner.

We see further advantages as follows. There are great benefits in speed of operation since file structures and access modes are designed specifically, and since much redundant housekeeping which a DBMS would perform, is discarded. In addition the application programmer's job is made much easier by having access to subroutines which do just the operations required, rather than a much more general set of access subroutines which are typically supplied with a DBMS package.

We do lose out by not having immediately accessable all the facilities a DBMS provides. To some extent we can offset this loss since we have code-level access to CATALIST, a CACI data management and report generation package, which we will be able to incorporate as an application package using the AEGIS subroutine library. Some facilities - such as simultaneous write access to a coverage by more than one user - we will just have to do without.

The Data Model

Basic Entities

There are four basic entity types

1) Points
2) Line Segments
3) Primitive Region
4) Compound Objects

In geometrical terms, Line Segments start and end at points, and may have any number (zero or larger) of intermediate co-ordinates. Note that the intermediate co-ordinates are not point entities. The term 'point' will henceforth always mean a point entity, and is distinct from an 'intermediate co-ordinate'. Primitive Regions are contiguous uncut, bounded regions. Unlike the other entity types, primitive regions cannot be added or deleted by editing operations. They are created only by a polygon build process. Compound objects are, in general, lists of entities of any of the four types.

This framework, although developed independently of the Ordnance Survey small scale proposals, is similar to the structure described by Haywood (1986).

Attributes.

The treatment of attributes is of central importance.

Any entity may, and usually will, have a list of attributes attached to it. All attributes have the same structure - there is a numeric attribute type, which in the present implementation must be in the range 0 to 16383, followed by further information, which may be of any length and of any format.

Attributes are used for two purposes.

1) To record information about the object it belongs to - for example that a point represents a railway station, that the railway station is called St Margaret's, or that a line segment represents a section of railway line.

2) To record geometrical and other relationships between objects for example that St Margaret's railway station is one of the end points of the above mentioned line segment. Typically such attributes will identify other entities by their internal reference number.

It is important to note that the attribute list is the only means of holding information about an entity, other than the basic information which is held for all entities of a particular type.

In implementation terms, all attribute types exactly divisiblê by ten (ie numbers ...0) are reserved by the system and have specific preallocated meanings. Some of these may not be manually added or deleted by the user. For example point attribute type 0 is used to record the fact that the point is a node of a line segment - it is added or removed only by the ADD_SEGMENT and DELETE_SEGMENT subroutines.

All other attribute types are user definable, and may be used for whatever purposes, and have whatever meanings, the user wants.

Housekeeping and General Support

In addition to the basic operations described in the following sections, a number of general housekeeping functions are undertaken automatically. In summary the AEGIS subroutine library provides the following support:

a) Database consistency is ensured, except in the case of a machine crash (in which case all data is recoverable).

b) Two separate point features cannot have the same location.

c) There is unlimited feature-sharing using attribute lists.

d) There are automatic checks for knots as segments are added.

e) Null segments cannot be added (if a segment has start and end node the same, it must have at least two intermediate co-ordinates).

f) Update histories are automatically recorded.

g) Application programs are independent of physical file structure.

h) Douglas-Peucker line generalisation is supported.

i) A comprehensive set of basic operations on spatial data is easily accessable to the application programmer.

Points, and Operations Upon Them

The basic information held for each point is as follows:

Unique internal reference number
Location specified by two floating point numbers
The degree of the point (ie. the number of segments for which it is a node).

The basic operations supported are ADD, FIND (ie. locational search), READ, READ_SEQUENTIAL, ADD_ATTRIBUTE, READ_ATTRIBUTE, DELETE_ATTRIBUTE.

Within these operations there is automatic support for one particularly important consistency check - namely that it is impossible for the user to cause two point entities with the same location as each other to be entered into the coverage.

There is no operation to delete a point. This is because a point is considered meaningless unless it has at least one attribute. A point without attributes is termed a 'ghost point' and is effectively deleted: read operations will not read such a point, but it can be activated by addition of an attribute immediately after its creation, or less commonly at a later time if its reference number is known. This approach avoids the problem of coping with delete-protection for points which are nodes.

Line Segments, and Operations Upon Them

The basic information held for a line segment is as follows:

Unique internal reference number
Start and end nodes (which are references to point entities)
Length
Area (cf Wade (1986))
Envelope
Date and Time created
Date and Time deleted (if it has been deleted)
XY location and a generalisation code, for every intermediate co-ordinate.

The basic operations supported are ADD, READ, READ_SEQUENTIAL, DELETE, ADD_ATTRIBUTE, READ_ATTRIBUTE, DELETE_ATTRIBUTE, JOIN, SPLIT.

The ADD operation automatically adds type 0 attributes to the point entities which are the start and end nodes. These provide pointers from the nodes to the segment. DELETE_SEGMENT will remove these attributes.

Unlike points, line segments may be deleted. This is because a line segment without attributes is more meaningful than a point without attributes. One may wish to create a coverage containing only local authority boundaries, for instance, without bothering to add any attributes because the meaning of all line segments in this coverage is already known. To do the same with a point coverage is less satisfactory since there may have to be other points (such a digitiser reference points) in the same coverage. In addition, the problem with delete-protection does not arise, because primitive regions are invalidated whenever any segment is added or removed from a coverage.

The DELETE operation does not remove the basic information for a segment. The segment is marked as deleted, and the date and time of deletion recorded. This allows the segment/node structure of a database as it was at any specified past time to be recovered, provided no subsequent garbage collection has taken place.

The ADD_SEGMENT operation automatically adds generalisation codes, calculated using the Douglas-Peucker algorithm, to all intermediate co-ordinates, and READ operation can filter intermediate co-ordinates using these codes. The user can, alternatively supply generalisation codes, or even use this facility to store height information for each intermediate co-ordinate.

Primitive Regions, and Operations Upon Them

These are a little more complicated. The reader is referred to Visvalingam (1986) and Wade (1986) for background and definitions.

The basic objects held are actually boundaries, but attributes are used to form a structure in which primitive regions are (almost) as easily accessable.

For each boundary, the following basic information is held:

Unique internal reference number
Type (either enclosing boundary, or hole, or null boundary)
Area
Perimeter length
Envelope
Inside (described further below)

The basic operation are BUILD, READ, READ_SEQUENTIAL, READ_ATTRIBUTE, READ_POLYGON_SEGMENTS

The BUILD operation is in two stages. The first stage builds boundaries, and identifies each boundary as either an enclosing boundary, or a hole. Each boundary comprises a list of segments, and these are held as attributes. Each such attribute contains the internal reference number of a line segment. This stage also adds an attribute to each line segment, containing the back pointers - ie. the internal reference numbers of the boundaries on the left and right of the segment.

The second stage forms primitive regions by allocating each hole to an enclosing boundary. The reference number of the enclosing boundary is entered into the INSIDE field of the hole, and an attribute containing the reference number of the hole is added to the enclosing boundary. To make everything tidy, there is a 'global enclosing boundary' with reference number zero, which has no segments, but has attached to it all the outermost holes of the structure.

Once this has been done, it is possible to access primitive regions by ignoring records corresponding to holes, and remembering when accessing segments for an enclosing boundary to also access segments for holes inside that boundary.

The third stage of Wade's boundary building algorithm has not yet been implemented because of lack of a good reason for doing so, but the data structure allows for it - each enclosing boundary (except the global enclosing boundary) would be marked INSIDE a hole, and each such allocation would cause an attribute to be added to the hole, giving the reference number of the enclosing boundary.

Compound Objects, and Operations Upon Them

Compound objects are lists of entities held as attribute lists. To a great extent, therefore, a compound object is nothing more than its attribute list. There is a little basic information held, however:

Internal reference number
Keys - four integers each in the range -32768 to 32767
Name
Envelope

The basic operations supported are ADD, READ, READ_SEQUENTIAL, READ_KEYED, DELETE, ADD_ATTRIBUTE, READ_ATTRIBUTE, DELETE_ATTRIBUTE, ADD_ENTITY_TO_COMPOUND_OBJECT, REMOVE_ENTITY_FROM_COMPOUND_OBJECT.

The ADD_ENTITY_TO_COMPOUND_OBJECT operation adds an entity of any type to the components of a specified compound object. It is just a double ADD_ATTRIBUTE function - one attribute added to the component list of the specified compound object indicating which object has been added, and one to the added entity indicating which compound object it is now a component of.

The most frequently used means of attaching a primitive region to a compound object is via an <u>area seed.</u> The general method is that area seeds are components of compound objects, and an <u>area allocation</u> procedure finds, for each seed, which primitive region it lies within. The primitive region is then attached to the compound object.

The reason for the keys is to provide a permanent unique identifier for each compound object which may be specified by the user, for instance, to specify a county, district, ward or enumeration district using standard OPCS area keys. This method can be used to represent hierarchical relationships between compound objects.

If permanent area objects, say counties, are to be stored they will be stored as compound objects. For example, the 'Islands Region' of Scotland would be a compound object (keys 65,0,0,0 if following OPCS codes) which would have many primitive regions attached to it.

Summary

We currently have an interactive digitising package acting directly upon these data structures via the AEGIS subroutines, together with a rather basic mapping capability. It is expected that all spatial data used by CACI Market Analysis will be held in this form, and accessed only via the AEGIS library, before the end of 1986.

We intend that a far more powerful mapping capability will be developed shortly, together with data dictionary and query language facilities.

No final decision has yet been made on whether the software will be made commercially available. The authors would be interested in hearing from any parties who feel that they could either contribute to the ongoing development programme, or benefit from it.

References

[1] Haywood, P.E. (1986). Proposal 1:50000 digital data specification. Ordnance Survey

[2] Visvalingam, M., Wade, P., Kirby, G.H. (1986). Extraction of area topology from line geometry. Proc. Auto-Carto London.

[3] Wade, P., Visvalingam, M., Kirby, G.H. (1986). From line geometry to area topology. CISRG discussion paper, University of Hull.