THE UNIVERSITY OF HULL

Cartographic Information Systems Research Group
C.I.S.R.G.

Hon. Co-ordinator:
Dr. M. Visvalingam
Department of Computer Science
The University
HULL HU6 7RX

Tel. (0482) 465295/465951
Telex 592592 KHMAIL G,
f.a.o. HULIB375
Fax (0482) 466666

Special Issue No. 3:

Management of Digital Map Data
Using a Relational Database Model

by

M Visvalingam and N M Sekouris

This project was undertaken on behalf of the Ordnance Survey of Great Britain and this end-of-grant report has been reproduced with their permission.

March 1989

List of CISRG reports:

Discussion Papers

1. Wade, P., Visvalingam, M. and Kirby, G.H. (1986)
   From Line Geometry to Area Topology, 48 pp.

2. Visvalingam, M. (1987)
   Problems in the Design and Implementation of a GKS-based User Interface for a Graphical Information System, 18 pp.

3. Visvalingam, M. (1988)
   Standardising Basic Spatial Units : Problems and Prospects, 19 pp.

4. Whyatt, J. D. and Wade, P. R. (1988)
   The Douglas-Peucker Algorithm for Line Simplification: an introduction with programs, 17 pp.

5. Visvalingam, M. (1988)
   Cartography, Geographical Information Systems and Maps in Perspective, 16 pp.

Special Issues

1. Visvalingam, M. (ed) (1986)
   Research based on Ordnance Survey Small-Scales Digital Data Proceedings of a meeting held as part of the 1986 Annual Conference of the Institute of British Geographers (Reading, Jan 1986), **sponsored by the Ordnance Survey,** 79 pp.

2. Visvalingam, M. and Kirby, G. H. (1987)
   Directory of Research and Development based on Ordnance Survey Small Scales Digital Data, **sponsored by the Ordnance Survey,** 38 pp.

3. Visvalingam, M. and Sekouris, N. M. (1989)
   Management of Digital Map Data Using a Relational Database Model, **sponsored by the Ordnance Survey,** 50 pp.

Previous Reports to Ordnance Survey

Visvalingam, M., Kirby, G. H. and Wade P. (1985)
Extraction of a Complete Description of Hierarchically Related Area Objects from Feature-coded Map Details, 28 pp.

Kirby, G. H., Wade, P. and Visvalingam, M. (1986)
Storage and Retrieval of Topographic Data using a Relational Database Management System, 37 pp.

Sekouris, N. M. (1987)
A relational Database Model Applied to Digital Maps, Final Year Project Report, Department of Computer Science, University of Hull, 37 pp.

ACKNOWLEDGEMENTS

This project was funded by the Research and Development Division of the Ordnance Survey. We would like to record our special thanks to Don Proctor and Neil King for their support and encouragement in the initial stages of the project, to Liz Bowell and Neil Smith for technical assistance and discussions and to John Farrow for permission to disseminate the results through a CISRG Special Issue.

Our sincere thanks also to Peter Williams of the Thames Water Authority, Phillip Gibb of the National Freight Consortium, Rob Ward of the Derbyshire County Council and Roger Moore of the Institute of Hydrology for comments on the relational approach and the utility of small scale digital map data.

This work could not have been undertaken without the facilities and technical support provided by the Department of Computer Science, University of Hull. We are particularly grateful to Dr G H Kirby for his continued encouragement, discussions, comments and support throughout the project.

M Visvalingam and N M Sekouris
March 1989

C O N T E N T S

1. INTRODUCTION

This end-of-grant report formally concludes the OS/CISRG collaboration on the one year project entitled "Management of Digital Map Data Using a Relational Database Model". This introductory section provides the background to the project, an overview of the project as undertaken and a summary of the major conclusions.

1.1 Background

The one-year project was set up to provide sponsorship for Nick Sekouris to pursue research towards an MSc degree in Computer Science on the topic "Management of Digital Map Data Using a Relational Database Model" (see Appendix 1). The funding was awarded on the strength of Sekouris' final year project report on this theme (Sekouris, 1987); a copy of this report was sent to the Ordnance Survey. Sekouris was also awarded the Star Award Scholarship for database management excellence by Relational Technology Ltd which enabled him to attend the 13th International Conference on Very Large Databases in Summer 1987. The one-year grant from the Ordnance Survey has enabled him to further extend his knowledge and expertise in this and related areas.

The main output of this project will be Sekouris' Masters thesis; a copy of which will be sent to the Ordnance Survey after it has been formally examined and passed. Appendix 2 lists the contents of his thesis; all chapters exist in typescript form. The purpose of this end-of-grant report is to provide a succinct and discussive account of those aspects of the thesis which are of interest to the Ordnance Survey and additional comments where appropriate.

1.2 Overview of Project

1.2.1 Initial Stage (1-10-87 to 31-12-87)

Neil King and Liz Bowell of the Ordnance Survey kindly provided some documents on the relational model for the 1:50 000 Small Scales Demonstrator on 15-10-87. These documents appear in Appendix 3. The data for Sheet 111 (Sheffield and Doncaster) was received on 29-10-87.

Since the documentation was minimal, Sekouris had to gain understanding of the model, structure, contents and meaning of database entities and attributes by inspection and hypothesis-based exploration of the database. During this period he had to learn to use Ingres from scratch and substantially extend his knowledge of GKS, Unix and spatial data models. Since our understanding of the OS model was based on inference, rather than on full documentation, we invited Neil King and Liz Bowell to the University of Hull to discuss our conclusions. This took place on 1-12-87 when Sekouris presented a seminar on "Management of Digital Map Data Using a Relational Data Model" and later asked questions within the context of a demonstration. Sekouris and Visvalingam in turn visited the Ordnance Survey on 8-12-87. During this visit Neil King and Liz Bowell explained the objectives of the 1:50 000 database experiment.

The primary aim was to formulate an appropriate data model which would facilitate :

* economical storage of small scales data
* speedy and error-free update of such data
* rapid retrieval of subsets of data conforming to spatial and/or aspatial criteria.

This experimental data is also used for demonstrating the content and utility of small scale data to potential clients. Liz Bowell had written a graphical front-end to the 1:50 000 database to illustrate the scope for :

* interactive exploration of the data by pointing to elements on the map or on pop-up menus
* building linear and areal objects from point, line and text features. The objects in the 1:50 000 Demonstrator database were directed at supporting route planning and area shading in mapping.

We were asked to offer a second opinion on the current OS data model for small scales as adopted for Sheet 111 (Appendix 3)

1.2.2 Conceptual and Functional Analysis (1-1-88 to 30-4-88)

This period focused on two types of activities, namely a conceptual analysis of the OS model and a survey of user requirements. Sekouris

had to resolve outstanding problems of interpretation of the OS model by telephone but was able to make some progress with the conceptual analysis during this phase.

Neil King provided us with a list of actual and potential customers of small scales data, both 1:625 000 and 1:50 000, from which a short list was drawn up. Since this information is confidential (and it is therefore not included in the Appendices), Neil King established that the short-listed Users were agreeable to Sekouris telephoning, and then later visiting, them. The telephone interviews revealed that relatively few users were willing to talk about their experience and indeed that few had comments based on experience with small scales data.

Consequently, site visits were limited to the following organisations,

Thames Water	(22-2-88)
National Freight Consortium	(2-3-88)
Derbyshire County Council	(26-3-88)
Institute of Hydrology	(25-4-88)

The main conclusions are summarised in Section 5.

1.2.3 Performance and Automation of Object Creation (1-5-88 to 30-9-88)

During this phase Sekouris paid more attention initially to issues relating to performance. Attendance at the Ingres Kaleidoscope Conference (6-5-88) proved useful.

The User survey and Sekouris' exploration of the 1:50 000 data led us to believe that there was a need to at least semi-automate the extraction of objects from the feature-based input data. This was welcomed by Liz Bowell and Neil King. Sekouris was thus encouraged to initiate some research into the formulation of rules for automatic recognition of objects.

1.3 Overview of Our Conclusions

1.3.1 Use of the Relational Model

At the current, still exploratory, state-of-the-art, the relational approach is convenient for modelling spatial data. Being flexible, it

encourages experimentation with alternative database designs. The relational approach is also being promoted by the emergence of database machines with parallel architectures.

A pure relational approach is wasteful for storage of pre-ordered data, such as the internal co-odinates of links and time series data captured at fixed time intervals. However, systems developers are aware of such deficiencies and accept the need for using hybrid data models. Until uniform schemes emerge for storage and manipulation of ordered data, users are forced to take advantage of system-specific schemes for coping with such data.

1.3.2 Defects in the OS Data Model for Small Scales Data

The relational model used in the 1:50 000 Small Scales demonstrator database has some weaknesses. In Section 2, we describe our understanding of the existing model, critically analyse it in conceptual terms and suggest means by which the model may be made more comprehensible, economical and efficient.

1.3.3 Errors in the 1:50 000 Database for Sheet 111

The 1:50 000 data for Sheet 111 contains numerous positional and semantic errors. At least one potential user, who had acquired the database for demonstration of his own software, was put off by these errors. We believe that OS should formulate and apply procedures for checking the consistency of semantic and topological relationships. In Section 3, we discuss the issue of errors and describe some error-trapping procedures. There is a need for further research in this area.

1.3.4 Performance Issues

Once the conceptual model is rationalised into a minimal and efficient form, further gains in performance can be achieved by taking advantage of system-specific features and a knowledge of applications. Also, experiments conducted at Hull suggest that the OS practice of indexing all four envelope fields is wasteful of storage. In Section 4, we describe a system and application independent scheme for improving the performance of spatial search with a much smaller index.

1.3.5 User Opinions

There appear to be few users of Sheet 111 at present. Relatively few users agreed to being interviewed. The four users, interviewed by the student, foresaw potential uses for small scales data, particularly at the 1:50 000 scale. Their comments are summarised in Section 5.

1.3.6 Need for Objects

The present OS policy is to exclude objects from small scales digital products. Objects were created by OS for purposes of demonstration only. However, a feature-only archive is insufficient for maintaining logical consistency of the database during update operations. Indeed, errors in feature coding were only detected by cross-checking them against object definitions. The current manual chaining of objects smooths over errors in input data whilst itself being error-prone. Also, the linear and areal objects, manually created by OS for demonstration, are also required by users. The Institute of Hydrology has made considerable progress in automating the definition of river nets and watershed boundaries although the OS 1:50 000 cartographic base is inadequate for this purpose. We too believe that it is possible to at least semi-automate object building and would encourage OS to devote resources to automatic recognition of those objects whose presence and geometry is already indicated on the map. This is discussed further in Section 6.

2. COMMENTS ON THE CURRENT OS RELATIONAL MODEL FOR SMALL SCALES DATA

As explained in Section 1.2.1 our appreciation of the OS Relational Model is based on the documentation in Appendix 3; full documentation was not available. Our understanding of the model is based on hypothesis-based verification and exploration of the data and of the model by the student. Liz Bowell kindly answered questions arising from such explorations and checked our inferences.

2.1 OS Requirements and Implications

In Section 1.2.1, we outlined the major requirements of the relational model. These are:

* economy of storage
* speed of satisfaction of spatial and aspatial queries
* efficient and error-free update

Although the input data consists of point, line and text features, the OS model provides for the definition of linear and areal objects.

Before making detailed comments on the OS model, it is useful to make the following general observations. Given the size of OS data holdings, the addition or deletion of individual fields in a table would have a major impact on storage. For example, in the largest table in the 1:50 000 Demonstrator, namely the Coords relation with 120,520 records, each 4 byte integer field results in half a Mbyte of storage. There is thus a need to scrutinise every field before inclusion within a database for Great Britain. In relational theory, normal forms not only facilitate our understanding of the structural relationships between database entities, but they can also result in space savings. In addition, integrity of update of the database is easier to achieve if as many relations as possible can be held in one of the higher normal forms as appropriate. Thus, conceptual analysis, update and storage are assisted by use of normal forms.

Unfortunately, normal forms increase the number of join operations, resulting in a loss of performance, during data retrieval. Thus, retrieval may be optimised at the expense of storage and update or vice versa. Such conflicts may be compromised by duplication of data but this is not recommended given the storage (and cost) implications and the problems of ensuring consistent update. A more acceptable solution is to compromise the main model towards storage and retrieval and accept the use of an alternate model as and when necessary, for example for update or for chaining networks and boundaries during object definition.

2.2 An Overview of the OS Data Model

The OS data model has the following characteristics:

* In theory, it is a link and node structured model. In practice, the presence of errors in the data contradicts this assumption.

* It is basically a feature-based model with point, line and text features.

* It has been extended to model named and unnamed point, line and area objects. However, the model does not explicitly encode information relating to area topology although such information may be extracted by spatial data processing.

* Areal entities are represented using an object-, rather than region-, defining approach. This approach facilitates rapid retrieval of the geometry of pre-defined objects for mapping. It is not intended to support more complex spatial queries involving the relationships between objects. In our conceptual analysis of the OS model, we therefore do not comment upon the limitations of this approach since this is outside our brief.

2.3 Analysis of the OS Data Model and Suggestions for its Improvement

Spatial data models tend to be complex. But, the experimental model is unnecessarily complex. Different schemes are utilised for performing the same functions, thereby adding to the complexity of the model. Some relations hold duplicate information; these not only have space implications but could also result in update anomalies. Some information is missing and the logical structure could be improved to economise on storage and/or improve speed of access.

In the following sections, we review relevant parts of the OS model and offer our opinions. The percentage figures quoted are based on a subset of the relations in the SSD Database. This subset of 2,934,834 bytes excludes the following relations:

Coords : This occupies 46 % of the database, making the other relations look insignificant. As this relation is not subject to analysis and will be held in a system-specific way, it should not feature in the comparisons.

History, Id_Table, Window_Data, Window_Desc, Area_Tie : These five relations were not provided.

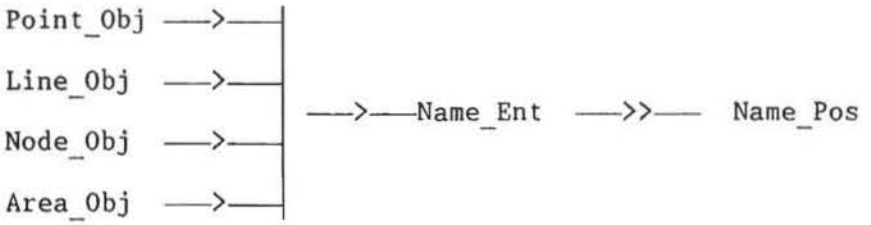
Graphics : This relation appears to be out-of-date and inadequate. It is also very small.

The size of the subset includes the two fields, history and appl. They are not shown in the following examples since we have not eliminated them in our suggestions.

2.3.1 Relations Referring to Names

An overview of the OS Small Scales Demonstrator Data Model is given in Sheets 5 and 6 of Appendix 3.4. The relations, referring to names, are shown in Figure 1.

FIGURE 1 : RELATIONS REFERRING TO NAMES



where

—>— indicates a preferred direction of travel
—>>— indicates a one-to-many relationship

The common key, name_id, is used to obtain the name (from Name_Ent) and its attributes (from Name_Pos) of named objects in the database. Name_id is a global field with unique values across the object relations. Name_id is not unique within Name_Pos since the same name may be displayed several times in different places. For example, the name M1 is displayed along several stretches of the motorway. Name_Pos may therefore be used to define different locations and/or orientations for the given name. In interactive graphics, a map may also be displayed at reduced scales. Name_Pos thus enables the text attributes, such as font and character size, to be defined for a range of scales.

As it stands, the Name_Pos relation is incomplete for representing text attributes and for interactive graphics. Indeed, the OS model in general is incomplete with respect to graphics since it does not provide for explicit modelling of all the graphic attributes of objects. Assuming that OS intends that the formulation of data structures for graphics is the user's responsibility, we have not included detailed comments on Name_Pos.

2.3.2 Object Identifiers instead of NAME-IDs

The following observations may be made.

* Figure 1 indicates that, given the name_id within an object relation, it is easy to locate the text of the name. However, if the database is accessed through a name, then all four object relations will have to be searched since there is no explicit data on the type of object the name refers to. No doubt, the type of object can often be inferred. But, this is an unnecessary burden on the system, given that the space overheads for coding the object type are minimal (it can be as little as two bits per text label).

* The OS model assumes that text is an attribute of geometrically defined objects. However, a map database can include free-standing text. Names, such as "The Pennines" or "English Channel", refer to areas with fuzzy boundaries which are not explicitly indicated on the map. In the OS model, free-standing text can only be recognised as such when its name_id does not appear in the other object relations. Thus, there is an unnecessary overhead involved in recognising and accessing free-standing text.

* There is a further problem. Spatially defined objects need not have names. The current data model assigns a value of zero to the name_id of unnamed objects in an object relation. This is wasteful since the vast majority of area objects do not have names associated with them.

We suggest the following.

[1] Use a composite prime key in place of name_id within Name_Ent.

The Name_Ent relation will now consist of:

obj_id obj_type text

where: obj_id — may be unique in the database.
obj_type — is enumerated as Point and Node, (see Section 2.3.3 below), Line, Area or Text. The space overheads are implementation dependent but a two-bit field is sufficient for the proposed usage.
text — is the text as in the OS model

[2] The object relations will only use obj_id instead of name_id.

This schema offers the following:

* Given an existing name, we can immediately identify its type and thus the access path to its geometry.

* Given an obj_id, it would be possible to establish whether the object has a name and access its text.

* The obj_id of unnamed objects will not appear in the Name_Ent relation.

* If there is a need for storing some other information about free-standing text, a FreeText_Obj relation can be created. Since Name_Pos provides all relevant information on free-standing text, and the type of the latter is indicated by obj_type, there is no reason for creating a FreeText_Obj relation at the moment.

* Use of obj_id in place of name_id offers space savings (see Section 2.4).

2.3.3 Point and Node Objects

The OS model uses two relations for point objects, namely Point_Obj and Node_Obj (see Sheets 3 and 4 of Appendix 3.4). Point objects, which are detached from line features, are recorded in Point_Obj,

whilst those which happen to fall on a line feature are recorded in Node_Obj. The following relations exist for Point and Node objects.

Point_Obj

po_ref	:	Morten key
x, y	:	National Grid co-ordinates
z	:	height of a point
name_id	:	foreign key to Name_Ent relation
model_key	:	foreign key to Model relation

Node_Obj

node_ref	:	Morten key
name_id	:	foreign key to Name_Ent relation
model_key	:	foreign key to Model relation

Node_Geom

node_ref	:	Morten key
x, y	:	National Grid co-ordinates
z	:	height of a node

Node_Str

node_ref	:	Morten key
line_id	:	foreign key to Link_Geom
link_sort	:	{function not clear}
level	:	level of link for route planning
azimuth	:	angle

The following observations can now be made.

* All detached points have semantic attributes.
* Node Objects are points, with semantic attributes, which fall on a line. Thus features, such as bridges, electricity pylons, chimneys and spot heights, may be classed as either Point_Obj or Node_Obj.

Smith and Turner (1987) considered point features to be either a separate class of entity, or to be degenerate links with no length or to be nodes with no associated links. This interpretation encourages the separation of point and node objects on the one hand, and the designation of some point objects as a subclass of node objects on the other. The resulting schema is not only untidy and difficult to comprehend, but it also detracts from performance. Two separate queries are needed to retrieve all objects in one feature class. Thus, to retrieve all objects (e.g. spot heights or bridges), both Point_Obj and Node_Obj will have to be searched. Also, it will be necessary to join Node_Geom and Node_Obj for retrieving the geometry of Node Objects.

We believe that there are conceptual and practical advantages in imposing an alternative interpretation which treats node objects as specialisations (a subclass) of point objects instead of vice versa.

* Node objects were defined to facilitate applications, such as route planning and in-car navigation. Thus, representative points for towns, villages, suburban areas and other settlements, have been located on existing, or specially created, nodes on the network of road links. The names of these settlements are accessed through the name_id (now obj_id) attribute of Node_Obj. Given the name, it is also possible to find the line objects to which the node object is attached through the Node_Str relation, using the key field node_ref. Node_ref may also be used to find node objects from the link relations and from Node_Str during routing applications.

  Thus, the retrieval of the geometry of Node_Objs, and the establishment of the existence of semantic information for topological nodes, require a join of Node_Obj and Node_Geom. Since only a small proportion of all topological nodes have semantic attributes recorded in Node_Obj, this is an unnecessarily time consuming operation.

* The settlement names accessed by Node_Obj and Area_Obj (through area seeds) form mutually exclusive sets. It is not possible, therefore, to infer whether a node object has an associated boundary (i.e. an areal entity). In some situations it may be possible to establish this correctly through spatial analysis, such as point-in-polygon tests. But, there are problems of interpretation since node objects may be no more than point abstractions, which happen to fall within a built-up area. This would most certainly be the case with suburban nodes. Thus, as indicated on Sheet 6 of Appendix 3.4, there is no direct connection between Node and Area objects. Since the OS model is orientated towards selective applications, we have not pursued the implications of this observation.

Based on these observations we suggest the following:

[3] Combine Point_ and Node_Obj into one relation, called Point_Objs. The Point_Objs relation has the following fields:

obj_id : this replaces node_ref in Node_Obj, point_ref in Point_Obj and name_id in both relations. Note that obj_id can be made a unique key; Morten keys cannot be assumed to be unique especially when user-defined coverages are overlayed.

x, y, z : as before (the x,y,z values for nodes are extracted from Node_Geom).

model_key : as before

n_p : a new field which distinguishes between point and node objects.

This schema is more comprehensible, compact, flexible and extensible since it allows other types of point objects to be distinguished if necessary. The only overhead is the n_p field, which requires minimal storage in the current application. The po-ref/node_ref field may be retained as po_ref if the system supports spatial sorting and clustering.

In the Node_Geom relation, height values were only recorded for node objects, such as spot heights and bridges. The majority of nodes do not have height values. By moving height values to Point_Objs as suggested above, some 23 kbytes of storage may be saved.

[4] Delete the Node_Geom relation as suggested by Smith and Turner (1987, p6).

If the node height values are transferred to Point_Obj, as suggested above in [3], then all the information in Node_Geom is redundant and can be extracted when necessary from Link_Geom and Coords. This will save circa 285 kbytes (304 bytes less circa 19 kbytes for duplicating the x and y co-ords in Point_Objs as suggested in [3]). Unless po_ref is retained in Point_Objs, the start_ and end_node fields of Link_Geom must use x and y co-ordinates instead of Morten keys.

[5] The structure of the Node_Str relation can be changed to economise on storage.

Most of the information held in Node_Str is duplicated or calculated from other relations in the database. The Link_Geom relation records start_ and end_nodes for each link. These are held in inverted form in Node_Str, where for each node, there is a list of all the links that start or end there. The azimuth field of Node_Str can be derived from the co-ordinates of the link.

The only extra information in Node_Str is provided by the level field, which records the height of the link where it joins the node. This information can be retained by using a trimmed version of Node_Str, namely Node_Level, which has the following structure.

Node_Level

x,y (or node_ref)	line_id	level

This structure trims storage horizontally (by removing fields) and vertically (by removing records) since relatively few tuples provide information on level.

The redundant elements of Node_Str are useful for the chaining of links into line objects and boundaries, for edit operations and applications which involve network analysis. However, an ephemeral Node_Str relation may be created for localised subsets of data, as and when required, in real time.

[6] The ephemeral Node_Str relation must include a direction field.

The direction field, currently missing in the OS model, indicates whether a node enters or leaves the node. For each link, which loops back on itself (there are 739 closed links in Node_Str), the Node_Str relation contains only one record. It is impossible to establish from the database whether the azimuth relates to the angle at which the node enters or leaves the node.

The suggestions, relating to Point_Obj, Node_Obj, Node_Geom and Node_Str, result in a more elegant and storage-efficient structure for archival purposes.

The size of the original relations were:

Relation	Size in bytes	Percentage of SSD Data, excluding Coords
Point_Obj	3,400	
Node_Obj	52,552	
Node_Geom	311,234	
	372,186	12.7 %
Node_Str	914,688	31.2 %
TOTAL	1,286,874	43.8 %

The size of the proposed relations:

*	Point_Objs - po_ref	57,454	2.0 %
	Node_Level	61,200	2.1 %
	TOTAL	118,654	4.0 %
	SAVINGS	1,168,220	

* 77,438 if po_ref is included, adding 0.6%

There is a time penalty for some operations and applications. But, the creation of anticipated ephemeral relations is a task which could be undertaken in parallel in the future without noticeable degradation of performance.

2.3.4 Line Objects

In the OS model, line objects are named, ordered collections of links. The relations corresponding to line objects are shown in Sheet 2 of Appendix 3.4. The relations under scrutiny are:

Line_Obj (provides access to attributes of line objects)

line_obj_id name_id envelope

Line_Tie (provides access to ordered collection of links)

line_obj_id lt_item line_id lt_sort

Line_Ent (permits access to feature codes for links)

line_id model_key

Note the following observations:

* Each line object has a unique key number, line_obj_id, for each object.

* A foreign key, name_id, is used to reference the name of each line object.

* The geometry of the line objects, in Link_Geom and Coords, are accessed using two fields in Line_Tie, namely line_id and lt_sort; the latter indicates the order in which lines should be retrieved.

* The field lt_item in Line_Tie is used to indicate higher level objects, which consist of other objects. These super objects will be indicated by a lt_item value of '0'.

During our visit of December 8th 1987, the proposal to replace Line_Obj and Line_Tie with Line_Ent, for reasons of space savings, was outlined. The structure of Line_Ent is as follows:

line_id model_key name_id ....

The analysis of this proposal takes account of the following:

- Every line object in the SSD database has a text label. As discussed in Section 2.3.2, it is possible to record unnamed line objects, if necessary, by using an obj_id which does not appear in the Name_Ent relation.

- However, only 21.5 % of links in the SSD database have text labels. Examples of links without text labels are some woodland boundaries and electricity transmission lines.

- The model must provide for multiple feature codes.

- The sort order, lt_sort, is useful for speedy retrieval and display of line objects as one entity. The display of fragmented

lines in an ad-hoc order not only detracts from the quality of the map user interface in human-computer interaction, but it also has performance implications at the levels of both software (increase in number of polyline calls) and hardware (increase in pen-up and pen-down movements).

- In this section, we accept the OS decision to remove the envelope data, whilst recognising their value for spatial search (see Section 4).

We therefore believe that the proposed changes have the following weaknesses.

* There is an implicit, incorrect assumption that the distinguishing property of line objects is their name. A line object is essentially an ordered collection of links, which need not have a name. For example, unnamed roads and paths form legitimate and useful objects.

* This false assumption leads to the proposition that it is possible to retrieve a line object's constituent links by the name_id. This poses problems. For, all unnamed line objects will have the same name_id, namely zero, and cannot be distinguished.

* This schema would only be appropriate if the majority of links have names. As pointed out earlier, the majority of links are unnamed. Thus, this schema will increase, rather than save, space requirements because the name_id value has to be recorded for all links, including unnamed links. Whereas Line_Obj and Line_Tie together occupied 75,190 bytes, Line_Ent now requires an additional 77,190 bytes to record the 4-byte name_id.

* The above schema allows named line objects to be retrieved as a collection, but not as an ordered collection. The information on ordering, recorded by lt_sort, is lost. Fairly exhaustive and time consuming processing therefore will have to performed repeatedly for some basic operations, such as finding the start of a line or identifying the branches of a network. Thus, the elimination of lt_sort will diminish both performance and the quality of the user interface as explained earlier.

* Multiple feature coding will reduce Line_Ent to second normal form since the name_id field will be repeated even if it is null, leading to a requirement for further unnecessary storage.

We suggest the following:

[7] The line object should be viewed as an ordered collection of links, with the emphasis on ordered.
These objects may be named or unnamed. This definition is consistent with the existing OS view of point and, especially, area objects (see Section 2.3.5).

[8] Retain the original Line_Ent relation.

[9] Combine the two relations Line_Obj and Line_Ent into a smaller one.
The new relation, Line_Obj_Tie has the following fields:

obj_id lt_sort line_id

Obj_id forms a unique key for each line object, which may also be used to reference its name. If the object has no name, then its obj_id will not appear in Name_Ent. In the given data set, this relation requires 66,912 bytes, compared with an additional overhead of 77,190 for the new, proposed version of Line_Ent.

[10] Use a new relation, namely Hg_Line_Obj, to model super objects.
Hg_Line_Obj has the following fields:

obj_id sub_obj_id

The storage implications of these changes can now be assessed.

Size of original relations:

Line_Ent	233,112	
Line_Obj	4,096	
Line_Tie	71,094	
TOTAL	308,302	10.5 %

Size of relation proposed by OS:

Line_Tie	310,816	10.6 %

Size of relations proposed by us:

Line_Ent	233,112	
Line_Obj_Tie	66,912	
Hg_Line_Obj	0	
TOTAL	300,024	10.2 %
SAVINGS	8,278	(compared with original model)
	10,792	(compared with new proposal)

2.3.5 Area Objects

Within the SSD database, an area object is viewed as a named or unnamed, ordered collection of links, which forms the outer boundary of a simply connected region. As observed in Section 2.2, the model does not encode area topology and there is no provision for explicitly modelling holes (i.e. excluded regions). Disjoint regions are regarded as sub-area objects with identical names.

Area objects are modelled using the following relations (In Sheet 1 of Appendix 3.4, relation Bndry_Ent_Tie has been erroneously omitted):

Area_Obj (records attributes of area object)

area_ref x y model_key name_id bndry_id envelope

Bndry_Ent_Tie (provides access to ordered collection of links in boundary)

area_ref bndry_id bndry_sort l_r line_id model_key

Area_Tie (for accessing disjoint parts of area objects)

area_ref sub-area_ref

The following observations were made.

* The area object is taken to mean the area within an enclosing boundary. This is conceptually inaccurate but was regarded as sufficient by OS staff. Areas with holes (an example is the built-up area of Sheffield, which includes one hole) are sometimes

represented by bridging links, which connect the outer boundary to the hole. This method has some weaknesses since it is inadequate for modelling complex regions with several holes. Islands in lakes, which are geometrically holes within the lake, remain unconnected and independent within the SSD database. The enclosing boundary is represented by bndry_id.

* The key, area_ref, is an 8-byte Morten key formed of the x,y co-ordinates of an area seed. It is used to sort the Area_Obj relation so that data for adjacent areas on the map will tend to cluster together on disc, resulting in more hit records and fewer I/O operations.

* Using model_key and name_id, a feature code and text label can be associated with the area_seed.

* Bndry_Ent_Tie is used to access constituent links, using the key field bndry_id. Bndry_id is unique in Area_Obj but each bndry_id in Area_Obj has only one corresponding value in Bndry_Ent_Tie. The remaining records form an ascending progression of bndry_ids and a safety gap of two numbers flags the end of each boundary. The order in which links should be traversed in indicated by bndry_sort. The values for bndry_id correspond directly to those of bndry_sort. Bndry_Ent_Tie is thus sorted on bndry_sort within each boundary.

* The l_r field in Bndry_Ent_Tie indicates the direction of digitising, by indicating whether the area object lies on the left or right of the link. This information is used to reverse the link when chaining boundaries for area calculations and shading. This field does not indicate the direction of the boundary (see Visvalingam et al, 1986), which is missing in the OS model; this field cannot therefore be used to distinguish between enclosing boundaries and holes.

* The field, model key, in Bndry_Ent_Tie records the feature code of the boundary. It appears to replicate the model_key recorded in Area_Obj.

We suggest the following:

[11] Drop model_key in Bndry_Ent_Tie.

It reduces Bndry_Ent_Tie to second normal form without adding any information. Since this relation does not have an envelope field it can only be used to extract all boundaries belonging to a feature type in the database. This saves 43,592 bytes.

[12] Use a re-defined bndry_id as the key for all three relations.

There are two unique keys in Area_Obj, namely area_ref and bndry_id. Area_ref is an 8-byte field which replicates the x,y information. If a database system supports clustering, then Area_Obj will be sorted on area_ref. Since Bndry_Ent_Tie and Area_Tie need not be sorted on area_ref, its usage in these two relations cannot be justified, especially since it is an 8-byte field. Bndry_id can be constrained to be unique but area_ref need not be unique when maps are digitally overlayed.

Bndry_id is therefore a more useful key if its usage in Bndry_Ent_Tie is redefined. The ascending progression of bndry_id values in this relation, which basically replicates information in bndry_sort, is extremely inconvenient. It is difficult to access records belonging to the same boundary using bndry_id. It is not usable within a relational query and its value is diminished because it cannot be used to join Bndry_Ent_Tie and Area_Obj. Also this schema can lead to inconsistencies during update. For if more than two lines are added to a boundary, then a new bndry_id will have to be used to start a new progression and all instances of the old bndry_id will have to be replaced in the other tables which record it.

Bndry_id should have a unique value for each boundary in Bndry_Ent_Tie, which equals the value in Area_Obj. Bndry_id can now be substituted by obj_id in all relations.

This allows us to redefine the relations as follows:

Area_Obj

obj_id x y model_key envelope

Bndry_Ent_Tie

obj_id bndry_sort l_r line_id

Area_Tie

obj_id sub-obj_id

As a result,

* Area_Obj has been slimmed; if preferred, area_ref may be retained for purposes of sorting.

* It is now possible to record unnamed area objects (and there are several) without wasting space unnecessarily. The original Area_Obj relation took 109,200 bytes. Our version (without area_ref) takes 75,600 bytes.

* Space is economised in Bndry_Ent_Tie since both area_ref and model_key have been omitted. Storage reduces from 316,042 bytes to 141,674 bytes.

* By using obj_id instead of area_ref, storage for the key field can be halved for Area_Tie. Although the model provides for detached areas we note, for example, that the three detached parts for the County of Derbyshire have been bridged at the map edge to form one boundary with one area seed.

The storage implications of these changes may be summarised.

Size of original relations:

Area_Obj	109,200		
Bndry_Ent_Tie	316,042		
Area_Tie	0		
TOTAL	425,242	bytes	14.5 %

Size of proposed relation:

Area_Obj	92,400		
Bndry_Ent_Tie	141,674		
Area_Tie	0		
TOTAL - area_ref	234,074	bytes	8.0 %
(TOTAL + area_ref	250,874)	bytes	8.5 %
SAVINGS	191,168	bytes	

2.4 Concluding Comments

1. We believe that the primary advantage of the proposed changes lies in the uniformity and simplicity of the resulting model. Relations which perform similar tasks assume a similar structure.

   Hg_Line_Obj and Area_Tie (which form composite objects) have the structure,

   obj_id sub-obj_id

   Line_Obj_Tie and Area_Obj_Tie (which chain links) have the following structures,

   obj_id lt_sort line_id
   obj_id bndry_sort line_id l_r

   Thus, the model is easier to understand and learn compared with the current OS model. Consequently, it would be easier for new staff to become productive in a shorter time.

2. The proposals result in a space-saving model. Ignoring Link_Geom and Line_Ent, which remain unchanged, the savings are as follows.

	OLD	NEW
Point Objects	1,286,874	118,654
Line Objects	308,302	300,024
Area Objects	425,242	234,074
TOTAL	2,020,418	652,752

The amended relations require 32.31% of the storage required by the original relations. This reduction in storage has been achieved by eliminating redundancy. Performance can be improved by retaining some redundancy. However, the relevant relations may be generated at run time and need not be recorded in the archive.

3. We have examined the conceptual implications of the OS model and have suggested a conceptual schema appropriate to stated internal requirments within OS. Other performance related adjustments are discussed in Section 4. We wish to stress that the revised model is still a pragmatic model, and that it is inadequate for general purpose digital cartography, let alone Geographical Information Systems.

4. The database should make explicit the presence of holes in objects. This may be achieved by an automatic process which always chains enclosing boundaries in anticlockwise, and holes in a clockwise direction.

5. Although the SSD includes manually extracted objects and multiple attribute codes, it remains a cartographic base mainly because low priority features are not shown in their entirety. Inclusion of information on the relative priority of features would not only assist mapping but it is also essential for automatic object creation.

3. ERRORS

Since we did not receive comprehensive documentation of the OS 1: 50 000 SSD model, our understanding of the data and of the model was reliant upon experimental exploration of the data by the student. Some errors in the SSD database caused problems of interpretation and impeded this process. We recognise that a catalogue of errors within an experimental database is of limited value. However, some of the errors are listed below for two reasons. Firstly, at least one organisation, which had acquired and had carried out detailed trials with the database, decided against using it within its demonstration suite because of errors. We believe that OS should endeavour to supply correct experimental data since the purpose of creating such

data is otherwise undermined. Secondly, we hope that the description of some of the procedures which led to the detection of errors would assist OS staff with the difficult task of validating data. It must be stressed, however, that our brief did not include validation of data and the observations presented here are by-products of processes devised to meet other project objectives.

3.1 Errors in Feature Coding

Eight water feature links form part of A roads.

River objects include 3 links which are feature coded as railway dismantled, land boundary fenced and land boundary unknown respectively. The last of these could belong to multiple objects but we could not easily check this since we did not receive multiple attributes.

3.2 Replicated values of bndry_id in Bndry_Ent_Tie and duplicated area seeds

* Bndry_ids 49153 - 49212 and 49281 - 49305 are duplicated for two different boundaries.

* Bndry_id 49306 occurs three times

* The data suggest that there are 2023 boundaries because there are 2023 unique area-ref values in Bndry_Ent_Tie, which are all accessible from Area_Obj and there are as many records in Bndry_Ent_Tie with a bndry_sort value of one. But, Area_Obj has 2026 unique non-zero values of bndry_id; only 2004 of them correspond to bndry_id values in Bndry_Ent_Tie (after adjustment to equivalent values in Area_Obj). Thus 22 values of bndry_id are recorded incorrectly in one or the other of the two relations.

* It appears that some of the 2023 bndry_ids in Bndry_Ent_Tie have no corresponding values in Area_Obj. This can be verified. For example, a relation was created to contain the following fields, namely bndry_id from Area_Obj and other fields from Bndry_Ent_Tie, using area_ref as the common field. This is based

on the assumption that area_ref is correctly recorded in both relations. This relation contained 10,862 records (36 less than its counterpart, Bndry_Ent_Tie) and 2013 unique bndry_ids (10 less than in Bndry_Ent_Tie). But, because of the way this relation was created, all records are accessible from Area_Obj.

* Bndry_ids do not always form an ascending progression as specified by the model. Most of the larger boundaries, namely administrative areas and built-up areas (e.g. Sheffield City) have errors in the chaining of links. These areas would not area fill correctly. When the recorded sequence of boundaries was checked using topological rules for chaining, 46 of the 2013 objects failed the test. These erroneous chains include seven of the ten administrative area boundaries and the Sheffield City boundary (48793). This boundary also contains two area seeds.

3.3 Error prone method for chaining boundaries

Boundary chaining requires reversal flags. The 1:50 000 data model uses a l_r field to indicate whether the object is to the left or right of the link. However, the boundaries themselves do not have a consistent direction. Instead, the l_r code for the first link in the chain acts as a local reference and subsequent links are reversed if their l_r codes do not correspond to that of this first link. Consequently, the l_r codes do not have an independent and consistent meaning.

Although, the automatic re-organisation of manually chained boundaries is a trivial task, the existing rules are more difficult to apply manually in a consistent manner. The presence of numerous errors in l_r coding supports this proposition. If boundary chaining is to remain a manual process, then boundaries should be chained so that the object being defined is always to the left. This would mean that all links, with an l_r value of 'R' will be always reversed; at present, the meaning of these values is context dependent and thereby introduces unnecessary complexity.

If such a rule were adopted, then it would be easy to include software which would automatically generate the l_r codes for the

given set of ordered links. This rule will also allow the OS to upgrade the data model at a later stage to include descriptions of holes in areal objects (see Visvalingam et al, 1986; Wade et al, 1986). For, then all enclosing boundaries and holes will have anticlockwise and clockwise direction respectively. This rule may be used with object defining, and not just region defining, models.

3.4 Violation of the link and node structure

The following types of errors were detected by an automatic polygon chaining process, described in Wade et al (1986) and Kirby et al (1989), when applied to a 6 x 6 km square in the north west corner of Sheet 111. This sample covers 2.25 % of the area of Sheet 111 and includes 8.8 % of the nodes in the SSD database. The process attempted to form primitive regions and check the relationship of area seeds to primitive regions.

3.4.1 Non-coincident values for end points of links

There were several instances of this.

3.4.2 Undershots and overshots

These could be due to errors in digitising or restructuring. Errors of about 3.16 metres occur, for instance, when a side road stops at the edge rather than centre of a dual-carriage way. Whereas it is difficult to judge whether undershots of this kind constitute an error, overshots are definitely errors since they violate the link and node structure. The existence of such errors does not reflect too well on the data re-formatting package.

3.4.3 Two seeds in the same polygon

This also appears to be due to errors.

3.5 Recommendations

[13] OS should commission software for validation of the line topology as suggested by Visvalingam et al (1985) and for automatic

chaining of primitive regions as suggested by Wade et al (1986) and Kirby et al (1989) **to ensure that the database does indeed conform to a link and node structure.** Ideally, such validation should be undertaken by the data restructuring package.

[14] **OS should at least implement some simple procedures for cross-checking the logical consistency of semantic data.** This could be done with a database query language after object creation. All that is needed is the extraction of links belonging to specific object classes to check that their feature codes are in a valid/invalid list. If trivial errors could be detected and eliminated automatically, manual effort may be concentrated on the more obscure errors.

[15] **Boundaries should be chained in a consistent manner with the area being defined always to the left of the direction of chaining.** This will ensure that a l_r value of 'R' always flags a need to reverse the link and that the model may be easily upgraded to record holes in area objects at a later stage.

4. SOME PEFORMANCE ISSUES

One of the aims of the project was to optimise the performance of load, search and update operations. This requirement was specified in July 1987 when RDB was used for the 1:50 000 database. By December 1987 experience, with the large-scale pilot study on the Britton-Lee system, revealed that previous performance problems were due to the limitations of the RDB system. The original specifications relating to performance were no longer considered to be of high priority given the capabilities and power of the Britton-Lee system. Nevertheless, these discussions prompted the student to take note of performance related issues. These include:

* the impact of rationalisation and of application-orientated adaptation of the data model (see Section 4.1)
* stategies for partitioning a spatial database (Section 4.2)
* strategies for structuring individual relations (Section 4.3)
* Structuring and indexing for spatial search (Section 4.4)
* Strategies for coping with ordered tuples (Section 4.5)

4.1 Rationalisation and application-orientated adaptation of the Data Model

The rationalisation of the OS model and the introduction of obj_id in Section 2, not only result in space savings, but also in some improvements in:

a) the performance of name-based searches

b) retrieval of links forming area boundaries.

Performance may be further improved by **composition** or **de-normalisation** if the patterns of usage of the database are known or predictable. De-normalisation involves the collection of fields from several relations into one relation so as to avoid join operations. Composite tables of this type are also known as stored-join relations.

For example, within the VAX 8200 Ingres environment, Sekouris (1989, Section 6.2) conducted a series of experiments using the embedded query language EQUEL to establish the best strategy for retrieving the geometry of multiple features. This involves a query of the form:

```
range of m is Model
range of le is Line_Ent
range of lg is Link_Geom
range of c is Coords

retrieve into tmp_lines (m.feature_code, lg.line_id, lg.no_coords,
                         c.seqno, c.x, c.y)

where     lg.xmin < XMAX and lg.xmax > XMIN
     and lg.ymin < YMAX and lg.ymax > YMIN
     and (   m.feature_code = "AB"
          or m.feature_code = "RD"
          or m.feature_code = "WR" )
     and m.model_key = le.model_key
     and le.line_id  = lg.line_id
     and lg.line_id  = c.line_id
```

This is a common query in mapping applications and it is time consuming since it involves a join and search over four relations, three of which (Link_Geom, Line_Ent and Coords) are relatively large relations.

The search is optimised by changing the Line_Ent relation. New_Line_Ent now contains the following fields:

line_id : primary key as before
model_key : as before
envelope : from Link_Geom
feature_code : from Model

The envelope fields may now be deleted from Link_Geom. It must be stressed that the inclusion of the envelope fields in New_Line_Ent makes the latter unsuitable for representing multiple feature codes. It would result in replication of the envelope fields for each model_key value and result in undue wastage of space. However, within the 1:50 000 database we received, each link has only one semantic attribute, namely that corresponding to its display. Other attributes of links are implied when links become constituent parts of other linear or areal entities. For example, when a road which is also a woodland boundary is displayed as a road, the model key corresponding to the type of road is recorded in Line_Ent. The fact that the link also forms part of the boundary of an adjoining woodland is digitally and visually implied by its presence in Bndry_Ent_Tie and the adjoining area shading respectively.

If this approach is to be retained, then for mapping applications New_Line_Ent is to be preferred to Line_Ent. It avoids a join over four relations. Instead, a more efficient three step retrieval may be adopted, namely:

(1) Search the single relation New_Line_Ent, for links within the specified envelope and put them in a temporary relation. This is because Ingres scans the whole relation for each required feature_code. The temporary relation is often likely to be much smaller than New_Line_Ent.

(2) Search the temporary relation for the required features.

(3) Join the resulting records with Coords to retrieve the geometry.

Sekouris (1989, Chapter 6) describes the experiments which led to this strategy and similar experiments should be repeated by OS

since data base management systems vary in their behaviour and our observations cannot be generalised with confidence beyond the limits of the VAX 8200 Ingres environment. Also as demonstrated by Sekouris, temporary files expedite search in some queries but diminish retrieval times in other contexts. However, with sufficient knowledge and contrived experience of an environment, it should be possible to predict relative performance and formulate optimal strategies.

4.2 Partitioning the Database

The database may be structured at two levels, using different methodologies at each level. It is partitioned at a high level into sets which are mapped onto different physical regions on storage media. At a lower level, appropriate storage structures have to be selected for each relation. This section deals mainly with high level partitioning of spatial databases.

Spatial data exist within the multi-dimensional framework of space (which may be two or three dimensional), time and other aspatial dimensions. (In the OS 1:625 000 experimental database, three aspatial dimensions were used, namely administrative boundaries and water features, relief and settlement and communications.) The problem of partitioning is one of identifying suitable subsets for mapping onto different areas of storage. These may be devised using idiographic criteria or systematic rules. Some of the considerations involved in the design of basic spatial units (Visvalingam, 1988a & 1988b) are relevant to the partitioning of databases.

Some Geographic Information Systems, based on spatial statistics, tend to partition the database on the time dimension since thematic databases tend to correspond to snapshots in time. But, many Land Information Systems need chronological accounts of spatial change. OS requirements for maintenance of the topographic archive justify the partitioning of the database on spatial and thematic criteria as proposed in the 1:625 000 experiment. The present OS strategy is to time stamp records; deleted records are not removed from the database. There is,

thus, scope for inclusion of efficient procedures for the management of change information when the archive is periodically rationalised.

Having selected the principal dimensions on which data is to be partitioned, it is then necessary to identify the specific criteria to be used. The 1:50 000 thematic data cannot be partitioned in the same way as the 1:625 000 data because some features, such as rivers, do not form a complete network and need to be viewed in the context of other features, such as settlements and communications. Consequently, the most useful dimension for partitioning the database, is the spatial dimension. But space may be segmented on different criteria, both ad-hoc and systematic. The framework provided by traditional map sheets is unsuitable mainly because they overlap. The 1:625 000 database was partitioned into libraries corresponding to 100 kilometre squares on the National Grid. This will result in partitions of different sizes on disc, owing to the varying density of data. This is not a serious problem at the level of partitioning data; but, others are experimenting with data adaptive schemes for distributing data over physical blocks at lower levels. However, a data adaptive scheme yields irregular and unpredictable boundaries; i.e the boundaries cannot be calculated and have to be explicitly recorded.

Several researchers are now experimenting with the use of linear quadtrees (Gargantini, 1982)- ie. quartenary, as opposed to denary divisions, of space - for partitioning, ordering and physically blocking vector data (Abraham, 1988, personal communication; Ibbs and Stevens, 1988). Quadtrees have been used successfully for compact storage and speedy processing of raster data. However, we are inclined to agree with OS staff (Neil Smith, personal communication) that there is insufficient evidence to commend the use of quadtrees with vector data to the OS at present.

All systematic divisions of space, whether denary or quarternary, result in arbitrary boundaries. Quadtree partitioning of space may be no more convenient than other schemes. Advantages are believed to accrue from the observation that quadtree addresses, consisting of Morten codes, map onto the Peano curve. The Peano

curve is one of a family of space-filling curves which pass once through every location of data space and which thus transform space into a line and vice versa. This property was used by Morten for ordering spatial data on disc so as to reduce access time. Space filling curves are useful therefore for mapping spatial entities onto computer storage, which is essentially linear. But, these anticipated advantages have not been convincingly demontrated with respect to vector data.

Quadtree indexing of vectors involves unrealistic overheads. For each cell of the tesselation, every edge (which starts, ends or passes through the cell) has to be directly or indirectly indexed. Sekouris has calculated that Ibbs and Stevens (1988) use 3 Megabytes of storage for a data adaptive quadtree for the 6166 links for just roads and rivers in the south west quadrant of Sheet 111 at the 1:625 000 scale. As pointed out in Section 2.4, all line objects in the entire 1:50 000 database occupy only 300,024 bytes.

Another point to note is that such detailed indexing of vectors is advantageous when there is a need to retrieve relatively few segments from long links. In small scale applications, using the DoE/SDD 1: 50 000 database of the hierarchy of administrative areas, for example, quadtree indexing could speed up zoom and window operations. However, the OS database contains a large proportion of very short links and the cost/benefits of indexing OS data at the level of individual line edges, or even Morten-sorted individual vertices (Abraham, 1988, personal communication) are not immediately apparent from published studies. Ibbs and Stevens (1988) did not sytematically compare the performance of windowing operations but have merely provided retrieval times for the whole database.

[16] In our opinion, the present OS strategy for partitioning data on a denary basis constitutes a reasonable approach but quadtree-based ideas for organisisng data should be investigated and compared.

4.3 Strategies for structuring individual relations

Strategies that are used for structuring partitioned data are now considered. Performance of a database is a function of the number of I/O operations or page accesses. When relations are over a critical size, which is system dependent, sequential search becomes inefficient and storage structures are used for locating hit records. Different data base management systems provide different facilities; the VAX 8200 Ingres system only provides heap, hash and ISAM methods for structuring records on fields of primary interest. This is because the intervention of the UNIX file system prevents precise control over the physical location of data. Thus, B-trees and clustering are not supported on our system.

Different storage structures favour different types of retrieval and specific methods, appropriate to projected use, have to be selected for each relation in the database. The reasons for selecting a storage structure must be made explicit. Since a relation is structured to the advantage of retrievals based on one field primarily, it is equally necessary to make explicit the reasons for choosing that field. Sekouris (1989) discussed his selection of structures for the relations described in Section 2 of this report. This is not described here since they are specific to our environment and because he was mainly concerned with optimising retrievals for exploring the database and for mapping; i.e. for effecting retrievals from a static database. This was because he had not completed his work on automatic object creation, which would have provided useful information for checking the integrity of updates. Consequently, no experiments for optimising update were conducted.

However, Sekouris did undertake some fruitful investigations on strategies for optimising spatial search, using envelopes. The results are discussed below.

4.4 Indexing and Spatial Search

The use of storage structures is sufficient when a relation is accessed primarily on one field. Searches on other fields are

improved by creation of secondary index tables. These index tables, in turn, can be structured to provide rapid access to the hit records in the primary relation. On executing a query, a database management system will seek to optimise performance by searching an index table rather than the primary table.

A relational database may be fully indexed but index tables optimise search at the expense of storage and update. Large index tables pose problems during insertion and deletion of records. For every record inserted/deleted in the primary relation, an entry has to be managed in its correct position in the corresponding index. Thus, storage for indices must be justified; i.e. they must yield significant gains in performance of functions. On some systems, such as the Britton-Lee system, indices may be created reasonably quickly as and when required and deleted when no longer needed.

Whether indices are permanent or transient, they need to be designed. The following section (4.4.1) provides some general comments on the OS strategy for indexing the 1:50 000 database in a RDB environment. An alternative approach to the indexing of spatial fields is described in Section 4.4.2.

4.4.1 The OS Strategy for Indexing the 1:50 000 Database

1) In the OS data model, there are three relations with envelope fields, namely Area_Obj, Line_Obj and Link_Geom. The OS strategy places all four envelope fields (xmin, xmax, ymin, ymax) in **one** index table. This is an error since this effectively means that only the most significant field will determine the efficiency of spatial search and some spatial queries will not be optimised in any way by the existence of this index. An alternative strategy for indexing spatial fields is outlined in Section 4.4.2.

2) Bndry_Ent_Sort, which contains area_ref, bndry_sort and line_id, is an index to Bndry_Ent_Tie. This scheme results in wasteful duplication of data. The strategy used with the VAX 8200 Ingres is based on the following lines of reasoning.

For mapping, the geometry of the links which form an area object need to be accessed in the correct order. Thus, it is more useful

to structure (hashed at Hull) Bndry_Ent_Tie on object_id, followed by bndry_sort. With hash structuring, Ingres relations are automatically sorted with the significance of fields decreasing from left to right. This guarantees that the line_ids will be retrieved in the correct sort order. This makes Bndry_Ent_Sort redundant. Note, however, that this has been made possible by the changes made to the conceptual model in Section 2. Also note that the retrieval of all area objects to which a given line_id belongs is not optimised by this structure.

To facilitate the retrieval of the geometry of composite area objects, the Area_Tie relation should be hashed on object_id to enable efficient join operations with Area_Obj and Bndry_Ent_Tie. Similarly, to retrieve constituent links of a line object in the right order, Line_Obj_Tie should be hashed on obj_id (most significant) and lt_sort. The Coords relation was hashed on line_id (most significant) and seq_no to optimise joins with Link_Geom and Bndry_Ent_Tie.

The Names relation was modified to ISAM on the text field to facilitate retrieval by name. Retrieval of text, given an area seed, requires a join of Area_Obj and Names over obj_id. This join is facilitated by indexing Names on obj_id and hashing the index table. This benefits all relations that contain an obj_id field and which need to join with Names to access the text field.

3) Other unnecessary fields, including primary keys, were also placed in indices but this was due to the deficiencies in RDB.

Sekouris calculated that all the indices generated by the OS occupy over 2 million bytes; i.e. they add up to over 81.3 percent of the 1:50 000 database excluding Coords; or half the size of the entire database. Using empirical results, Sekouris argues that the existing performance of spatial search can be maintained with only three indices, requiring a total of 268,616 bytes. This is outlined below.

4.4.2 Spatial Search and the Envelope Fields

Whilst optimisation of performance through storage structures is system dependent, it appears that there may be system independent strategies for improving spatial search. As described in Section 4.2,

quadtrees are believed to offer one such solution. Research on this topic should be pursued through a PhD, rather than a Masters, research programme and Sekouris was not encouraged to explore this theme. He has concentrated, instead, on use of the envelope fields for 'windowing' onto spatial subsets.

Windowing involves the retrieval of the geometry of features which either occur within, enter or cross the specified search window, using a conditional test (see Section 4.1). If we consider the New_Line_Ent relation described in Section 4.1, we could create four indices - one each for xmin, xmax, ymin, ymax - to optimise 'windowing'. (Note that this strategy is different from that used in the OS scheme which placed all four fields in one index). The four index tables would require a minimum of 621,632 bytes for a relation that occupies only 505,016 bytes. Although this would yield performance gains, the overheads involved are exhorbitant. Thus, alternative strategies were considered.

Experiments undertaken by Sekouris suggest that relations with envelope fields need only be ISAM structured on XMIN, YMAX and indexed on XMAX and YMIN to optimise spatial search.

1) Initial Experiments and Results

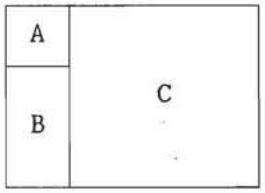
An EQUEL program was written to retrieve all the records within a 6 x 6 kilometre square in the north west corner of Sheet 111. This area represents 2.25 percent of the area of Sheet 111 but it contains 8.3 percent of the nodes in the database. The Ingres query is:

```
range of l is new_line_ent
range of c is coords
retrieve (l.line_id, c.seq_no, c.x, c.y)
where       l.xmin < XMAX and l.xmax > XMIN
      and   l.ymin < YMAX and l.ymax > YMIN
      and   l.feature_code = FCODE
      and   c.line_id = l.line_id
```

The query involves four range searches and a selection in the New_Line_Ent relation and a join with Coords. As described in Section 4.4.1, Coords was hashed on line_id and seq_no. The structure of the New_Line_Ent relation was modified and

performance statistics were collected using the **time** command in UNIX. Note that the statistics were collected in a multiuser environment and provide only approximate indications of performance. The average of three observations, for each of a variety of experiments, are summarised in Table 4.1.

The most significant improvement in performance resulted from experiment 3, in which New_Line_Ent was structured on XMIN. Thus, records which fall outside the vertical strip containing the window can be quickly excluded; i.e. region C is excluded when searching for A in strip A+B in the diagram below.



The records in this strip were then searched serially since the YMIN and YMAX fields were not structured. Note that structuring New_Line_Ent on the two X fields in experiment 5 offered no advantage whilst structuring the relation on XMIN and then YMAX in experiment 6 only produced a slight improvement in performance. This is because the gain in performance accrues mainly from the sort order of the most significant key, XMIN. When the relation was modified to CISAM on XMIN and indexed on YMAX in experiment 7, a further significant gain in response time was observed, mainly as a result of reduced system calls.

2) Further Experiments on Spatial Search

Further experiments were conducted to establish the efficiency of spatial search for other windows and as a result it was found that New_Line_Ent, Line_Obj and Area_Obj (relations with envelope fields) should be structured on fields XMIN, then YMAX with CISAM structured index files for XMAX and YMIN.

Table 4.1 Spatial Search using Envelope Fields - Timing Statistics

Experiment	real	user	system
1	175.2	78.80	9.2
2	136.7	76.00	9.1
3	100.0	52.60	6.6
4	129.5	71.75	9.1
5	102.0	52.40	6.7
6	98.8	52.50	6.1
7	71.0	51.30	5.5
8	103.9	49.70	6.5

Notes on experiments

1 No structure or indices for New_Line_Ent

2 New_Line_Ent modified to CISAM (Compressed ISAM) structure on field XMAX; comparable results when replaced by YMIN

3 New_Line_Ent modified to CISAM structure on field XMIN; comparable results when replaced by YMAX
Note that the improvement on experiment 1 is significant.

4 New_Line_Ent modified to CISAM structure on two fields - XMAX, then XMIN

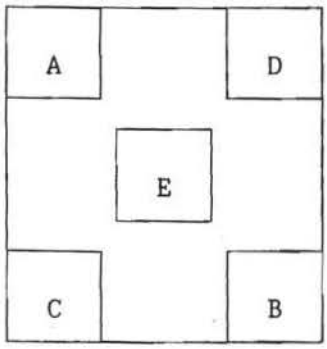
5 New_Line_Ent modified to CISAM structure on two fields - XMIN, then XMAX
Note that the results are comparable to those of experiment 3. Thus, there is no advantage in sorting a relation on more than one envelope field.

6 New_Line_Ent modified to CISAM structure on XMIN, then YMAX.
Note that the timing statistics are only slightly different from those for experiment 3.

7 New_Line_Ent modified to CISAM structure on XMIN, indexed on YMAX
Note that this involves fewer system calls.

8 New_Line_Ent modified to CISAM structure on XMIN, indexed on XMAX. Note the increase in system calls relative to experiment 7.

Using this strategy, the retrievals for windows A and B in the diagram below were similar and comparable to the best results from the foregoing experiments. The results for windows C, D and E were slightly poorer. The statistics for A, C and E are quoted below.



Statistics

Window	real	user	system
A	82.1	48.9	6.1
C	102.0	51.0	6.9
E	106.8	50.5	6.5

Sorting New_Line_Ent on XMIN, YMAX and indexing on XMAX, YMIN favours the extraction of vertical strips from which hit records are found mainly by serial search. If the area covered by the database is elongated in the Y direction, the extraction of horizontal strips would yield better performance and the relation should be sorted on YMIN, XMAX and indexed on YMAX, XMIN.

In this section we outlined the type of experiments which may be conducted to evaluate the value of alternative structures and, especially, of secondary index tables.

[17] **We recommend that the OS undertakes similar experiments to evaluate the value of alternative structures and, especially, of secondary index tables.**

[18] **It would be helpful if OS staff could establish whether the space-efficient indexing scheme proposed by Sekouris provides a system independent strategy for spatial search.**

4.5 Storage of ordered tuples

The relational model requires the explicit recording of the line_id and seq_no of co-ordinates, which are normally implied or calculated when using direct access files. This necessitates space overheads and increases I/O not only because of the large volumes of data, but also because data cannot be directly moved as arrays. This same problem also applies to other ordered lists, such as time series data and other relations within the OS model itself, namely Line_Obj_Tie and Bndry_Ent_Tie. The inadequacy of the relational model for storage and retrieval of data has led some vendors to offer non-relational facilities as part of the RDBMS. These include :

(1) variable length character string fields.

Such a facility is already in use at OS for storing co-ordinates.

(2) access to tuple identifiers

Tuple identifiers (tids) refer to tuple addresses in terms of a page number and offset. When a table is in a heap, tids always increase in value and may therefore be calculated and are never stored. This means that co-ordinates may be stored as in a direct access file but within a RDBMS. Problems can arise with update since the insertion and deletion of records cause changes to tid values. However, records in the OS database are never deleted. Moreover, if the Coords relation is within the DBMS, it is possible to constrain the use of the table to APPEND operations only.

The alternative is to manage the ordered tables outside the RDBMS in direct access files. The advantage lies in the scope that this provides for efficient storage and retrieval of this data directly by applications but there are several disadvantages. If the tables are outside the RDBMS, they cannot be used within the RDBMS's query language and in any relational operations. Also the file cannot be covered by the DBMS's recovery mechanism and the DBMS cannot enforce integrity constraints on the file.

4.6 Summary

The specific contributions of Sekouris may be summarised as follows:

1. With Ingres in the VAX 8200 UNIX environment, Sekouris has established that mapping applications can be assisted by re-distribution of fields in the link relations and use of a de-normalised version of the Line_Ent relation.

2. Sekouris has provided some constructive comments on the OS strategy for indexing.

3. In particular, he has established that spatial search can be expedited by structuring relations with envelopes on one X and one Y field and indexing the relation on the other X and Y fields.

5. USER SURVEY

It was evident from telephone conversations with potential users, introduced by the OS, that very few had obtained Sheet 111 and that even fewer had the time and/or technical skills to evaluate it. The exception was Rob Ward of Derbyshire County Council, who had already submitted a report to the OS. Another user, close to commercial exploitation of an in-house developed system, did not wish to be interviewed. Sekouris visited four organisations, namely the National Freight Consortium (NFC), Thames Water Authority, the Institute of Hydrology (IoH) and Derbyshire County Council (DCC); Visvalingam accompanied him on the two latter visits. In addition, comments from other contacts, who did not wish to be named, are also considered in the following summary of users' perception of the suitability of the relational model and of small scales map data, their expressed requirements and potential uses.

5.1 The Relational Model

Users reactions to a RDBMS and to the Graphical Kernel System (GKS) are similar in many respects. GKS is valued for its conceptual advances. It is seen to have distinct advantages when there is a need to port applications over a diversity of graphics hardware of varying

intelligence and to support a range of styles for graphic interaction. The trend towards firmware and hardware implementation of aspects of this standard and evolution of other complimentary standards at lower and higher levels act as further pull factors. However, where efficiency is paramount, users tend to opt for turnkey systems and manufacturer's packages. The pre-existence of applications software also influences the decision.

Reactions to RDBMS are similar. Very few users understand fully the differences between the various database models. Naive users tend to choose the popular option. The simple tabular user interface, the ease of experimentation with alternative data models, the standardisation of SQL and adoption within database machines favour the relational approach. This approach is optimal for data processing departments in large organisations which support a diversity of applications departments. However, some organisations specialising in network-type applications have chosen, after experimenting with the relational model, to forego its advantages and use more conventional file handling methods for more efficient storage, retrieval and manipulation of ordered data, where such data form the bulk of their database. This decision has been prompted partly by limitations of specific systems and the requirements of pre-existing software but it also reflects the nature of their applications and the known limitations of RDBMS.

Thus, whereas Thames Water and DCC were keen on RDBMS, others including the IoH and NFC were more partial towards other solutions. Some vendors of microcomputer based systems are also opting for ad-hoc solutions to economise on storage and improve speed. Wade et al (1986) were unsure of the appropriateness of the relational model for cartographic data. More recently, others have noted the limitations of RDBMS for CAD/CAM, computer-aided publishing and computer-aided software engineering (CASE), i.e. appliations which integrate text and graphics (Computer Weekly, 1989).

Although the RDBMS is convenient for management of a static archive, it is not ideal for map update operations.

5.2 Small Scales

Small scale data were welcomed as useful for some functions. Thames Water and others (see Visvalingam and Kirby, 1987) believe that the 1:50 000 scale is appropriate for information required at the management, rather than the operational, level. Roger Moore of the IoH expressed that this represents the most useful compromise for hydrologic applications. Previous modelling of the hydrological cycle were based on generalisations at the catchment level. The 1:50 000 base allows the catchment to be subdivided into more genuinely homogeneous units. With the present technology, it would be difficult for the IoH to cope with the volume of data which would result from a larger scale. The DCC also believe that the 1:50 000 scale would provide usable base maps in areas where there was no large-scale digital map coverage for recording phenomena such as mineral sites, derelict land, a Biological Sites Register and Special Landscape Areas. All respondants expressed that this scale was particularly useful for network applications. Since the Military Survey is generating a DTM at this scale, it is reasonable to assume that realistic terrain modelling would also require land-use information for this scale. Some others, for example Mason et al (1988) have already found this scale useful for interpretation of satellite images.

5.3 Content

5.3.1 Communications and urban areas

Since these are high priority features on the 1:50 000 database, the data are fairly complete and appear to be satisfactory for a number of applications from route scheduling and in-car navigation systems to Finance Office checking of claims for travel expenses. The generalised representation of the 1:625 000 database within AutoRoute appears to satisfy a number of users. DCC see potential use of these features for management of mobile libraries and for siting of depots for servicing street lights. Both DCC and NFC welcome the association of settlement names with nodes on the road network. DCC and others also value road centre lines for calculation of drive time contours and catchment area definitions. But, there are other sources of road

data, which also provide further types of information required by these applications, such as width, height and weight restrictions and average speeds on sections.

Route-related applications also anticipate a requirement for a scale-integrated road network which will allow use of generalised data over long distances and more detailed data in urban locations (Chapman, 1988).

5.3.2 Rivers and Water Features

Thames Water, DCC, IoH and others have a well defined interest in these features but these form relatively low priority features on the 1:50 000 map. Although this scale is useful for extracting the network (centre lines) of rivers, the database was regarded as grossly incomplete. The IoH is using the DTM, generated by the Military Survey and information from Water Authorities to extend and fill in the river network and flow information. The DTM is also useful for including aspect in snow melt calculations. A complete flow-directed network is also useful to local authorities; DCC was disappointed that the 1:50 000 data was limited by its cartographic, rather than topographic, model.

5.3.3 Contours and Heights

DCC and several others expect that contour and height information will be useful for DTM-based applications, using software such as MOSS. The need for height limits on bridges has already been described in the context of road related applications. Military and civil aviation would presumably require heights of chimneys, woodland, church spires and other hazards, for example, pylons, to low-flying aircraft

5.3.4 Woodlands

These are also regarded as high priority by many, including the DCC and IoH.

5.4 Accuracy and consistency of data

DCC and IoH have reservations about the accuracy of the 1:50 000 data. Only Rob Ward has used the SSD database to any great extent and our

experiences were slightly different because we did not receive identical databases. DCC and others were concerned about geometric inaccuracies, especially in settlement locations, missing links, polygons which did not close, discontinuous contours and inconsistent data, particularly in the south west quadrant of Sheet 111. Digital maps will be used within GIS for metric analysis, for example for calculating frequency of occurrence, lengths and areas. Since some polygons would not close, they could not be area filled nor used in GIS applications. One user expressed that the quality of the SSD database was inadequate for demonstration of his software.

Further, when DCC overlayed data on mineral exploitation and reclamation sites (from 1:25 000), road centre lines (from 1:10 000) and built-up land (from 1:50 000), spurious polygons were generated. The need for proper scale integration was emphasized.

5.5 History

Land Information Systems require chronological data.

5.6 Structure

Since small scale data are valued primarily for network and management applications, a link and node structure is essential. Users also welcomed objects within the database. Object definition automatically generates multiple attribute codes for linear features. DCC saw prospects for using a 1:50 000 database as a base for a GIS. By extending the feature codes, heights and other information from disparate sources, such as the Biological Sites Register and Sites and Monuments Records, may be linked to this base. Thames Water also saw a need to link population data for suitably defined area objects. IoH will also be including data on a variety of phenomena affecting the hydrological cycle, such as geology, soils, rainfall and land use.

5.7 Conclusion

There appears to be a latent market for a variety of features on small scale digital maps within several GIS applications. However, the cost, presence of errors and omissions (in the cartographic base) are encouraging users to investigate other specialised sources of data.

Higher level object definitions and multiple attributes are now widely recognised as essential for GIS. Since users expect to link data from various sources, the problems of scale integration and geometric registration will need to be resolved.

6. FURTHER INVESTIGATION

In this section, we would like to advance two topics for further investigation. These relate to automatic abstraction of objects from a link and node structured database of features and to the suitability of the relational model for map editing.

1) GIS applications require that the database contains appropriate objects. Users were positive in their requirement for objects depicted on small scale maps and are already investigating potential sources of digital data (see Section 5). Object creation also assists in the identification of geometric, topological and semantic errors (see Section 3). But, the manual creation of objects is tedious, costly and error prone. Although the automatic identification of objects on small scale maps is also error prone, especially since the cartographic source is incomplete, it is possible to formulate rules for automatic elimination of classes of errors and achieve a product of consistent quality. Manual editing tends to focus on individual occurrences, rather than classes, of errors and is thus unreliable. Research into the identification of objects on specific scales of maps would at the very least result in :

   a) an intelligent system for assisting with the manual definition of objects

   b) development of procedures for identifying some classes of errors in the database

   c) knowledge of objects, representations and methods, which could further the long-term goal of scale-free digital mapping

   Such research is unlikely to result in a comprehensive set of rules for use in the short term within systems for the creation

and maintenance of the archive. However, usable results should be forthcoming within a period of 3 to 5 years to assist with validation tasks. Given scanned images of area fill symbolism on small scale maps, considerable progress may be made towards development of intelligent software for extracting object definitions.

2) Sekouris adapted Wade's software, written in Fortran for implementing the Disassociative Area Model (Visvalingam et al, 1986; Kirby et al, 1989), for use with a small subset of the SSD database. Numerous errors deterred further work towards automatic object creation and cross validation across manually extracted descriptions. Automatic object creation would have involved update problems at the level of object structures. This would have enabled the student to investigate how update should be managed using a relational DBMS. But, lack of time and an improved understanding of the nature of spatial data processing led Sekouris to abandon use of the relational model and opt instead for continued use of software using linked lists.

However, there is a need to establish empirically the extent to which spatial data processing, as described in Kirby et al (1989), may be undertaken using a RDBMS, to quantify time penalties and to isolate specific operations for which the relational approach is particularly inapt.

We therefore suggest the following.

[19] OS should actively encourage further research on the management of map data within a relational database model and on the automatic extraction of objects.

7. CONCLUSION

Our brief was to provide a second, independent opinion on the OS relational data model used within the 1:50 000 SSD database of Sheet 111. This model is already being used with satisfaction within the OS. We have therefore tried to retain the basic structure of the model and be constructive with our comments.

In Section 2, we identified some conceptual weaknesses of the model and suggested some modifications. Our aims were to abstract a simpler, more consistent and therefore more comprehensible schema, which is also more economical of storage and more efficient for retrieving data in various ways from a static archive. The amended relations require only a third of the storage required by the original relations.

Some suggestions for detecting obvious errors in the database were then described in Section 3. In Section 4, we illustrated how the conceptual model could be adapted for specific applications, such as mapping. We also outlined the type of experiments which could be conducted to evaluate empirically the benefits of alternative schemes for indexing and suggested a system independent strategy for indexing the envelope fields in a frugal but effective manner.

In Section 5, we provided a collated account of comments made by users in unstructured interviews. These tend to suggest that it is the presence of appropriate but geometrically generalised objects which make small scales map data particularly suitable for management functions and network analysis. Given that a scale free strategy is not feasible at present, users forsee a need for scale integration at the geometric level so that small scales data could be used more effectively with data from other sources within a GIS.

Two directions for further research are outlined in Section 6.

REFERENCES

Chapman, D. (1988) "The derivation of a road network database from Ordnance Survey Digital Map Data", Unpublished paper presented at the British Cartographic Society 25th Annual Symposium, Nottingham, 16-18 September.

Computer Weekly (1989) "Data dictionaries - failings of RDBMS", 16-2-89, 25 - 27.

Gargantini, I. (1982) "An effective way to represent quadtrees" **CACM**, 25 (12), 905 - 910.

Ibbs, T. J. and Stevens, A. (1988) "Quadtree storage of vector data", **Int. J. G.I.S.**, 2(1), 43 - 56.

Kirby, G. H., Visvalingam, M. and Wade, P. (1989) "The recognition and representation of polygons with holes", **Computer Journal**, in press.

Mason, D. C., Corr, D. G., Cross, A., Hogg, D. G., Lawrence, D. H., Petrou, M. and Taylor, A.M. (1988) "The use of digital map data in the segmentation and classification of remotely-sensed images", **Int. J. G. I. S.**, 2 (3), 195 - 215.

Sekouris, N. M. (1989) "Management of digital map data using a relational database model", Draft MSc Thesis, Department of Computer Science, University of Hull.

Smith, N. and Turner, J. (1987) "The Ordnance Survey Topographic Database; Implications for the Input and Output of Map Data", in Earnshaw, R (ed.) **State of the Art in Computer Cartography**, British Computer Society, Unpublished proceedings of Meeting held in November 1987 (London).

Visvalingam, M. (1988a) "Some issues relating to Basic Spatial Units", **Mapping Awareness** 2 (3), 40 - 42.

Visvalingam, M. (1988b) "Some issues relating to Basic Spatial Units", **Mapping Awareness** 2 (4), 42 - 45.

Visvalingam, M., Wade, P. and Kirby, G.H. (1986) "Extraction of area topology from line geometry", in Blakemore, M. (ed.) **Auto Carto London**, volume 1, 156 - 165.

Visvalingam, M. and Kirby, G.H. (1987) Directory of Research and Development based on Ordnance Survey Small Scales Digital Data, Cartographic Information Systems Research Group, University of Hull.

Wade, P., Visvalingam, M. and Kirby, G. H. (1986) "From Line Geometry to Area Topology", Discussion Paper 1, Cartographic Information Systems Research Group, University of Hull.

Appendix I

Project Title

Management of Digital Map Data Using a Relational Database Model

1. Investigators

Drs. M. Visvalingam and G.H. Kirby Department of Computer Science
University of Hull
HULL HU6 7RX

2. General Aims

The project will provide an independent evaluation of the present OS strategy for managing small-scales digital map data within a relational database model and propose an alternative approach if necessary. The investigation will address the need to:

a) ease update of the database;
b) maintain logical consistency of the database during update operations;
c) optimise performance of load, search and update operations;
d) minimise the size, and thus cost, of creation and maintenance of the database given the excessive projected requirements.

3. Project Duration and Finance

4. Structure of project

4.1 Conceptual Analysis (approx. 3 months)

The student will undertake a conceptual analysis of the small scales digital data and comment upon the implications of the current OS model. There is a requirement for a hybrid data model which will accommodate the handling of both the feature-based archive and of areal and linear objects derived from these features.

OS will provide full documentation on the current set of relations and background material, in the form of written documents if possible. OS will also provide a suitable subset of the 1:50 000 Sheffield/Doncaster experimental sheet on magnetic tape with relevant documents on tape format, structure, content etc.

4.2 Review of functional requirements and of current practices (approx. 3 months)

OS will provide a clear statement of its present and anticipated in-house use of this database and of its current methods for loading, searching and updating of the database. OS will also provide a list of current users of the database, who may be willing to discuss their requirements.

The student will contact system developers to obtain information on strategies they use. All parties will co-operate in locating relevant contacts, literature and algorithms.

4.3 Identify a suitable strategy for use with small scales data (3 to 4 months)

This stage will depend on a functional analysis of the database undertaken during the preceding stage. The proposed strategy will be independent of features specific to current proprietary systems.

The primary requirement is for a strategy which, given change information in the current feature-based format, will allow update of Users' established databases without disruption. The strategy will have to operate with the established OS practice which relies on date stamps, rather than deletions, for update and for keeping a history of the Archive.

If time permits, there is also a need to consider alternative criteria for partitioning the database and to suggest elegant solutions to improve performance and size requirements. Published algorithms may be compared on the Britton-Lee system at OS.

4.4 Preparation of Masters Thesis (approx. 6 weeks)

4.5 Holidays (not exceeding 6 weeks)

Such holidays will be taken at times as agreed with the academic supervisors.

5. Results and Publications

All developed software (including adequate documentation to enable its use but not copyright or other rights to software), reports, publications and thesis produced as a result of this studentship will be made freely available to the Ordnance Survey.

Appendix 2 (NMS angle [illegible])

CONTENTS

SMALL SCALES (RDB) DATA STRUCTURE

User Relations in Database with filename SSDDB:SSDEMO_DB

- AREA_OBJ
- AREA_TIE
- BNDRY_ENT_TIE
- GRAPHICS
- HISTORY
- ID_TABLE
- LINE_ENT
- LINE_OBJ
- LINE_TIE
- LINK_GEOM
- MODEL
- NAME_ENT
- NAME_POS
- NODE_GEOM
- NODE_OBJ
- NODE_STR
- POINT_OBJ
- WINDOW_DATA
- WINDOW_DESC

Fields for relation AREA_OBJ

AREA_REF	G_floating
AO_X	signed longword scale 0
AO_Y	signed longword scale 0
NAME_ID	signed longword scale 0
BNDRY_ID	signed longword scale 0
MODEL_KEY	signed longword scale 0
AO_XMAX	signed longword scale 0
AO_YMAX	signed longword scale 0
AO_XMIN	signed longword scale 0
AO_YMIN	signed longword scale 0
APPL	text size is 4
HIST	signed longword scale 0

Fields for relation AREA_TIE

AREA_REF	G_floating
SUB_AREA_REF	G_floating
HIST	signed longword scale 0

Fields for relation BNDRY_ENT_TIE

AREA_REF	G_floating
BNDRY_ID	signed longword scale 0
LINE_ID	signed longword scale 0
BNDRY_SORT	signed longword scale 0
L_R	text size is 1
MODEL_KEY	signed longword scale 0
HIST	signed longword scale 0

```
Fields for relation  GRAPHICS
    MODEL_KEY                   signed longword scale  0
    L_S                         text size is  1
    LINE_TYPE                   signed word scale  0
    LINE_THICK                  signed word scale  0
    PLOT_COL                    signed word scale  0
    SCREEN_COL                  signed word scale  0
    TXT_FONT                    signed word scale  0
    TXT_SIZE                    signed word scale  0


Fields for relation  HISTORY
    HIST                        signed longword scale  0
    HIST_DESCR                  text size is  80
    OPERATOR                    text size is  20
    DATE                        Date


Fields for relation  ID_TABLE
    ID_NUMBER                   signed longword scale  0
    DATE                        Date
    ID_FLAG                     text size is  1


Fields for relation  LINE_ENT
    LINE_ID                     signed longword scale  0
    MODEL_KEY                   signed longword scale  0
    HIST                        signed longword scale  0


Fields for relation  LINE_OBJ
    LINE_OBJ_ID                 signed longword scale  0
    NAME_ID                     signed longword scale  0
    LO_XMAX                     signed longword scale  0
    LO_YMAX                     signed longword scale  0
    LO_XMIN                     signed longword scale  0
    LO_YMIN                     signed longword scale  0
    APPL                        text size is  4
    HIST                        signed longword scale  0


Fields for relation  LINE_TIE
    LINE_OBJ_ID                 signed longword scale  0
    LT_ITEM                     text size is  1
    LINE_ID                     signed longword scale  0
    LT_SORT                     signed longword scale  0
    HIST                        signed longword scale  0
```

```
Fields for relation  LINK_GEOM
    LINE_ID                        signed longword scale  0
    LG_XMAX                        signed longword scale  0
    LG_YMAX                        signed longword scale  0
    LG_XMIN                        signed longword scale  0
    LG_YMIN                        signed longword scale  0
    LG_LENGTH                      signed longword scale  0
    START_NODE                     G_floating
    END_NODE                       G_floating
    LG_NO_COORD                    signed word scale  0
    LG_COORD_LIS                   segmented string of type 8
                                     segment_length 512
    HIST                           signed longword scale  0


Fields for relation  MODEL
    MODEL_KEY                      signed longword scale  0
    FEATURE_CODE                   text size is  2
    ATTRIBUTE_LIST                 text size is  48
    MODEL_NAME                     text size is  6
    ITEM_DESCRIPTION               text size is  80


Fields for relation  NAME_ENT
    NAME_ID                        signed longword scale  0
    NE_TEXT                        text size is  70
    HIST                           signed longword scale  0


Fields for relation  NAME_POS
    NAME_ID                        signed longword scale  0
    NP_X                           signed longword scale  0
    NP_Y                           signed longword scale  0
    NP_SIZE                        signed word scale  0
    NP_FONT                        signed word scale  0
    NP_ORI                         signed word scale  0
    APPL                           text size is  4
    HIST                           signed longword scale  0


Fields for relation  NODE_GEOM
    NODE_REF                       G_floating
    NG_X                           signed longword scale  0
    NG_Y                           signed longword scale  0
    NG_Z                           signed word scale  0
    HIST                           signed longword scale  0


Fields for relation  NODE_OBJ
    NODE_REF                       G_floating
    NAME_ID                        signed longword scale  0
    MODEL_KEY                      signed longword scale  0
    APPL                           text size is  4
    HIST                           signed longword scale  0
```

```
Fields for relation  NODE_STR
    NODE_REF                    G_floating
    LINE_ID                     signed longword scale  0
    LINK_SORT                   signed longword scale  0
    NS_LEVEL                    signed word scale  0
    NS_AZI                      signed word scale  0
    HIST                        signed longword scale  0

Fields for relation  POINT_OBJ
    PO_REF                      G_floating
    PO_X                        signed longword scale  0
    PO_Y                        signed longword scale  0
    PO_Z                        signed word scale  0
    NAME_ID                     signed longword scale  0
    MODEL_KEY                   signed longword scale  0
    APPL                        text size is  4
    HIST                        signed longword scale  0

Fields for relation  WINDOW_DATA
    WINDOW_CODE                 text size is  1
    DATA_CLASS                  text size is  1
    MODEL_KEY                   signed longword scale  0
    SPATIAL_REF                 G_floating
    ID_NUMBER                   signed longword scale  0

Fields for relation  WINDOW_DESC
    WINDOW_CODE                 text size is  1
    WINDOW_NAME                 text size is  10
    WD_XMAX                     signed longword scale  0
    WD_YMAX                     signed longword scale  0
    WD_XMIN                     signed longword scale  0
    WD_YMIN                     signed longword scale  0
    HIST                        signed longword scale  0
```

Appendix 3.2

DATA MODEL FOR THE SMALL SCALES DEMONSTRATOR

User Fields in Database:

FIELD	DATA TYPE	DESCRIPTION
APPL*	text size is 4	APPLICATION LABEL FOR FEATURES
ATTRIBUTE_LIST	text size is 48	LIST OF SECONDARY ATTRIBUTES FOR FEATURE
COORD*	signed longword scale 0	COORDINATE VALUE (X,Y OR E,N)
DATA_CLASS	text size is 1	ALPHA CODE DESCRIBING CLASS OF DATA HELD
DATE	Date	DATE AND TIME OF ACTION
DUMMY_ATT	signed longword scale 0	DUMMY NODE ATTRIBUTE FIELD
FEATURE_CODE	text size is 2	PRIMARY CODING OF FEATURE
HEIGHT*	signed word scale 0	TOPOGRAPHIC HEIGHT
HIST*	signed longword scale 0	NUMERIC KEY TO HISTORY INFORMATION
HIST_DESCR	text size is 80	DESCRIPTION OF ACTION CARRIED OUT
ID_FLAG	text size is 1	INDICATES LOCKED (1) OR USABLE CURRENT ID (0)
ID_NUMBER*	signed longword scale 0	NUMERIC KEY
ITEM_DESCRIPTION	text size is 80	DESCRIPTION OF ITEMS IN A MODEL LIST
LG_COORD_LIS	segmented string of type 8 segment_length 512	COORDINATE PAIR LIST FOR LINK
LG_LENGTH	signed longword scale 0	LENGTH OF LINK
LG_NO_COORD	signed word scale 0	NUMBER OF COORD PAIRS HELD FOR LINK
LINE_THICK	signed word scale 0	
LINE_TYPE	signed word scale 0	
LT_ITEM	text size is 1	FLAG FOR OBJECT /(E)NTITY OR OBJECT/(O)BJECT TIE
L_R	text size is 1	AREA ENCLOSED TO (L)EFT OR (R)IGHT
L_S	text size is 1	DEFINITION IS (L)INE OR (S)YMBOL
MODEL_KEY*	signed longword scale 0	KEY TO FEATURE DESCRIPTION INFO
MODEL_NAME	text size is 6	DATA MODEL
NAME_ID*	signed longword scale 0	NUMERIC KEY TO NAME INFORMATION
NE_TEXT	text size is 70	NAME TEXT STRING
NP_FONT	signed word scale 0	FONT FOR NAME TEXT
NP_ORI	signed word scale 0	ORIENTATION OF NAME TEXT
NP_SIZE	signed word scale 0	SIZE OF LETTERING FOR TEXT
NS_AZI	signed word scale 0	AZIMUTH OF LINK AT A NODE
NS_LEVEL	signed word scale 0	RELATIVE HEIGHT OF LINK AT NODE
OPERATOR	text size is 20	WHO CARRIED OUT THE ACTION
PLOT_COL	signed word scale 0	
SCREEN_COL	signed word scale 0	
SORT*	signed longword scale 0	RELATIVE ORDER OF ITEM
SPATIAL_REF*	G_floating	SPATIAL KEY (PEANO)
TXT_FONT	signed word scale 0	
TXT_SIZE	signed word scale 0	
WINDOW_CODE	text size is 1	SINGLE CHARACTER KEY TO WINDOW DATA
WINDOW_NAME	text size is 10	THE USERS NAME FOR THE DATA WINDOW

(* - global field definition)

User Relations in Database:

AREA_OBJ

AREA_TIE

BNDRY_ENT_TIE

DUMMY_REL - created for demonstration purposes only

GRAPHICS

HISTORY

ID_TABLE

LINE_ENT

LINE_OBJ

LINE_TIE

LINK_GEOM

MODEL

NAME_ENT

NAME_POS

NODE_GEOM

NODE_OBJ

NODE_STR

POINT_OBJ

WINDOW_DATA - not currently used

WINDOW_DESC - not currently used

Fields for relation AREA_OBJ

AREA_REF	based on global field SPATIAL_REF
AO_X	based on global field COORD
AO_Y	based on global field COORD
NAME_ID	
BNDRY_ID	based on global field ID_NUMBER
MODEL_KEY	
AO_XMAX	based on global field COORD
AO_YMAX	based on global field COORD
AO_XMIN	based on global field COORD
AO_YMIN	based on global field COORD
APPL	
HIST	

Fields for relation AREA_TIE

AREA_REF	based on global field SPATIAL_REF
SUB_AREA_REF	based on global field SPATIAL_REF
HIST	

Fields for relation BNDRY_ENT_TIE

AREA_REF	based on global field SPATIAL_REF
BNDRY_ID	based on global field ID_NUMBER
LINE_ID	based on global field ID_NUMBER
BNDRY_SORT	based on global field SORT
L_R	
MODEL_KEY	
HIST	

Fields for relation DUMMY_REL

NODE_REF	based on global field SPATIAL_REF
DUMMY_ATT	

Fields for relation GRAPHICS

MODEL_KEY
L_S
LINE_TYPE
LINE_THICK
PLOT_COL
SCREEN_COL
TXT_FONT
TXT_SIZE

Fields for relation HISTORY

HIST	
HIST_DESCR	
OPERATOR	
DATE	

Fields for relation ID_TABLE

ID_NUMBER	
DATE	
ID_FLAG	

Fields for relation LINE_ENT

LINE_ID	based on global field ID_NUMBER
MODEL_KEY	
HIST	

Fields for relation LINE_OBJ

LINE_OBJ_ID	based on global field ID_NUMBER
NAME_ID	
LO_XMAX	based on global field COORD
LO_YMAX	based on global field COORD
LO_XMIN	based on global field COORD
LO_YMIN	based on global field COORD
APPL	
HIST	

Fields for relation LINE_TIE

LINE_OBJ_ID	based on global field ID_NUMBER
LT_ITEM	
LINE_ID	based on global field ID_NUMBER
LT_SORT	based on global field SORT
HIST	

Fields for relation LINK_GEOM

LINE_ID	based on global field ID_NUMBER
LG_XMAX	based on global field COORD
LG_YMAX	based on global field COORD
LG_XMIN	based on global field COORD
LG_YMIN	based on global field COORD
LG_LENGTH	
START_NODE	based on global field SPATIAL_REF
END_NODE	based on global field SPATIAL_REF
LG_NO_COORD	
LG_COORD_LIS	
HIST	

Fields for relation MODEL

MODEL_KEY	
FEATURE_CODE	
ATTRIBUTE_LIST	
MODEL_NAME	
ITEM_DESCRIPTION	

Fields for relation NAME_ENT

NAME_ID	
NE_TEXT	
HIST	

Fields for relation NAME_POS

NAME_ID	
NP_X	based on global field COORD
NP_Y	based on global field COORD
NP_SIZE	
NP_FONT	
NP_ORI	
APPL	
HIST	

Fields for relation NODE_GEOM

NODE_REF	based on global field SPATIAL_REF
NG_X	based on global field COORD
NG_Y	based on global field COORD
NG_Z	based on global field HEIGHT
HIST	

Fields for relation NODE_OBJ

NODE_REF	based on global field SPATIAL_REF
NAME_ID	
MODEL_KEY	
APPL	
HIST	

Fields for relation NODE_STR

NODE_REF	based on global field SPATIAL_REF
LINE_ID	based on global field ID_NUMBER
LINK_SORT	based on global field SORT
NS_LEVEL	
NS_AZI	
HIST	

Fields for relation POINT_OBJ

PO_REF	based on global field SPATIAL_REF
PO_X	based on global field COORD
PO_Y	based on global field COORD
PO_Z	based on global field HEIGHT
NAME_ID	
MODEL_KEY	
APPL	
HIST	

Fields for relation WINDOW_DATA

WINDOW_CODE
DATA_CLASS
MODEL_KEY
SPATIAL_REF
ID_NUMBER

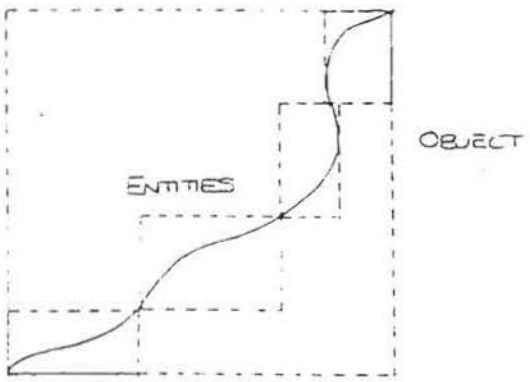
Fields for relation WINDOW_DESC

WINDOW_CODE	
WINDOW_NAME	
WD_XMAX	based on global field COORD
WD_YMAX	based on global field COORD
WD_XMIN	based on global field COORD
WD_YMIN	based on global field COORD
HIST	

Appendix 3.3

SSDEMONSTRATOR - DATA MODEL

LINK_~~STR~~ GEOM - geometry of the line
LINE_ENT - semantics
LINE_OBJ - collection of entities



LINK_~~STR~~ GEOM → LINE_ENT ≡ one → one
LINE_ENT → LINE_OBJ ≡ many → one
Must make a connection between the object and its entities
use LINE_TIE

LINE_TIE - can create an object from entities, or, an object from objects
eg M1 (object) is formed of links (entities)
British Motorways (object) is formed of motorways eg M1 (objects)

POINTS - fairly straightforward

NODES - more complex as nodes can also be points eg service stations
NODE_GEOM - geometry of the node
NODE_OBJ - eg service station - node object may not exist for all points
Node also has links attached
NODE_STR - gives record for every link at a node one → many
Azimuth - first / last vector at node.
Level - relative levels of links at node

COORDINATE STORAGE
Segmented strings - held in link_~~str~~ geom relation
- can only have 1 segmented string field in a relation, this though can be as long as 64K
- variable length text string
Each coordinate X - I4 } converted into a segment, character *8 by
Y - I4 } interpreting each byte into a character
- each segment is a coordinate pair stored as an 8 byte character
up to 64K for each link.
X Y Z Q ≡ character * 16
Cannot easily look at coordinates though

Can apply constraints to a field

eg DEFINE FIELD SPATIAL_REF

DESCRIPTION .....

DATATYPE .....

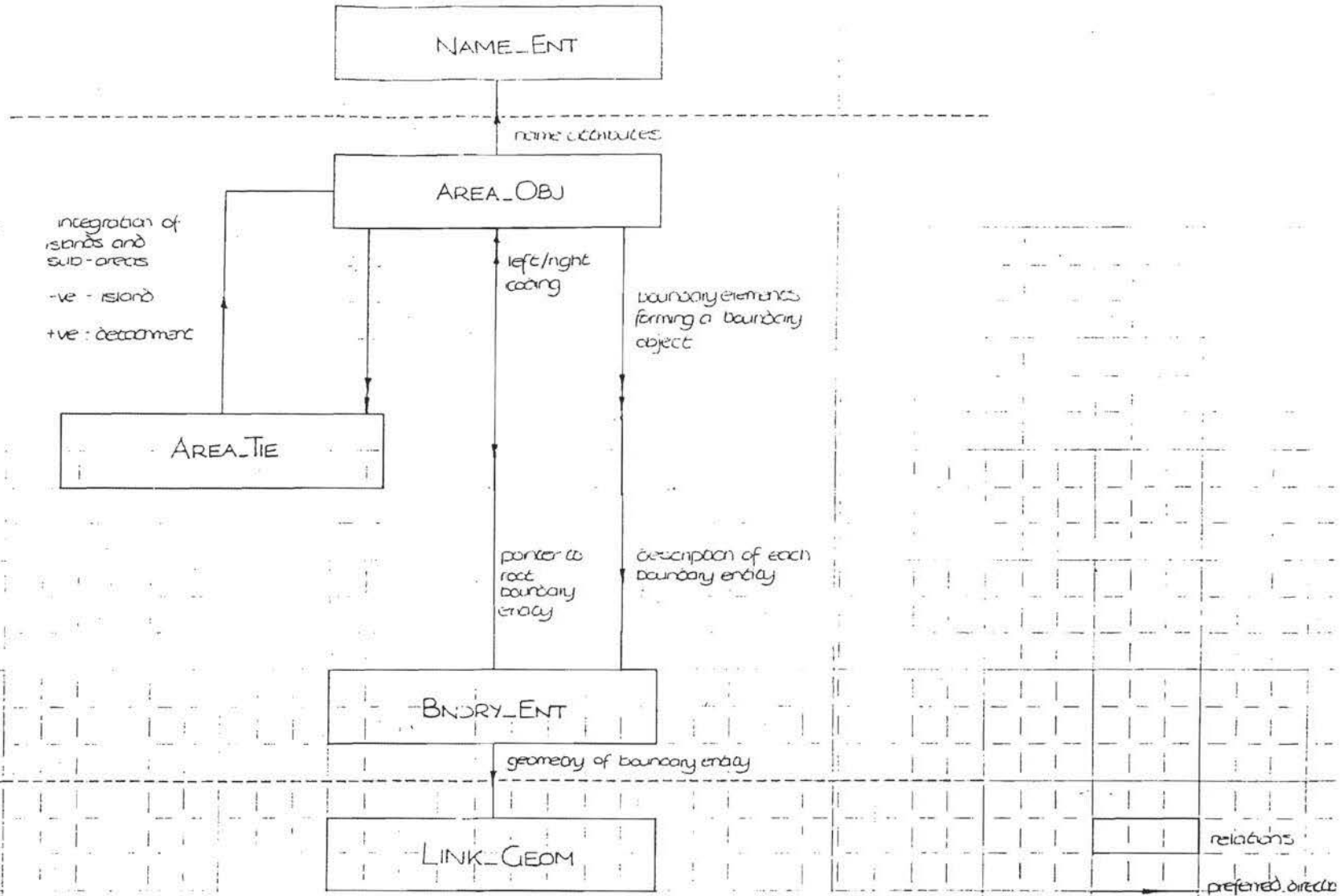
VALID IF SPATIAL_REF NOT MISSING

This applies on overhead though as each spatial_ref must be checked to see that it has a value

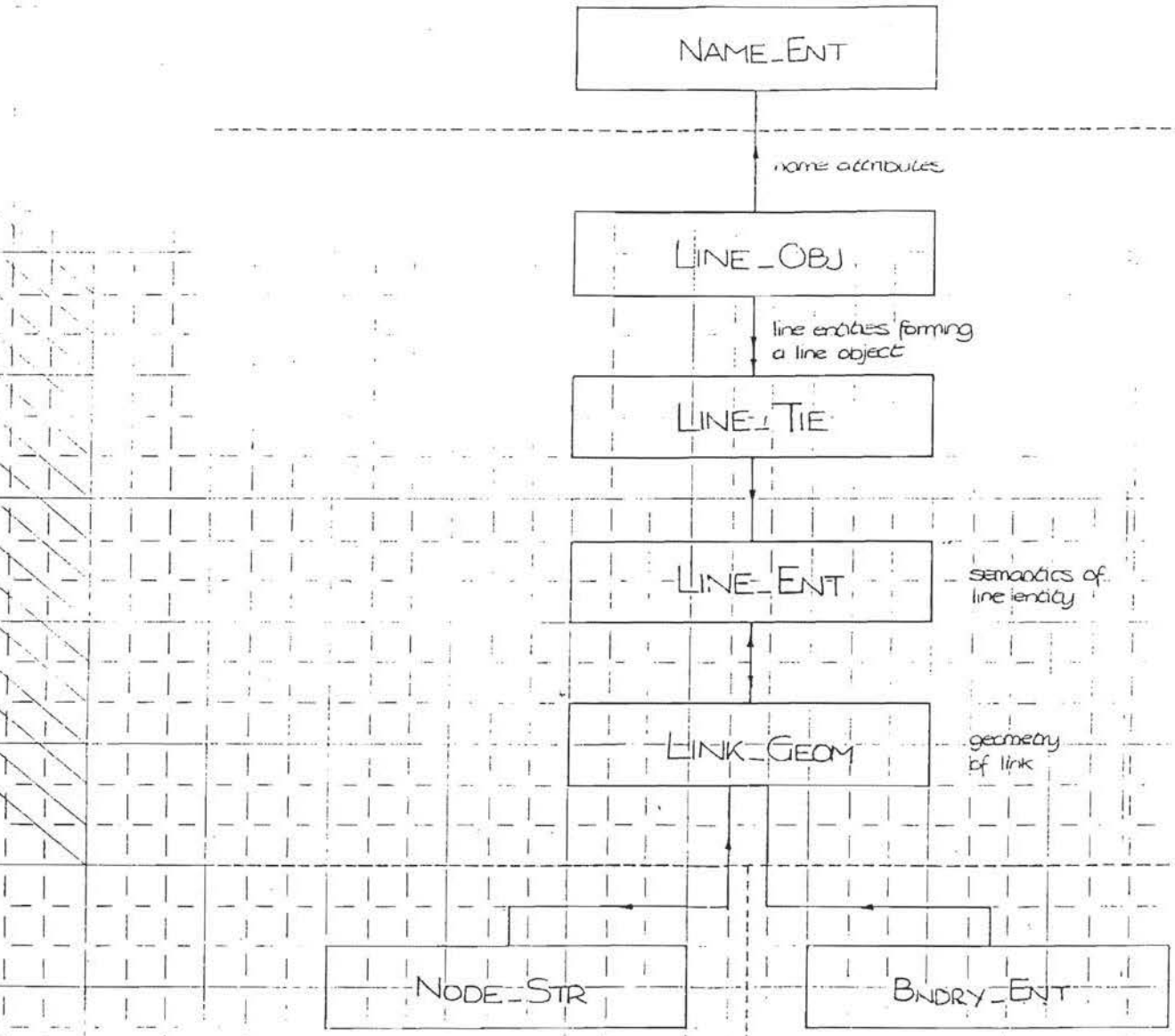
AREAS - DATA MODEL

Appendix 3.4

SSD | E. DOWELL



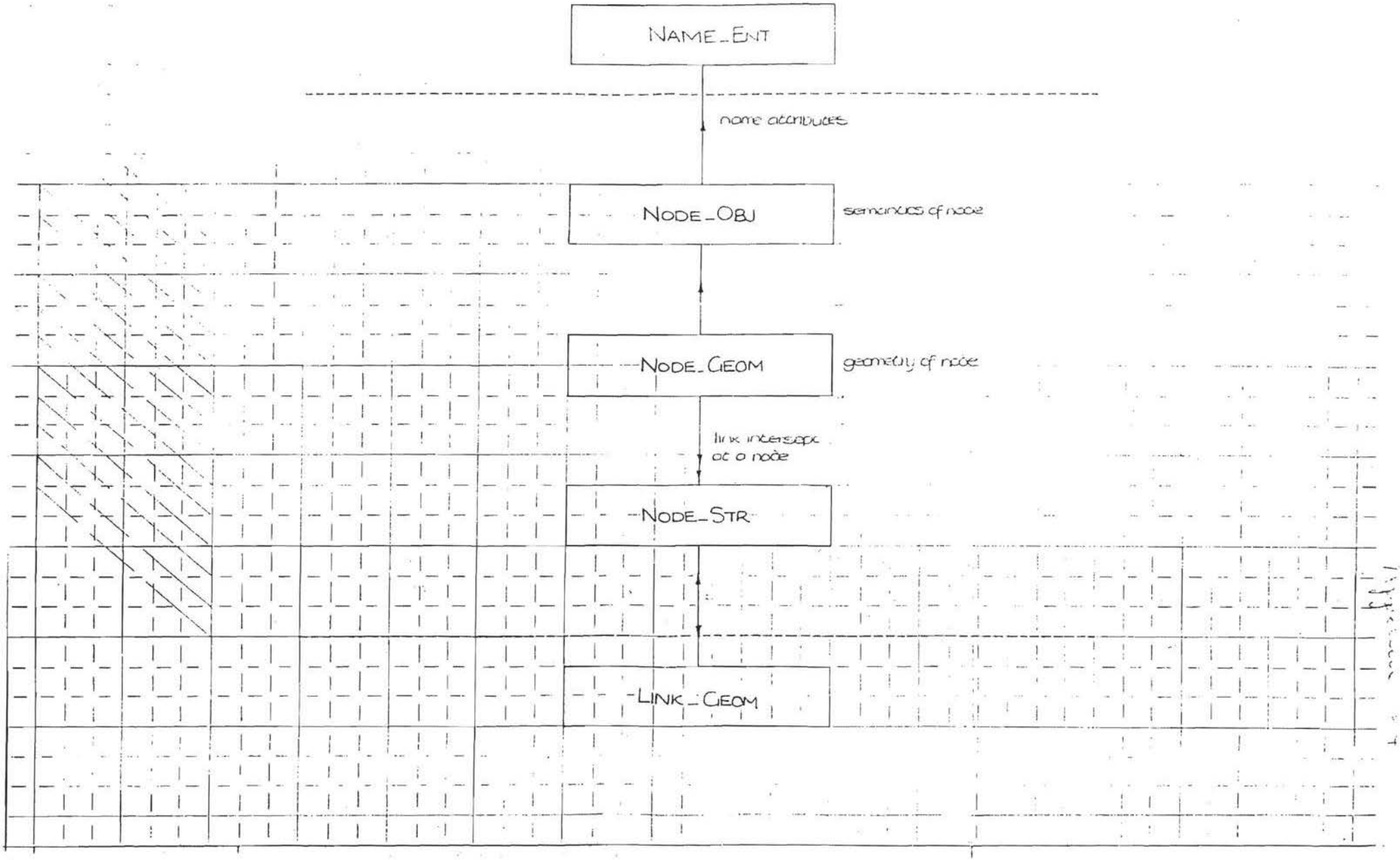
A3.4(1)


NAME_ENT
name attributes
LINE_OBJ
line entities forming a line object
LINE_TIE
LINE_ENT
semantics of line entity
LINK_GEOM
geometry of link
NODE_STR
BNDRY_ENT

A 3.4 (2)

NODES - DATA MODEL

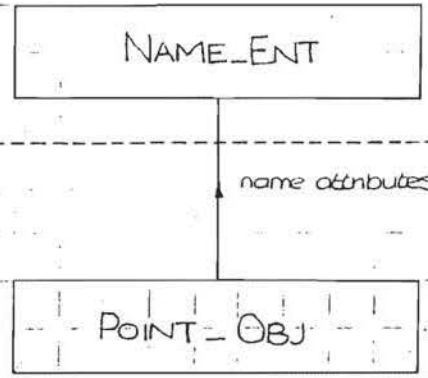
	SSD	E. BOWELL	3



Title

POINTS - DATA MODEL

System	Document	Author	Sheet
	SSD	E. BOWELL	4

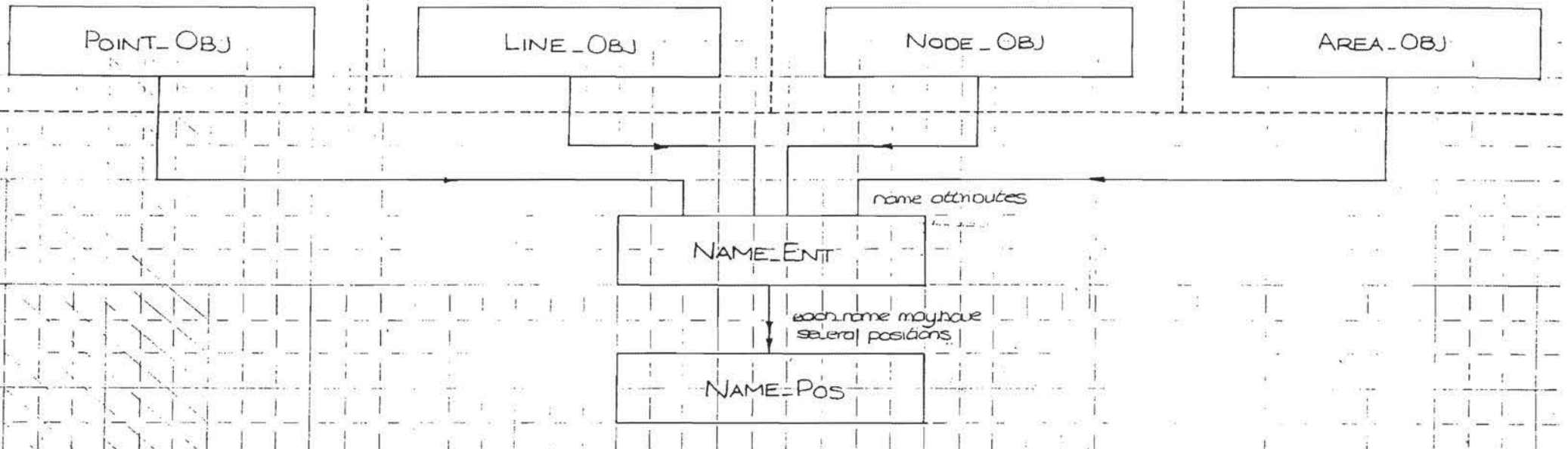


A3.4 (4)

Title

NAMES - DATA MODEL

System	Document	Name	Sheet
	SSD	E. BOWELL	5

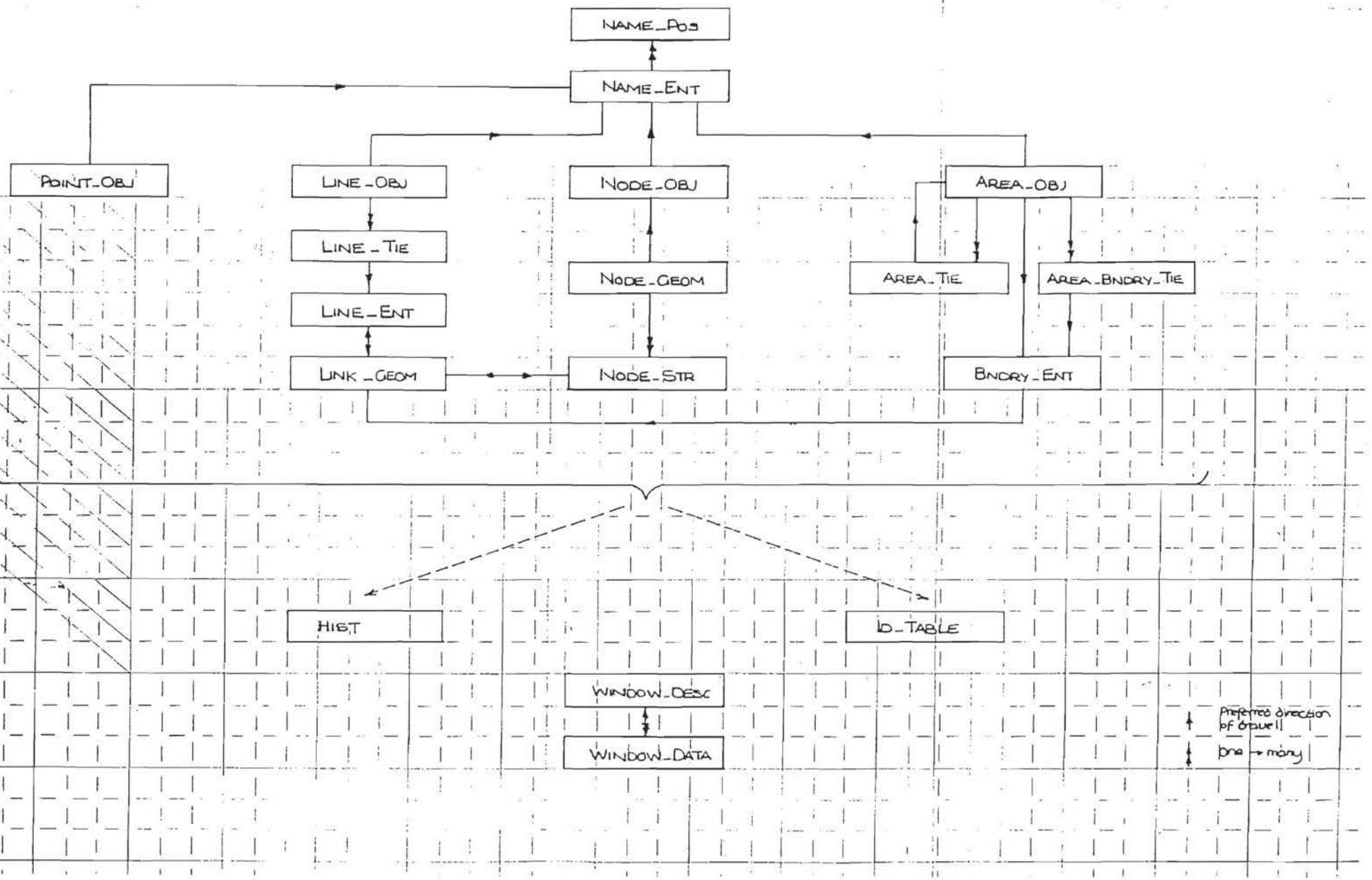


A 3.4 (5)

Title

SMALL SCALES DEMONSTRATOR - DATA MODEL

System | Document: SSD | Name: E. BOWELL | Sheet: 6



A3.4(6)