

# UC Santa Barbara

## Core Curriculum-Geographic Information Systems (1990)

### Title

Unit 30 - Storage of Complex Objects

### Permalink

<https://escholarship.org/uc/item/3mv1h090>

### Authors

Unit 30, CC in GIS  
Douglas, David H.

### Publication Date

1990

Peer reviewed

# UNIT 30 - STORAGE OF COMPLEX OBJECTS

## UNIT 30 - STORAGE OF COMPLEX OBJECTS

Compiled with assistance from David H. Douglas, University of Ottawa

- [A. INTRODUCTION](#)
- [B. REPRESENTATION OF SIMPLE SPATIAL OBJECTS](#)
- [C. STORAGE OF OBJECT ATTRIBUTES](#)
- [D. REPRESENTATION OF TOPOLOGY](#)
  - [Relationships in networks](#)
  - [Relationships between areas](#)
  - [The CanSIS data structure](#)
- [E. DISADVANTAGES OF ARC-BASED REPRESENTATIONS](#)
- [E. OTHER ISSUES ABOUT DATA STRUCTURES](#)
- [REFERENCES](#)
- [DISCUSSION AND EXAM QUESTIONS](#)
- NOTES

## UNIT 30 - STORAGE OF COMPLEX OBJECTS

Compiled with assistance from David H. Douglas, University of Ottawa

### [A. INTRODUCTION](#)

- previous units have been concerned with specifying and transforming locations
- GIS deals with objects such as lines and areas occupying extended locations, and with the complex relationships between them
- spatial data normally represented in vector systems as:
  - objects (points, lines and areas)
  - attributes associated with objects
  - relationships between objects
- this unit considers how to construct objects out of sets of coordinates, and how to create digital representations of attributes and relationships?

- many alternatives exist for structuring spatial data within a digital store
- here we review some of the most common which have been proven useful by years of experience and application

## B. REPRESENTATION OF SIMPLE SPATIAL OBJECTS

- spatial objects - points, lines, areas - can be coded as x,y coordinate pairs:
  - point: (x,y)
  - line: (x1,y1), (x2,y2), ... , (xn,yn)
  - area: (x1,y1), (x2,y2), ... , (xn,yn)
  - note that the digital representation of the three spatial objects is identical, n=1 in the first case
  - note the convention used throughout this unit:
    - the name of the record type, followed by a colon, then the items forming the record
- to construct a line or area, we simply connect each consecutive pair of points with straight lines
- in the case of an area object we might insist that the last point be the same as the first, or alternatively assume that the last point is connected to the first by a straight line to close the area
- points need not always be connected by straight lines - see later discussion in this unit

## C. STORAGE OF OBJECT ATTRIBUTES

- attributes of objects can be stored as tables
- for points, the coordinates can be included as two additional attributes for each object, so that the entire data structure can be a simple table
- this is not possible for lines and areas because of the variable number of coordinates
  - the data structure usually consists of two parts:
    - coordinates in one file, each set representing a single object identified by a unique ID
    - attributes in a table with one attribute identifying the objects to which each is linked
- in various GIS products are a number of different names used for these associated files:
  - Attributes: Descriptive Data Set (DDS), Polygon Attribute Table (PAT)
  - Coordinates: Geometry, Image Data Set (IDS), Locational Data. Geography
- databases of this type, populated by objects and their attributes, are common in cartographic or CAD (computer assisted design) databases.
  - many common packages for mapping use this structure
    - SAS/GRAPH and ATLAS (from Strategic Locations Planning) are examples

## D. REPRESENTATION OF TOPOLOGY

- the key to a GIS data structure, as distinct from cartographic databases, is the emphasis on the coding of relationships between objects
  - in GIS, the term topology is used to refer to these relationships between objects
- however, the term topology has a much more precise meaning in mathematics
  - topological properties are those which are preserved when an object is stretched or distorted, and are therefore distinct from geometrical properties
    - e.g. a circle can be stretched to form any shape of polygon, but no amount of distortion will make it into a cube
- there is an enormous range of possible relationships between objects (see Unit 12 for a detailed discussion of relationships)
  - simple examples include "nearest to", "crosses", "is connected to"
  - these expressions can be used to relate two objects together
    - for example, each object might be given an attribute which is the ID of the nearest other object in the same class, thus coding a relationship between pairs of objects
- two specific types of relationships are often coded in GIS databases:
  - relationships in networks
  - relationships between areas

### Relationships in networks

- networks consist of two types of objects:
  - lines, also known as links, edges or arcs
  - nodes, also known as intersections or junctions
- a simple way to code relationships between links and nodes is to give each link two additional attributes - the IDs of the nodes at each end (to-node and from-node)
- overhead and handout - Relationships in networks
- thus there will be two types of records:
  - 1. arc coordinates:  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
  - 2. arc attributes: to-node, from-node, length, attributes
- using this structure, it is possible to navigate from link to link by searching for links with matching node numbers
  - the DIME datasets created by the Bureau of the Census for the 1970 Census used this concept to code US street networks
    - each node or intersection was given a unique ID
  - the TIGER line files do not have unique node IDs, but the network can be navigated by matching the  $(x, y)$  coordinates of link endpoints
- however, searching for matches is not particularly efficient

- a better data structure would list the links at each node
  - this could be done by adding a third type of record:
    - 3. node: (x,y), adjacent arcs (positive for to- node, negative for from-node)
      - see overhead
- finally, it is awkward to have to store a variable number of arc IDs for each node, so it might be better to use two files: 1. a simple ordered list which compresses the node file into a string of arc IDs 2. a table in which the position of the first arc in the list is stored for each node

positionarc nodeposition 1 1 a 1 2 -5 b 3 3 3 c 6 4 2 d 9 5 -1

6 -4 7 -2 8 5 9 4 10 -3

### Relationships between areas

- knowing adjacency is important when working with area objects
  - many programs are more efficient if we know which areas share common boundaries
- many systems store boundaries as several individual arcs and include arc attributes (pointers) which indicate which polygon falls on each side of the arc
- by storing common boundaries, instead of complete polygon boundaries, can avoid:
  - duplication in digitizing
  - problems which arise when the two versions of each common boundary do not coincide
- many systems would store this set of three areas using three datasets:
  - overhead/handout - Relationships between areas
  - a polygon attribute table
  - an arc attribute table
  - a set of (x,y) pairs representing the arc geometry
  - note: in ARC/INFO these are referred to as the .PAT, .AAT and .ARC files respectively
- disadvantages:
  - to construct polygons, must search for arcs with correct polygon IDs and then match node numbers
    - for polygon B above, the result would be arcs 3, 4 and 5, with 5 in reverse order
  - this data structure cannot represent area objects which are fragmented - islands, for example

### The CanSIS data structure

- an example of a more fully developed data structure is the database of the Canadian Soil Information System (CanSIS)

developed by the Canadian Department of Agriculture in the 1970s

- has four interrelated datasets, with pointers
- a very simple summary of the CanSIS structure's four datasets is: 1. Object: attributes, first-polygon, last- polygon
  - soil types would be coded as objects
  - an object can describe many discontinuous polygons sharing the same attributes 2. Polygon: object ID, next-polygon, first-arc, last-arc
  - here "object" is the object of which the polygon is a part 3. Arc: R-polygon, L-polygon, next-R-arc, next-L- arc, previous-R-arc, previous-L-arc, first-point, last-point
  - the arc pointers are to the next arcs around the left and right polygons

diagram

- first-point and last-point identify the first and last (x,y) pairs of this arc in the point data below 4. Point: (x,y)
  - the points owned by each arc are stored in sequence in this dataset
- note how each type of record points to records of other types
  - e.g. each object points to the first and last polygons forming the object
  - e.g. each arc points to the polygons on its left and right, and also to other arcs

#### E. DISADVANTAGES OF ARC-BASED REPRESENTATIONS

- areas do not always exhaust the space
  - the method may be inefficient for coding data sets which consist of isolated polygons, e.g. woodlots in an agricultural area, various types of land use, house footprints on an urban map
- areas often overlap
  - a database of old burns in a forest contains polygons which may overlap and do not exhaust the space, so there are few if any common boundaries
- although the great majority of programs work better for arcs than for polygon representations, it is sometimes necessary to rebuild complete polygons from arcs, e.g. for display when a polygon is to be filled

#### F. OTHER ISSUES ABOUT DATA STRUCTURES

- the network and area data structures discussed above reflect common practice in existing GIS, but are far from comprehensive
- a data structure must be chosen to balance the need for: 1. efficient processing
  - arcs are more efficient than polygons for many operations 2. accurate modeling of reality
  - objects are abstractions of reality; the conditions imposed, e.g. non-

overlapping polygons, will affect the accuracy of the abstraction

- the conceptual structure of the data which the system presents to the user need not be closely related to the actual data structure
  - the simple structures described above can be used to present much more complex views to the user

Example: some systems allow the user to work with complex features, which are aggregates of simple features

- a simple feature such as a point can be part of several complex features
- this idea is useful in utilities applications, where it may be necessary to group together several objects, such as a house, land parcel, pipe, shutoff valve and gas meter, into a complex object ("account")

Example: analysts of spatial information must often deal with the fact that reporting zones, such as counties, change from time to time

- Great American History Project - to analyze the spatial distribution of the US population by county since 1800 requires a database which can present the user with different views of the set of counties at different times, as boundaries change
- one solution is to define a common set of arcs, but to build them selectively into area objects at each time period
  - the arcs list contains every line which has ever been a part of a US county boundary
  - the boundaries of objects (counties) are defined differently at each time period
  - an arc is part of the network of boundaries at time period  $t$  if the polygon IDs on its right and left belong to different objects at time  $t$
- data structure would have these record types: 1. Object: attributes at time  $t$  2. Polygon: objects to which polygon belongs at each time 3. Arc: L-polygon, R-polygon

## REFERENCES

Burrough, P.A., 1986. Principles of Geographical Information Systems for Land Resources Assessment, Clarendon Press, Oxford. See Chapter 2.

Haralick, R.M., 1980. "A Spatial Data Structure for Geographic Information Systems," in H. Freeman and G.G. Pieroni, eds., Map Data Processing, Academic Press, New York.

Peucker, T.K., and N. Chrisman, 1975. "Geographic Data Structures," American Cartographer 2(1):55-69.

van Roessel, J.W., and E.A. Fosnight, 1984. "A relational approach to vector data structure conversion," Proceedings, International Symposium on Spatial Data Handling, Zurich, pp. 78-95.

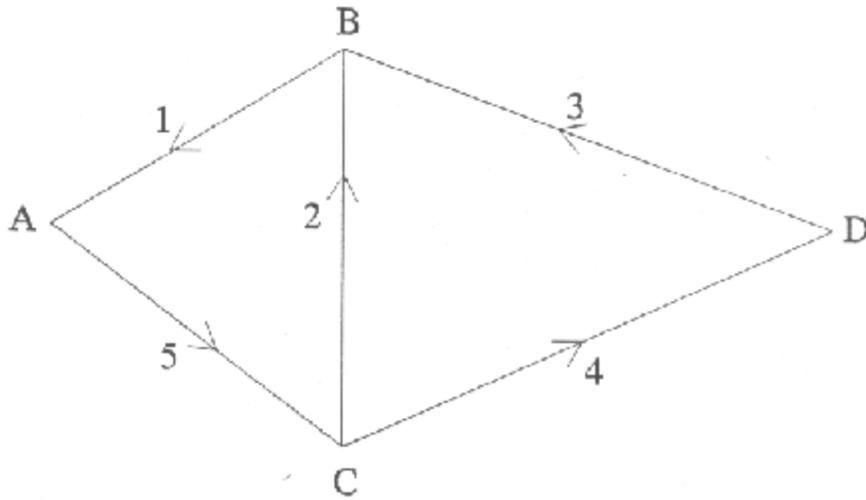
## DISCUSSION AND EXAM QUESTIONS

1. Make a list of the kinds of relationships which can exist between pairs of spatial objects, for each pair of points, lines and areas, e.g. point to point, point to line, area to point etc. Are there any examples of relationships between triples of objects, e.g. point-point-point?
2. Write out the CanSIS data structure for a simple map of three or four polygons, forming an equal or smaller number of objects (include the x,y coordinate pairs) (need to include a drawn example).
3. The GIS industry has traditionally provided data models which assume that within any one layer of the database, polygon objects do not overlap, and exhaust the space available. Comment on the degree to which this assumption has limited the application of GIS databases in specific areas. Are these sufficiently significant to warrant a change of data models in the future?
4. Discuss areas of application in which the concept of a complex feature type would be useful. What operations would you want to perform on complex and simple features respectively?

---

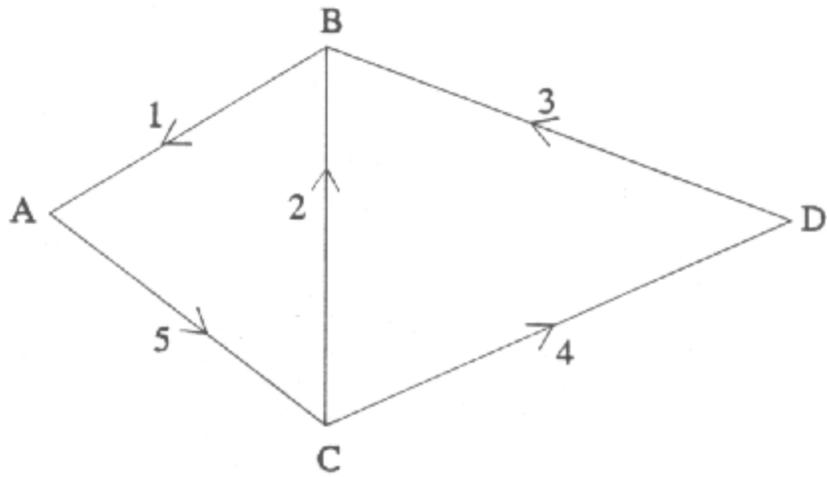
*Last Updated: August 30, 1997.*

# UNIT 30 IMAGES



Arcs	To	From	Attr <sub>1</sub>	Attr <sub>2</sub>
1	B	A		
2	C	B		
3	D	B		
4	C	D		
5	A	C		

Nodes	Arcs		
A	1	-5	
B	3	2	-1
C	-4	-2	5
D	4	-3	

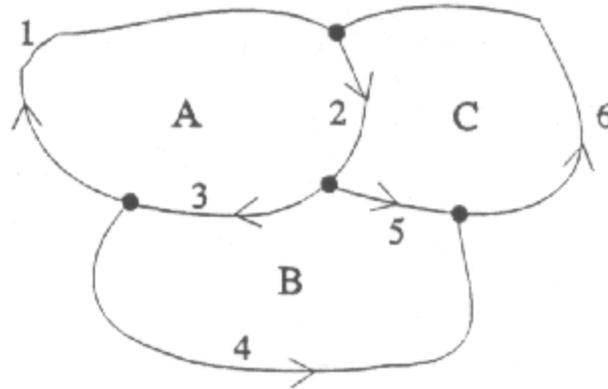


Arcs	To	From	Attr,	Attr,
1	B	A		
2	C	B		
3	D	B		
4	C	D		
5	A	C		

Nodes	A r c s		
A	1	-5	
B	3	2	-1
C	-4	-2	5
D	4	-3	

## Relationships In Networks

## Relationships In Networks



Polygon Attribute Table

ID	attributte 1	attributte 2	attributte 3
A			
B			
C			

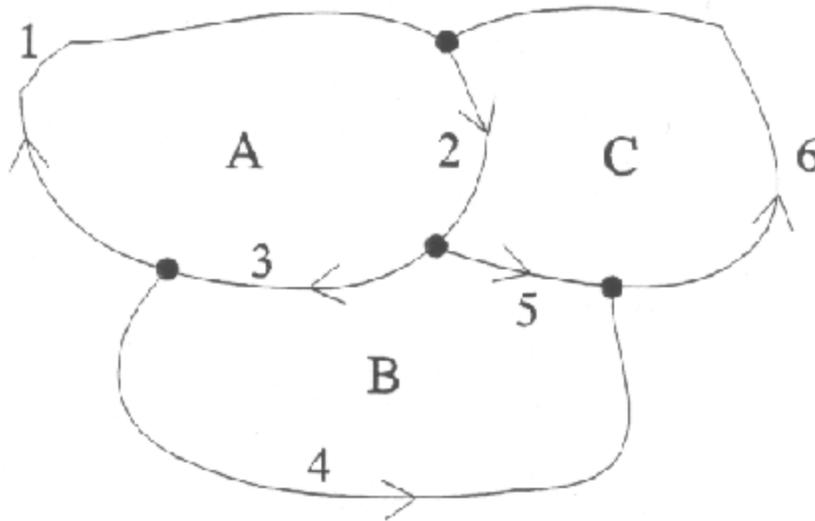
Arc Geometry

1	(x1,y1), (x2,y2), ... , (xn, yn)
2	(x1,y1), (x2,y2), ... , (xn, yn)
etc.	

Arc Attribute Table :

Arcs	R	L	From	To
1	A	0	a	c
2	A	C	c	b
3	A	B	b	a
4	0	B	a	d
5	B	C	b	d
6	0	C	d	c

(0 is the outside world)



Polygon Attribute Table

ID	attributte 1	attributte 2	attributte 3
A			
B			
C			

Arc Attribute Table

Arcs	R	L	From	To
1	A	0	a	c
2	A	C	c	b
3	A	B	b	a
4	0	B	a	d
5	B	C	b	d
6	0	C	d	c

(0 is the outside world)

Arc Geometry

1	(x1,y1), (x2,y2), ... , (xn, yn)
2	(x1,y1), (x2,y2), ... , (xn, yn)
etc.	