# The Geographic Database - Logically Continuous and Physically Discrete

Peter Aronson

Environmental Systems Research Institute, Inc.
380 New York Street
Redlands, CA 92373

## ABSTRACT

In conventional database terminology a distinction is made between the physical description of the database which refers to how the data is actually organized on the machine, and the logical description of the database which refers to how the data appears to be organized to theuser or applications programmer (Martin, 1977). This distinction maybe usefully applied to the geographic database as well.

The total sum of data manipulated by a GIS, both locational and descriptive, can be collectively referred to as the geographic database- the map database or library. These geographic databases can varygreatly in quantity of data employed - from the limited project basedon a single map sheet and one or two layers of data, to the detailed national database based on thousands of map sheets with hundreds of layers. And while the former case can be handled simply by a single simple data set, the latter case presents additional problems.

For a large geographic database, it is important that the logical view of the data should be continuous - without artificial breaks or storage artifacts. But for the same large geographic databases it is equally necessary that the physical storage scheme used allow for fast random reading and writing of map elements. With current storage technology, such access requires storage of the map data as subsets discrete by locational and descriptive criteria. This paper will discuss the issues raised by such a geographic database architecture and thesolutions arrived at in the ARC/INFO GIS.

## INTRODUCTION

The tool used by the ARC/INFO GIS to access, manage, and maintain large geographic databases is the LIBRARIAN subsystem (smaller geographic databases are handled by the core GIS in a more ad hoc fashion). The design of the LIBRARIAN subsystem was begun six years ago and made public five years ago (Aronson and Morehouse, 1983). Since that time, the product has evolved considerably. This paper is in part a report on that evolution and in part a discussion of the problems of geographic database *creation, maintenance, and access.*

The paper is divided into four basic sections: the first section defines the basic problems of the GIS geographic database; the second section details the solutions to those problems used in the LIBRARIAN subsystem; the third section describes the evolution of the LIBRARIAN software over the last five years; and the fourth section considers future directions.

## THE GEOGRAPHIC DATABASE PROBLEMS

Geographic databases, like all databases, must be constructed in, alas, a less than perfect world. Data arrives from multiple sources in multiple formats in multiple scales, projections, and with varying data extents; all of which somehow must be integrated into a single geographic database. The geographic database must be organized in such a fashion to make the organization's normal operations sufficiently quick that the organization can use the geographic database to meet its goals. The geographic database must be maintained and updated as required - a task somewhat more complicated than the maintenance of the traditional database.

Very few organizations collect all of the data that they use themselves. Much of the data is either purchased from a commercial data source or, obtained from some government agency. In the Dane County, Wisconsin example described by Chrisman (Chrisman and Niemann, 1985), the seven layers in the database were provided by five organizations: two federal, one state, and two county. This is typical of land records information in this country. In the commercial sector the situation can be even more complicated, since data is often purchased from multiple service bureaus.

This outside data may be supplied in any variety of format, scale, and projection. The areal extent of one supplier (say seven and one-half minute quandrangles) may not match that of another (SMSAs). It may be updated at different intervals (say every two years against every ten). Data integration is a continuing task all through the existence of a geographic database.

Some geographic information systems organize the data themselves; most require that the user make certain decisions on how the system will organize their data. In either case, the data must be organized in a fashion that: allows the data to be useful (that is, supports the organization's access to the geographic database); allows the data to be accessed with reasonable speed; and allows the data to be maintained without great difficulty. There are always trade-offs involved in these decisions - speed versus ease-of-update, ease-of-use versus flexibility, and so forth (anyone who tells you otherwise is a salesman).

Maintenance and update are where geographic databases can be the most difficult. In normal database work, a unit of work is referred to as a transaction. A typical transaction might consist of updating all salary fields of managers in an employee table. This is simple atomic operation in a conventional database management system. In a GIS, geographic database transactions are not so simple.

The type of transaction that we have in GIS geographic database maintenance is what is referred to as a long transaction. In a long transaction, slices of the database (the geographic database in our case) are extracted for update, worked on, then reinserted (Beller, 1988). Where a conventional database transaction lasts (typically) seconds, a GIS long transaction routinely requires days to complete. Simply making the involved data unavailable during the transaction may be unacceptable due to the length of the operation. A more complete solution is required.

Once you've solved the problems involved in building a geographic database, there is still the question of how it is to be used. In general, there

are three purposes for which a geographic database is used: 1) generation of routine products; 2) query; and 3) the generation of ad hoc products.

Routine products, such as yearly forest harvest maps or zoning update maps, are those that are produced according to regular procedures at either known times (spring) or in response to a predictable stimulus (zoning changes). These products are the raison d'etre for most geographic databases, and the bread and butter of the organizations maintaining them.

The term "query" describes a whole host of applications, including (but not limited to): informal database examination; ad hoc generation of graphics and reports; simple modeling; and examination of geographic database status for management purposes. Their common characteristic is that their subject cannot be predicted in advance, and their output is ephemeral. These are common applications for users such as planners and database administrators. For most organizations, query applications are an insufficient reason to build a geographic database on their own merit, but are considered a valuable secondary benefit from constructing the geographic database.

Ad hoc products lie in a region between query and routine products - like queries, their subject cannot be predicted in advance, but like routine products, there is concrete output. For those organizations with complex responsibilities, these can be an important type of application, even to the degree of justifying the geographic database's existence. For organizations with simpler responsibilities, ad hoc products may be much the same as queries - not vital, but a valuable side effect of having built the geographic database.

## THE ARC/INFO SOLUTION

The form that medium-to-large geographic databases take in ARC/INFO is the map library. The tool used to manage and maintain these map libraries is the LIBRARIAN subsystem. Access to map libraries is either via the cartographic and editing subsystems (ARCPLOT and ARCEDIT) or via copies created by LIBRARIAN.

The basic design principle of the LIBRARIAN is normal use - that is, the LIBRARIAN is designed to support best the operations that are performed the most. The highest priority is hence given to tools that aid in the production of routine products, the next priority to tools for query, and lowest to tools for ad hoc products. Not to say all three kinds of operations are not supported; they are, but map library is designed to give the fastest results in production of the routine product, then query, and then all other operations (the production of ad hoc products is not necessarily slow, just not as optimized).

The map library is a device which allows geographic data to be organized into a large, complex geographic database. Coverages (a digital form of a single topic map) are organized simultaneously in two dimensions -- by subject or content into super-coverages called layers and by location into tiles (see Figure 1).

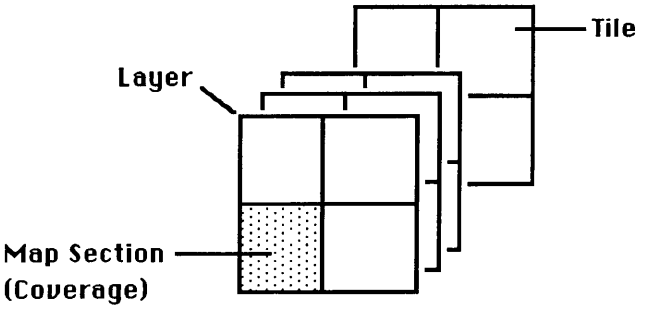


Figure 1: Structure of a Map Library

Tiles. The geographic area represented by a map library is divided into a set of non-overlapping tiles. Although tiles are generally rectangular (e.g., 30' squares), they may be any shape (e.g., counties or forest administration units). Tiles are a digital analogue for the map sheets of a conventional map series. All geographic information in the map library is partitioned by this tile framework. The tile layout is determined by the map library administrator at creation time.

Layers. A layer is a coverage type within the library. All data in the same layer have the same coverage features and feature attributes. Examples of layers are land sections, roads, soil types, and wells. A layer is logically a coverage, but can be physically multiple coverages - the user deals with the layer as single entity although the software may actually deal with multiple entities.

Map Sections. Once a layer has been subdivided into tiles, it consists of a set of individual units called map sections. A map section is simply a coverage as defined previously. Map sections are a physical storage entity, not a logical one.

The tiles are defined in a special INDEX coverage, where each polygon in the INDEX coverage represents a single tile in the library. Layers are defined by defining the feature classes present and the thematic data associated with each feature class.

Map libraries are used for four basic geographic database operations: 1) maintenance (including creation); 2) supplying data for routine products; 3) query; and 4) supplying data for ad hoc products. Each of these topics is discussed below.

Mapbase Maintenance

The basic organizational concept of geographic database maintenance in LIBRARIAN is the long transaction. The software supports long transactions by means of a data object called a named transaction. When a named transaction is created, it owns a set of map sections. These map sections may not be modified except by that particular named transaction. Data is checked out by the named transaction when extracted from the map library, and checked back in when returned after update.

A typical use of a named transaction would be to update a group of street maps. First, the library manager would use the SETTILES and SETLAYERS commands to restrict the tiles and layers affected, like this:

```
Librarian: LIBRARY URBAN

You have entered transactional library URBAN with MANAGER access.

Librarian: SETTILES LOWERTOWN UPPERFALLS WESTEND
Librarian: SETLAYERS NAME STREET
```

Having selected the area and topic of interest, the library manager would then begin a named transaction, extract the data (so that it can be edited), then leave LIBRARIAN to do the actual editing, like this:

```
Librarian: TRANSACTION BEGIN UPROADS/89 Updating STREETS ~
Librarian: in Traffic Zone #02 for 89 road repairs
Librarian: EXTRACT DISSOLVE WORK>DATA>EDIT
Librarian: QUIT
```

Beginning a transaction marks all the map sections specified by the set layers and set tiles as belonging to that transaction. Performing the extract marks those map sections within the transaction that contain data as checked out. An entry for one of the map sections in the map sections involved in this transaction would read:

```
TILE:         LOWERTOWN
LAYER:        STREETS
TRANSACTION:  UPROADS/89
STATUS:       OUT
```

With there being one such entry per map section marked by the transaction.

After leaving LIBRARIAN, the extracted data would be edited. When editing is complete, then the library manager would run LIBRARIAN again, set the proper transaction, re-insert the data, and then end the transaction (assuming it is all done), like this:

```
Librarian: LIBRARY URBAN

You have entered transactional library URBAN with MANAGER access.

Librarian: SETTILES LOWERTOWN UPPERFALLS WESTEND
Librarian: TRANSACTION SET UPROADS/89
Librarian: INSERT STREETS STREETS
Librarian: TRANSACTION END All done and checked off
Librarian: QUIT
```

If there had still been data checked out, then the transaction end operation would have failed, with the error message listing the errant map sections.

Routine Products

Routine geographic database products can be divided into two basic catagories: those requiring modeling, and those that only require data selection and symbolization. Those that require modeling, LIBRARIAN handles by extracting them from the map library, so that the modeling can be performed in the user's local workspace. This keeps the temporary data sets generated by the modeling process out of the permanant geographic database. Those products that do not require modeling are produced directly out of the geographic database.

As stated above, the basic design principle of LIBRARIAN is normal use. This is most critical in the case of routine products, which are by definition normal use. It is here where the user-defined tiles can be particularly valuable. By selecting a tile grid or tessellation that matches the boundaries required by a majority of the routine products, operations on the geographic database can be extremely efficient. (For a more complete discussion of how the user determines the tile boundaries, see Keegan and Aronson, 1983.)

To support this approach, LIBRARIAN supplies two sets of commands: coverage based commands that use the outer boundary supplied from a specified polygon coverage; and tile based commands that operate on tiles specified by name.

| Operation | Coverage based | Tile based |
|---|---|---|
| Determine working set | SETCOVER | SETTILES |
| Extract from database | EXTRACT | GETTILE |
| Insert into database | INSERT | PUTTILE |

The tile based commands operate on tile name and require very little computation or processing. The coverage based commands require calculation of tiles overlapped and the assembly of separate pieces or the splitting into pieces of the data. Hence, this approach yields fast response on the normal use case, but still supports other operations.

For those routine products that don't require modeling, ARC/INFO's graphic output engine, ARCPLOT, is capable of operating directly from the map library. Since ARCPLOT not only displays and symbolizes data, but can produce subsets based on spatial and thematic criteria, change projections, locate by address, and perform simple statistical modeling, a reasonable percentage of products can be produced directly from the map library. And as an added benefit, ARCPLOT serves the front end for cartographic publication product generation - output from ARCPLOT can be sent to PostScript typesetters or to a Scitex graphic production system.

Query

Because the two functions have considerable overlap, the primary query engine for ARC/INFO is its graphic output engine, ARCPLOT. ARCPLOT can produce graphics of all sorts, reports, tables, statistical analysis, and limited derived data sets. Combined with front-end programs written in ARC Macro Language (AML), it can be customized to perform specialized or routine queries. Features can symbolized and data accessed from tables stored in

ORACLE, INGRES, or SQL/DS, as well as from INFO. For the vast majority of query processes, no actual extraction from the map library is required.

An additional query-like operation is performed in ARC/INFO's coverage editor, ARCEDIT, where data from the map library can be used as a backdrop during the digitizing and editing process. This can help ensure that data involved in a long transaction matches the data still in the map library where desired.

It should be noted that while data is involved in a long transaction, it is still available for query purposes. Given the length of these transactions, it would be undesirable to shut the library down until they were complete.

Ad Hoc Products

Like routine products, ad hoc geographic database products can be divided into those requiring modeling and those that can be produced solely by use of the graphic output engine. Those requiring modeling supply a more interesting problem, as the other case is essentially identical to query.
To support the production of ad hoc modeling products, LIBRARIAN supplies several data extraction options. The area to be extracted can be defined by either the outer boundary of a coverage or by a list of tiles. Data can be extracted by whole tile or clipped to fit a boundary. Features split by the storage scheme can be aggregated to their original form or left divided.

An example of the use of the map library to support the creation of an ad hoc product might consist of the extraction, clipping and aggregation of two layers over an arbitrary region.

```
Librarian: LIBRARY URBAN

You have entered transactional library URBAN with MANAGER access.

Librarian: SETCOVER RIVER_BASIN
Librarian: SETLAYERS PARCELS SOIL
Librarian: EXTRACT DISSOLVE * CLIP
Librarian: QUIT
```

This operation would create two coverages (a soils coverage and and a parcels coverage) that contained all the polygons contained within the river basin, clipped to the boundary of the river basins, with the tile breaks removed. The data extracted from the map library is logically identical to that stored within it, even if it may be stored in a physically different manner.

## THE EVOLUTION OF LIBRARIAN

The evolution of the LIBRARIAN subsystem is primarily of interest on account of the lessons learned during the process. LIBRARIAN was formally released as part of version 3.0 of ARC/INFO in early 1985, and has undergone considerable changes in the four years since. Most of these changes have come about in response to user requirements. Nothing matures software like actually being used for real applications.

The LIBRARIAN software described five years ago was a smaller, simpler

system. There were no named transactions, no tile based operations, no access from ARCPLOT or ARCEDIT. It was just the bare bones of a simple map sheet management system. Since that time, every release of ARC/INFO has contained improvements to the subsystem. Described below are some of the changes and the requirements that brought them about.

Tile Based Operations

The original requirement for extraction from the map library was that the user had to have a coverage of the spatial extent of the area to be extracted. However, this is a strange requirement for users who have done a careful tile layout to support their normal use - they usually want one tile at a time, and they know its name. Some of our natural resource users went so far as to produce programs to generate a coverage for extracting whose boundary was identical to a particular tile's border! The same problems applied on insertion as well. Adding these commands made operations far simpler for many of our users.

Graphics and Query Access

The original version of LIBRARIAN included a module called QUERY which was a simple graphics and listing generator that operated on map libraries. QUERY, however, had very limited symbolization capability, and was yet another program to learn. By integrating map libraries into ARCPLOT, the full capability of ARC for graphic production is available straight from the map library. This limits the amount of data that needs to be actually extracted from the map library.

Macro Language Support

The addition of a macro language to ARC/INFO came at version 4.0, released in 1986. At this time the LIBRARIAN subsystem was reorganized in part to make use of AML to write menu-driven front ends more practical. Since then, users have requested (and received) additional functions in AML to support the writing of macros that drive LIBRARIAN, allowing non-technical users to access the map library without really knowing that it exists.

Spatial and Attribute Indexes

While not added specifically to LIBRARIAN, these additions to ARC/INFO very positively impact the use of map libraries. The spatial indexes consist of an improved quad-tree structure which speeds up spatial queries and subsetting. The attribute index is a B+ tree structure that improves thematic selection and scans of lookup tables. These indexes made casual query usable interactively even on the largest map libraries.

Transactions

Transactions were added to deal with highly dynamic libraries, such as a parcel database for a very large city, where there will be many changes per day, and hence some form of collision management is required. In such an organization, there will often be several individuals or even groups responsible for geographic database update, and a mechanism was definitely required to make sure that the process flowed smoothly.

## FUTURE DIRECTIONS

While LIBRARIAN is a mature product it is not a senile one. By not senile, I mean that it has not become so bastardized and rococo that further improvements are difficult or even impossible. There are future extensions of LIBRARIAN planned, due to both the ARC/INFO design team's plotting and requests from our users (who as a group are not noted for either shyness or an unwillingness to speak their minds).

Currently a layer and a coverage are much alike. There is no particular reason (aside form some work required) why they couldn't become functionally identical. Anywhere a coverage is accepted, a map library layer would be accepted as well.

One concept we would like to add is that of geographic database views. In SQL, a view is a table derived from another table or tables by way of an expression that may contain joins, selection, aggregation, etc.. A geographic database join would be a coverage or layer that is derived from other coverages or layers via selections, projections and transformations, and possibly spatial joins. Depending on the amount of processing allowed in the view actualization, nearly all modeling could be performed using this mechanism.

## CONCLUSIONS

Large geographic databases present problems somewhat different from other types of databases. Maintenance problems may be dealt with by use of long transactions. Production of routine products can be added by structuring the geographic database in such a way as to facilitate efficient operations by the spatial divisions required by those products. Query and cartographic production is facilitated by integrating the geographic database into the graphic output engine. Ad hoc products require flexibility in accessing the geographic database.

The ARC/INFO map library is a data structure designed to meet the above goals. It has been in general use for four years now, and has been upgraded to meet user requirements in that time. It is a mature software product, but not yet a senile one.

## REFERENCES

Aronson, P. and Morehouse, S., 1983, The ARC/INFO MAP LIBRARY: A Design for a Digital Geographic Database, Proc. Auto Carto 6, 1983.

Beller, A., Concurrency and Recovery for GIS Databases, unpublished paper, 1988.

Chrisman, N. and Niemann, B., Alternative Routes to a Multipurpose Cadastre: Merging Institutional and Technical Reasoning, Proc. Auto Carto 7, 1985. p. 84-93.

Keegan, H. and Aronson, P., Considerations in the Design and Maintenance of a Digital Geographic Library, Proc. Auto Carto 7, 1985. p. 313-321.

Martin, J. 1977, Computer Data-Base Organization, 2nd edition, Prentice-Hall, New Jersey.