A Fully Integrated Geographic Information System

Dr. John R. Herring
INTERGRAPH Corporation
One Madison Industrial Park
Huntsville, Alabama 35807-2180

ABSTRACT

The major problem in GIS implementation is the diversity of data types and data base management systems that can carry geographically related data. A fully integrated GIS must be able to handle data in various formats (vector, topology, attribute, raster, grids, TINs, etc.), providing a environment where they can co-exist, and interact. It must also provide standard interfaces to external data bases; foreign in structure, schema, DBMS, and machine environment. This paper describes the some requirements for and approaches to a fully integrated GIS.

INTRODUCTION

The last thing the Geographic Information System (GIS) community needs is yet another definition for GIS. Some current definitions are "that which exists" (a current useful product in geographic analysis) or "that which I do" (based on a particular product or application). This paper tries to remove this bias and establish a definition and basic requirements for "that which should be."

To begin, let us distinguish between data management and applications. The need in the GIS user community is for applications, which perform tasks for data collection, extraction, analysis, and product generation. But, neither singly nor collectively, do applications support the data management tasks most effectively done by a DBMS, usually a Relational Data Base (RDB), in the business community. If we make the same distinction, defining a GIS as "a DBMS for data having geographic significance", or "a DBMS upon which geographic applications can be built"; then a GIS must:

- o provide a common interface to organize, load, extract, and report on all types of geographic data.
- o provide a generic query environment to perform analysis common to most applications.
- o maintain ("serve, protect, and defend") the integrity and the validity of geographic data

The rest of this paper draws conclusions about what such a GIS must be, and presents requirements and potential solutions to fundamental problems. Since no restatement can truly begin unencumbered by the past, several technical controversies on GIS implementation are addressed directly.

WHO USES A GIS

A GIS manages geographically significant data, and supplies combined spatial and attribute analysis tools. Given this, who are the users of a GIS? We can classify the handling of geographic data into four categories (not necessarily mutually exclusive): collect, merge, validate, analyze, and produce products.

To collect data is to extract it from non-data-base sources, verify its consistency with that source and convert it to the appropriate data base format. The immensity and the importance of the verification task requires a number of automated and semi-automated tools to discover potential errors before the source is released. This includes such diverse problems as attribute range, consistency and validity checking; and geometric anomaly prevention, detection and correction. These tools require a GIS. In fact, the verity of the collected data is so pivotal to the validity of the results of any GIS application, and analysis is so important to verification, that it should be emphasized that no on needs GIS functionality more than the person collecting data.

To merge data is to combine multiple sources of collected data into a single environment appropriate for further manipulation. The merge process must be able to recognize when two feature representations are in fact different versions of the same feature, separated by collection method, statistically reasonable error, or time; and to be able to combine these multiple representations in a statistically valid compromise. A GIS is needed.

To validate data is to assure its internal consistency. To analyze data is to derive implicit information from explicit data. Combined spatial-attribute validation and analysis are the classic GIS applications, often confused with the GIS itself. They certainly require a GIS.

To produce products from data is to convert the data base information into a form directly consumable by a client (either a human or other digital process). As such, it can involve a massive reshaping of the data. For example, the production of paper or digital maps must contend with generalization, agglomeration, aggregation, and conflict detection and resolution. Even in a semi-automated environment, this very complex spatial analysis requires a GIS.

So the answer is "anyone who handles geographic data."

DATA TYPE COEXISTENCE

The first major separation between GIS's and other DBMS's, is the data itself. No non-spatial, and few non-geographic data bases have such a wide variety of data types. These types include, but are not limited to the following:

- o feature attribute information;
- o structured (topological) vector graphics;
- o unstructured vector graphics;
- o raster representations maps, aerial and satellite photography (grey-tone, color and multi-spectral);
- o TINS (triangulated irregular networks), grids, contours and other elevation models;
- o non-elevation TINS, grids, and raster information for the representation of analytical surfaces (e.g. soil permeability, cost functions, demographics);
- o 3 and higher dimensional equivalents for TINS (simplicial complexes) for subsurface geology;
- o projection, transformation and other coordinate information (including projection parameters, datum; primary and alternate units of measures; digitizer setups (table to screen); co-registration parameters

(e.g. best fit functions between raster-vector, raster-raster, or vector-vector data);
- o relational information linking all of the other data types together;
- o schema information, describing the application specific parameters for the other types of data (attribute names, types, and ranges; feature types, representation rules and display parameters; etc.).

The first coexistence problem is to co-register any of the various data types to common geographic coordinate frames of reference; but this is only the beginning. The disparate data types must be simultaneously manipulable; all data from a single geographic area, regardless of format or content, must be read and write accessible within a single GIS process; subprocesses such a commands must manipulate as many of the data types as logically feasible.

Structured Versus Non-structured

There is a continuing controversy in the GIS community between two seemingly disparate philosophies; between "non-ambiguous structured data model" and "a flexible and simplified data model." To explain this, let us investigate some of the major points of contention.

Real-time topology versus on demand structuring: No one disagrees that topologically structuring of vector data is a boon to spatial analysis (Herring-86), but there is a controversy as to the manner in which the system maintains such a structure. The two logical alternatives are real-time maintenance (topology is always valid) and on-demand updating (topology is frozen at user chosen points and bulk updated or recalculated as necessary for analysis). These two alternatives are solutions to different problems. In older and less powerful workstations or systems, the real-time load for topological maintenance quickly uses unacceptable amounts of system resources. In the newer workstations or system (with 0.5 MIPS or better per user), the excess capacity of the system can be used to take advantage of a real-time maintenance system. These advantages include at least the following:
- o ad-hoc combined spatial-attribute query and analysis
- o real-time geometric anomaly detection and prevention
- o better system utilization (% use of machine cycles)
- o intelligent user-feedback based on automatic analysis

The most obvious recipients of these benefits are the data collection tasks (with the real time validation and verification) and what-if-analysis tasks (faster turn-around on spatial queries or analysis based upon recent and tentative geometric changes). In summary, real-time systems provide a more flexible environment, and are preferable assuming sufficient system resources to maintain interactive response times. In non-interactive processes, maintenance is always less expensive than reconstruction.

Relational versus Object-Oriented Data Model: This is a true non sequitur for a very simple reason: almost all GIS's (even those using a RDB) are already object oriented (see Ullman-88). An "object-oriented system" is one which supports an abstract concept "object" ("entity", "feature") having existence independent of any attributes that entity may or may not have. The opposite of this is a "value-oriented system" which models only attributes. For example, in a valued-oriented system such as a "classical" RDB, the entity (and any tuple representing it) owns it existence to a non-null key attribute combination, and any pair of entities (sets of tuples) having equal (key) attributes are equal (duplicates are eliminated). Such pure value-oriented systems can create confusion, such as two employees with the same name getting each others paychecks. To prevent this, most applications using RDB's assign "employee numbers." This same natural approach appears in linked graphics-attribute systems, where graphics are given identifiers ("graphic link", or "feature id"). This moves the GIS from the "value-oriented" world to the "object-oriented" world, since the system now gives meaning and existence to the graphic or feature "objects" independent of their spatial and non-spatial attributes. This does not mean that Object Oriented Data Bases (OODB's) are being used, since other requirements are levied against such system (at least encapsulation and abstraction), but it does imply that, taking use into consideration, many of the object-oriented concepts are natural extensions of the relational model. In fact, some classical problems in RDB design, referential integrity and normal forms, lead into object concepts. For example as early as 1980, three rules for converting an entity-relation model to a relational one were laid down by Wong and Katz (Chap. 21, Stonebreaker-86) ("()" added to identify equivalent OODB concept):

1) each entity set (object) has an explicit identifier (object id) which represents it globally in the relational model
2) the identifiers (object id's) of a primitive object (class) together with all the primary functions (attributes) of the primitive object are grouped in the same relation in the relational schema
3) there is one and only one primitive object (class) per relation of the relational schema

They go on to show that these "mapping rules" lead to fourth normal form (4NF) RDB implementations. Looking back on this from the OODB point of view, we see that Wong and Katz have essentially proven that a straight forward, formal implementation of an object model in a RDB gives a 4NF relational data base schema. Since this early parallelism, further work has brought the two data models closer together. For example, Rumbaugh-87 proposes including relations in formal OODB models; Blaha-88 suggest using object-oriented models to design RDB schemas, formalizing what now occurs naturally (see above); and Ullman-88 proposes an extended entity-relation model as a super-model for both RDB's and OODB's.

A complete GIS system, regardless of its implementation details, must be able to communicate with all data bases capable of storing geographic data, including both RDB's and OODB's. Since DBMS can only rarely exchange raw data, this implies compliance with standard exchange formats and protocols (possibly based on object-oriented extensions of SQL standards, see Herring-88).

Single-content Multiple Layers versus Multiple-content Single Layer: (whether to merge all the data into a single integrated geometric structure or to maintain separate layers, merging only for analysis). These two options are actually the opposite ends of a broad spectrum of possible data-base schemes involving various levels of integration. Each application, and each user has differing needs based upon the particular workflow, analysis requirements, processing power, inter-department interfaces, etc. Even then, the requirements may vary between subsystems within the users' DP community. The considerations in deciding which data types (themes) to aggregate into single topologies (layers) are quite varied. They include at least the following:

- o the amount of shared geometry between the themes,
- o the degree in which they are combined in the usual course of GIS processing,
- o the common sourcè and maintenance responsibility.
- o data complexity and storage requirements within a theme, and
- o the geometric or non-geometric quality of a theme.

Sliver and gap detection, elimination and prevention is very costly, so that themes which tend to share a great deal of their geometry should probably be integrated at the master data base level to prevent high initial access time for applications requiring integrated data. The same is true for data that is often used in combination during GIS processing, again to save the repeated cost of the merge process. Common source and maintenance responsibility allows for easier integration.

So each user must be able to select from the broad spectrum of data structures, most often deciding on a master data base with multiple layers each with multiple content.

Implicit Data versus Explicit Data: In a RDB, some contend that relationships are only implicit since they are "derived" through the join process. While technically correct, this view ignores the fundamental problem of referential integrity. Suppose for example, that in a Wong-Katz RDB implementation, there are tables for road "segments" and other tables for "highways," and that these entities are linked via a specific ownership relation. Thus, one of the segment tables would have an attribute for highway-number, acting as a foreign key into the highways tables' primary key. If the user wishes to report on "all road segments that belong to particular highways," then the needed join on highway-number is valid since a common value means ownership (is valid semantics). If the data base exhibits referential integrity, then each value of highway-number in a segment table corresponds to a valid and correct highway-number in a highway table. On the other hand, if the join where done on other integer valued fields (such as segment.width-in-meters = highway.age-in-years) the results might be nonsensical. Thus, even though the tables appear to be without structure, there is an implied structure based upon the semantic meanings of attributes.

The distinction then between structured and unstructured data is whether or not the DBMS system is aware of the structure that naturally exists within the data. In a GIS, this includes common attributes and common geometry. In a RDB environment, this means that the system is aware of any foreign key attribute value and support procedures to maintain referential integrity; in a OODB, this means that any foreign key is implemented through object relations.

Valid non-causal joins (those based upon attribute relations not involved in foreign-key to primary-key linkage), represent a extra-system (user) interpretation of the data. Thus, the user is placing an interpretation on the data base not considered during its data structure design phase. Whether this query derives valid information depends upon the correlation between the semantic interpretation of the data analyst and the data extractor.

Thus, a GIS, during normal operations, should maintain the correct interpretation of the data (referential integrity). But, in non-causal joins such as used in statistical analysis, the GIS must allow the user to override the default interpretations.

Integrated versus Linked Attribute-Graphics

The most obvious problem in a system supporting such a wide variety of data types is to minimize the number of lower-level data management systems involved in the GIS. The GIS must hold and manipulate several types of stored (persistent) data as listed above, in addition to non-persistent, application data (point buffers, the temporary results of query, display information, window information, temporary command information, etc.).

The most controversial element is the linkage between vector (or topological) data and feature attribute data. The classical approach is to split the two data types at the graphics-to-attribute juncture. This creates a gap that must be hurdled every time a combined spatial-attribute edit or query is executed. The extreme alternative is to place all attribute information in the same DBMS as the graphic information. This is not generally feasible in the RDB world due to its value-oriented programming implementation. Graphic operations (such as simple display) require the access of large amounts of diverse data (an average graphic window might contain 500 kilobytes of information distributed among 2,500 data items of differing types). This would require an RDB to access multiple tables, and return large amounts of structured data. Currently, commercial RDB's are simply incapable, by several orders of magnitude even on the best systems, of reasonable response times in such situations. Further complicating the process, RDB's usually require 1NF (first normal form, all column data types are simple), while geometric data in inherently not 1NF (e.g. coordinate lists). OODB's by their very definition, and some of the more theoretical RDB's, support abstract data types and alternate access methods such as triggers (Stonebreaker-86, Andrews-87) which render such problems solvable.

On the other hand, a single data management system for attributes and geometry simplifies the system. making real-time topological maintenance (see above) and integrated spatial-attribute ad-hoc query (see below) possible. This implies, as in the real-time topology discussion, the implementation of the geometry-attribute linkage is a function of the performance level and sophistication of the system.

When large amounts of static, persistent data are linked with the geographic data (such as well logs in petroleum applications), data size can become a problem. Therefore, the application or user must decide which data to integrate and which to segregate (to other nodes in a distributed system, see below). The criteria that affect this decision are basically the same as the layering criteria below (i.e. some of the layers are non-geometric). Some data such as raster, and dense grids are so storage space intensive, they probably should be stored in the most efficient manner possible such as indexed run-length-encoded or quadtree structured files. Even so, foreign linkages must exist in main DBMS to hold integration information such as co-registration parameters (see below).

In summary, some degree of tight integration between attributes and geometry is required to support full GIS capabilities; but, the degree of integration between layers of the data should be user and application controllable system configuration decisions.

Communication

Once coexistence has been achieved, the next most important criteria is communication between the data types. For example, the passing of geometric descriptions should be possible between any two domains. For example, the raster subsystems must be able to perform classification algorithms based upon vector area criteria; and raster line-following and polygon classification must be usable in vector digitization and spatial query. This requires that each of the data subsystems support common protocols for the transfer of at least geometric information. Across diverse systems this implies universally acceptable exchange standards. Within a single multi-content system, internally defined protocols are more efficient.

SPATIAL QUERY

It is in spatial query that a GIS is most distinct from standard DBMS implementations. For example, in a RDB there is no interpretation of the data in spatial terms, since such interpretations depend upon the particular abstract representation of space chosen by the application. Thus no reasonable set of spatial operators can exist in the pure form of the RDB model. On the other hand, a GIS data base system sole purpose is to incorporate spatial operations into the other more conventional DBMS functions.

This leads to a nearly insolvable problem. Spatial extent is different from other data types: non-1NF, non-declarative geometric algorithms even for simple comparison (e.g. point-in-polygon), etc. In fact, many spatial data bases separate the spatial and non-spatial data into two systems (see above). This usually means that an integrated spatial query language is impossible, and such systems must rely on a tool-box approach to spatial analysis, implementing spatial operators as separate procedural code which the user alternates with the non-spatial query language to do analysis. This is the ultimate (worst) in procedural approaches, forcing the user to specify each transition from the spatial arena to the non-spatial arena ("procedural" queries specify how data is manipulated, "non-procedural" or "declarative" query specify what results are required, leaving the procedural decisions to the DBMS). Spatial operators are thus object-oriented and essentially procedural, and out of the usual domain of a declarative query language.

This leads to an interesting paradox. GIS's built on RDB's should expect to gain the advantage of the non-procedural, declarative query language, but do not due to the nature of the spatial data; leading to a procedural spatial query and analysis environment. GIS's built on OODB's, which are naturally procedural (see Ullman-88), supply as applications those procedures needed to do spatial query and analysis, thus allowing the GIS users and applications to reside in a declarative, non-procedural environment (see Herring-88).

FOREIGN DATA BASE LINKAGES

The data size in GIS's, and the need to access large volumes of non-spatial data, require that the question of distributed, non-homogeneous data bases be addressed. Since much has been written on the general problem of distributed data bases (e.g. Ullman-88), we will concentrate on the mechanism to link the data within the data bases together.

Geographic Linkages: The most common way of linking GIS data is common geographic location. For disparate data contents (different layers), this is sufficient since most co-location is not causal and subject to some statistical interpretation anyway (this data-layer independence is a measure of the success of the layering, see above). For common data content (dependent layers or adjacent data collection cells), it is not sufficient. Absolute error is much larger than the possible relative error, and usually larger than the micro-structure of the geometry (such as real road misalignment at intersections).

Explicit Linkages: In a RDB, explicit linkages, as internal foreign keys, are implemented by a set of common attribute values. In a GIS implemented upon the RDB, these linkages can be logically either value-oriented or object-oriented, depending upon whether the system allows direct application or user access to primary and foreign keys. In a OODB, such linkages are maintained by the DBMS itself through linkages based upon internal object id's. Value-oriented linkages are still possible through common attribute values.. As within a single DBMS, concerns of referential integrity suggest that the GIS maintain object-oriented linkages.

THE DATA SERVER

The environment of the GIS with its multiple data formats, partitioned data, multiple foreign data linkages, places heavy requirements on the management of data at the macroscopic level. This management system, here called a data server, must fulfill at least the following functions:

- o maintain data on the content, accuracy, format, geographic extent, and storage location of all data sets, including versioning;
- o maintain source utilization and production histories;
- o translate standard exchange formats to and from internal formats;

o control access to data;
o manage schema and view information

Much of these are standard DBMS concerns, so we shall limit ourselves to items which take on a special meaning due to the geographic nature of the data.

The Geographic Index: The geographic index necessary for a distributed GIS is a GIS itself, since the partitions are associated to their spatial extent. Spatial query and analysis tools used directly on the data are needed in the index to support production management and distributed analysis (analysis covering some number of partitions, executable from the index, which distributes processing and data to the appropriate partitions, possibly distributed).

Schema Management: In any non-homogeneous distributed data base, the schemas of partitions may vary based on layer, data format and type, etc. Further, data collection and various analysis tasks may require different schemas (simple for collection, complex for analysis), which in turn differ from master data base schemas. The data server must provide a set of schema management tools such as:

o a generic schema definition interface independent of underlying DBMS's (facilitating inter-DBMS exchange);
o the association of partitions to appropriate schemas;
o update functions for maintaining consistency between the schema and all partitions of a single layer;
o a schema merge capability to allow multiple layers to be combined into single data sets for analysis;
o translation functions between different schemas and user views to allow inter-application sharing of data

This schema management requirement is independent of whether the GIS is based upon a RDB or an OODB. Currently, most non-geographic systems avoid this problem by insisting upon "all data in a single data base," an unacceptable approach for GIS's due to data volume and diversity.

Access Control Locking and Concurrency: Access control requirements in a GIS differ widely from those in conventional DBMS's. First, standard record, or table locking is nearly useless, since data is usually accessed based upon common location, not data type. Some form of "area locking" is more appropriate. This is especially true for topologically structured layers, where a few data types (face (polygon), edge (arc) and node (point)) are evenly distributed so that a single table lock could bring all but one user to a halt. This is even a greater problem in a RDB which locks tables after a threshold number of tuples have been locked. There are two basic solutions: partitions locking or object-oriented locking. In the first, a partition is locked completely while a user modifies it, resulting in a long transaction (hours or days). In the second, side effect locks can be controlled at the object class level. In a topological data base (Herring-87), standard locks could be used on feature data, and proximity locks on topology (not allowing two users to modify the same or adjacent faces simultaneously).

SUMMARY

The GIS implementations problems are much more complex than found in any non-spatial DBMS. Thus, not only should GIS research investigate the forefront of data base technology, it should also be driving RDB and OODB research to investigate specific geographically-related problems. Further, we must recognize the inevitability of diverse GIS data and GIS implementations and concentrate on a rational set of exchange standards capable of supporting diverse, real-time, distributed, geographic processing.

ACKNOWLEDGEMENT

The issues and solutions presented in this paper represent six years of continuous interaction, within Intergraph and the GIS community. I wish to thank everyone who has participated in any of these discussions, especially those directly involved with TIGRIS requirements analysis, design, and implementation.

REFERENCES

Andrews, Timothy, and Craig Harris; "Combining Language and Database Advances in an Object-Oriented Development Environment"; OOPSLA'87 Proceedings; ACM; October 4-7, 1987; pp 430-440.

Blaha, Michael R., William J. Premerlani, and James E. Rumbaugh; " Relational Database Design Using an Object Oriented Methodology"; Communications of the ACM; vol 31:no 4; ACM; April 1988; pp 414-427.

Herring, John R., "TIGRIS: Topologically Integrated Geographic Information System"; Proceedings of AutoCarto8, March 1987, Baltimore, Maryland, pp. 282-291.

Herring, John R., Robert C. Larsen, and Jagadisan Shivakumar; "Extensions of the SQL Query Language to Support Spatial Analysis in a Topological Data Base"; GIS/LIS'88 Proceedings; ACSM, ASP/RS, AAG, URISA; November 30 - December 2, 1988; pp 741- 750.

Rumbaugh, James; "Relations as Semantic Constructs in an Object-Oriented Language", OOPSLA'87 Proceedings; ACM; October 4-7, 1987; pp 466-481.

Stonebreaker, M. 1986, ed. The INGRES Papers: Anatomy of a Relational Database System, 1986, Addison-Wesley, Reading, Massachusetts.

Ullman, Jeffery D.; Principles of Database and Knowledge-Base Systems, Vol 1; Computer Science Press; Rockville, Maryland; 1988.