ACCESSING SPATIOTEMPORAL DATA IN A TEMPORAL GIS

Gail Langran
Department of Geography, DP-10
University of Washington
Seattle, WA 98195
bitnet: langran@uwavl

ABSTRACT

This paper evaluates ways to boost the performance of spatiotemporal data access in a temporal GIS via strategic partitioning and indexing. Using a specific conceptual model as an example, the discussion describes a taxonomy of access methods and explores the methodogical options available.

INTRODUCTION

Effective access methods for spatiotemporal data are vital to the development of effective temporal geographic information systems. Because the access scheme of an atemporal GIS has major performance impacts, we can expect the spatiotemporal corollary to be similarly influential upon temporal GIS performance.

Although many alternate methods of representing temporal geographic data surely exist, this discussion builds upon a specific conceptual model that is described elsewhere in some detail (Langran and Chrisman 1988), and which was first suggested by Chrisman (1983). This model, called a space-time composite, represents spatiotemporality by accumulating geometric change into one integrated topological description. Each successive change causes the changed objects to break from parent objects, creating new objects with histories distinct from those of their neighbors. In other words, the representation decomposes over time into the area's greatest common spatiotemporal units; each unit's history is described by a variable-length list of attribute sets bracketed by effective dates.

Figure 1 shows a space-time composite of temporal census-tract data. Polygons 1, 2, 3, and 4 are the greatest common spatiotemporal units of Census Tracts A, B, and C. Until 1980, only three polygons represented the three census tracts because Polygon 3, being part of Tract B, was incorporated into Polygon 2. Then in 1980, the area represented by Polygon 3 moved from Tract B to Tract C, creating a fourth polygon with a distinct history from its neighbors.

We can reconstruct any time slice from the space-time composite by referencing the attribute histories of its objects to find the attribute sets that were current on the requested date. This tack obviously works for small numbers of objects with shallow histories, but it is reasonable to ask whether the data processing burden is manageable given realistic geographic data volumes. Clearly, a complex query could swamp a system without an access scheme to boost performance.

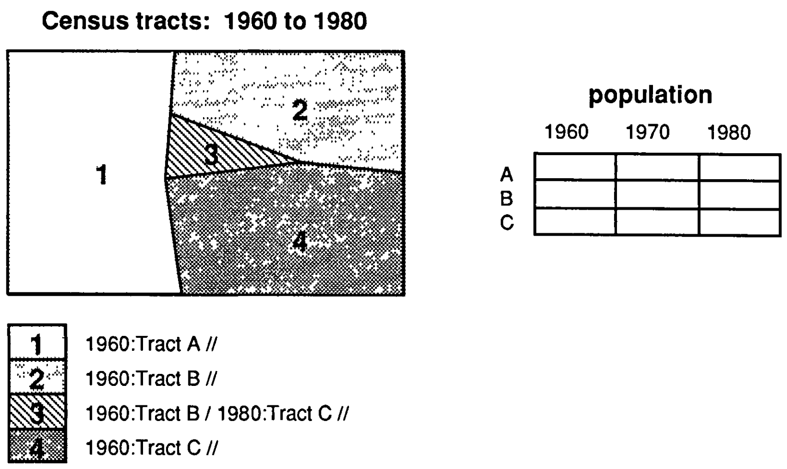

Figure 1. Temporal census-tract data represented as a space-time composite. Polygons 1, 2, 3, and 4 are the greatest common spatiotemporal units. Polygon 3 was part of Tract B until 1980, when it moved to Tract C. Census statistics such as population require no special treatment because time is symmetric with respect to other attributes.

QUERIES TO A TEMPORAL GIS

Table 1 lists potential queries to a temporal GIS. The listing distinguishes between forays into the past and future because the mechanisms for "examine" and "extrapolate" could differ depending on whether data are stored or computed.

Table 1. Temporal GIS queries.

Examine an object's history.
Extrapolate an object's future.
Examine a single time slice.
Examine an object's history; when the object meets some criteria, examine that time slice.
Extrapolate an object's future; when the object meets some criteria, examine that time slice.
Examine a single time slice; examine the histories of objects meeting some criteria.
Examine a single time slice; extrapolate the histories of objects meeting some criteria.
Examine the histories of all objects.
Extrapolate the futures of all objects.
Examine time slices, going backward through time.
Extrapolate time slices, going forward through time.

If we ignore extrapolated information and concentrate on accessing stored data, four primitive queries lie at the root of Table 1's eleven queries:

simple temporal query, i.e., what is the state of an object at time t?
temporal range query, i.e., what happens to an object over a given period?
simple spatiotemporal query, i.e., what is the state of a region at time t?
spatiotemporal range query, i.e., what happens to a region over a given period?

Access Mechanisms for Query Response

The space-time composite describes all temporality via time stamps in the attribute database, which permits us to treat time aspatially and space atemporally. Changes to geometric objects spawn new objects and break existing objects; these fragments replace the previous unbroken versions. By definition, each object in the space-time composite has a single geometric and topological description throughout time.

Thus, the temporal access mechanism for a space-time composite operates primarily on an attribute database, which is cross-referenced to the spatial representation. For the sake of simplicity, this discussion assumes that the attribute database is relational (here called an RDB) and that tuple versions correspond to object versions. Current RDBs can be fortified by a suite of indexing methods which, unfortunately, are almost solely one-dimensional (Freeston 1987). By stepping through the algorithms required to respond to the four primitive queries, we can comprehend how their requirements exceed standard RDB and GIS accessing capabilities.

Simple Temporal Query. The goal of the simple temporal query is to find an object version that was current on a specified date. Ideally, the system could search for the tuple whose "Object = ID" and "Time = T." But a temporal database is event-driven so a tuple will not necessarily have a time value to match every T. In essence, an object lifespan can be considered a chain whose nodes are the object's birth and death, and whose vertices are the points where the object changed. To time slice the object is to locate the value of a point along that chain based on the time stamp that equals *or immediately precedes* the requested time. This implies that ordering an object's versions within storage would be helpful.

Temporal Range Query. To respond to a temporal range query, the system must locate all versions of the desired object that were current during any part of the specified time span, i.e., where Object = ID, Time < Maximum Time, and Time > Minimum Time. The system could select all qualifying tuples; or if tuples are temporally ordered, the system could locate a tuple at one end of the desired time range then "walk" through time-sorted object versions until reaching the other end of the range.

Spatiotemporal Queries. The preceding two queries focus on single objects that meet temporal criteria. In contrast, spatiotemporal queries request all objects within specified spatial and temporal ranges. To respond to a simple spatiotemporal query, the system clips the desired region from the space-time composite, locates all attribute records for the desired region as of the desired time, then dissolves the chains that separate polygons of like attributes (i.e., recomposes the greatest common spatiotemporal units into greatest common spatial units). Alternately, the system can first access the required attribute records, then match them to the space-time composite. Responding to a spatiotemporal range query is the same as a simple spatiotemporal query except the system seeks attribute records falling within a space-time range.

The Search Space of the Four Queries

While the data space of these four queries is three-dimensional (two space and one time dimension), a closer look reveals that the queries define ranges of zero, one, two, and three dimensions, respectively. Query One defines a degenerate range, the point in data space that describes an object's state at a given time (for example, the location of a tuple that references the attributes current at that time). The data that satisfy Query Two lie along a vector that traces the changes undergone by a specific object over a specific period--a one-dimensional range. The response to Query Three occupies an orthogonal plane--a two-dimensional range. And the objectives of Query Four are located in a cube embedded within the GIS data space defined by a one-dimensional time range and a two-dimensional space range (Figure 2).

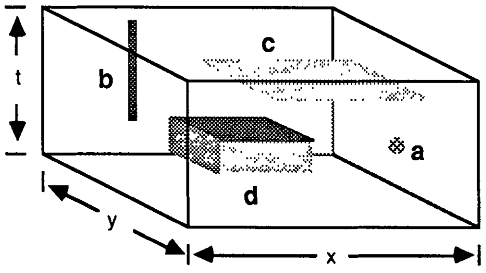

Figure 2. The search space of four primitive geographic queries in a three-dimensional data space. (a) A simple temporal query references a single point in space. (b) A temporal range query references a vector along which the desired data lie. (c) A simple spatiotemporal query references a plane within which the desired data lie. (d) A spatiotemporal range query references a cube within which the desired data lie.

RESPONDING TO TEMPORAL GIS QUERIES

Understanding the number of dimensions involved in a query is helpful in selecting the most effective means of response. Noronha (1988) and Chrisman (1988) provide recent discussions of multidimensional data access methods. Unfortunately, many methods that are termed "multi-" or "k-dimensional" can access only zero-dimensional ranges in a multidimensional data space. While such methods hardly seem worthy of their name to those accustomed to working with multidimensional ranges in multidimensional space, their intent is to facilitate "composite key" or "multikey" retrievals. Such retrievals seek a subset of records in k-dimensional space, one dimension per attribute. Table 2, drawing on Noronha's discussion, lists methods designed to access data in a multidimensional data space according to the number of ranges each can treat.

Given the accessing options available for zero-, one-, and higher-dimensional ranges, the next step is to examine the methods that treat k-dimensional ranges in k-dimensional space (Table 2's third column), since these are the methods that potentially provide a single means of rapid response for the four primitive queries of a temporal GIS.

Table 2. Accessing ranges of zero, one, and more dimensions in k-dimensional space.

Zero-dimensional	One-dimensional	K-dimensional
K-d tree[1]	Strip tree[10]	R-tree[11]
K-d-b tree[2]		R+ tree[12]
Multikey hashing[3]		Packed R-tree[13]
Extendible hashing[4]		Cell tree[14]
Point quadtree[5]		Grid file[15]
Multidimensional trie[6]		BANG file[16]
Multidimensional directory[7]		BSP tree[17]
Log log n structure[8]		Region quadtree[18]
Quintary tree[9]		EXCEL[19]
		Field tree[20]
		Quad-CIF tree[21]

---

[1] Bentley 1975
[2] Robinson 1981
[3] Rothnie & Lozano 1974
[4] Fagin et al. 1979
[5] Finkel & Bentley 1974
[6] Orenstein 1982
[7] Liou & Yao 1977
[8] Fries et al. 1987
[9] Lee and Wong 1980
[10] Ballard
[11] Guttman 1984
[12] Roussopoulos & Leifker 1985
[13] Faloutsos et al. 1987
[14] Gunther 1986
[15] Nievergelt et al. 1984
[16] Freeston 1987
[17] Samet 1984
[18] Fuchs et al. 1980
[19] Tamminen 1981
[20] Frank 1983
[21] Kedem 1982

A Taxonomy of Access Methods

A taxonomy of access methods would be useful to truly understand the options available. Several writers have attempted to classify data access schemes. Nievergelt et al. (1984) define two broad classes: a scheme can organize the data themselves or partition the embedding space. For example, the actual locations of objects in data space determine the branching of k-d trees and R-trees. Conversely, a grid file or quadtree subdivides when a predetermined sector of data space exceeds a predetermined maximum capacity. The Nievergelt framework is quite useful conceptually but enough hybrid schemes exist to make it something less than a taxonomy. Specifically, the quad-CIF tree associates the minimum bounding rectangles of its objects with cells of a recursively subdivided embedding space (Noronha 1988). Other schemes that resist the Nievergelt framework are the BANG file, cell tree, and extendible hashing.

Freeston (1987) defines a pragmatic classification of access schemes: tree structures, extendible hashing, and grid files. But where the Nievergelt framework is perhaps too conceptual, this classification is too technical. The strengths and weaknesses of quadtrees and grid files, and those of R-trees and BANG files, are more similar than those of quadtrees and R-trees, or grid files and BANG files. Yet Freeston's classification results in the latter two dissimilar groupings based on common data structuring mechanics.

Noronha, too, defines a classification scheme for access methods (1988). This scheme distinguishes hierarchical vs. nonhierarchical and regular vs. object-oriented subclasses to highlight the major performance differences among methods. However, Noronha's goal is description and exposition, and he purposely avoids the rigor of a taxonomy.

The ideal taxonomy should produce clear distinctions between classes, and members of a taxonomic class should share strengths and weaknesses. The terminology used here departs from earlier conflicting, and potentially confusing, usages. The "access methods" described here have been variously termed "partitioning" (by Noronha), "indexing" (by Chrisman) and "file structuring" (by the bulk of computer scientists). This taxonomy distinguishes indexing from partitioning because these two operations are associated with separable functions and ramifications. Membership in more than one indexing class is permitted, but the partitioning class is uniquely defined, since herein lies the greatest performance distinction.

Indexing. The two major indexing methods are search trees and hashing. A search tree stores physical locations of entities according to some order; hashing methods use functions to compute storage locations. Topological navigation is a third indexing method that is somewhat peculiar to geographic applications. If data records have an innate order and supply pointers to neighbors, accesses within neighborhoods can use these "topological" data as stepping-stones to navigate from from one object to another. The DIME file editor (White 1974), the ETAK automobile navigation system (White 1987), and the TIGRIS editor (Herring 1987) demonstrate topological navigation in a spatial system. Lum et al. (1984) demonstrate navigation in a temporal (and aspatial) RDB.

Partitioning. The most complex portion of the taxonomy describes partitioning. The wide array of partitioning strategies and problems are discussed at some length in both Chrisman 1988 and Noronha 1988. Partitions can be at one level or hierarchical. Each can use regular or irregular units. Irregular units can intersect or not within a level. Successive levels of a regular hierarchy might nest fully or not. Figure 3 depicts the relationship of these subclasses.

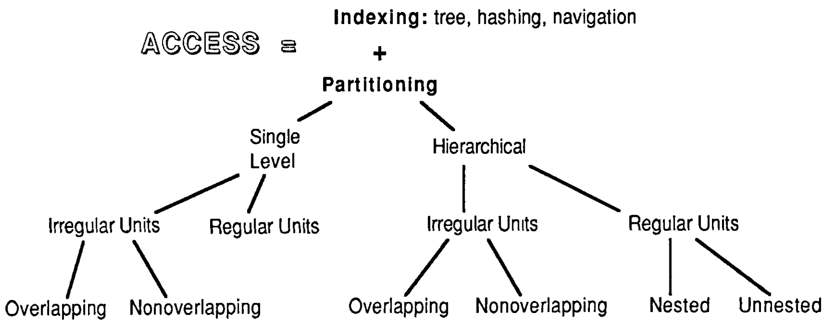

Figure 3. A taxonomy of multidimensional access schemes, which provides separate indicators for indexing and partitioning method.

Using the Taxonomy
The usefulness of the taxonomy is in evaluating a given access scheme for a given purpose. Since an access scheme inherits the strengths and weaknesses of its methodological components (as defined by the taxonomy), we can quickly judge a scheme by knowing its classification.

For example, irregular subdivisions permit us to cluster data as desired and thereby avoid splitting individual objects. Objects that are split by regular cells can be difficult to reconstitute and may produce erroneous replies to analytical queries concerning size or duration. Conversely, irregular cells require more complex heuristics to build and greater storage overhead to describe than do regular cells.

CONCLUSIONS

Four primitive queries whose ranges are zero-, one-, two-, and three-dimensional lie at the root of more sophisticated requests for temporal GIS information. A set of multidimensional data access schemes exist that are theoretically capable of boosting temporal GIS performance. These queries and access schemes are two endpoints from which to proceed to untangle the temporal GIS problem that lies between them.

This paper presents a taxonomy that groups access methods according to performance traits. Each subclass of the taxonomy is associated with a set of strengths and weaknesses, which are inherited by its members. The question, then, remains: what strengths would be most useful to a temporal GIS application, and what weaknesses would be intolerable? The cursory analysis described here indicates that using a topological navigator to supplement a tree or hashed index would assist in time-slicing temporal GIS data. How to partition spatiotemporal data is far more problematic and requires further examination.

ACKNOWLEDGMENTS

I would like to thank Intergraph Corporation's Advanced Projects Group for their generous support of this work. I would also like to thank Nick Chrisman for many enthusiastic conversations on this topic.

REFERENCES

Ballard, Dana H. (1981). "Strip Trees: A Hierarchical Representation for Curves." **Comm ACM** 24, May, 310-321.

Bentley, Jon Louis (1975). "Multidimensional Binary Search Trees Used for Associative Searching." **Comm ACM** 18, September, 509-517.

Burton, Warren (1977). "Representation of Many-Sided Polygonal Lines for Rapid Processing." **Comm ACM** 20, March, 166-171.

Chrisman, Nicholas R. (1988). "Spatial Indexing Schemes for GIS." Unpublished paper, Dept of Geography, Univ of Washington, October.

Fagin, R.; Nievergelt, J.; Pippenger, N.; and Strong, R. (1979). "Extendible Hashing: A Fast Access Method for Dynamic Files." **ACM Transactions on Database Systems** 4, 3, 315-344.

Faloutsos, C.; Sellis, T.; and Roussopoulos, N. (1987). "Analysis of Object-Oriented Spatial Access Methods." Proceedings of SIGMOD '87, 426-429.

Finkel, R. A. and Bentley, J. L. (1974). "Quad Trees: A Data Structure for Retrieval on Composite Keys." **Acta Informatica** 4, 1-9.

Frank, A. (1983). "Storage Methods for Space-Related Data: The Field Tree." Institut fur Geodasie und Photogrammetrie, ETH, Zurich, Nr 71.

Freeston, Michael (1987). "The BANG File: A New Kind of Grid File." Proceedings of SIGMOD '87, 260-269.

Fries, O.; Mehlhorn, K.; Naher, S.; and Tsakalidis, A. (1987). "A log log n Data Structure for Three-Sided Range Queries." **Inf Proc Ltr** 25, 269-273.

Fuchs, H.; Kedem, Z.; and Naylor, B. (1980). "On Visible Surface Generation by A Priori Tree Structures." **Computer Graphics** 14, 3.

Gunther, Oliver (1986). "The Cell Tree: An Index for Geometric Data." Electronic Research Lab, UCB/ERL M86/89. UC Berkeley, December.

Guttman, Antonin (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching." Proceedings of SIGMOD '84, 47-57.

Herring, John R. (1987). "TIGRIS: Topologically Integrated Geographic Information System." Proceedings of Auto-Carto 8, 282-291.

Kedem, G. (1982). "The Quad-CIF tree: A Data Structure for Hierarchical On-Line Algorithms." Proceedings, Design Automation Conference, 352-357.

Langran, Gail and Chrisman, Nicholas (1988). "A Framework for Spatiotemporal Information." **Cartographica** 25, 3.

Langran, Gail (1988). "Temporal GIS Design Tradeoffs" Proceedings of GIS/LIS '88 Volume 2, 890-899.

Liou, J. H. and Yao, S. B. (1977). "Multidimensional Clustering for Database Organizations." **Information Systems** 2, 4, 187-198.

Lee, D. T. and Wong, C. K. (1980). "Quintary Trees: A File Structure for Multidimensional Database Systems." **ACM Trans DB** 5, 3, 339-353.

Lum, V.; Dadum, P.; et al. (1984). "Designing DBMS Support for the Temporal Dimension." Proceedings of SIGMOD '84, 115-126.

Nievergelt, J.; Hinterberger, H.; and Sevcik, K. C. (1984). "The Grid File: An Adaptable, Symmetric Multikey File Structure." **ACM Trans DB** 9, 1.

Noronha, V. (1988). "A Survey of Hierarchical Partitioning Methods for Vector Images." Proceedings of the International Symposium on Spatial Data Handling, Sydney.

Orenstein, J. A. (1986). "Spatial Query Processing in an Object-Oriented Database System." Proceedings of SIGMOD '86.

Robinson, J. T. (1981). "The K-D-B Tree: A Search Structure for Large Multidimensional Dynamic Indexes." Proceedings of SIGMOD '81.

Rothnie, J. B. and Lozano, T. (1974). "Attribute-Based File Organization in a Paged Environment." **Comm ACM** 17, 2, 63-69.

Roussopoulos, N. and Liefker, D. (1985). "Direct Spatial Search on Pictorial Databases Using Packed R-Trees." Proceedings of SIGMOD '85, 17-31.

Samet, H. (1983). "The Quadtree and Related Hierarchical Data Structures." **ACM Computing Surveys** 16, 2, June.

Tamminen, M. (1981). "The EXCEL Method for Efficient Geometric Access to Data." Acta Polytech. Scandinavia. Mathematics and Computer Science Series, 34, Helsinki.

White, Marvin (1975). "Map Editing Using a Topological Access System." Proceedings of Auto-Carto 2, 422-429.

White, Marvin (1987). "Digital Map Requirements of Vehicle Navigation." Proceedings of Auto-Carto 8, 552-561.