Geol. Jb.	A 122	101—107	3 fig.	1 tab.	Hannover 1992

Indexing Methods for Area-of-Interest Retrieval from World-Scale GIS

CLIFFORD A. SHAFFER

Geographical Information System (GIS), geoid, spatial data, data handling

Abstract: This paper compares approaches to extracting a user's window of interest from a geographical database representing large portions of the earth's data. Such a database is characterized by two atypical features: its size (we assume in the range of a few hundred megabytes) and the effects of the earth's spherical shape. In such a system, data manipulation should be viewed as two separate phases: (1) extraction of the user's area of interest as specified by a rectangular window, and (2) further processing of the data within this window. If we wish to use one of the many popular hierarchical spatial indexing schemes, we must first convert the data from the earth's spherical reference system to a plane. We evaluate such conversions in terms of their ability to efficiently extract the data from secondary storage, such as a hard disk or CD-ROM. The primary contribution of this paper is to present an objective means for comparison of world-scale indexing methods.

[Indizierungsmethoden für die Auswahl eines gewünschten Datenbereiches aus dem globalen GIS]

Kurzfassung: In dieser Abhandlung werden Annäherungen an die Auswahl eines gewünschten Benutzerfensters aus einer geographischen Datenbasis, die eine große Anzahl von Erddaten enthält, verglichen. Eine solche Datenbasis ist durch zwei atypische Merkmale gekennzeichnet: ihren Umfang (vermutlich im Bereich von ein paar hundert Megabytes) und die Auswirkungen der Kugelform der Erde. In einem solchen System sollte die Manipulation von Daten als zweiphasig betrachtet werden: (1) die Auswahl des den Benutzer interessierenden Bereichs in ein rechteckiges Fenster, und (2) die weitere Bearbeitung der Daten innerhalb dieses Fensters. Wenn wir eines der vielen gebräuchlichen hierarchischen, räumlichen Indizierungsschemata benutzen wollen, müssen wir zunächst die Daten aus dem Referenzsystem für die kugelige Form der Erde für eine Ebene umrechnen. Diese Umrechnungen werden in Hinblick auf ihre Fähigkeit, Daten effizient aus zweiter Speicherung herauszuziehen (z.B. Festplatte oder CD-ROM), ausgewertet. Als Hauptziel dieser Arbeit soll ein objektiver Weg für den Vergleich von globalen Indizierungsmethoden aufgezeigt werden.

Contents

Author's address: Dr. C. A. SHAFFER, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 24061 USA.

1 Introduction

A world-scale GIS requires careful design, due to the massive amounts of data to be organized. To illustrate the potential hazards, consider that a database stored on a CD-ROM would require at least 72 minutes to extract data if its organization required that the entire disk be read, since this is the amount of time required by the read head to scan the entire disk. Fortunately, at any given instant the user will want to process only a fraction of the total database. For example, the user may wish to view data only at a particular scale, or only a subset of the feature types available. Additional complications result if we attempt to represent large portions of the earth in the database (since the earth is spherical) when most known spatial indexing schemes are inherently planer.

Manipulating an entire database of the target size is much too expensive. For example, if the user wants to compute a landuse and contour overlay for a small area of interest (AOI) within the database, the data cannot be extracted by examining the entire database for the desired AOI and then processing. Instead, the manipulation of data in world-scale GIS should be viewed as two separate phases: (1) extraction of the user's AOI as specified by a rectangular window, and (2) further processing of the data within this window. In our example, the user would extract the contour map for an AOI, extract the landuse map for the same AOI, and perform the polygon overlay operation on the extracted portions. Optimally, an AOI will be used many times before being abandoned by the user. This is not to say that such an extraction process cannot be hidden from the user. What is important is that processing be performed only on the data within the user's AOI. In fact, we argue that efficiency of AOI retrieval is **the** dominant issue in selecting a world-scale indexing method.

Given that we wish to retrieve an AOI from a large database stored on disk or CD-ROM, we must organize the data so that windows of interest are covered by as few extents as possible on the disk. This will minimize search time (particularly odious on a CD-ROM). When evaluating indexing methods for such databases, we must consider the ability of the indexing method to extract areas of interest and the effects of conversion of spherical data from the earth's surface to the indexing method and associated effects on query windows.

2 Indexing Methods

Two distinct approaches to indexing spherical data have been suggested by the research community. The first converts from spherical to planer data by means of some projection technique. We then store the projected data using a planar indexing scheme. Note that we will have all of the problems of traditional paper map cartography in terms of distortion. Many projections will also distort the shape and size of the AOI query rectangle. Once an AOI is retrieved from secondary storage, any desired operations can be performed, including reprojection.

Another approach is to represent the earth's surface by means of successive refinement of a regular polyhedron inscribed within the earth's sphere, with the earth's surface projected onto the faces of the refined polyhedron. There are only five regular polyhedra, three of which have equilateral triangles for their faces. In particular, the cube (CHAN & O'NEILL 1975) might seem like a promising approach since it appears to give «square» faces, ideal for a hierarchical indexing scheme applied to each face. However, many problems arise. Of greatest significance, the corners of each cube face meet with two other faces, with the angles at the meeting points being 120 degrees in the projection. The geometry at the corners is different than anywhere else on the cube; at all other points four square pieces meet at their corners. The corner cells of each face still have four side neighbors (two in the same face, one in each of the other faces), but have no diagonal neighbor in the direction of the neighboring faces. Because of these singularities in the geometry, the angles of the corners making up quadrants on any given face change in the projection depending on their position — from 90 degrees in the center of the cube's faces to 120 degrees at the corners. Thus, the areas of the quadrants are different. In short, the projection of the resulting grid cells are not regular.

Each of the three regular solids constructed from triangles are composed of equilateral triangles. Equilateral triangles can be broken down to four smaller equilateral triangles by joining the midpoints of the edges to form a triangle. Hierarchical indexes based on triangles are well known. Polyhedral approaches based on octahedrons have been suggested by DUTTON (1984). But all of the problems associated with the cube projection are also associated with the triangular projection.

While polyhedron-based methods certainly appear be a viable approach, our main interest in the rest of this paper is to study projection effects on area-of-interest retrieval. Some of these issues are also relevant to polyhedon-based indexing.

Many approaches have been studied for the indexing of geographic data stored in the 2D-plane. Currently, the methods considered most efficient are based on some form of tree indexing. Such methods include MORTON code-based schemes, such as the linear quadtree (GARGANTINI 1982; SHAFFER et al. 1990), the grid file (NEIVERGELT et al. 1984), hB-trees (LOMET & SALZBURG 1990), and the Fieldtree (FRANK & BARRERA 1989).

As an example of one popular approach to spatial indexing in the plane (though certainly not the only approach), we describe MORTON coding (MORTON 1966). Given a 2^n x 2^n square grid, a MORTON code addressing scheme can be derived by simply interleaving the bits from the binary representation of the x and y coordinates for each grid cell. Note that thinking of indexing in terms of a grid in no way limits the database to a region (raster) representation. The grid cells can be viewed either as the minimum resolution for vertices, or the grid cells can be viewed as bins for indexing data.

Given a MORTON code addressing scheme, the data in each cell can then be laid out on the disk by MORTON code order. This has several advantages. First, given a square window of a certain size that is aligned to the grid axes, and grid cells the same size as the window, the window can fall on four grid cells at most. If the window is not aligned to the grid axes, it can fall on at most six grid cells. If all data in a grid cell (at any resolution) is placed as a sequential block on the disk, at most four (or six) sections of the disk need be searched. Since not all data in these four (or six) grid cells fall within the window, there is a certain amount of wasted data being read. This can be reduced using various schemes to match the actual MORTON code cells of various sizes to the window (ABEL & SMITH 1984). Furthermore, a MORTON code addressing scheme can be extended to support multiple resolutions for data generalization by storing data with the internal nodes of the corresponding quadtree.

3 A Comparison of Projection Methods

Previous work has discussed projection methods for world-scale geographic databases indexed by hierarchical data structures. In particular, MARK & LAUZON (1985) suggest using the UTM coordinate system as the basis of an indexing method. However, this is not a regular decomposition of the earth's surface, resulting in a disjoint collection of hierarchies. TOBLER & CHEN (1986) provide a good study of many of the issues discussed in this paper; those wishing further background should consult this reference. In particular, TOBLER & CHEN consider several projections from the sphere onto the plane for use as the basis of a regular hierarchical decomposition. Their conclusions include the desirability for each node at a particular level of the hierarchy to represent the same surface area on the sphere. However, no compelling reasons are given for this conclusion and, in particular, their comparison criteria for various projections do not consider AOI retrieval.

Issues involved in picking a projection include the cost of the projection calculation, the distribution of the data on the plane, and the shape of the projected query window. When discussing AOI query windows, we assume that the original window is rectangular on the sphere's surface (i.e., before projection). After projection, this window may become distorted in both size and shape, with the amount of distortion dependent on the position in the projected plane. Thus, two query windows of identical size and shape on the sphere may not be the same size or shape under the projection.

The most straightforward projection is to use a rectangular grid to divide the world by longitude on the x-axis and latitude on the y-axis, as shown in Figure 1. This is known as an equirectangular projection. One advantage of the equirectangular projection is that coordinates described by longitude and latitude require no translation to calculate their projection onto the plane. This will make geographic operations more efficient, and simplify the task of writing software to convert to other projections. The earth is 360 degrees around and 180 degrees high giving an aspect ratio of 2 : 1, but this is easily solved by making the first level of a four-way branching hierarchy contain two empty nodes (TOBLER & CHEN 1986).

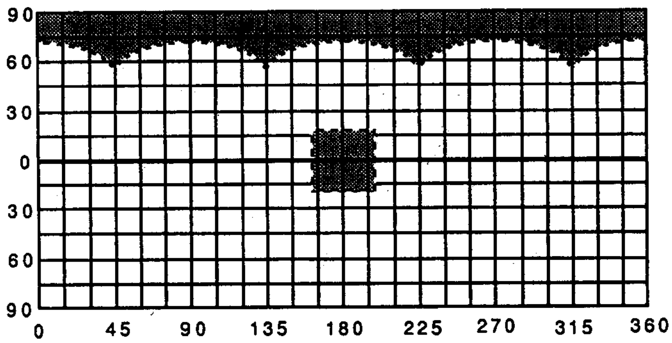

Fig. 1: Graticule for the equirectangular projection. Shaded areas illustrate equatorial and polar query windows.

The grid cells at any given level of resolution will represent different areas and shapes under the equirectangular projection, depending on the corresponding position on the sphere. In particular, grid cells near the poles represent much less spherical surface area. One result may be clustering of the data as seen on the plane, resulting in degraded performance by the indexing mechanism. In practice, this should not occur for two reasons. First, geographic data are not uniformly distributed, so further clustering due to projection is not likely to be noticeable. Second, good indexing techniques can handle clustered data without loss of performance. Of greater concern may be the effect of the projection on AOI query windows. With equirectangular projection, query windows of a particular size on the sphere will become wider towards the poles, while being more rectangular at the equator. As the window approaches the poles, it covers more and more grid cells. A window that includes the North (or South) pole covers all grid cells at the top (bottom) of the grid. That may correspond to a large number of disjoint blocks on the disk, and thus may require more search time. On the other hand, polar grid cells probably contain less data than equatorial grid cells, since these cells have a small surface area. Another problem is that the query window is no longer rectangular, as shown in Figure 1.

An equal-area projection can easily be obtained by simply modifying the equirectangular projection to use authalic coordinates, where the projected latitude is the sine of the true latitude. The effect will be to make sections at the poles correspond to shorter cells at the top and bottom of the array, as shown in Figure 2. Most important, this modification will make all query windows of a given size on the sphere have the same area (i.e., cover the same number of grid cells) under projection. However, the shape of a query window will still be distorted depending on position, as shown in Figure 2.

We can eliminate the problem of distorted query windows by using a conformal projection, such as the Azimuthal Stereographic Projection, shown in Figure 3. In this example, two projections would be used, one for the northern hemisphere, and one for the southern hemisphere. Such a projection ensures that all fixed-size rectangular query windows on the sphere will be rectangular in the plane.

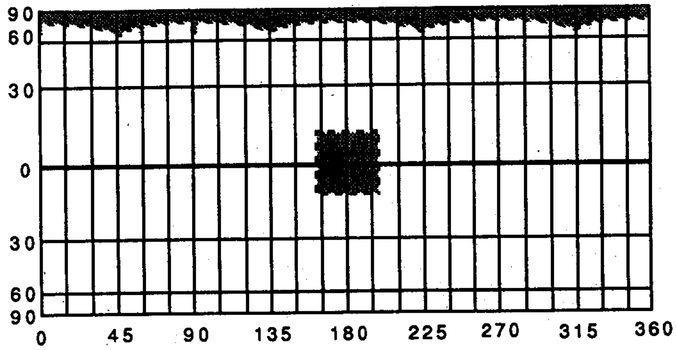

Fig. 2: Graticule for the equirectangular projection using authalic coordinates for latitude. Shaded areas illustrate equatorial and polar query windows.

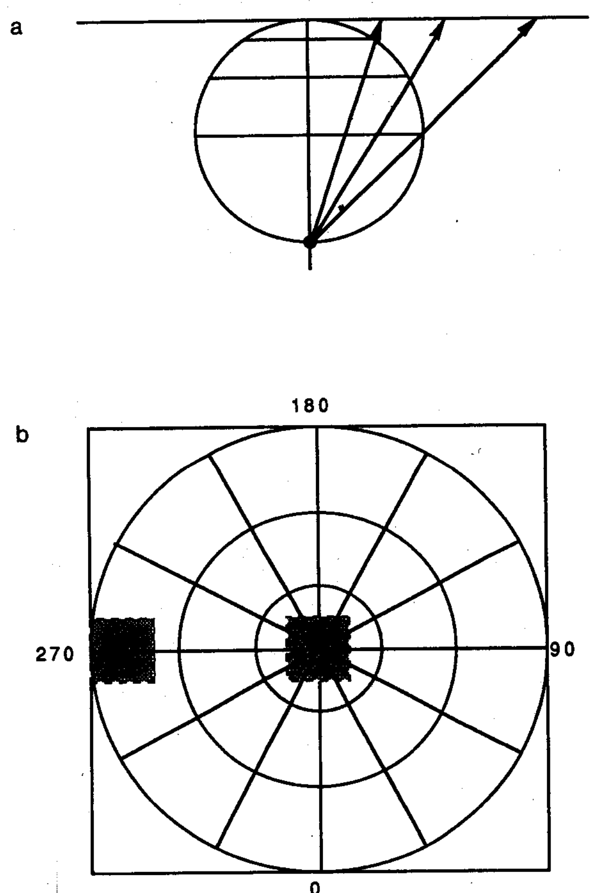

Fig. 3: Azimuthal stereographic projection for one hemisphere. (a) The projection is formed by passing a line from the pole, through the data point, to the plane. (b) The resulting graticule in the indexing plane. Shaded areas illustrate equatorial and polar query windows.

To test for the effects of size distortions on query windows, we conducted the following experiment. First, a large number of points (about 400,000) were randomly generated, with a uniform distribution on a spherical surface. These points were then projected onto a plane of resolution 16,384 x 16,384 cells using the equirectangular projection. The points were then stored using a hierarchical indexing method, specifically, the QUILT GIS system (SHAFFER et al. 1990). This system uses a linear quadtree for its index, with a B-tree to organize retrieval from disk. A series of pairs of query windows were then produced, and their time for retrieval was compared. In each case, a window centered on the equator was compared with a window centered on the north pole. The north pole window stretched (in projected space) all across the range of x-coordinates. The height of the polar query window was selected so that the number of points retrieved was a close as possible to the number of points retrieved from the equatorial query window. No shape distortions were considered (which would increase the programming complexity and possibly the time for polar queries). Naturally, the area (in projected space) of the polar query was considerably larger than that of the equatorial query. How much greater depends on the size of the windows — smaller windows have a greater proportional distortion.

Table 1 shows the results of this experiment. The experiment was run on a Macintosh II computer with A/UX (a UNIX derivative). All times are in seconds, obtained by adding «u» time and «s» time as measured by the standard UNIX timing functions. As can be seen, there was no significant difference in the time required to fetch a particular number of points, regardless of the size (and corresponding density) of the query window. In other words, the hierarchical indexing method has adequately compensated for the density of the data.

Table 1: Query times for locating points in query rectangles

Equatorial	Equator query		Polar query			Size
rectangle size	points	time	points	time	rows	differential
256 x 256	669	1.0	671	1.3	45	11.2
1024 x 1024	5,661	7.4	5,666	7.8	370	5.8
2048 x 2048	16,100	20.9	16,095	21.5	1041	4.1

4 Conclusions

On the basis of the results of our experiment, we conclude that size distortions in the query window due to projection are not significant. In particular, use of an equal-area projection for making all query windows of the same size will not affect retrieval time. Shape distortions in the query window may be significant since they may result in data being read from disk that do not actually fall within the window. A more complicated range query algorithm will be required to minimize the number of such points retrieved. If all such false retrievals can be avoided, then shape distortions will not be significant for AOI retrieval. Experiments to test this hypothesis are more difficult to arrange, since they rely on the quality of the search algorithm as well as the indexing technique. Such experiments are planned for future work.

Our work provides an objective criterion for comparison of world-scale indexing methods — namely, the ability to do AOI retrieval of various sizes at arbitrary positions in the database. This is particularly helpful to implementors, since they can focus on those issues that most affect database performance.

Acknowledgement

This work was partially supported by General Dynamics Electronics Division and the Virginia Center for Innovative Technology.

5 References

ABEL, D.J. & SMITH, J.L. (1984): A Data Structure and Query Algorithm for a Database of Areal Entities. — The Australian Computer J., **16**: 147—153; Kingston, Australia.

CHAN, F. & O'NEILL, E. (1975): Feasibility Study of a Quadrilateralized Spherical Cube Earth Data Base. — EPRF, TR 2-75; Silver Spring, MD. (Computer Sciences Corporation).

DUTTON, G. H. (1984): Geographic Modelling of Planetary Relief. — Cartographica, **21**: 188—207; New York.

FRANK, A.U. & BARRERA, R. (1989): The Fieldtree: A Data Structure for Geographic Information Systems. — In: BUCHMANN, A. & GÜNTHER, O. & SMITH, T. R. & WANG, Y.-F. (Eds.): Symp. on the Design and Implementation of Large Spatial Databases, Lecture notes in Computer Science, **409**: 29—44; Berlin (Springer).

GARGANTINI, I. (1982): An Effective Way to Represent Quadtrees. - Communications of the ACM, **25**: 905—910; New York.

LOMET, D.B. & SALZBERG, B. (1990): The hB-tree: A Robust Multiattribute Indexing Method with good guaranteed performance. — ACM Transactions on Database Systems, **15**, 4: 625—658.

MARK, D.M. & LAUZON, J.P. (1985): Approaches for Quadtree-based Geographic Information Systems at Continental or Global Scales. — Proc. Auto-Carto 7, Washington: 355—364; Washington, D.C.

MORTON, G.M. (1966): A Computer-Oriented Geodetic Data Base and a New Technique in File Sequencing. — Internal memorandum, March, IBM Canada.

NEIVERGELT, J. & HINTERBERGER, H. & SEVCIK, K.C. (1984): The Grid File: An Adaptable, Symmetric Multikey File Structure. — ACM Transactions on Database Systems, **9**: 38—71; New York.

SHAFFER, C.A. & SAMET, H. & NELSON, R.C. (1990): QUILT: A Geographic Information System Based on Quadtrees. — The Internat. J. of Geogr. Inform. Systems, **4**: 103—131; London.

TOBLER, W. & CHEN, Z.T. (1986): A quadtree for global information storage. — Geographical Analysis, **18**: 360—371; Columbus, OH.