Storing And Using Multi-Scale Topological Data Efficiently In A Client-Server DBMS Environment

MAARTEN VERMEIJ, PETER VAN OOSTEROM, WILKO QUAK AND THEO TIJSSEN.

Faculty of Civil Engineering and Geosciences, Department of Geodesy, section GIS Technology,
P.O. Box 5030, 2600 GA Delft, The Netherlands.
Tel +31 15 2783756;
Fax +31 15 2782745;
Email: quak@geo.tudelft.nl

Biography

Maarten Vermeij is a master's student at the Delft University doing his final project on the storage and retrieval of scale-less maps. Professor van Oosterom is the leader of the section GISt, that carries out research and provides services in the field of technological aspects of geographic information systems.

(Note: The procedure presented here is still under development and has not yet been fully implemented and tested. Implementation and testing should be finished before the full paper is due.)

Introduction

When working with a large-scale dataset interactively, a user often zooms out to get an overview of the map. The result of this is often that all map details of that big part of the data are retrieved from the database and drawn on the screen. In many cases this is not wanted; the user is only interested in an overview map, and too much data is transferred from the client to the server. This will especially be a problem if the client is connected to the server over a small bandwidth connection. The solution to this problem is to perform generalization at the server side to reduce the amount of data that has to be sent to the client. However generalization is a computational intensive task, which would put a great strain on the server if generalization needs to be calculated separately for every request. It is necessary to be able to store generalization information in such a way that generalized datasets can easily be retrieved at a range of levels of detail (l.o.d.'s). The procedure uses a data structure to efficiently store geometric and thematic information with additional information for the creation of generalized data sets. The use of this data structure requires some dedicated procedures at both server- and client side. Both are discussed in this article. Also the server-side routines for the creation of the generalization data are discussed. The procedure is tested with a large-scale dataset. The article ends with conclusions on the presented procedure.
In this paper, a procedure has been developed that allows the on the fly creation of a generalized map.

Data structure

The procedure uses a topological description of a planar partition using faces and their bounding edges. However, the geometric shape of the faces is not stored explicitly, but is to be reconstructed out of the appropriate edges at the client side. Generalization is achieved by omitting certain edges at smaller scale, or zoomed out, maps. Since edges represent the boundaries of faces, a reduction in the number of edges will result in larger faces. To enable the appropriate edges to be selected for map visualization at a certain level of detail, each edge record will contain, besides it's 2D geometric representation, a scale factor. An appropriate choice for this scale factor is the reciprocal value of the scale at which the edge should be removed from the map. E.g. if an edge should be visible in maps with a scale of 1:20,000 and larger, the scale factor would be 20,000. The term scale in on screen cartography does not correspond directly with the term scale as used on paper maps, but it nevertheless can give a useful indication for the amount of generalization required.

If a generalized map view is requested, the edges with a scale factor that is equal to, or larger then the requested scale factor, are to be retrieved from the database. These edges will enable the client to provide a visual partition at the required scale, however this does not contain any thematic information. The thematic information is stored in a separate table that contains face descriptions. As with the edges these faces are only visible on certain l.o.d.'s. This is accomplished by adding to each face a scale range, which indicates at which l.o.d's. or scales, a face should be present in the map display. In contrast to the edges, faces have an upper and a lower scale limitation, which allows the selection of only those faces that are present at a certain scale since large faces are replaced by a number of smaller ones at larger scales. To identify which polygon, created at client side, corresponds to which face, each face record contains a centroid, which is a point that lies within the geometric boundary of the face. Figure 1 shows the composition of a generalized map using the appropriate edges and centroids.

Map with generalization information(Original level of detail, 1:10,000)

Generalized map for scale 1:45,000

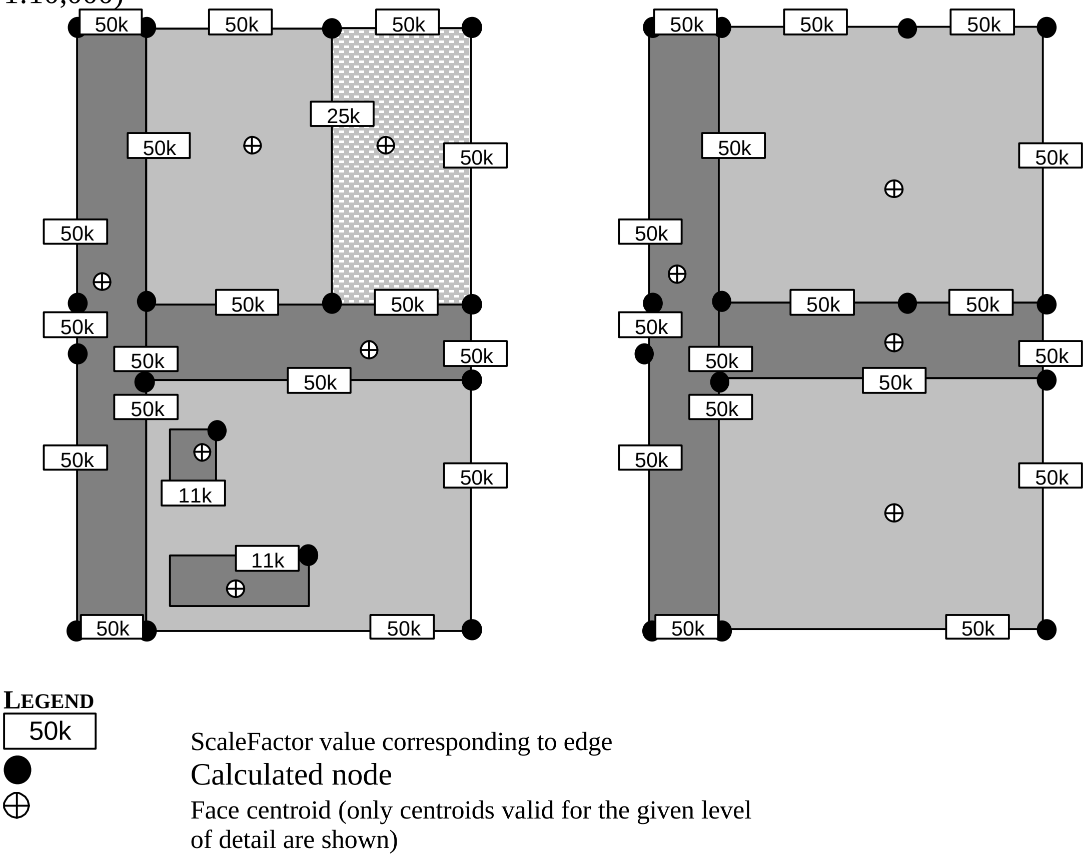

Figure 1: Example of generalization using the ScaleFactor value stored in the edges.

Representation within a database

The edges and faces of the data structure are stored using a spatial DBMS. In the research an Oracle 9i Spatial database is used, but others should also be possible. The Oracle DBMS allows the use of indices on more then 2 dimensions. This feature can be exploited by using the above-described scale factor as the third dimension. The consequence of this choice, is that the request for a generalized map can be transformed into a 3D spatial query. A generalized map is then created by requesting all edges that lie within, or intersect a 3D box , with the x,y extent equal to the query window, which is the area to be represented at the client, and the z extent from the maximum scale level down to the requested scale level. At the edges of the query window a problem arises, in the form that at those places not all lines enclosing a polygon are returned. This poses a problem since it is then not possible to ensure that the polygons that are created contain the centroid of the face it represents and thus can not be assigned a plane. To overcome this problem, first the face table needs to be queried.

The selection of the faces works somewhat different. All faces are selected of which the 3D bounding box intersects a face, with x-,y extent equal to the query window and a z-value equal to the scale level. All the edges that are necessary to completely reconstruct

these faces are present within the union of the bounding boxes of all these faces. Since this union might have a very irregular shape, which would have a great impact on the actual spatial query, it might by useful to use the bounding box of this union as a new query window. The query for all necessary edges would proceed as described above, but using the extended bounding box.

The following database tables are used to store the Scale-less map:

TABLE: EDGES

Column	**Type/description**
Oid	Integer, to uniquely identify an edge
Geometry	2D Geometric, description of an edge
Bbox	Flat 3DBounding Box, constructed using the 2D bounding box of an edge and the Scale Factor as the z-coordinate.
ScaleFactor	Float, the reciprocal value of the largest scale at which an edge is to be displayed.

TABLE: FACES

Column	**Type/description**
Oid	Integer, to uniquely identify a face.
Bbox	3DBounding Box, constructed using the 2D bounding box of a face and the ScaleRange for the third dimension.
Centroid	2D Point, that lies within the exact shape of the face.
ThematicInfo	Anything that is to be stored about the face.
ScaleRange	Array of Float[2]. Upper and lower Scale Factor to indicate scales at which the face is visible.

Creation of generalization information

The procedure was developed to allow the on-the-fly creation of generalized maps at a range of scales. In this situation it is important that computational intensive procedures are not necessary at query time, at least not at the server side. It is therefore important that the generalization information is calculated and stored before hand. The data structure described above supports this. What remains is the creation of this generalization data. The procedure used in the research follows closely the procedures used in the GAP-tree method (Oosterom, 1993), although it is expected that the data structure and query-time part of procedure, should be able to handle other generalization methods as well. The generalization method used, starts with selecting the least important face within a map. The selection of this face is based upon an importance value, calculated for example by: Importance=priority*Area; whereby priority is a value related to the type of face, e.g. road, building or water. The least important face is aggregated

with one of it neighbors. The appropriate neighbor is selected using the following formula as a criterium:
Collapse(a,b) = compatibility(a,b) * LengthCommonBoundary(a,b). The neighbor with the highest value for this function is selected for aggregation. The aggregation it self is performed by deleting the common boundary of the two neighboring faces. This boundary is represented by edge in the database. Although the boundary is deleted from the map it is not removed from the database. Instead the ScaleFactor column in the edge's record is set to a value relevant to the generalization level at the moment of it's removal from the map. Furthermore the upper limit of the ScaleRange of the aggregating faces is set to that value. Also a new face is created with the lower ScaleRange value set to the actual ScaleFactor and a valid value for the centroid. At this moment it is possible to supply the new face record with relevant thematic information, which could for example be based upon information of the two aggregated faces.

Testing

The entire procedure, including creation of generalization information client- and server side processing is tested using the scale 1:10000 large scale vector maps (TDN, 1998) from the Dutch Topographic Service (TDN) in the form of the TOP10Vector data. Testing will comprise timings, data amounts, both at server- and client side and during transfer, as well as other relevant factors.

References

TDN (1998) *Productbeschrijving TOP10vector*, Topografische Dienst Nederland, September, 1998

Peter van Oosterom (1993) *The GAP-tree, an approach to "On-the-Fly" Map Generalization of an Area Partitioning*,

P.van Oosterom (1990) *Reactive Data Structures for Geographic Information Systems*, TNO Physics and Electronics Laboratory, Leiden

Putten, J. van and Oosterom, P. van (1998) *New results with Generalized Area Partitionings*SDH'98, Vancouver Canada