Comparing DBMS-based approaches for representing 3D building objects

Hyeyoung Kim, Geunhan Kim and Chulmin Jun
Dept. of Geoinformatics, University of Seoul,
Jeonnong-Dong, Dongdaemu-Gu, Seoul 130-743, Korea
{mhw3n, nani0809, cmjun}@uos.ac.kr
Phone: +82-2-2210-2643 FAX: +82-2-2246-0186

Preferred track: URBAN 2009
Preferred topic: Algorithms for urban area applications

There is a growing interest in representing geographic objects in 3D. Modeling and visualizing terrains in 3D has been well known technique now and most commercial GIS packages accommodate tools to represent terrain data types (i.e. TINs). However, modeling techniques for buildings in 3D has less been established and are still being studied theoretically without explicit implementations. Current building modeling techniques can be categorized into two—CAD and GIS. CAD systems have diverse data types such as cylinders, cones and freeform shapes and have been extensively used to model complex shapes in architecture or mechanics fields. CAD-related software packages also provide abundant functionality for visualization. The other option is to use GIS systems. GIS is mainly designed to represent geographical features and use less number of data primitives than CAD, which are points, lines and polygons. 3D representation in current GIS is confined to 2.5D, which means that one location can have only one z value. 3D buildings in GIS are actually 2.5D since they are constructed by extruding 2D polygons as much as building heights. Although either approach has been used for years and suffices visualization purposes at certain application domains, still some issues remain to be resolved as follows:

- Neither CAD nor GIS for 3D objects supports topological structure. Topology is the key property implemented in 2D GIS such as adjacency and connectivity that enables diverse analyses. We also need such property in 3D buildings in order to define relationships between elements in a building (for example, what are the adjacent walls of this wall? or, what are the walls that share this edge?).
- In urban modeling, we frequently deal with large areas and, thus, we need to represent a larger number of buildings than a few. Most CAD systems use file-based formats, which are unfavorable for storing and visualizing many building objects due to the computational performances. In contrast, although GIS packages support both file and DBMS formats, they are mostly software-dependent and do not support 3D topology as of now.

Therefore, a solution to these problems would be using DBMS, which are widely accepted as reliable method for managing large amount of data. Current mainstream DBMS such as Oracle and PostgreSQL have spatial extensions such as Oracle Spatial and PostGIS that support 3D primitives. Although they allow us to store 3D data (i.e. z-values), they do not support 3D

topology and what we get from analyses on 3D objects in these DBMS are 2D. Thus, we have to partially rely on traditional relational schemas to represent topological relationships while storing 3D geometries in Spatial DBMS. The possibilities of using DBMS for 3D objects have been recently investigated by those researchers including Zlatanova, Van Oosterom, Stoter, Arens, Rahman, Tet-Khuan and Ellul. With some variations, they mostly suggested hierarchical relationships between 3D element classes, which are based on SOLID-FACE-EDGE-NODE in order to represent 3D topology. However, as of writing this paper, we have not found any literature that deals with large amounts of 3D objects. The major motivation of this paper was testing the applicability of DBMS approaches whether they can accommodate large amount of 3D data while supporting 3D topology. We first reorganized 3D topology data models suggested by others including our own model and they are as follows:

1. SOLID – FACE – EDGE – NODE
2. SOLID – FACE – EDGE
3. SOLID – FACE – NODE
4. SOLID – FACE

In each model here, the last class contains geometries and the rest are non spatial relations. For example, in the first model in the above list, the NODE class stores actual point data and each of SOLID, FACE and EDGE classes corresponds to a non-spatial table. The PostgreSQL/PostGIS was used for the test. After building schemas for each of the models, 3D buildings were created and stored. Then, a series of performance tests were carried out for each of the models to test differences in retrieval times by different range queries. The test is summarized as follows:

- Increasing numbers of artificial cubes were created and stored in DBMS (1,000~100,000 cubes by 1000 increment)
- Two different queries were defined and used in the test; one for retrieving non spatial solid-ids and the other for geometries.

The result showed that the models 1 and 2 are the fastest models almost equally for retrieving geometries followed by 3, then 4. For retrieving non-spatial solid-ids, the model 2 was the fastest in performance followed by 4, 1 and 3 in order. Using the queried geometries, we also carried out visualization tests using VRML and OpenGL.