Chapter 16
3D Solids and Their Management In DBMS

Chen Tet Khuan, Alias Abdul-Rahman, and Sisi Zlatanova

Abstract

3D spatial modeling is one of the most important issues in 3D GIS research. It involves the definition of spatial objects, data models, and attributes for visualization, interoperability and standards. Real world complexity leads to different modeling approaches, as seen in different GIS applications. This paper provides some review of the problems, challenges and issues pertaining to the 3D GIS problems, especially in the handling and managing of 3D solids in DBMS. The paper also describes 3D spatial operators in DBMS and presents results using a simulation dataset. At the end of the paper, we provide and highlight requirements and recommendations for future research.

16.1 Introduction

'True' 3D GIS require extensive effort, as revealed from the recent research output and workshop (see Abdul-Rahman, et al. 2006). It is interesting to note that work on fundamental aspects, like 3D spatial analysis, has not been addressed to the level where an operational 3D system could be realized. The aim of this paper is to review recent research on 3D spatial data modeling and describe our recent work on the management of 3D solids in geo DBMS. Recent research development shows that 3D spatial modeling is becoming very important for many advanced GIS applications and the scenario is being enhanced by the advancement of computer graphics (hardware

Department of Geoinformatics, Faculty of Geoinformation Science and Engineering, Universiti Teknologi Malaysia, Skudai, Malaysia
Delft University of Technology, OTB, section GIS Technology,
Jaffalaan 9, 2628 BX the Netherlands
kenchen, alias@fksg.utm.my, s.zlatanova@tudelft.nl

and software), visualization, etc. and also influenced by developments in the OpenGIS consortium. 3D visualization environments such as Google Earth or 3D navigation software have already made some contribution and enabled more and more users to utilize visualization technology. Until recently, only specialized applications were able to manage and analyze 3D spatial data. The third dimension was used primarily for visualization and navigation. However, users are looking for applications that have one or more 3D GIS functionality. Due to the complexity of real-world spatial objects, various types of representations (e.g. vector, raster, constructive solid geometry, etc.) and spatial data models (topology, and geometry) have been investigated and developed, including e.g. Pilouk, 1996; Zlatanova, 2000; and Kada et al, 2006.

A universal and practical spatial data model that is capable of addressing more than one application is not available. This is due to the complexity of real world objects and situations. On the other hand, different disciplines emphasize different aspects of information e.g. including different requirements and output. Thus, a data model could be considered appropriate for a certain application but not so appropriate for other tasks. Different aspects and characteristics of real objects have led to the existence of several variations in object definition. The solution for these problems has directly referred to GIS standardization.

Current 3D GIS offer 2D functionality with 3D visualization and navigation capability. However, promising developments were observed in the DBMS domain where more spatial data types, functions and indexing mechanism were supported. In this respect, DBMS are expected to become a critical component in developing of an operational 3D GIS. However, extensive research and development are needed to achieve native 3D support at DBMS level.

This paper reviews works on 3D DBMS, especially on the aspect of managing volumetric objects. It is organized in the following order – Section 2, a short discussion on the standard specifications for 3D GIS spatial data modeling by Open GIS Consortium. Based on the specifications, Section 3 discusses the implementation of maintaining 3D spatial objects in DBMS. Section 4 describes the previous research works on 3D spatial data modeling. A brief discussion for 3D visualization is given in Section 5 and the paper concludes with recommendations for future work in Section 6.

16.2 The OGC Abstract Specifications for 3D Solids

The Open Geospatial Consortium (OGC 1999) is a non-profit organization that deals with the development of standards for modelling real-world objects. These standards deal with conceptual schemes for describing and manipulating the spatial characteristics of geographic features. The desire to provide a standard specification for GIS was initially driven by the developers - due to

the difficulty in GIS interoperability. The specification, in short, defines three important areas, namely:

- Data types: the need to have data types that represent real world object is obvious. Different kinds of data types and different kinds of objects could be modelled within DBMS.
- Functions/operations: there must be functions and operators to support the management of multi-dimensional objects that work for spatial analysis in DBMS, e.g. objects intersection.
- Spatial index: the main purpose is to deal with spatial searching (query), and sometimes it implements in different operators to speed up the query process.

According to the Spatial Schema, spatial characteristics are described by one or more spatial attributes whose value is given by a geometric object (GM_Object) or a topological object (TP_Object). Geometry provides means for the quantitative description, by means of coordinates and mathematical functions, of the spatial characteristics of features, including dimension, position, size, shape, and orientation. The mathematical functions used to describe the geometry of an object depend on the type of coordinate reference system used to define the spatial position. Geometry is the only aspect of geographic information that changes when information is transformed from one geodetic reference system or coordinate system to another.

Topology deals with characteristics of geometric figures that remain invariant when space is deformed elastically and continuously – for example, when geographic data is transformed from one coordinate system to another. Within the context of geographic information, topology is commonly used to describe the connectivity of an n-dimensional graph, a property that is invariant under continuous graph transformation. Computational topology provides information about the connectivity of geometric primitives that can be derived from the underlying geometry.

This paper will further concentrate on Geometry.

16.2.1 GM_Solid

OGC defines 3D object as GM_Solid (OGC 2001) and it is a subclass of GM_Primitive and is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces. The boundary defines a sequence set of GM_Surfaces that limit the extent of this GM_Solid (see Fig. 1). These surfaces shall be organized into one set of surfaces for each boundary component of the GM_Solid. Each of these shells shall be a cycle (closed composite surface without boundary). In general, a solid in a bounded 3-dimensional manifold has no distinguished exterior boundary. In cases where 'exterior' boundary is not well defined, all shells of the GM_SolidBoundary shall be

listed as 'interior'. The GM_OrientableSurfaces that bound a solid shall be oriented outward – that is, the 'top' of each GM_Surface as defined by its orientation shall face away from the interior of the solid. To represent a 3D solid as a volumetric object, GM_Solid is the best abstract spesification defined by OGC. Other than the GM_Solid, some feature geometry such as GM_Composite also involves a 3D solid object with other primitives, e.g. point, line, and polygon.

There are some functions or operations that could be implemented using GM_Solid. The function/operations are:

- Area: the operation shall return the sum of the surface areas of all of the boundary components of a solid. For example: GM_Solid::area() : Area
- Volume: the operation shall return the volume of this GM_Solid. This is the volume interior to the exterior boundary shell minus the sum of the volumes interior to any interior boundary shell. For example: GM_Solid::volume() : Volume
- GM_Solid (constructor): since this standard is limited to 3-dimensional coordinate reference systems, any solid is definable by its boundary. The default constructor for a GM_Solid is from a properly structured set of GM_Shells organized as a GM_SolidBoundary. For example: GM_Solid:: GM_Solid(boundary : GM_SolidBoundary) : GM_Solid

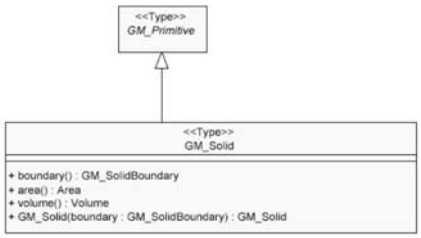


Fig. 16.1 GM_Solid data type defined by OGC

Although the OGC does not discuss some operations that refer to 3D solid e.g. 3D intesection between 2 solids, to extend to the third dimension, similar specifications could be given to the 3D operations, if the z-coordinate is considered. The notion for operations provided by OGC are as provided:

```
return-type type-1::operation(type-2, type-3 ... )
```

Example:

```
Double Precision Geometry 1::Distance(Geometry 2, Geometry 3)

operation(name-1 : type-1, name-2 : type-2, name-3 :
          type-3 ...) : return-type, ...
```

Example:

```
3D Intersects(A1:Geometry 1, A2:Geometry 2) : Geometry 3
```

There are other 3D objects being considered in the OGC specification, i.e. cone, sphere and, etc. Some 3D object are not considered volumetric solids, but still appear in 3D space, i.e. free-form curve and surface. Fig.2 denotes the complete list of 3D objects (with highlighted part) considered in OGC specification.

The OGC abstract specifications deal with geometry and functions. Spatial index is not mentioned in the abstract specification. Therefore, rule or specifications about developing spatial indexing is unavailable. However, the OGC provides for the implementation specification of R-Tree indexing according to the existing DBMS format, i.e. Oracle Spatial. The following section will discuss the basic idea of R-Tree index implemented within DBMS.

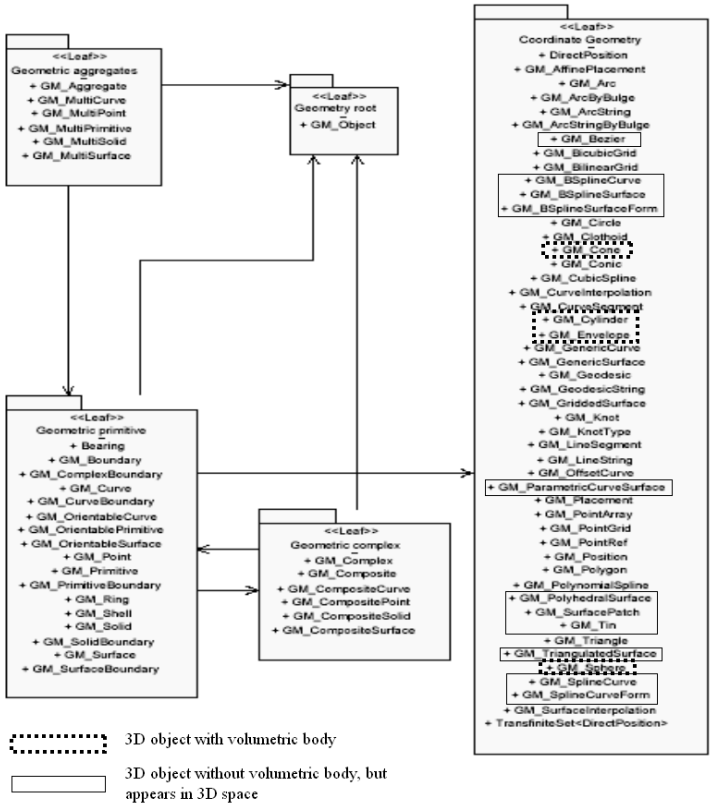


Fig. 16.2 Geometry package in OGC abstract specification

16.2.2 The OGC Implementation Specifications for DBMS

The GM_Solid has been defined by the OGC as a general 3D primitive in abstract specification (OGC 1999a). However, the existing implementation (for SQL) of 3D solid (e.g. polyhedron, tetrahedron) is not available due to the absent of 3D data type (as 3D primitive) within existing DBMS. A volumetric object could be modeled using a multi-collection of similar or different geometries. OpenGIS implementation specification for 3D solid objects can be referred to as *PolyhedralSurface* and *MultiPolygon*. A *PolyhedralSurface* is a contiguous collection of polygons that share common boundary segments. It is a subtype of *Surface*. The primitive of *PolyhedralSurface* and *MultiPolygon* are referred to as *Polygon* (see Fig. 5). The difference between these two geometries is that the polygons that construct *PolyhedralSurface* must share boundaries with the neighboring polygons. The *MultiPolygon* is flexible, i.e. share boundary may not exist for certain polygon(s). For each pair of polygons that 'touch', the common boundary shall be expressible as a finite collection of LineStrings. Each LineString shall be part of the boundary of at most 2 polygon patches. A TIN (triangulated irregular network) is a PolyhedralSurface consisting only of Triangle patches. For any two polygons that share a common boundary, the 'top' of the polygon shall be consistent. This means that when two LinearRings from these two Polygons traverse the common boundary segment, they do so in opposite directions. Since the Polyhedral surface is contiguous, all polygons will be consistently oriented. This means that a non-oriented surface shall not have single surface representations. Fig. 3 shows an example of such a consistently oriented surface (from the top). The arrows indicate the ordering of linear rings from the polygon boundary in which they are located. The methods of implementing the polyhedral surface in DBMS is given as below (see Fig. 4):

```
NumPatches (): Integer - Returns the number of including
                         polygons

PatchN (N: Integer): Polygon - Returns a polygon in this
                         surface, the order is arbitrary.

BoundingPolygons (p: Polygon): MultiPolygon - Returns the
                         collection of polygons in this
                         surface that bounds the given
                         polygon 'p' for any polygon 'p'
                         in the surface.

IsClosed (): Integer - Returns 1 (True) if the polygon
                         closes on itself.
```

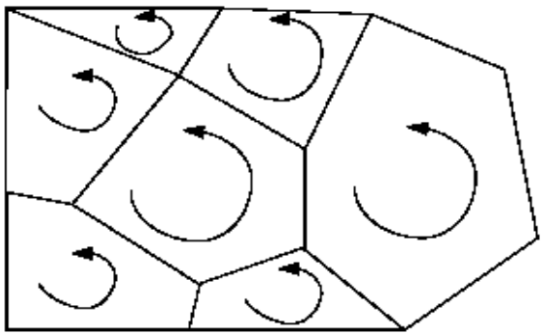


Fig. 16.3 *PolyhedralSurface* with consistent orientation

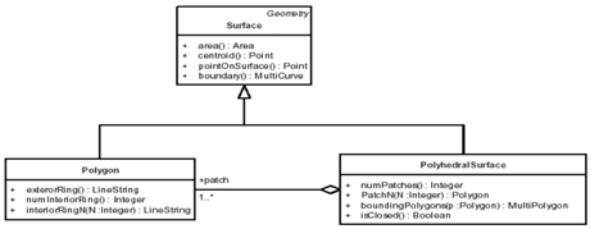


Fig. 16.4 Implementation specification for *PolyhedralSurface*

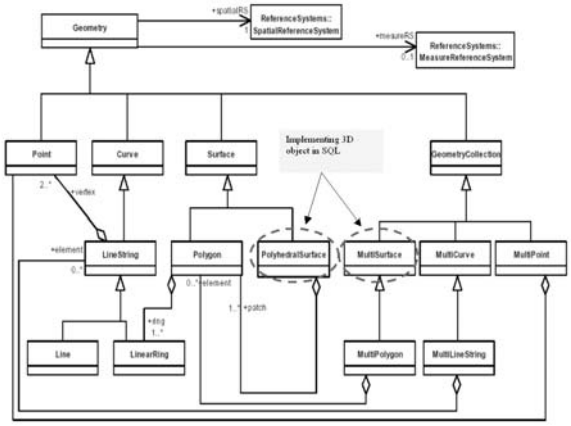


Fig. 16.5 SQL Geometry type hierarchy

In the implementation specification, OGC provides the geometry function that is not limited to any dimension. Only DBMS itself decides the implementation of the standard functions (specified by OGC) that considers the third dimension or not. Some of the standard functions given by OGC (Simple Feature Specification for SQL, Revision 1.1) are:

```
Intersection (g1 Geometry, g2 Geometry): Geometry
```

Return a Geometry that is the set intersection of geometries g1 and g2.

```
Difference (g1 Geometry, g2 Geometry): Geometry
```

Return a Geometry that is the closure of the set difference of g1 and g2.

```
Union (g1 Geometry, g2 Geometry): Geometry
```

Return a Geometry that is the set union of g1 and g2.

```
SymDifference(g1 Geometry, g2 Geometry): Geometry
```

Return a Geometry that is the closure of the set symmetric difference of g1 and g2 (logical XOR of space).

```
Buffer (g1 Geometry, d Double Precision) : Geometry
```

Return as Geometry defined by buffering a distance d around g1

```
ConvexHull(g1 Geometry) : Geometry
```

Return a Geometry that is the convex hull of g1.

Implementing the spatial index that follows the standard specification is not available with the OGC document. This is because the spatial index deals with the method of searching, which often involves mathematical algorithms, e.g. the implementation of R-Tree indexing. A R-Tree is a depth-balanced tree extending the B-tree for n-dimensions. The index stores the minimum bounding boxes as representations, not the objects themselves. It is equally referred to as a minimum bounding rectangle (MBR). A detailed documentation about the R-Tree could be found in Rigaux et al. (2002). There is no standard syntax/command/structure stated by OGC that enables any DBMS to be implemented. Only the DBMSs themselves provide their own syntax/command/structure that establishes the spatial index. The following examples are provided:

(For Oracle Spatial)

```
CREATE INDEX [index_name] on
<table_name>(geometry_column)

INDEXTYPE IS mdsys.spatial_index

PARAMETERS('sdo_indx_dims=3');  -- Dimension = 3
```

(For PostGIS)

```
CREATE INDEX [index_name] ON <table_name>

USING GIST <geometry_column>
GIST_GEOMETRY_OPS);
```

The concept of sample R-tree structure is given in Fig. 6, Fig. 7, & Fig. 8 in two and three-dimensions. The impact of the z-coordinate on 3D spatial indexing will influence the execution time because the indexing mechanism will search each of the (x, y) elements that relate to its z-coordinate. For example, 7 (x, y, z) points will search 7 times greater than 7 (x, y) elements.

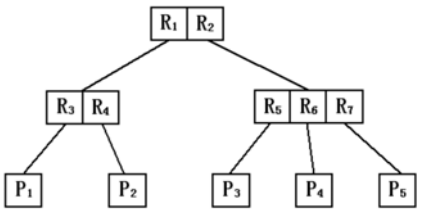


Fig. 16.6 Directory of R -Tree indexing

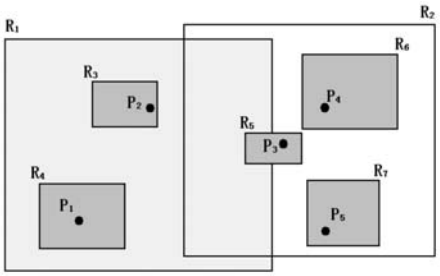


Fig. 16.7 A planar representation of an R-tree

Note that the Oracle Spatial provides the spatial index up to 4D and the dimensionality should be defined in the syntax. However, the GiST index is widely used for 2D data. The implementation of GiST is rather limited for 3D data. The research and application of 3D GiST is expected in the near future. The next section discusses some implementations of spatial indexes for the third dimension in DBMS.

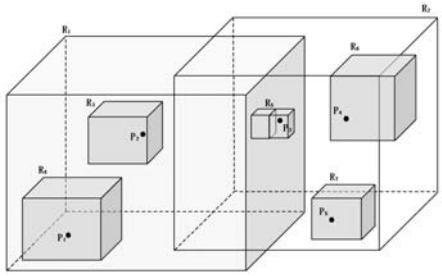


Fig. 16.8 A 3D representation of an R-tree

16.3 Some Implementations of 3D Solid In DBMS

Since the Implementation specifications do not recommend a 3D data type, most of the DBMS (except Oracle Spatial) have not implemented volumetric data types. However, 3D data can be stored in the database since the data types are embedded in 3D space, i.e. point, line and polygon can be represented with their 3D coordinates. The next section describes how 3D real-world objects can be stored in the DBMS using 3D multipolygon.

16.3.1 Modeling 3D Solid Using MultiPolygon

In the Oracle Spatial object-relational model, a 3D solid object from 3D primitive is not possible. However, it could be done by implementing the MultiPolygon that bounds a solid. The geometric description of a spatial object is stored in a single row and in a single column of object type SDO_GEOMETRY in a user-defined table. Any tables that have a column of type SDO_GEOMETRY must have another column, or set of columns, that define a unique primary key for that table. Tables of this sort are referred to as geometry tables.

Oracle Spatial defines the object type SDO_GEOMETRY as:

```
CREATE TYPE sdo_geometry AS OBJECT (
   SDO_GTYPE NUMBER,
   SDO_SRID NUMBER,
   SDO_POINT SDO_POINT_TYPE,
   SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
   SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);
```

An example of implementing a 3D multipolygon (where the geometry can have multiple, disjoint polygons in 3D) is provided below:

```
CREATE TABLE Solid3D (
```

```
   ID number(11) not null,
   shape mdsys.sdo_geometry not null);

INSERT INTO Solid3D (ID, shape) VALUES (
   1 SDO_GEOMETRY(3007,  -- 3007 indicates a 3D multipolygon
   NULL,                 -- SRID is null
   NULL,                 -- SDO_POINT is null
   SDO_ELEM_INFO_ARRAY(  -- the offset of the polygon
      1, 1003, 1,
      16, 1003, 1,
      31, 1003, 1,
      46, 1003, 1,
      61, 1003, 1,
      76, 1003, 1),
   SDO_ORDINATE_ARRAY(
      4,4,0, 4,0,0, 0,0,0, 0,4,0, 4,4,0,  -- 1st polygon
      4,0,0, 4,4,0, 4,4,4, 4,0,4, 4,0,0,  -- 2nd polygon
      4,4,0, 0,4,0, 0,4,4, 4,4,4, 4,4,0,  -- 3rd polygon
      0,4,0, 0,0,0, 0,0,4, 0,4,4, 0,4,0,  -- 4th polygon
      0,0,0, 4,0,0, 4,0,4, 0,0,4, 0,0,0,  -- 5th polygon
      0,0,4, 4,0,4, 4,4,4, 0,4,4, 0,0,4   -- 6th polygon
   )));
```

For PostGIS, the 3D solid as a primitive object is also not available. To create a 3D object that implements existing primitives, then a MultiPolygonM could be used. The three dimensions simply allow a z-coordinate to be stored for each point. The geometry column in PostGIS differs from Oracle Spatial. The description of geometry column is given below:

```
AddGeometryColumn(<table\_name>, <column_name_of_geometry>,
                  <srid>, <geomery_type>, <dimension>)
```

An example of implementing the MultiPolygonM is given below:

```
CREATE TABLE Solid3D (ID integer primary key,
                      NAME varchar (20) not null);}

SELECT AddGeometryColumn('Solid3D', 'shape',
                         423, 'MULTIPOLYGONM', 3);
```

Note that the table name, Solid3D, is given a geometry column named 'shape', with MULTIPOLYGONM type in third dimension. The following example denotes a real multipolygon stored in PostGIS.

```
INSERT INTO Solid3D (ID, shape) VALUES (
   2, -- ID
   GeometryFromText('MULTIPOLYGONM(
```

```
    (4,4,0, 4,0,0, 0,0,0, 0,4,0, 4,4,0) -- 1st lower polygon
    (4,0,0, 4,4,0, 4,4,4, 4,0,4, 4,0,0) -- 2nd side polygon
    (4,4,0, 0,4,0, 0,4,4, 4,4,4, 4,4,0) -- 3rd side polygon
    (0,4,0, 0,0,0, 0,0,4, 0,4,4, 0,4,0) -- 4th side polygon
    (0,0,0, 4,0,0, 4,0,4, 0,0,4, 0,0,0) -- 5th side polygon
    (0,0,4, 4,0,4, 4,4,4, 0,4,4, 0,0,4) -- 6th upper polygon
))));
```

The advantage of implementing the multipolygon in DBMS is that the integration between CAD and GIS is possible for 3D visualization, i.e. Oracle (or called Spatial) spatial schema is supported by MicroStation and Autodesk Map 3D. This is due to the geometry column provided by Spatial directly accesses the 3D coordinates of the object, which allow the display tools to retrieve spatial information from the geometry column. However, problems occur if the data volume is huge, i.e. more polygons are stored for a single 3D solid body. Data size will directly affect data retrieval and yield a slow dataset loading within visualization environment. This weakness could be overcome with the approach of implementing polyhedron as 3D data type in DBMS as proposed by Arens (2003), see Section 4.

Although the implementation of MultiPolygons and Multipatch could be done for 3D visualization, these objects do not represent real 3D objects. They define only a set of bounding surfaces that construct a 3D object. Thus, it is not suitable for 3D analysis. This is one of the main reasons why 3D analytical functions are limited.

16.3.2 Spatial Indexing

Another important aspect of 3D data management is spatial indexing. Spatial indexes are used in DBMS for fast search especially when spatial functions are applied. Without indexing, any searches for a feature would require a sequential scan of every record in the database. Indexing speeds up searching by organizing the data into a search tree that could be quickly traversed to find a particular record. There are several types of indexes within DBMS, e.g. PostGIS and Oracle Spatial: they are B-Tree indexes, R-Tree indexes, and GiST indexes.

- B-Trees are used for data, which can be sorted along one axis; for example, numbers, letters, dates. GIS data cannot be rationally sorted along one axis (which is greater, (0,0) or (0,1) or (1,0)?) so B-Tree indexing is of no use for GIS user.
- R-Trees break up data into rectangles, and sub-rectangles, and sub-sub rectangles, etc. R-Trees are used by some spatial databases to index GIS data, but the PostGIS R-Tree implementation is not as robust as the

GiST implementation. Oracle Spatial will implement the 3D R-Trees in the coming version 11g.

- GiST (Generalized Search Trees) indexes break up data into 'things to one side', 'things which overlap', 'things which are inside' and can be used on a wide range of data-types, including GIS data. PostGIS (2006) uses an R-Tree index implemented on top of GiST to index GIS data.

GiST indexes have two advantages over R-Tree indexes in PostGIS. First, GiST indexes enable the null value in the index columns. Secondly, GiST indexes could easily deal with GIS objects larger than the PostGIS 8K page size. The important part of an object in an index will only be considered within DBMS, e.g. in the case of GIS objects, just the bounding box. GIS objects larger than 8K will cause R-Tree indexes to fail in the process of being built. It could take a long time to create a GiST index if there is a significantly large amount of data in a table. Moreover, 3D indexing is not available within PostGIS.

Other DBMS, e.g. Oracle Spatial, are able to provide 3D indexing for 3D object (MULTIPOLYGON). For Spatial, the metadata that maintains the lower and upper bounds and tolerance of 3D object needs to be created. Later, a spatial index (R-tree in 3D) could be created on tables to speed up spatial queries. The following example denotes the sample in creating a 3D spatial index within Spatial.

```
-- Inserting metadata for 3D object: MULTIPOLYGON

INSERT INTO user_sdo_geom_metadata VALUES
   ('Solid3D', 'shape',
   mdsys.sdo_dim_array(
      mdsys.sdo_dim_element('X', 0, 100, 0.1),
      mdsys.sdo_dim_element('Y', 0, 100, 0.1),
      mdsys.sdo_dim_element('Z', 0, 100, 0.1))
   , NULL);

-- Creating 3D Spatial Index

CREATE INDEX Solid3D_I on Solid3D(shape)
   INDEXTYPE IS mdsys.spatial_index
   PARAMETERS(sdo_index_dims=3);       -- Dimension = 3

ANALYZE TABLE Solid3D COMPUTE STATISTICS;
```

16.3.3 Functions and Operations In DBMS

The 3D functions/operations in DBMS are mainly based on 2D objects that appear in 3D space, i.e. point, line, and polygon (in 3D). Most of the functions consider only the x,y coordinates of the data types although, and they may be given with 3D coordinates. However, there are some exceptions. Some of the 3D functions provided in PostGIS are:

- length3d(geometry): Returns the 3-dimensional length of the geometry if it is a linestring or multi-linestring.
- length3d_spheroid(geometry,spheroid): Calculates the length of of geometry on an ellipsoid, taking the elevation into account. This is just like length_spheroid except vertical coordinates (expressed in the same units as the spheroid axes) are used to calculate the extra distance vertical displacement adds.
- perimeter3d(geometry): Returns the 3-dimensional perimeter of the geometry, if it is a polygon or multi-polygon.
- MakeBox3D(<LLB>, <URT>): Creates a BOX3D defined by the given point geometries. LLB denotes lower left bottom, whereas URT denotes upper right top.
- xmin(box3d) ymin(box3d) zmin(box3d): Returns the requested minimum of a bounding box.
- xmax(box3d) ymax(box3d) zmax(box3d): Returns the requested maximum of a bounding box.

3D operations in existing DBMSs are hardly available. For example, due to the third dimension, Oracle Spatial is not considered in any function and operation, thus the 3D function and operation are not available. Maintaining objects with 3D coordinates are possible but the functions available within DBMS still do not consider the third-dimension. Some exceptions are only limited to geometry calculations, e.g. 3D length and 3D perimeter. The existing spatial functions are only based on the native geometry model, i.e. buffer for 2D polygon. The 3D operation for DBMS must focus on two directions:

- The existing operations have to be extended to the third-dimension, in which the z-coordinate must be involved, i.e. 3D intersection, 3D buffer, and etc.
- New 3D operations have to be developed based on topological models, i.e. 3D overlap, 3D meet that extended from 9-intersection model.

In the coming Oracle Spatial 11g, the 3D coordinate system will be implemented in DBMS environment. The 3D coordinate systems are all based on European Petroleum Survey Group (EPSG) specifications. The supported coordinate systems are: Vertical coordinate systems, Geocentric (3D Cartesian), Geographic (3D ellipsoidal), and Compound coordinate System.

16.4 Problems and Issues on 3D Data Modeling in DBMS

A number of works attempt to address the problem of spatial data modeling for 3D GIS where most of these efforts focused on polyhedron, tetrahedron, triangulated tetrahedron and even free-form curves and surfaces as a mechanism to formalize 3D spatial data modelling. The following section discusses some recent works on data modeling in DBMS.

16.4.1 Modeling 3D Solid in DBMS

16.4.1.1 Polyhedron

The modelling 3D spatial object and corresponding operations in a spatial DBMS has been investigated quite successfully by Arens (2003), and Arens et al. (2005). The basic idea was that a 3D polyhedron could be defined as a bounded subset of 3D space enclosed by a finite set of flat polygons, such that every edge of a polygon is shared by exactly one other polygon. The polygons are in 3D space because they are represented by vertices that appear in 3D space. The 3D primitive implemented by Arens was in a geometrical model with internal topology. The polyhedron was realized by storing the vertices explicitly (x,y,z) and describing the arrangement of these vertices in the faces of the polyhedron. This yields a hierarchical boundary representation (Aguilera 1998; Verbree and Zlatanova 2004). The sample of a polyhedron is illustrated in Fig. 9a, and the polyhedron storage is depicted in Fig. 9b.

The functions/operations given by Arens includes validation for polyhedron, spatial conversion, topological operation, and metric functions. To visualize 3D objects, it is necessary to use programs that actually show the third dimension. There are two options as proposed by Arens:

- GIS/CAD programs make a DBMS connection, for instance Microstation (Bentley 2007). These programs can only handle 3D objects that consist of multiple 2D objects. The 3D data stored as a 3D type needs a conversion before it can be visualised, e.g. splitting up the 3D object in multiple 2D polygons.
- VRML (Virtual Reality Modelling Language). When using VRML, there needs to be translation between the 3D type in the database and the VRML syntax.

These two representations have advantages and disadvantages. Displaying 3D objects using VRML require an extra step for 3D visualization. The polyhedron needs to be converted into a VRML file. First, the VRML file is stored as an SQL-loader file. Then, SQL-loader (from Oracle tool) load this file into DBMS environment to construct a table. The object's geometry will

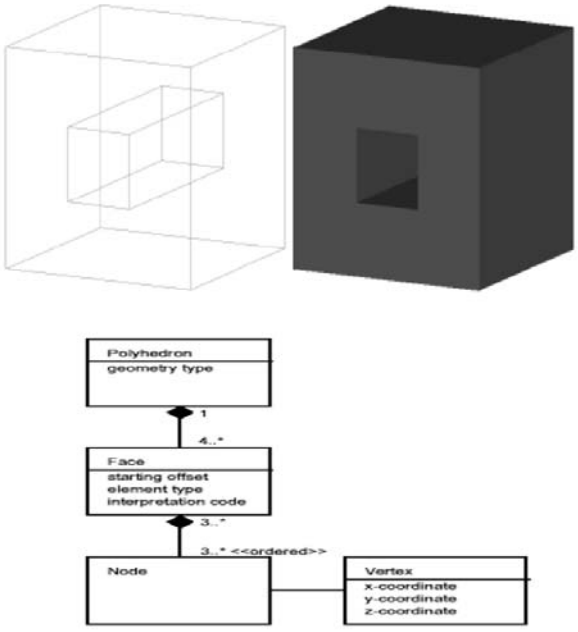


Fig. 16.9 Sample of polyhedron, and UML diagram of polyhedron storage (after Arens 2003)

be added into the table and the VRML file can be browsed on the Internet. Taking advantage of web display, the data exchange/transfer could be done easily by extracting the VRML file. However, the VRML file is not part of the DBMS environment. The 3D visualization becomes inefficient if the data volume is huge – this happens when the conversion of geometry (from DBMS) to VRML file is carried out. However, the weakness could be overcome by integrating DBMS and display tool directly, i.e. GIS/CAD integration. In this case, a CAD system, such as Microstation, could be connected directly to the DBMS and retrieve the 3D data for 3D visualizaton.

16.4.1.2 TEN

Another attempt to define 3D object has been reported by Penninga, 2005. The 3D object, i.e. tetrahedron, is used to represent 3D volumetric shapes. The tetrahedron is the simplest possible geometry in the 3D domain. The conceptual design was intended for implementation of both geometrical and topological models in topographic modeling.

Initially, Penninga (2005) attempted to implement the TIN/TEN(2.5D /3D) model approach for topographic modeling. The idea is that the earth's terrain can be modelled in 2.5D TIN. The complex object will be mapped on top or below this terrain. This leads to the combination of TIN/TEN model (TIN: Triangulated Irregular Network / TEN: Tetrahedronized Irregular Network). However, since problems appear at both the conceptual and implementation level, an alternative model was suggested, i.e. the full TEN model. The shift to the full 3D model avoids the complication of designing multiple data structures in both TIN and TEN models for different spatial objects (Penninga et al. 2006; Penninga and van Oosterom 2007).

In the TEN model, four types of topographic features can be determined in this integration: 0D (point features), 1D (line features), 2D (area features) and 3D (volume features). For each type, feature simplexes of corresponding dimension are available to represent the features with nodes, edges, triangles and tetrahedrons (see Fig. 10).

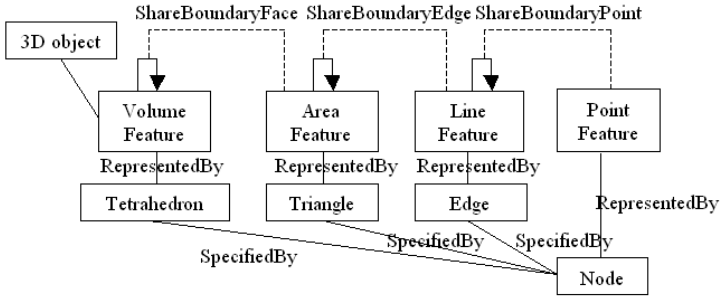


Fig. 16.10 Logical design of 3D TEN (after Penninga & van Oosterom 2007)

With this TIN/TEN integration, a minor drawback would occur if an object became more complex, such as a complex building block. The entire building could be modeled using triangles as a whole to complete the geometry. An undesirable side effect is that the data size may become rather large, because more faces have to be stored in the data structure. The triangulation approach produces more storage, as compared to the polyhedron approach depending on the complexity of 3D objects. Since the space is completely subdivided into tetrahedrons, the interiors of objects (e.g. buildings), as well as the open space, are also decomposed into tetrahedrons. These tetrahedrons, however, require additional algorithms to be developed as a whole building block. This leads to database size expansion (see Fig. 11) and longer response time for visualization. More information on this comparison (polyhedron and TEN) can be found in Zlatanova 2000.

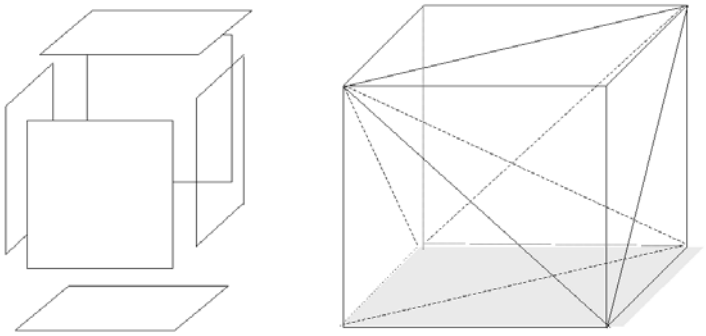


Fig. 16.11 Comparison of the total faces/triangles between polyhedron and TEN (after Zlatanova 2000)

16.4.1.3 Triangulated Polyhedron

The triangulated polyhedron was proposed by Ledoux and Gold (2004) and it was based on 3D Voronoi Diagram (VD) and Delaunay Tetrahedralization (DT). The Voronoi diagram for a set of points (in a given space, R is the partitioning of that space into regions such that all locations within any one region are closer to the generating point than to any other.

In 2D, this structure is defined by partitioning the plane into triangles (where the vertices of the triangles are points that generate each Voronoi cell) that satisfy the empty circumcircle test (a circle is empty when no points are in its interior, but more than three points can be directly on the circle). In any dimensions, the VD has a geometric dual structure called the Delaunay Triangulation. The two-dimensional DT is illustrated in Fig. 12 by the dashed lines. The Delaunay Triangulation is appropriate for modelling surfaces because among all the possible triangulations of a set of points, it creates one where the minimum angle in each triangle is maximized (triangles are as equilateral as possible), thus being useful for interpolation.

In three-dimension, a Voronoi cell generalizes to a convex polyhedron formed by convex faces, as shown in Fig. 13. The generalization to three dimensions of the Delaunay Triangulation is the Delaunay tetrahedralization: each triangle becomes a tetrahedron that satisfies the empty circumsphere rule. The DT is unique for a set of points, except when there are degenerate cases in the set (if five or more points are cospherical in 3D). In these cases, an arbitrary choice must be made among all the possible solutions. The number of tetrahedra in a DT constructed with n points depends on the configuration of these points.

It can be realized that triangulated polyhedron could be utilized for generating 3D spatial objects and eventually into DBMS.

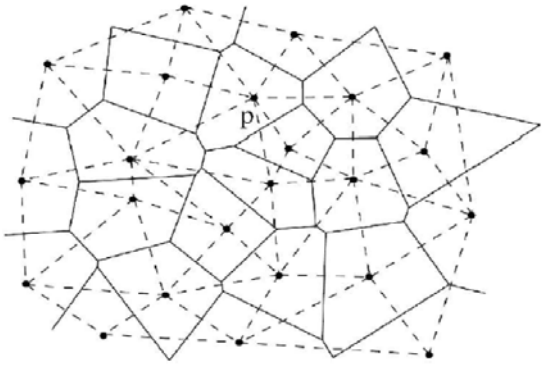


Fig. 16.12 Two-dimensional VD (bold lines) and DT (dashed lines) (after Ledoux & Gold 2004)

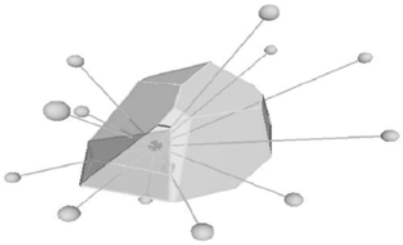


Fig. 16.13 The Voronoi cell in 3D (after Ledoux & Gold 2004)

16.4.1.4 Modeling 3D Freeform Curves and Surfaces

Complex geometry types such as freeform curves and surfaces can be implemented in DBMS. Many shapes in the real world are freeforms, i.e. not only contain points, linestrings and polygons, but also curves and curved surfaces, e.g. roads, building surfaces, and etc. Pu (2005) has created complex geometry data types that describe freeform curves and surfaces. Although freeform shapes can be simulated by tiny line segments/triangles/polygons, it is quite unrealistic and inefficient to store all these line segments/triangles/polygons into a DBMS especially when shapes are rather huge or complex. The freeform shapes discussed by Pu 2005, Pu and Zlatanova, 2006 are Bezier (Fig. 14), B-spline and NURBS.

A B-Spline surface is an expansion of B-spline curves (Fig. 15a) and B-spline curves are a generalization of Bezier curves, and the same applies for

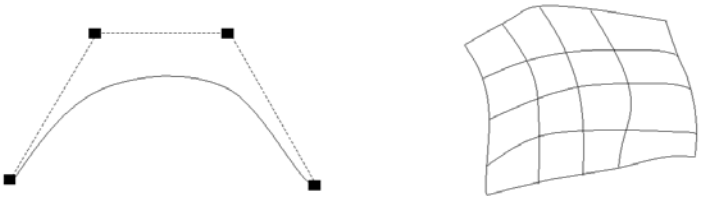


Fig. 16.14 (a) Bezier curve, and (b) bi-cubic Bezier surface (Pu, 2005)

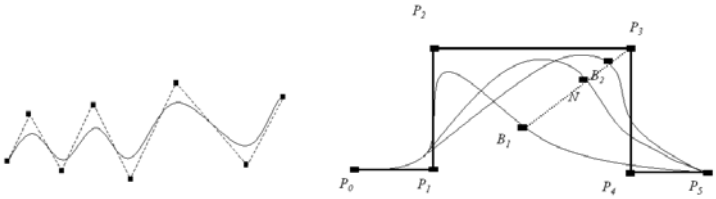


Fig. 16.15 (a) B-spline, and (b) NURBS curve (Les, 1991)

surfaces. NURBS (Non-Uniform Rational B-Splines) curve is generalized from B-Spline curve (Fig. 12b). The implementation was done in Oracle Spatial, where new data types for each Bezier curve, B-spline curve and NURBS curve were created separately. An alternative approach was to create a data type for NURBS curves, which also represented Bezier curves and B-spline curves by leaving some parameters of NURBS curve empty - because NURBS curve is actually the generalization of Bezier curve and B-spline curve. The final freeform datatypes could include:

- Three curve types: GM BezierCurve, GM BSplineCurve and GM NURBSCurve (see Fig. 16a),
- One surface type: GM NURBSSurface (see Fig. 16b), and
- Four supplementary types: GM PointArray, GM WeightArray, GM Knot Vector and GM Trim.

The freeform curve and surface developed by Pu 2005 could not represent a 3D solid object. Although these data types consider the z-coordinate, the objects do not bound a volumetric body. The research could be extended from freeform surface that able to envelope a solid body, but it yields greater complexity due to more complex mathematical algorithms are required. As a result, it will slow down the process of 3D visualization and spatial operation.

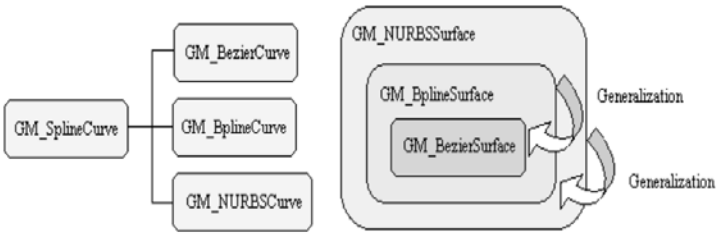


Fig. 16.16 (a) Freeform curve, and (b) Freeform surface datatypes

16.4.2 3D Spatial Indexing

Spatial searching is a fundamental primitive in non-traditional databases such as GIS, CAD/CAM and multi-media applications. With the rapid proliferation of these databases in the past decade, extensive research has been conducted on the design of efficient data structures to enable fast spatial searching. Several data structures have been developed in this context,including Quadtrees (Wang, 1991), R-trees (Guttman, 1984), hB-trees (Lomet and Salzberg 1990), and TV-trees (Lin et al. 1984). Subsequent research has improved these basic structures further by proposing new techniques for query processing (Berchtold et al. 2000; Ferhatosmanoglu et al. 2001), faster and better index creation (Garcia et al. 1998), and better split-strategies in dynamic updates (Beckmann et al. 1990; Berchtold et al. 1996). These techniques are especially effective for low-dimensional spatial data such as those in GIS and CAD/CAM applications.

For indexing low-dimensional spatial data, certain DBMSs allow users to choose between one of two spatial indexes: a (Linear) Quadtree or an R-tree. The Oracle implements these two kinds of spatial indexes and incorporates and enhances some of the best proposals from existing spatial indexing research. The PostGIS implements the GISt indexing for spatial query.

Most of the spatial indexes are extended from these two kinds of indexing methods. The Linear Quadtree (or Quadtree for short) computes tile approximations for geometries and uses existing B-tree indexes to perform spatial searches. This approach results in simpler index creation, faster updates and inheriting a built-in B-tree concurrency control protocol. The R-tree is implemented logically as a tree and physically using tables inside the database and the search involves recursive SQL for traversing the tree from root to relevant leaves. This approach may be more efficient for queries due to the enhanced preservation of spatial proximity but may be slow in updates or index creation and implements its own concurrency protocols on top of spatial DBMS table level concurrency mechanisms.

The conventional approach to support similarity searches in high-dimensional vector spaces can be broadly classified into two categories:

The first approach uses data-partitioning index trees. Neighbouring vectors are coveredby MBRs (minimum bounding rectangles) or MBSs (minimum bounding spheres), which are organized in a hierarchical tree structure. Many index tree schemes have been proposed. They include the R-tree, the R*-tree (Beckmann et al. 1990, the Hilbert R-tree (Kamel and Faloutsos 1994), and the SS-tree (White and Jain 1996). In addition, nearest neighbour search methods using such indices have been proposed (Henrich 1994; Hjaltason and Samet 1995). Two recently proposed indices, the X-tree (Berchtold et al. 1996) and the SR-tree (Katayama and Satoh 1997), are reported to offer good performance. The X-tree introduces the notion of a supernode and outperforms the R*-tree. The SR-tree has a unique feature in that it uses both MBRs and MBSs and is reported to outperform both the R*-tree and the SS-tree.

The second approach is the use of approximation files. Among the others, the VA-file (vector approximation file) (Weber et al. 1998) is a simple yet powerful scheme. The VA-file divides the data space into cells and allocates a bit-string to each cell. The vectors inside a cell are approximated by the cell and the VA-file itself is simply an array of these geometric approximations. When searching vectors, the entire VA-file is scanned to select candidate vectors. Those candidates are then verified by visiting the vector files. Weber et al. (1998) report that the VA-file outperforms both the R*-tree and the X-tree when dimensionality is high. In the field of spatial search of high-dimensional data, this problem looms larger and larger. Search methods that present an approximate answer (Arya et al.1994; Gionis et al. 1999), have been proposed to avoid the problem. Although these methods are useful, to overcome this problem, an A-tree index was proposed by Sakurai et al. 2002. Introduction of the A-tree is motivated by a comparison and analysis of the SR-tree and VA-file. The basic idea of A-tree is the introduction of virtual bounding rectangles (VBRs), which contain and approximate MBRs or data objects. The A-tree indexing is based on the following design structures:

- Tree structure: It adopts a tree index to limit the searching result from one phase to the next phase.
- Relative approximation: to overcome the problem of tree indices identified in evaluation results, a new notion (i.e. relative approximation) was introduced, which is a simple yet powerful approximation method utilizing the hierarchy of tree indices. In relative approximation, bounding regions or data points are approximated by their relative positions in terms of the parent's bounding region.
- Partial usage of MBSs: since the SR-tree is one of the best indices among the tree indices proposed so far, the SR-tree is used as the starting point in designing the A-tree. However, the effect of MBSs is limited when searching high dimensional data. Hence, MBSs are not stored in the A-tree. As a result, the centroid of data objects in a subtree is used only for insertion and deletion. The A-tree is a new index that applies the notion of relative approximation to the hierarchical structure of the SR-tree.

However, this application is not naive; A-tree's configuration is unique in that: i) each node contains an MBR and a representation of the relative approximation of its children; and ii) the centroid of data objects is used only for updating purposes.

16.4.3 3D Operations

Since the subject of implementing 3D topological operations for geometrical structure in a relational DBMS is a fairly unexplored area, some approaches will be considered in developing 3D spatial operation for DBMS:

- The 3D spatial operation will cover all necessary topological structures that define a complete solid object. In certain cases, not all primitives are needed, e.g. a polyhedron is defined by an ordered set of coordinate triplets for each polygon that bound a volumetric body, line will not be used in the data structure.
- Implementation of the 3D spatial operations will be tested within the DBMS environment.
- The results from 3D topological operations return to a Boolean form (TRUE/FALSE). It involves two spatial objects, polyhedron and polyhedron.

The topological operations presented here are based on the body-body relation (Zlatanova, 2000). Typically, the results given by this operation are in Boolean type, i.e. either TRUE or FALSE. The related operations include Overlap, Meet, Disjoint, Inside, Covers, CoveredBy, Contain, and Equal (see Fig. 17).

For topological operations in a geometrical model, a coordinate triplet of the vertex is used. Similar to computational-geometry operation from previous sections, the binary operation is divided into base and target object. However, the vertices from base object and polygons from target object will be discussed (see Fig. 18a). This topological operation involves vertices (from base object) and polygon (from target object). Therefore, the relation between these two objects will be examined. The location of base vertices relative to target polygon will be either outside, touch, or inside as has been implemented and discussed in Chen and Abdul-Rahman (2006). These relations will be used to determine how these two polyhedrons intersect each other.

The following table (Fig. 18b) denotes the complete relationship between base and target object. The 'X' sign represents the impossible intersection between two objects, whereas the 'check' sign represents the possible intersection for geometrical models.

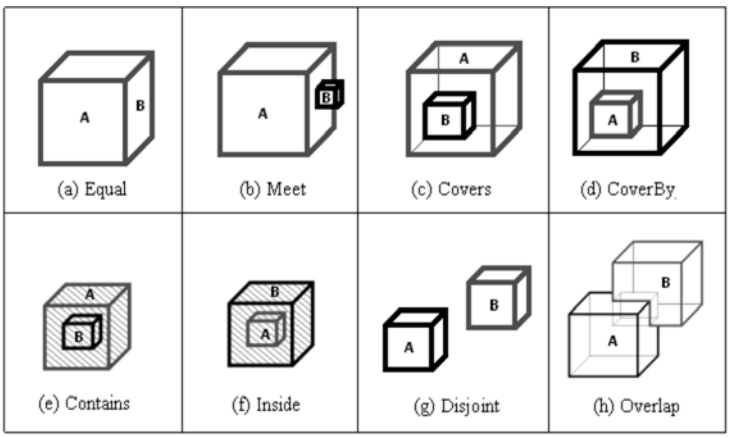
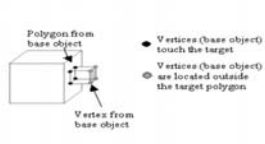


Fig. 16.17 Body and body relation (after Zlatanova, 2000)



3D topological operations	Inside	Outside	Touch
Equal	X	X	✓
Meet	X	✓	✓
Covers	✓	X	✓
CoveredBy	✓	X	✓
Contains	✓	X	X
Inside	✓	X	X
Disjoint	X	✓	X
Overlap	✓	✓	✓

Fig. 16.18 Vertices (base) touch the target polygon (3D Meet), and Conditions for topological operations (geometrical model)

The implementations of 3D topological operations involve two intersecting polyhedrons. This implementation was performed within the PostGIS environment. Below is the structure of polyhedron:

```
SELECT * FROM Solid3D WHERE PID = 1;

1,POLYHEDRON(PolygonInfo(6,24),SumVertexList(8),
SumPolygonList(4,4,4,4,4,4),
VertexList(100.0,100.0,100.0,400.0,100.0,100.0,400.0,
400.0,100.0,100.0,400.0,100.0,100.0,100.0,400.0,400.0,
100.0,400.0,400.0,400.0,400.0,100.0,400.0,400.0),
PolygonList(1,2,6,5,2,3,7,6,3,4,8,7,4,1,5,8,5,6,
7,8,1,4,3,2)) o,400.0,100.0,100.0,100.0,400.0,400.0,100.0,
400.0,400.0,400.0,400.0,100.0,400.0,400.0),
PolygonList(1,2,6,5,2,3,7,6,3,4,8,7,4,1,5,8,5,6,7,8,1,4,3,2))}
```

1. PolygonInfo(6,24) denotes 6 polygons and 24 IDs of polygon arrange in PolygonList,
2. SumVertexList(8) denotes the total vertices,

3. SumPolygonList(4,4,4,4,4,4) denotes total vertices for each of polygon (total polygon is 6, referred to (1)),
4. VertexList() denotes the list of coordinate-values for all vertices (with no redundant), and
5. PolygonList() denotes the information about each polygon from sets of ID.

The following SQL statement runs the 3D Overlap (see Fig. 19):

```
SELECT GMOVERLAP3D(a.POLYHEDRON,b.POLYHEDRON) AS GM_OVERLAP3D
FROM test a, test b where a.PID=1 and b.PID=2;
```

The result:

```
GM_OVERLAP3D
------------
 (TRUE)
```

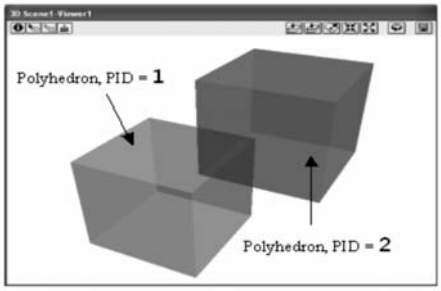


Fig. 16.19 3D Overlap

16.5 3D Visualization

Without visualization, any queries from database would be just numbers and characters – thus hard for users to decipher the meaning of the generated information. DBMS only provides a medium for data set management, and it certainly requires a front-end tool for visualizing the information as it is perceived in the real world. The data from DBMS needs to be integrated into a visualization tool so that it could be viewed as graphic. The 3D spatial data stores in the spatial column (within DBMS), and a connection needs to be built so that a display tool manages to access the spatial column and retrieve the data for 3D visualization.

It is also important to note that 3D objects need to be visualized in realism. With the benefit of the computer graphic technology, GIS could provide a good display with textures and colours. Some web application, e.g. Google Earth (GE) maintains the texture of spatial object over the Internet. GE is a dynamic 3D virtual globe application that contains high-resolution satellite and airborne images streamed through the Internet. GE uses a specific standard for external data sources called the Keyhole Markup Language (KML and KMZ is the zip version). KML is a file format used to display geographic data in an earth browser. A KML file is processed in much the same way that HTML (and XML) files are processed by web browsers. Like HTML, KML has a tag-based structure with names and attributes used for specific display purposes. Thus, Google Earth and Maps act as browsers for KML files. In Fig. 20, elements to the right on a particular branch in the tree are extensions of elements on the left. For example, Placemark is a special kind of Feature. It contains all of the elements that belong to Feature and adds some elements that are specific to the Placemark element.

Other than visualization in 3D, it is important to have spatial query and data updating based on the 3D data from the display tool. Some of the software, e.g. ArcGIS, and Microstation could provide data editing/update, and perform spatial query. Another advantage of integration between DBMS and the visualization tool is that posting data could be done from the display tool. Furthermore, the data will be stored and converted into the DBMS enviroment.

Another important element of 3D visualization is Level-Of-Detail (LOD). The concept of Levels of Detail (LOD) has been introduced to facilitate visualization of large scenes (see Clark 1976). The idea is to represent spatial objects that are compatible with the pixel size of the screen, relative of the observer's distance. This permits the original geometric representation to be replaced with a new low-resolution represebtation. Low-resolution representations require less memory and processing time for rendering and hence speeding-up the visualization process. The different representation is used by the visualisation system only if the object is far enough from the user. Closer objects are still represented in their full resolution. Moreover, if the distant object gets closer (as a result of the user's navigation through the model), the high-resolution representation is restored. The intentions are an unnoticeable switch between low and high levels of detail.

Currently, the CityGML (Kolbe et al. 2006) supports different LOD. It requires independent data collection processes with differing application requirements. In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualization of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated. The CityGML provide multiple kinds of LOD as given in below (see Fig. 21):

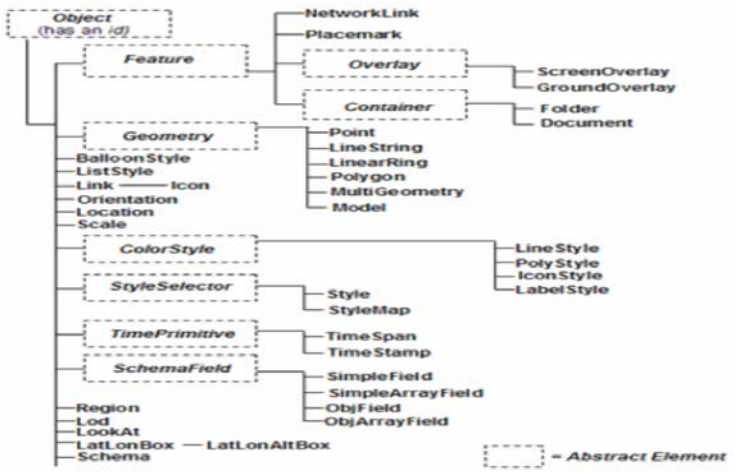


Fig. 16.20 Sample KML file format

- LOD0 is essentially a two and a half dimensional Digital Terrain Model. An aerial image or a map may be draped on the DTM.
- LOD1 is the blocks model representing the buildings with flat roofs.
- LOD2 is used to differentiate the roof structures among different building. It also used in differentiating surfaces thematically.
- LOD3 provide a building model with detailed wall and roof structures, balconies, bays and projections. Vegetation objects may also be represented in this level. High-resolution textures can be mapped onto these structures too.
- LOD4 completes a LOD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture.

LOD are used not only to speed up visualisation but also for different applications. For example LOD1 (CityGML) is perfect for air pollution analysis; LOD3 is good for realistic visualisation; LOD4 can be used for evacuation from buildings.

To maintain the LOD and colour/texture attributes could be performed in both the DBMS and display tool. Certain display tools, e.g. VRML browser, ArcGIS, and Microstation are able to maintain the colour and texture well. This is not necessary to maintain the texture and colour attribute within DBMS since these display tool manage to maintain these features with better interactive functions. However, the LOD is required to store within DBMS. This is because different LOD represent different kinds of geometry. For example, from CityGML, LOD1 mainly stores simple block models that represent buildings with flat roofs; LOD2 differentiate the roof structures among dif-

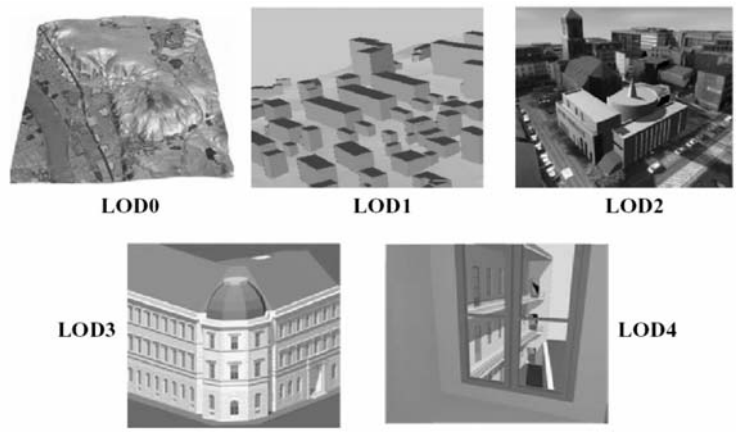


Fig. 16.21 The five levels of detail (LOD) defined by CityGML, 2006

ferent building. With the same building, different LOD represents different kind of roofs. The different LOD could only be stored in visualization tool, if and only if, it converts into different layer of graphical data.

16.6 Conclusions

A number of issues and challenges must be addressed to develop and manage 3D solid objects in database. The problems, challenges and issues could be summarized as:

1. An appropriate 3D datatype that defines 3D primitive for geometry and topology needs to be developed. Different kinds of 3D spatial objects deal with different applications, e.g. TEN deals with terrain modeling and polyhedron addresses with building structures. The 3D datatype should follow the standard specification provided by OGC, and store three-dimensional objects in an DBMS environment. The DBMS structure that stores 3D primitive in spatial column should be able to manage and maintain different kinds of LOD (texture/colour attributes are optional since certain display tools could manage these well). The performance in terms of the size of data storage and management efficiency need to be given attention because DBMS provides the medium for data management and should also be integrated with other aspects like 3D display. Therefore, an efficient DBMS that supports various kinds of 3D primitives is important for 3D spatial modeling.
2. Spatial indexing: It is a mechanism that is usually applied to accelerate the process of queries in the database by keeping some extra in-

formation. Many types of indexing methods have been cited such as the R-tree (Guttman 1984), the K-D-B-tree (Robinson 1981) or the Z-ordering (Orenstein 1986). Although other spatial indices that combine a tree structure and a capacity technique have been proposed (Seeger and Kriegel 1990; Berchtold et al. 2000), novel algorithms and structures that give very high performance for high dimensionalities (e.g. 3-dimension) need to be developed. Again, the performance of 3D indexing will also need to be evaluated. The 3D R-tree indexing available in Oracle Spatial, A-tree (for 3D spatial index by Sakurai 2002) and other 3D indexing should be compared.

3. Functions and operations: There are several geometrical algorithms that deal with spatial and attribute data manipulations for GIS analysis. The importance of this algorithm is directly referred to its application, e.g. calculation of volume for land subsidence, etc. Some DBMSs implement a wide range of functions for database management and spatial analysis. The spatial operations could be divided into several types and need to addressed as well:

   - Computational-geometry operations: functions that return a new geometry from two objects intersection, e.g. 3D Intersection, and 3D Union.
   - Topological operations: functions that return a Boolean result from 2 object intersection, e.g. extending 3D overlap.
   - Metric operations: functions that involve mathematical calculation, e.g. volume calculation of tetrahedron and polyhedron.

4. 3D visualization and interaction: Visualisation is mostly used in the context of display 3D graphics. Realism display of the objects is also an issue and could be categorized into two different approaches: texture/colour and Level Of Detail (LOD). One of the questions like 'how close to the actual real world' one could display and interact with the object. Another issue of 3D visualization is interaction. Ideas about how user-friendly the interactive tool needs to be for individuals to perform tasks in 3D GIS must be adressed. Different principles and applications lead to different approaches of visualization and interaction methods.

We reviewed and described a number of research works pertaining to the 3D solids associated with spatial data modelling and management in DBMS. The discussions cover the 3D datatype, spatial indexing, and functions/operations (from standard specification to implementation; from commercial to research/development). However, many issues must be addressed to improve the current situation of 3D spatial modeling. The most important issue for 3D spatial data modeling is the standardization and specification of GIS. Although some of the specifications (abstract specification) are discussed in this paper, many other standards need to be investigated as well, i.e. 3D operations (geometry and topology) for solid objects. The implementation of 3D operations could be done in DBMS. The spatial operators should

involve some procedures that can use, query, create, modify, or delete spatial objects.

Other challenges in the 3D GIS domain include interoperability between different applications, data model, integration between DBMS and visualization, and the link between data modelling and data acquisition.

References

Abdul-Rahman A, Zlatanova S, Coors V (2006) Lecture Note on geoinformation and cartography – Innovations in 3D Geo Information Systems, Springer-Verlag

Aguilera A (1998) Orthogonal polyhedra: study and application. Ph.D. Thesis, LSI-Universitat Politècnica de Catalunya

Arens CA (2003) Modelling 3D spatial objects in a geo-DBMS using a 3D primitives. Msc thesis, TU Delft, The Netherlands

Arens C, Stoter JE, van Oosterom PJM (2005) Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. In: Computers & Geosciences, 31:165-177

Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1994) An optimal algorithm for approximate nearest neighbor searching. In: Proc. ACM-SIAM Symposium on Discrete Algorithms, pp. 573-582

Beckmann N, Kriegel H, Schneider R, Seeger B (1990) The R* tree: An efficient and robust access method for points and rectangles. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 322-331

Bentley (2007) available at `http://www.bentley.com/`

Berchtold S, Keim DA, Kreigel HP (1996) The X-tree: An index structure for high dimensional data. In: Proc. of the Int. Conf. on Very Large Databases

Berchtold S, Keim DA, Kriegel HP, Seidl T (2000) A new technique for nearest neighbor search in high-dimensional space. IEEE Trans. In: Knowledge and Data Engineering, 12(1):45-57

Chen TK, Abdul-Rahman A (2006) 0-D feature in 3D planar polygon testing for 3D spatial analysis. In: Abdul-Rahman A, Zlatanova S, and Coors V (eds), Lecture Note on geoinformation and cartography – innovations in 3D

Geo information systems, Springer-Verlag. pp. 169-183

CityGML available at `http://www.citygml.org/`

Clark JH (1976) Hierarchical geometric models for visible surface algorithm. In: Communications of the ACM, 19(10), pp. 547-554

ESRI (2007) available at `http://www.esri.com/`

Ferhatosmanoglu H, Tuncel E, Agrawal D, Abbadi AE (2001) Approximate nearest neighbor searching in multimedia databases. In: Proc. Int. Conf. on Data Engineering, pp. 503-511

Garcia YJ, Leutenegger ST, Lopez MA (1998) A greedy algorithm for bulk loading R-trees. In: Proc. of ACM GIS

Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proc. 25th International Conference on Very Large Data Bases (VLDB), pp. 518-529

Guttman A (1984) R-trees: A dynamic index structure for spatial searching. In: Proceedings of ACM SIGMOD, International Conference on Management of Data, Boston, MA, pp. 47-57

Henrich A (1994) A distance scan algorithm for spatial access structures. In: Proc. ACM International Workshop on Advances in Geographic Information Systems, pp. 136-143

Hjaltason GR, Samet H (1995) Ranking in spatial databases. In: Proc. 4th Symposium on Spatial Databases, pp. 83-95

Ledoux H, Gold CM (2004) Modelling oceanographic data with the three-dimensional Voronoi diagram. In: ISPRS 2004-XXth Congress, Istanbul, Turkey,. Vol. 2, pp. 703-708

Kada M, Haala N, Becker S (2006) Improving the realism of existing 3D city model. In: Abdul-Rahman A, Zlatanova S, and Coors V (eds), Lecture Note on geoinformation and cartography – innovations in 3D Geo information systems, Springer-Verlag. pp. 405-415

Kamel I, Faloutsos C (1994) Hilbert R-tree:An improved R-tree using fractals. In: Proc. 20th International Conference on Very Large Databases, pp. 500-509

Katayama N, Satoh S (1997) The SR-tree: an index structure for high-dimensional nearest neighbor queries. In: Proc. ACM SIGMOD International Conference on Management of Data, pp. 369-380

Kolbe T, Groeger G, Czerwinski A (2006) City Geography Markup Language (CityGML). In: OGC, OpenGIS Consortium, Discussion Papers, Version 0.3.0

Les P (1991) On NURBS: a survey. IEEE Computer Graphics and Applications 11(1): 55-71

Lin KI, Jagdish HV, Faloutsos C (1994) The TV-tree: An index structure for high-dimensional data. VLDB Journal, 3:517-542

Lomet DB, Salzberg B (1990) The hB-tree: A multi-attribute indexing method with good guaranteed performance. Proc. A CM Syrup. on Transactions of Database Systems, 15(4):625-658

OGC (1999) Abstract specifications overview. Available at `http://www.opengis.org/`

OGC (1999a) OpenGIS simple features specification for SQL. Available at `http://www.opengis.org/`

OGC (2001) The OpenGIS™ Abstract specification, topic 1: feature geometry (ISO 19107 Spatial Schema) Version 5

Oracle Spatial 10g available at `http://www.oracle.com/`

Orenstein J (1986) Spatial query processing in an object-oriented database system. In: Proceedings of 1986 ACM SIGMOD International Conference on Management of Data, pp. 326-336

Penninga F (2005) 3D topographic data modelling: why rigidity is preferable to pragmatism. In: Spatial Information Theory, Cosit'05, Vol. 3693 of Lecture Notes on Computer Science, Springer. pp 409-425

Penninga F, van Oosterom PJM, Kazar BM (2006) A TEN-based DBMS approach for 3D topographic data modelling. In: Spatial Data Handling 2006

Penninga F, van Oosterom PJM (2007) A compact topological DBMS data structure for 3D topography. In: Fabrikant S, Wachowicz M (eds.), Lecture Notes in Geoinformation and Cartography. ISBN: 978-3-540-72384-4

Pilouk M (1996) Integrated modelling for 3D GIS. PhD Thesis, ITC, The Netherlands

PostGIS (2006) available at `http://postgis.refractions.net/`

Pu S (2005) Managing freeform curves and surfaces in a spatial DBMS. Msc Thesis, TU Delft

Pu S, Zlatanova S (2006) Integration of GIS and CAD at DBMS level. In: E. Fendel E, Rumor M (eds), Proceedings of UDMS'06 Aalborg, Denmark, TU Delft, pp 9.61-9.71

Rigaux P., Scholl M, Voisard A (2002) Spatial databases - with application to GIS. Morgan Kaufmann Publishers, San Francisco

Robinson J (1981) The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 10-18

Sakurai Y, Yoshikawa M, Uemura M, Kojima H (2002) Spatial indexing of high-dimensional data based on relative approximation. The International Journal on Very Large Data Bases, 11(2), pp. 93-108

Seeger B, Kriegel HP (1990) The Buddy tree: an ef[FB01?]cient and robust access method for spatial data base systems. In: Proc. 16th International Conference on Very Large Data Bases (VLDB), pp. 590-601

Vebree E, Zlatanova S (2004) 3D-modeling with respect to boundary representations within geo-DBMS. GISt report No.29, TU Delft

Wang F (1991) Relational-linear quadtree approach for two-dimensional spatial representation and manipulation. IEEE Trans. on Knowledge and Data Engineering, 3(1):118-122

Weber R, Schek HJ, Blott S (1998) A quantitative analysis and performance study for similarity-search methods in high dimensional spaces. In: Proc. 24th International Conference on Very Large Data Bases (VLDB), pp. 194-205

White DA, Jain R (1996) Similarity Indexing with the SS-tree. In: Proc. IEEE 12th International Conference on Data Engineering, pp. 516-523

Zlatanova S (2000) 3D GIS for urban development. PhD thesis, ITC, The Netherlands