Chapter 17
Implementation alternatives for an integrated 3D Information Model

Ludvig Emgård[1,2] and Sisi Zlatanova[1]

Abstract

The 3DIM (3D Integrated Model) is an information model under development which intends to integrate geographic features on the earth surface as well as above and below the earth surface into a common semantic-geometric model. We present and discuss two alternative implementations of the information model for DBMS. In the first alternative semantics are separated from geometry and organized into two table groups while in the second alternative semantic tables incorporates the geometry of the objects.

17.1 Introduction

Semantic models describing geographic features are common in many fields. Geo-scientists, geologists, constructors, architects and urban planners have been developing various semantic models to be able to better define objects, their representations and important relationships, which might be of importance for a particular application. However, generic semantic models that include a broad range of geographic features without emphasis on a specific application exist mostly in international or national GIS standards or ontology research. For example, the INSPIRE initiative (INSPIRE 2007) deals with harmonization of topographic features, the North American Data Model (NADM 2004) is focused on geological features while the CONGOO (Pantazis,1997), Towntology project (Caglioni 2006) and CityGML (Gröger et. al. 2006) concentrate on city environments. Although not related to a spe-

[1]Delft University of Technology, OTB, section GIS Technology,
Jaffalaan 9, 2628 BX the Netherlands
[2]SWECO Position AB, Sweden
`ludvig.emgard@sweco.se, s.zlatanova@tudelft.nl`

cific application, these frameworks focus on a set of real-world features (e.g. above, on the surface or under surface). Furthermore, semantic models hardly discuss spatial representations of semantic features. Those dealing with the spatial aspect consider only 2D geometries in 2D space. The most extensive contribution in semantic spatial models focusing on three-dimensional representations of real world is found in the information model of CityGML. The information model takes care of the semantic respectively thematic properties, taxonomies and aggregations of Digital Terrain Model (DTM), sites (including buildings, bridges, tunnels, etc.), vegetation, water bodies, transportation facilities, and city furniture. The semantic part of the model is complemented with geometry corresponding to the Simple Feature Specifications (Herring 2001). Special focus is put on the building features, which are represented in five levels of detail (LOD). The current version of CityGML does not include underground features, however.

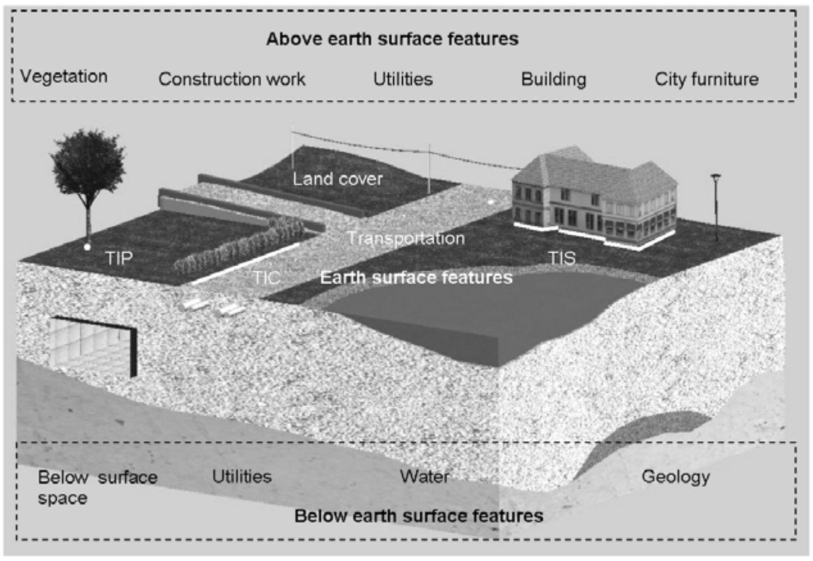

Fig. 17.1 3DIM subdivision of real-world features

The section for GIS technology at the Delft University of Technology is currently developing a 3D Information Model (3DIM), which intends to integrate features on surface, above and below the surface (Figure 1). We have presented an initial conceptual model in a previous publication (Emgård & Zlatanova, 2007). This article concentrates on the implementation of 3DIM in a DBMS. In the next section we briefly outline the major concepts of the information model. The third section gives a short overview on possibilities for implementing 3D geometry in a DBMS. We present and compare two alternatives for DBMS implementation of the conceptual model. The last section concludes the paper with a discussion on advantages and disadvantages of the two alternatives and outlines future research.

17.2 3D Integrated Model concept

The 3D integrated information model is intended to provide a generic model for 3D environments, which can be used by different domains, but being less specific and avoiding semantics and attributes that are very specific for a certain domain (and thus not of interest for many applications).

3DIM is intended to be used as a data model and contains thematic semantics and mapping to geometry data types for all man-made and natural real-world features on the surface, above and bellow (Figure 1). The model adopts several concepts presented in CityGML but also introduces stricter general rules as follows:

- The features are classified into above surface, integrated in surface and below surface
- The earth surface is fully partitioned. One part of the surface can only be occupied by one feature. Fictional features (Zlatanova 2000, Billen & Zlatanova, 2003) such as thematic land use can also be additionally attached i.e. residential or industrial area or another administrative attribute (not elaborated in the current version of the model).
- The surface accommodates all the intersections (touch) between the features above and below the surface and the surface itself. The idea of *TerrainIntersectionCurves (CityGML)* is therefore extended to include terrain intersection *surfaces, curves* and *points.*

The top-level classes are subdivided as follows (Figure 2):

- The Earth surface is represented by a fully partitioned surface consisting of *Transportation*, *Landcover* or *TerrainIntersectionSurface.* The first two classes represent objects on the earth surface. The *TerrainIntersectionSurface* is a special type of class since it represents the intersections of objects above and below the surface with the terrain.
- The above ground features are classified into *Building, Vegetation, City Furniture, AboveSurfaceUtilitiy and ConstructionWork.*
- The underground features are classified into *BelowSurfaceSpace, Geology, Water* and *BelowSurfaceUtilitiy*

The class *Building* is entirely adopted from the CityGML information model while *ConstructionWork*, *Geology*, *BelowSurfaceSpace*, *AboveSurfaceUtility and BelowSurfaceUtility* are new developments within this model.

The spatial extent of features may be defined either by a geometry or a topology model. In this paper we concentrate strictly on the geometry model.

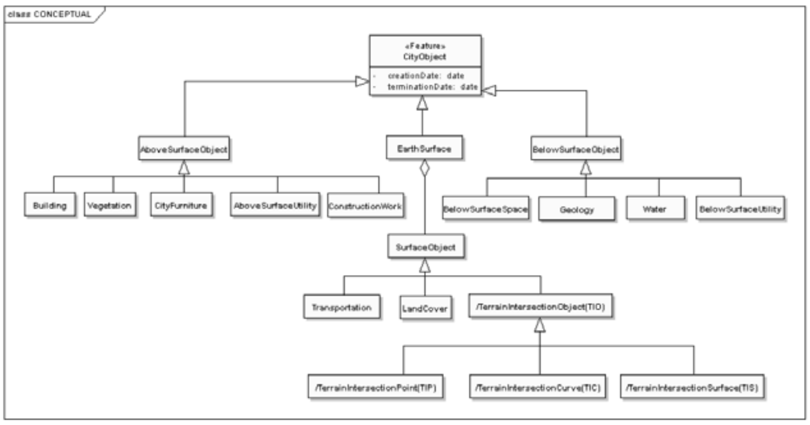

Fig. 17.2 UML class diagram of top level feature hierarchy in the 3DIM

The possible geometry data types in the model are those available in a DBMS, i.e. point, curve, surface and solid. These are to be represented by simple geometries as described in the Spatial Schema (Herring 2001) and Simple Feature Specifications for SQL (OGC 1999). This means that volumetric objects can be represented only by tetrahedrons and polyhedrons. Freeform curves and surfaces, Constructive Solid Geometry solids, and other complex geometry representations are currently not discussed in the model.

We believe that the subdivision of real-world objects into three groups is defendable. The features bellow and above the surface have rather distinct nature, i.e. geological features are generally continues phenomena, while features above the surface have crisp boundaries. Therefore the modelling approaches differ. Features below the earth surface are preferable modelled in a full partition of space while above ground the features are embedded in the 3D space (i.e. air is not commonly modelled). The features on the earth surface are most important since they allow establishing relationships between all features and considering them in their integrity. Traditionally, the earth surface has also been central in 2D maps, since map features are in most of the cases projections of above- and below- surface features on the earth surface. Very often these projections have been represented on separate 2D maps, i.e. topographic maps (for above and on the surface features) and geologic maps (for below surface features).

The strong subdivision of the objects into above, below and earth surface objects also brings disadvantages. Using the earth surface as a separator may increase the amount of geometric features describing the surface model. In addition, the fact that 3DIM integrates features from many different domains makes it much richer of geometry than a traditional map created for one purpose and thus more complex.

17.3 LOD (Level of Detail)

A very important concept in 3DIM is the LOD. In traditional 2D maps scale has been the factor defining the features on the map and for 3D environments the concept of level of detail somehow replaces the concept of scale. The 3DIM includes five levels of detail (LOD0-4), which allow geometry of features to be represented with different accuracy and detail. Features represented in LOD0 use representations that correspond to earth surface features. This mean that above and underground objects are given with their corresponding terrain intersection point, curves and surfaces. Practically, LOD0 is a 2.5 D map and can be integrated with available 2D maps if the z-value is not considered.

Features represented in LOD1 are created by simple geometric shapes (e.g. box models for buildings) consisting of surfaces, some of which are earth surfaces (see Figure 3). The polygons that constitute the boundary of the features are not semantically classified. For example, the polygons constituting a building are not classified into wall-polygons and roof-polygons. LOD2 is more detailed as it includes textures for geometric features and allows semantic classification. LOD3 contains more detailed surface geometries and texture is compulsory for each feature polygon. LOD3 contains the highest resolution that is available for the outside representation of features. LOD4 adds a different type of resolution, i.e. it handles indoor environments of two classes *Building* and *BelowSurfaceSpace*.

The concept of LOD in 3DIM has been adopted from CityGML and the newly introduced features are incorporated in the concept. The *ConstructionWork* and *BelowSurfaceSpace* features are similar to *Building* and therefore use the same definition of LOD. LOD1 describes a simple extruded representation, LOD2 a textured representation where individual polygons are classified, LOD3 a detailed geometric representation and LOD4 indoor environments.

The *AboveSurfaceUtility* and *BelowSurfaceUtility* features are represented by curves and points in LOD0-4. In higher LODs (2-4) the points are replaced for visualization by symbolic surface representations and lines are replaced by the necessary shape. In addition, larger utility objects are that can not be replaced by symbols are stored using surfaces in LOD1-4 The *Geology* feature is represented by borehole TIP points describing observations on the surface in LOD0 as well as a defined borehole datatype to express the observations of the borehole below the surface. In higher LODs the *Geology* features are represented by surfaces and solids.

The point and curve representation in LOD0 is in 3DIM also applied for *Vegetation* and *CityFurniture* features using the terrain intersection concept. The earth surface features *Transportation* and *LandCover* are presented by surface geometries only in all LODs.

17.4 Rules

3DIM also contains a set of rules as defined below:

1. A semantic feature must have a geometric representation. The geometry of a feature must be defined before (or simultaneously) with the semantic feature it describes.
2. A semantic feature can have only one geometry representation with respect to a LOD.
3. Texture images, color coding and symbols that are used for visualization of features must be created before (or simultaneously) they are referenced by a feature.
4. The earth surface parts *TIS*, *LandCover* and *Transportation* must together form a fully partitioned surface.
5. If a terrain intersection point, curve or surface is represented, a corresponding geometry must exist for the same feature.
6. A surface geometry or a combination of earth surface geometry and surface geometry must exist for a feature that is defined as a solid
7. Surfaces describing the exterior boundary of a building in LOD1, LOD2 or LOD3 and corresponding earth surface must be specified for each semantic building feature. If an earth surface is defined in one LOD the corresponding surface object for the LOD must also be specified.
8. The relation *TerrainIntersectionSurface* between the *Geology* feature and the earth surface may only exist when the *Geology* feature is to be seen in the open, for instance a mountain outcrop or a beach

More details about the conceptual model can be found in Emgård and Zlatanova (2007).

17.5 3D management of geometry in DBMS

Since the mid-nineties, several solutions for DBMS storage of 3D city models have been described in the literature i.e. (Kofler, 1998, Köninger & Bartel 1998, Stoter & Zlatanova 2003, Coors 2003). Köninger and Bartel (1998) presented a DBMS schema for 3D city models based on boundary representation in three levels of details using vertex index arrays to represent faces also including image mappings to texture images stored as BLOBs. Other examples can be found in Coors (2003) and Stoter & Zlatanova 2003 where buildings are stored as sets of faces. Within geology, 3D features are represented as 2.5D surfaces and 3D volumetric primitives for storage of stratum boundary surfaces, folded strata, ore bodies etc. in an object oriented DBMS (Breuning & Zlatanova 2006).

However, natively supported 3D data types within DBMS were not developed until recently. Prototype representations of polyhedron are reported by

Arens et. al. (2005) and of tetrahedron by Penninga et. al. (2006). Commercial support of 3D data types and operations are also expected to be available shortly (Oracle 2007).

At the moment, data types are restricted to point, curves and polygons with 3D coordinates. With some exceptions (e.g. PostGIS) the operations on these data types are 2D dimensional (Zlatanova & Stoter 2006).

Generally, within 3D database research, emphasis has been mostly given to topology and geometry and less work is completed on mapping between thematic semantics and 3D geometry. Some examples of DBMS implementation of a semantic model with 3D geometry are the 3D-geodatenbank Berlin (Plümer et. al. 2007), based on CityGML and the GeoBase21 (Haist & Coors 2005).

17.6 Implementations

Since most DBMSs follow the Simple feature specification for SQL (OGC 1999), the initial implementation alternatives we present are restricted to usage of simple features: point, curves and polygons. Given the simple feature types, surfaces and solids are created from a collection of polygons or a multi-polygon data type. Solids are in addition expressed by solid data type, assuming commercial DBMS will be able to maintain solids soon. TINs are expressed by multi-polygons containing only triangular polygons. Texture is either mapped to each polygon in a collection of polygons or draped over a surface consisting of a multi-polygon geometry.

The following thematic semantic features are selected for implementation in the database model: In general, each semantic class as given in Figure 2 is

Above surface features	**Earth surface features**	**Below surface features**
Building	Transportation	Geology
Construction Work	LandCover	BelowSurfaceSpace
Vegetation		Utility
CityFurniture		Water

implemented as a table and the attributes correspond to a column in a table. The superclass *TerrainIntersectionObject* is not implemented as a table, since it is not a feature class. *AboveSurfaceUtility* and *BelowSurfaceUtility* are merged in one table *Utility* due to the similar properties. An attribute shows whether the *Utility* is located above or below ground. If a utility feature is intersecting the earth surface, the feature is split into two geometries and a *TerrainIntersectionPoint* appears on the earth surface.

Given the chosen semantic entities, two alternatives of implementation are described. In the first alternative the common geometric representations of all entities are identified and created as separate tables. The tables containing the thematic semantic objects are linked to the geometric tables. In the second alternative the thematic semantic tables integrate the geometries. Geometric tables are not shared except data types describing symbols and textures, which features have in common.

Both suggested implementations of the 3D Information Model include an extended amount of semantic features. It should be noticed that some solutions are adopted from the Berlin model because (as mentioned above) some features are adopted from the CityGML.

The two implementation approaches presented here are intended for Oracle Spatial and its object-relational data model. Therefore data types as given by SDO_GEOMETRY and ORDSYS.ORDIMAGE are used in the descriptions.

17.6.1 Implementation alternative I

The first implementation alternative strictly separates semantics from geometry into two table groups. The common geometries are organised in four relational tables (point, curve, surface, earth surface) corresponding to simple geometry data types (point, curve, surface), one compound table giving maintenance of solids and supplementary tables for maintaining textures (Figure 3). The semantic entities are modelled as separate tables each referring to a semantic class and linked to tables containing the geometry.

The geometry tables are divided into point, curve, polygon, multi-polygon, solid and texture tables where the SURFACE GEOMETRY table represents polygons, the EARTH SURFACE table multi-polygons and the COMPUND GEOMETRY table solids. The SURFACE GEOMETRY tables contain polygons with texture or colour that is defined on one or both sides of the polygon while the EARTH SURFACE GEOMETRY table contains multi-polygons that are textured with a draped image or a repeated draped image. In this way a simple colour mapping, a low resolution ortho photo or a high resolution façade texture can be used depending on the semantic feature class. Since two textures can be referenced by a SURFACE GEOMETRY polygon, two relations exist between the SURFACE GEOMETRY table and the TEXTURE IMAGE table. A solid representation of the feature is defined in the COMPUND GEOMETRY table that is referring to a collection of SURFACE GEOMETRY polygons and EARTH SURFACE GEOMETRY multi-polygons. A surface from the SURFACE GEOMETRY table can appear only in one solid. For example, in the case of a building with a common surface the surface is stored twice in the SURFACE GEOMETRY table. In contrast, a surface from EARTH SURFACE GEOMETRY appears only once in the table. A surface from the EARTH SURFACE GEOMERY table can be a part of a feature above or below the surface. When composing the COMPOUND GEOMETRY table (see below), the orientation of the surface has to

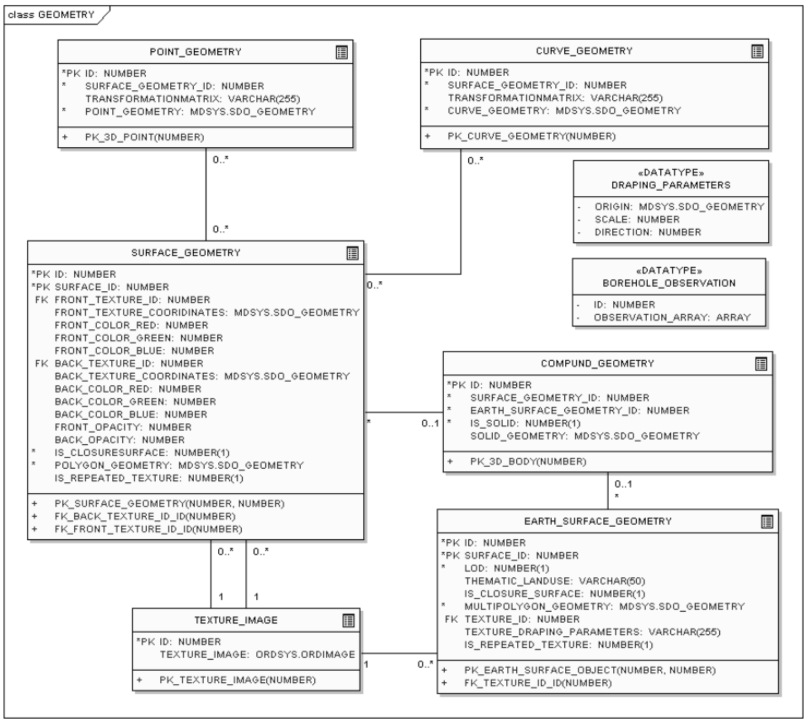
class GEOMETRY
POINT_GEOMETRY
*PK ID: NUMBER
* SURFACE_GEOMETRY_ID: NUMBER
TRANSFORMATIONMATRIX: VARCHAR(255)
* POINT_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_3D_POINT(NUMBER)
CURVE_GEOMETRY
*PK ID: NUMBER
* SURFACE_GEOMETRY_ID: NUMBER
TRANSFORMATIONMATRIX: VARCHAR(255)
* CURVE_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_CURVE_GEOMETRY(NUMBER)
0..*
0..*
0..*
0..*
«DATATYPE»
DRAPING_PARAMETERS
- ORIGIN: MDSYS.SDO_GEOMETRY
- SCALE: NUMBER
- DIRECTION: NUMBER
«DATATYPE»
BOREHOLE_OBSERVATION
- ID: NUMBER
- OBSERVATION_ARRAY: ARRAY
SURFACE_GEOMETRY
*PK ID: NUMBER
*PK SURFACE_ID: NUMBER
FK FRONT_TEXTURE_ID: NUMBER
FRONT_TEXTURE_COORIDINATES: MDSYS.SDO_GEOMETRY
FRONT_COLOR_RED: NUMBER
FRONT_COLOR_GREEN: NUMBER
FRONT_COLOR_BLUE: NUMBER
FK BACK_TEXTURE_ID: NUMBER
BACK_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
BACK_COLOR_RED: NUMBER
BACK_COLOR_GREEN: NUMBER
BACK_COLOR_BLUE: NUMBER
FRONT_OPACITY: NUMBER
BACK_OPACITY: NUMBER
* IS_CLOSURESURFACE: NUMBER(1)
* POLYGON_GEOMETRY: MDSYS.SDO_GEOMETRY
IS_REPEATED_TEXTURE: NUMBER(1)
+ PK_SURFACE_GEOMETRY(NUMBER, NUMBER)
+ FK_BACK_TEXTURE_ID_ID(NUMBER)
+ FK_FRONT_TEXTURE_ID_ID(NUMBER)
COMPUND_GEOMETRY
*PK ID: NUMBER
* SURFACE_GEOMETRY_ID: NUMBER
* EARTH_SURFACE_GEOMETRY_ID: NUMBER
* IS_SOLID: NUMBER(1)
SOLID_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_3D_BODY(NUMBER)
*
0..1
0..1
*
EARTH_SURFACE_GEOMETRY
*PK ID: NUMBER
*PK SURFACE_ID: NUMBER
* LOD: NUMBER(1)
THEMATIC_LANDUSE: VARCHAR(50)
IS_CLOSURE_SURFACE: NUMBER(1)
* MULTIPOLYGON_GEOMETRY: MDSYS.SDO_GEOMETRY
FK TEXTURE_ID: NUMBER
TEXTURE_DRAPING_PARAMETERS: VARCHAR(255)
IS_REPEATED_TEXTURE: NUMBER(1)
+ PK_EARTH_SURFACE_OBJECT(NUMBER, NUMBER)
+ FK_TEXTURE_ID_ID(NUMBER)
0..*
0..*
1
1
TEXTURE_IMAGE
*PK ID: NUMBER
TEXTURE_IMAGE: ORDSYS.ORDIMAGE
+ PK_TEXTURE_IMAGE(NUMBER)
1
0..*

Fig. 17.3 Geometric table group containing all used geometric feature tables in implementation alternative I

be adapted to create a valid solid. This is to describe that for above features it has to be flipped.

The symbols (e.g. trees, bus stops, streetlights and utility elements) consist of a collection of intersecting or non-intersecting polygons coloured or textured to be placed at different locations in the scene. These locations are maintained in the point and the curve geometry tables. The SURFACE GEOMETRY table is used to store the surfaces of the symbols as well. The relation between a feature and a symbol is many-to-many and it is established by the IDs of the feature and the symbol. For example one point refers to a SURFACE GEOMETRY ID, which may consist of many polygons having different IDs. In this manner, symbols can be referenced by several points and the points can use symbols using several polygons. Symbols can be placed along a curve described in the CURVE GEOMETRY table if the distance between the symbols is specified in the semantic tables. Such cases are possible for *Vegetation* and *CityFurniture* semantic classes.

The COMPUND GEOMETRY table is included to be able to define and validate solids. This actually creates a relation between the SURFACE GE-

OMETRY and the EARTH SURFACE GEOMETRY. It should be noticed that the primary storage tables for the geometries of the features are still the SURFACE GEOMETRY and EARTH SURFACE GEOMETRY tables. The geometry stored in the COMPUND GEOMETRY table is a copy of the geometry but in a *solid* data type, which allows to perform validation and/or other 3D spatial operations (e.g. volume).

Image textures are stored using the ORDSYS.ORDIMAGE data type referred from the TEXTURE IMAGE table. Parameters for texture placement are included in each polygon of the surface geometry table and for each multipolygon in the earth surface object.

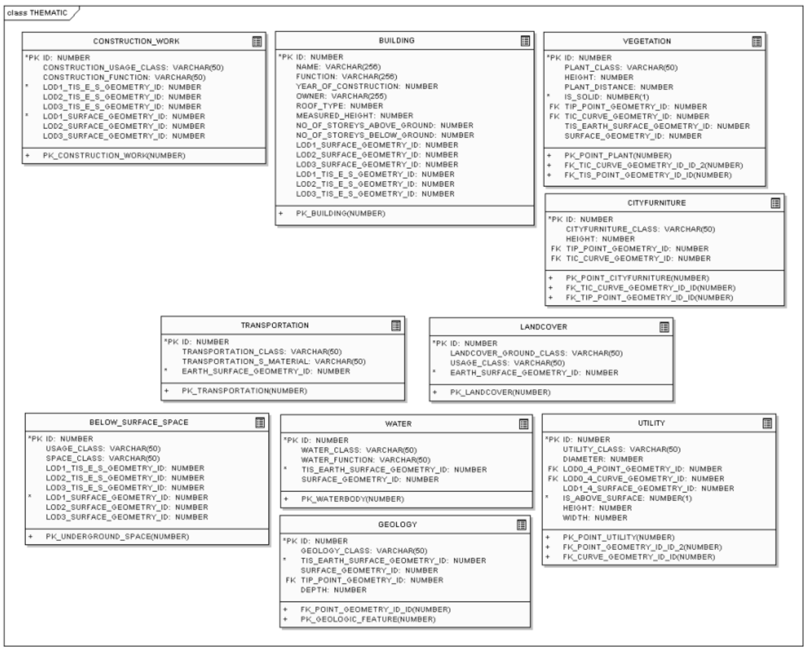
class THEMATIC
CONSTRUCTION_WORK
*PK ID: NUMBER
CONSTRUCTION_USAGE_CLASS: VARCHAR(50)
CONSTRUCTION_FUNCTION: VARCHAR(50)
* LOD1_TIS_E_S_GEOMETRY_ID: NUMBER
LOD2_TIS_E_S_GEOMETRY_ID: NUMBER
LOD3_TIS_E_S_GEOMETRY_ID: NUMBER
* LOD1_SURFACE_GEOMETRY_ID: NUMBER
LOD2_SURFACE_GEOMETRY_ID: NUMBER
LOD3_SURFACE_GEOMETRY_ID: NUMBER
+ PK_CONSTRUCTION_WORK(NUMBER)
BUILDING
*PK ID: NUMBER
NAME: VARCHAR(256)
FUNCTION: VARCHAR(256)
YEAR_OF_CONSTRUCTION: NUMBER
OWNER: VARCHAR(255)
ROOF_TYPE: NUMBER
MEASURED_HEIGHT: NUMBER
NO_OF_STOREYS_ABOVE_GROUND: NUMBER
NO_OF_STOREYS_BELOW_GROUND: NUMBER
LOD1_SURFACE_GEOMETRY_ID: NUMBER
LOD2_SURFACE_GEOMETRY_ID: NUMBER
LOD3_SURFACE_GEOMETRY_ID: NUMBER
LOD1_TIS_E_S_GEOMETRY_ID: NUMBER
LOD2_TIS_E_S_GEOMETRY_ID: NUMBER
LOD3_TIS_E_S_GEOMETRY_ID: NUMBER
+ PK_BUILDING(NUMBER)
VEGETATION
*PK ID: NUMBER
PLANT_CLASS: VARCHAR(50)
HEIGHT: NUMBER
PLANT_DISTANCE: NUMBER
* IS_SOLID: NUMBER(1)
FK TIP_POINT_GEOMETRY_ID: NUMBER
FK TIC_CURVE_GEOMETRY_ID: NUMBER
TIS_EARTH_SURFACE_GEOMETRY_ID: NUMBER
SURFACE_GEOMETRY_ID: NUMBER
+ PK_POINT_PLANT(NUMBER)
+ FK_TIC_CURVE_GEOMETRY_ID_ID_2(NUMBER)
+ FK_TIS_POINT_GEOMETRY_ID_ID(NUMBER)
CITYFURNITURE
*PK ID: NUMBER
CITYFURNITURE_CLASS: VARCHAR(50)
HEIGHT: NUMBER
FK TIP_POINT_GEOMETRY_ID: NUMBER
FK TIC_CURVE_GEOMETRY_ID: NUMBER
+ PK_POINT_CITYFURNITURE(NUMBER)
+ FK_TIC_CURVE_GEOMETRY_ID_ID(NUMBER)
+ FK_TIP_POINT_GEOMETRY_ID_ID(NUMBER)
TRANSPORTATION
*PK ID: NUMBER
TRANSPORTATION_CLASS: VARCHAR(50)
TRANSPORTATION_S_MATERIAL: VARCHAR(50)
* EARTH_SURFACE_GEOMETRY_ID: NUMBER
+ PK_TRANSPORTATION(NUMBER)
LANDCOVER
*PK ID: NUMBER
LANDCOVER_GROUND_CLASS: VARCHAR(50)
USAGE_CLASS: VARCHAR(50)
* EARTH_SURFACE_GEOMETRY_ID: NUMBER
+ PK_LANDCOVER(NUMBER)
BELOW_SURFACE_SPACE
*PK ID: NUMBER
USAGE_CLASS: VARCHAR(50)
SPACE_CLASS: VARCHAR(50)
LOD1_TIS_E_S_GEOMETRY_ID: NUMBER
LOD2_TIS_E_S_GEOMETRY_ID: NUMBER
LOD3_TIS_E_S_GEOMETRY_ID: NUMBER
* LOD1_SURFACE_GEOMETRY_ID: NUMBER
LOD2_SURFACE_GEOMETRY_ID: NUMBER
LOD3_SURFACE_GEOMETRY_ID: NUMBER
+ PK_UNDERGROUND_SPACE(NUMBER)
WATER
*PK ID: NUMBER
WATER_CLASS: VARCHAR(50)
WATER_FUNCTION: VARCHAR(50)
* TIS_EARTH_SURFACE_GEOMETRY_ID: NUMBER
SURFACE_GEOMETRY_ID: NUMBER
+ PK_WATERBODY(NUMBER)
GEOLOGY
*PK ID: NUMBER
GEOLOGY_CLASS: VARCHAR(50)
* TIS_EARTH_SURFACE_GEOMETRY_ID: NUMBER
SURFACE_GEOMETRY_ID: NUMBER
FK TIP_POINT_GEOMETRY_ID: NUMBER
DEPTH: NUMBER
+ FK_POINT_GEOMETRY_ID_ID(NUMBER)
+ PK_GEOLOGIC_FEATURE(NUMBER)
UTILITY
*PK ID: NUMBER
UTILITY_CLASS: VARCHAR(50)
DIAMETER: NUMBER
FK LOD0_4_POINT_GEOMETRY_ID: NUMBER
FK LOD0_4_CURVE_GEOMETRY_ID: NUMBER
LOD1_4_SURFACE_GEOMETRY_ID: NUMBER
* IS_ABOVE_SURFACE: NUMBER(1)
HEIGHT: NUMBER
WIDTH: NUMBER
+ PK_POINT_UTILITY(NUMBER)
+ FK_POINT_GEOMETRY_ID_ID_2(NUMBER)
+ FK_CURVE_GEOMETRY_ID_ID(NUMBER)

Fig. 17.4 Semantic table group containing above surface, earth surface and below surface feature tables in implementation alternative I

Each semantic table include relations to one or more of the geometry tables (Figure 4). BUILDING, CONSTRUCTION WORK, BELOW SURFACE SPACE and WATER tables are all related to both SURFACE GEOMETRY and EARTH SURFACE GEOMETRY tables. As mentioned above, features that belong to these classes can be represented by surfaces and solids. TRANSPORTATION and LAND COVER tables are typically surface types of features and can be related only to the EARTH SURFACE GEOMETRY table. The VEGETATION table is related to all geometric tables, since the

Vegetation can be represented by point, curve, surface and solid. The CITY FURNTIURE and UTILITY tables are related to point and curve geometry tables, as they can be points and curves. The UTILITY table is also related to the SURFACE GEOMETRY table since a utility feature can be a large object under ground connected to several other utility features and the geometry of the feature is too complex and individual to be replaced by a symbolic geometric feature.

A *Geology* feature that is not intersecting with the earth surface is stored as a solid referring only to the SURFACE GEOMETRY table. Faults and stratums are maintained as surfaces. Borehole geometries are stored using the POINT GEOMETRY table to store the TIP of the Borehole. In addition a data type borehole is created for storage of the observations in the borehole.

BUILDING, CONSTRUCTION WORK and BELOW SURFACE SPACE tables can have different surface models depending on the level of detail. Therefore each LOD is related to a set of polygons in the SURFACE GEOMETRY table. To fulfil the more extended concept of LOD2-3 for buildings our implementation must be complemented with the tables: THEMATIC SURFACE GEOMETRY, THEMATIC SURFACE and BUILDING OPENING as described in the database schemas of 3D-Geodatenbank Berlin (Plümer et. al. 2007).

Implementation of rules

The rules for the conceptual model are implemented as foreign keys and user specifications, which may be further implemented as methods and triggers. Rules 1 and 3 are implemented as foreign keys between some of the defined features i.e. between semantic tables and the POINT GEOMETRY and CURVE GEOMETRY tables. With the foreign keys defined, the given TIP POINT GEOMETRY ID of a semantic point feature i.e. a streetlight must also be defined as an ID in the POINT GEOMETRY table. This fulfils rule 1 defining that the geometry of the feature has to be created before the semantic information can be added. Similar constraints are also applied for the tables containing *TerrainIntersectionObject*. A foreign key is also added to control that a texture that is referenced exists in the TEXTURE IMAGE table (rule 3).

Since the *TerrainIntersectionSurface* identifier ID for the semantic tables are referring to one or several rows in the EARTH SURFACE GEOMETRY table, a foreign key can not be created for EARTH SURFACE GEOMETRY or for SURFACE GEOMETRY (rule1). This constraint is therefore implemented as a user rule instead of a database constraint. Rules 2 and 4-8 are not implemented as constraints in the database.

17.6.2 Implementation alternative II

The second implementation alternative is based on complete semantic subdivision where the geometry columns are integrated in the semantic tables.

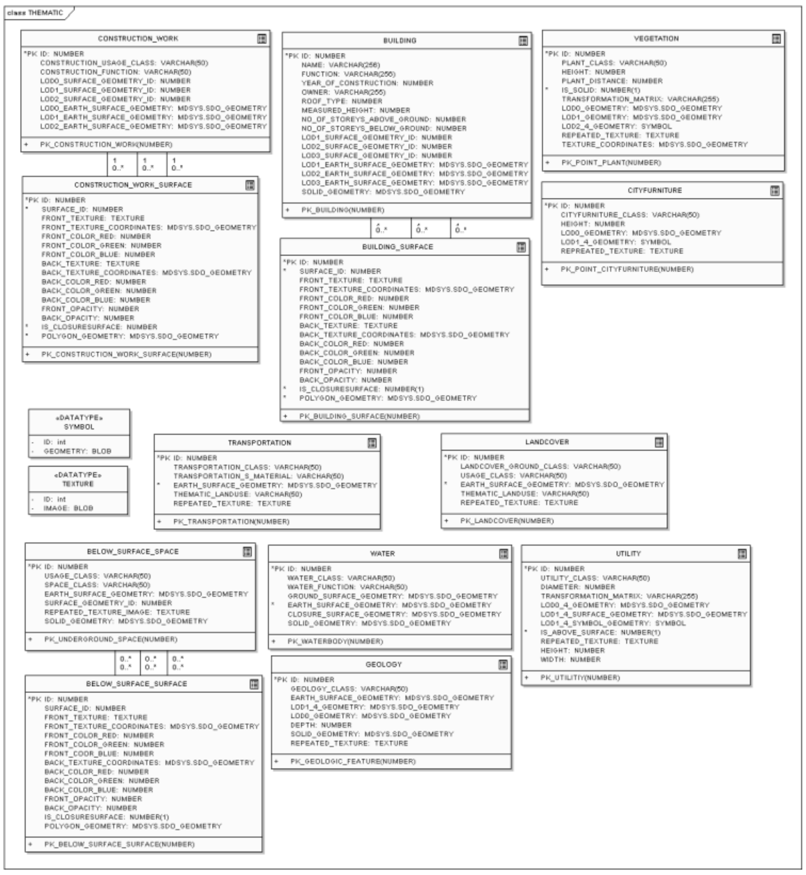
class THEMATIC
CONSTRUCTION_WORK
*PK ID: NUMBER
CONSTRUCTION_USAGE_CLASS: VARCHAR(50)
CONSTRUCTION_FUNCTION: VARCHAR(50)
LOD0_SURFACE_GEOMETRY_ID: NUMBER
LOD1_SURFACE_GEOMETRY_ID: NUMBER
LOD2_SURFACE_GEOMETRY_ID: NUMBER
LOD0_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD2_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_CONSTRUCTION_WORK(NUMBER)
1 0..*
1 0..*
1 0..*
CONSTRUCTION_WORK_SURFACE
*PK ID: NUMBER
* SURFACE_ID: NUMBER
FRONT_TEXTURE: TEXTURE
FRONT_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
FRONT_COLOR_RED: NUMBER
FRONT_COLOR_GREEN: NUMBER
FRONT_COLOR_BLUE: NUMBER
BACK_TEXTURE: TEXTURE
BACK_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
BACK_COLOR_RED: NUMBER
BACK_COLOR_GREEN: NUMBER
BACK_COLOR_BLUE: NUMBER
FRONT_OPACITY: NUMBER
BACK_OPACITY: NUMBER
* IS_CLOSURESURFACE: NUMBER
* POLYGON_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_CONSTRUCTION_WORK_SURFACE(NUMBER)
BUILDING
*PK ID: NUMBER
NAME: VARCHAR(256)
FUNCTION: VARCHAR(256)
YEAR_OF_CONSTRUCTION: NUMBER
OWNER: VARCHAR(255)
ROOF_TYPE: NUMBER
MEASURED_HEIGHT: NUMBER
NO_OF_STOREYS_ABOVE_GROUND: NUMBER
NO_OF_STOREYS_BELOW_GROUND: NUMBER
LOD1_SURFACE_GEOMETRY_ID: NUMBER
LOD2_SURFACE_GEOMETRY_ID: NUMBER
LOD3_SURFACE_GEOMETRY_ID: NUMBER
LOD1_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD2_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD3_EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
SOLID_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_BUILDING(NUMBER)
0..*
0..*
0..*
BUILDING_SURFACE
*PK ID: NUMBER
* SURFACE_ID: NUMBER
FRONT_TEXTURE: TEXTURE
FRONT_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
FRONT_COLOR_RED: NUMBER
FRONT_COLOR_GREEN: NUMBER
FRONT_COLOR_BLUE: NUMBER
BACK_TEXTURE: TEXTURE
BACK_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
BACK_COLOR_RED: NUMBER
BACK_COLOR_GREEN: NUMBER
BACK_COLOR_BLUE: NUMBER
FRONT_OPACITY: NUMBER
BACK_OPACITY: NUMBER
* IS_CLOSURESURFACE: NUMBER(1)
* POLYGON_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_BUILDING_SURFACE(NUMBER)
VEGETATION
*PK ID: NUMBER
PLANT_CLASS: VARCHAR(50)
HEIGHT: NUMBER
PLANT_DISTANCE: NUMBER
* IS_SOLID: NUMBER(1)
TRANSFORMATION_MATRIX: VARCHAR(255)
LOD0_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD2_4_GEOMETRY: SYMBOL
REPEATED_TEXTURE: TEXTURE
TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
+ PK_POINT_PLANT(NUMBER)
CITYFURNITURE
*PK ID: NUMBER
CITYFURNITURE_CLASS: VARCHAR(50)
HEIGHT: NUMBER
LOD0_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_4_GEOMETRY: SYMBOL
REPREATED_TEXTURE: TEXTURE
+ PK_POINT_CITYFURNITURE(NUMBER)
«DATATYPE» SYMBOL
- ID: int
- GEOMETRY: BLOB
«DATATYPE» TEXTURE
- ID: int
- IMAGE: BLOB
TRANSPORTATION
*PK ID: NUMBER
TRANSPORTATION_CLASS: VARCHAR(50)
TRANSPORTATION_S_MATERIAL: VARCHAR(50)
* EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
THEMATIC_LANDUSE: VARCHAR(50)
REPEATED_TEXTURE: TEXTURE
+ PK_TRANSPORTATION(NUMBER)
LANDCOVER
*PK ID: NUMBER
LANDCOVER_GROUND_CLASS: VARCHAR(50)
USAGE_CLASS: VARCHAR(50)
* EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
THEMATIC_LANDUSE: VARCHAR(50)
REPEATED_TEXTURE: TEXTURE
+ PK_LANDCOVER(NUMBER)
BELOW_SURFACE_SPACE
*PK ID: NUMBER
USAGE_CLASS: VARCHAR(50)
SPACE_CLASS: VARCHAR(50)
EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
SURFACE_GEOMETRY_ID: NUMBER
REPEATED_TEXTURE_IMAGE: TEXTURE
SOLID_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_UNDERGROUND_SPACE(NUMBER)
0..* 0..*
0..* 0..*
0..* 0..*
BELOW_SURFACE_SURFACE
*PK ID: NUMBER
SURFACE_ID: NUMBER
FRONT_TEXTURE: TEXTURE
FRONT_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
FRONT_COLOR_RED: NUMBER
FRONT_COLOR_GREEN: NUMBER
FRONT_COOR_BLUE: NUMBER
BACK_TEXTURE_COORDINATES: MDSYS.SDO_GEOMETRY
BACK_COLOR_RED: NUMBER
BACK_COLOR_GREEN: NUMBER
BACK_COLOR_BLUE: NUMBER
FRONT_OPACITY: NUMBER
BACK_OPACITY: NUMBER
IS_CLOSURESURFACE: NUMBER(1)
POLYGON_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_BELOW_SURFACE_SURFACE(NUMBER)
WATER
*PK ID: NUMBER
WATER_CLASS: VARCHAR(50)
WATER_FUNCTION: VARCHAR(50)
GROUND_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
* EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
CLOSURE_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
SOLID_GEOMETRY: MDSYS.SDO_GEOMETRY
+ PK_WATERBODY(NUMBER)
GEOLOGY
*PK ID: NUMBER
GEOLOGY_CLASS: VARCHAR(50)
EARTH_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_4_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD0_GEOMETRY: MDSYS.SDO_GEOMETRY
DEPTH: NUMBER
SOLID_GEOMETRY: MDSYS.SDO_GEOMETRY
REPEATED_TEXTURE: TEXTURE
+ PK_GEOLOGIC_FEATURE(NUMBER)
UTILITY
*PK ID: NUMBER
UTILITY_CLASS: VARCHAR(50)
DIAMETER: NUMBER
TRANSFORMATION_MATRIX: VARCHAR(255)
LOD0_4_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_4_SURFACE_GEOMETRY: MDSYS.SDO_GEOMETRY
LOD1_4_SYMBOL_GEOMETRY: SYMBOL
* IS_ABOVE_SURFACE: NUMBER(1)
REPEATED_TEXTURE: TEXTURE
HEIGHT: NUMBER
WIDTH: NUMBER
+ PK_UTILITIY(NUMBER)

Fig. 17.5 Table schema representing all tables in implementation alternative II

In this implementation, the geometry column may contain different geometries. Specific tables are defined for geometry only when explicitly required. For example, CONSTRUCTION WORK, BELOW SURFACE SPACE and BUILDING tables require individual textures for each surface in higher LOD. Therefore, the features can not be implemented using the multi-polygon data type. Instead, three corresponding surface tables where created to handle the texturing where the CONSTRUCTION WORK SURFACE table, the

BUILDING SURFACE table and BELOW SURFACE SPACE table follow the same concept as the surface geometry table in implementation Alternative I storing one polygon in each row. The tables also have three relations to the geometry table, one for each LOD. In the other cases geometries are stored as point, curve or multi-polygon features as an attribute in the table. The earth surface features do not require individual texturing for each polygon and are therefore modelled as attributes within the feature they are part of. Symbols and textures are defined in new data types storing images and geometry as BLOBs.

Implementation of rules

No geometric constraints are implemented in alternative II since the geometry is mostly organised as an attribute to a particular semantic feature. The only exceptions are the BUILDING, CONSTRUCTION WORKS and BELOW SURFACE SPACE tables, which refer to separate tables for the geometry surfaces. This approach is used to allow texture mapping per each individual polygon. An alternative option will be to create a special data type, which would encompass geometry and texture. This alternative, however, complicates the spatial index.

The rules 1- 3 can be fulfilled by developing functions and/or triggers. While the triggers can check the validity during data import (Louwsma et al 2006), functions control the rules after data are loaded. Similarly to the first alternative, rules 4-8 are not implemented.

17.6.3 Comparison of implementations

The two proposed implementations can be strictly compared only after testing with different data sets. Here only some initial expectations are discussed. In general, from a user's point of view, Alternative II has advantages due to the clear and simpler structure (where all features are integrated in the semantic tables). However, Alternative I provides more robust database management, since some of the consistency check can be performed by the DBMS.

Loading data into the database is more straightforward for alternative II, since at least two tables have to be filled for each feature in alternative I. In Alternative I the geometric feature have to be created before the semantic feature while in Alternative II they can be loaded simultaneously with the insertion of the semantics. Therefore we expect better performance (faster import) in the second alternative.

Alternative II may cause more redundancy in geometric storage. A surface cannot be shared by two features in some cases as in Alternative I. For example, a geologic body that is touching another body shares several polygons. Alternative I allow the user to make reference to the same polygons (surfaces) from the two different geological features. This concept is however not allowed for buildings. In Alternative II, surfaces will be stored per feature and therefore twice. However, Alternative II allows for more elaborated use of data types (solid, multi-polygon) at lower LOD when, no texture mapping is applied.

Operations like edit and update would be faster in one or the other Alternatives with respect to the attributes to be changed. If the changes are related to features, the second alternative will be faster. Changes in geometry will benefit from alternative I. It should be noticed that also the implementation of Rule 4 (full partitioning of earth surface) will be rather complicated for the Alternative II.

A query based on semantics of a single feature is less complicated in Alternative II, while always a join of tables is required in Alternative II. In addition less geometry is traversed since all geometries of one class are in the same table. For example a query of all utilities defined as points is less complex in Alternative II. On the other hand a pure geometric selection is less complex in Alternative I since the geometry can be found in a single table depending on geometry type (point, curve and surface). For example, a query of all features represented by surfaces within a specified area will be much faster in Alternative I. Similarly, spatial queries investigating relationships between objects in many cases would be simpler in Alternative I. For example, the query 'find all the neighbouring features of Building 77' can be performed on only the geometry tables. Alternative II would always require a traverse of all the semantic tables.

In Alternative I symbolic representations are stored using individual textured polygons while in Alternative II the symbols are expressed by BLOBs. To assemble the symbols from individual polygons in general has a negative effect with respect to performance compared to symbols stored in BLOBs. This yet has to be tested to be proved true.

17.7 Conclusion and future work

We have presented two alternative implementations of the 3D Integrated Model and have discussed advantages and disadvantages. The first approach is more beneficial for geometric queries while the other one is more promising concerning semantic queries. What implementation to choose is much dependent on the application or the purpose. Generally, most of the cases would require both semantic and geometric attributes. This means the query schema (order of querying) should be well-thought to achieve needed performance.

As mentioned above, the model is under development. Some of the classes in the model have to be further elaborated with respect to LOD, for example

Geology, *BelowSurfaceSpace* and *ConstructionWork*. The *Utility* attributes have to be extended with domain specific attributes.

The two alternatives will be tested in a case study of the Campus area of Delft University of Technology.

References

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

Billen R, Zlatanova S (2003) Conceptual issues in 3D Urban GIS, In: GIM International, Vol. 17, No. 1, January 2003, pp.33-35

Breuning M, Zlatanova S (2006) 3D Geo-DBMS, In: 3D large scale data integration: challenges and opportunities, CRC Press, Taylor&Francis Group, pp. 87-135

Caglioni M (2006) Ontologies of Urban Models, Technical report n°4, Short Term Scientific Mission Report, Urban Ontologies for an improved communication in urban civil engineering projects, Towntology Project 5p, `http://www.towntology.net/Documents/STSM-Caglioni.pdf` [last accessed 2007-08-27]

Coors V (2003) 3D-GIS in networking environments, Computers, Environment and Urban Systems, Volume 27, Number 4, July 2003, pp. 345-357(13)

Emgård L, Zlatanova S (2007) Design of an integrated 3D information model, in: Coors, Rumor, Fendel&Zlatanova UDMS Annual 2007, Taylor & Francis, 26th proceedings of UDMS, 10-12 October 2007, Stuttgart, (in press)

Gröger G, Kolbe T, Czerwinski A (2006) OpenGIS CityGML Implementation Specification. `http://www.citygml.org/docs/CityGML_Specification_0.3.0_OGC_06-057.pdf` [last accessed 2007-08-27]

Haist J, Coors V (2005) The W3DS-Interface of CityServer3D In: Proceedings of International Workshop on Next Generation 3D City Models 2005, Bonn pp.63-67

Herring J (2001): The OpenGIS Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), Version 5. OGC Document Number 01-101. `http://www.opengeospatial.org/standards/as` [last accessed 2007-

08-27]

INSPIRE, The European Parliament (2007) Directive of the European Parliament and of the Council establishing an Infrastructure for Spatial Information in the European Community (INSPIRE) (PE-CONS 3685/2006, 2004/0175 (COD) C6-0445/2006) http://www.europarl.europa.eu/oeil/FindByProcnum.do?lang=2&procnum=COD/2000/0175 [last accessed 2007-08-27]

Kofler M (1998) R-trees for the visualisation of large 3D GIS databases, PhD thesis, TU, Graz, Austria.

Köninger A, Bartel S (1998) 3D-GIS for Urban Purposes. GeoInformatica 2:1,pp 79-103 (1998)

Louwsma J, Zlatanova S, van Lammeren R, van Oosterom P (2006) Specifying and implementing constraints in GIS – with examples from a geo-virtual reality system, *GeoInformatica* 10 (2006): pp 531-550

NADM (2004) North American Geologic Map Data Model Steering Committee Conceptual Model 1.0—A conceptual model for geologic map information: U.S. Geological Survey Open-File Report 2004-1334, 58 p., accessed online at URL http://pubs.usgs.gov/of/2004/1334. Also published as Geo-logical Survey of Canada Open File 4737, 1 CD-ROM.

OGC 1999 OpenGIS Simple Features Specification For SQL, Revision 1.1 http://www.opengeospatial.org/standards/sfs [last accessed 2007-08-31]

Oracle (2007) Oracle Spatial 11g Planned features presentation material

Pantazis D (1997) CON.G.O.O. : A conceptual formalism for geographic database design. In Geographic Information Research, Bridging the Atlantic (London: Taylor & Francis), pp. 348-367.

Penninga, F., van Osteroom, P. & Kazar, B. 2006. A Tetrahedronized Irregular Network Based DBMS approach for 3D Topographic Data Modeling. *Progress in Spatial Data Handling*

Plümer L, Kolbe T, Gröger G, Schmittwilken, J & Stroh V (2007) 3D-Geodatenbank Berlin, Dokumentation V1.0. http://www.3dcitydb.org/index.php?id=259 [last accessed 2007-08-27]

Stoter J, Zlatanova S (2003)Visualisation and editing of 3D objects organised in a DBMS, J., In: Proceedings of the EuroSDR Com V. Workshop on Visualisation and Rendering, 22-24 January 2003, Enschede, The Netherlands, 16p

Zlatanova S, (2000) 3D GIS for urban development, PhD thesis, ISBN 90-6164-178-0, ICG, TUGraz, Austria, ITC publication 69, ISBN 90-6164-178-0

Zlatanova S, Stoter J (2006) The role of DBMS in the new generation GIS architecture, in: Rana&Sharma (eds.) Frontiers of Geographic Information Technology, Springer-Verlag, Berlin Heidelberg ISBN-1- 3-540-25685-7, pp. 155-180