3D Cadastres Best Practices, Chapter 4: 3D Spatial DBMS for 3D Cadastres

**Karel JANEČKA, Czech Republic, Sudarshan KARKI, Australia,
Peter VAN OOSTEROM, The Netherlands, Sisi ZLATANOVA, Australia,
Mohsen KALANTARI, Australia, and Tarun GHAWANA, India**

Key words: 3D Spatial Database Management System, 3D Cadastre, 3D Representation, 3D Spatial Indexing and Analysis

SUMMARY

Subdivision of land parcels in the vertical space has made it necessary for cadastral jurisdictions to manage cadastral objects both in 2D as well as 3D. Modern sensor and hardware capabilities for capture and utilisation of large point clouds is one of the major drivers to consider Spatial Database Management Systems (SDBMS) in 3D and organisations are still progressing towards it. 3D data models and their topological relationships are two of the important parts of 3D spatial data management. 3D spatial systems should enable data models that handle a large variety of 3D objects, perform automated data quality checks, search and analysis, rapid data dissemination, 3D rendering and visualisation with close linkages to standards. This chapter asserts that while there has been work done in defining 2D and 3D vector geometry in standards, it is still not sufficient for 3D cadastre purposes as 3D cadastral objects have a much more rigorous definition. The Land Administration Domain Model (LADM), which is an ISO Standard, addresses many of the issues in 3D representation and storage of 3D data in a database management system (DBMS). The chapter further discusses the various approaches to storing 3D data such as through voxels, or point cloud data type and elaborates on the characteristics of a 3D DBMS capable of storing 3D data. Approaches for spatial indexing to improve the fast access of data and the various available options for a 3D geographical database system are presented. Several spatial operations on and amongst 3D objects are illustrated with linkages to the current standards including the LADM. Next, construction of 3D topological and geometrical models based on standards and including their characteristics is discussed. Current 3D spatial database managements systems and their characteristics, including some comparison between selected DBMS including the hardware capabilities are elaborated in detail. Finally, the chapter proposes a 3D topology model based on Tetrahedron Network (TEN) synchronised with LADM specifications for 3D cadastral registration. This topological model utilises surveying boundaries to generate 3D cadastral objects with consistent topology and rapid query and management capabilities. The definition for validation of 3D solids also considers the automatic repair of invalid solids. Point cloud and TEN related data structures available in SDBMSs are also investigated to enable storage of non-spatial attributes so that database updates would store all spatial and attribute information directly inside the spatial database.

3D Cadastres Best Practices, Chapter 4: 3D Spatial DBMS for 3D Cadastres

**Karel JANEČKA, Czech Republic, Sudarshan KARKI, Australia,
Peter VAN OOSTEROM, The Netherlands, Sisi ZLATANOVA, Australia,
Mohsen KALANTARI, Australia, and Tarun GHAWANA, India**

1. INTRODUCTION

With the advancements in computing and spatial science based technologies, the generation and usage of 3D data is now possible with much ease than before.
Boss and Streilein (2014) observed four major technology and business drivers for 3D:

1. There are massive new sensor hardware capabilities, such as automated data capture and model creation on the sensor side, LIDAR with masses of point clouds and automated photogrammetric workflows and processes.
2. 3D visualisation has now come into the mainstream, but 3D analysis has not. But there is as yet no mass market with consumer-focused systems.
3. Managing 3D data in enterprise workflows with improved performance and scalability of existing workflows and bridging the gap between point cloud surveys, GIS, CAD, BIM. Traditional file handling moves to database management.
4. There is a necessity for 3D data, where 2D data is not sufficient to describe our world and the consumer expectation demands three dimensions, as we all live and act in a three dimensional environment.

For cadastral organizations, who traditionally describe their cadastral data in two dimensions and hold their information in 2D (often graphical) files, concepts for entering the third dimensions are not yet available, mainly due to the facts that (Boss and Streilein, 2014):

- 3D modelling is much more heterogeneous and complex compared to 2D modelling,
- Converting 2D data to 3D data on an operational level, with not just adding a Z-coordinate onto each planimetric pair of coordinates, is quite cumbersome and there is no ‘best’ solution obvious, as the existing datasets are usually quite specific,
- One has to migrate from simple data structures to complex data structures,
- One has to deal with the economic and sustainability issues of handling and storing high data volumes compared to (relatively) low data volumes in the current years, and
- User-friendly tools for 3D analysis are still missing.

The technologies for creating and using 3D models have matured over the past ten years. People are accustomed to use 3D technologies in their daily life, ranging from watching TV and movies in 3D, gaming and 3D printing to navigating through 3D maps. Still 3D technologies are not common to solve location-based issues: spatial planning is still mainly done based on 2D maps

and databases with geo-information that support location-related policies (like INSPIRE, building registers, land use plans, cadastral maps) are mainly 2D (Stoter et al., 2016).
In our contemporary social context, the development of land use has subdivided land parcels into three-dimensional (3D) spaces according to certain property rights, especially in metropolitan areas with dense population. This results in 3D parcels (ISO, 2012) above or below the land surface. In such circumstances, the local government needs to construct and manage 3D cadastral objects to be able to manage the development of real urban 3D spaces appropriately (Ying et al., 2015).
Constructing 3D data models and their topological relationship are two important parts of 3D cadastre (Ying et al., 2011). 3D Spatial Systems should then enable (Ravada et al., 2009):

- Data Model to handle a variety of 3D Objects
- Data quality control
- Geo-Referencing
- Comprehensive Location based search and Analysis
- Handling level of detail for seamless operation
- High Performance dissemination of 3D data
- Support High performance real-time 3D rendering
- Support for 3D Standards

This chapter addresses several topics: the different types of 3D spatial representations (vector, voxel and point cloud) (Section 2), 3D spatial indexing and clustering (Section 3), 3D geometries and 3D operations (Section 4), 3D topology structures (Section 5), from theory to practice (Section 6), recent development of spatial databases (Section 7), analysis, what is available and what is needed (Section 8). The chapter ends with conclusions.

2. 3D SPATIAL REPRESENTATION

2.1 Vector representation

Practically most of the work on geometry models has been completed by the Open Geospatial Consortium Inc. (OGC, formerly the Open GIS Consortium) (Lee and Zlatanova, 2008). ISO has also independently from OGC developed ISO/TC 211 19107:2003 (ISO, 2003), Geographic information – Spatial Schema (Hering, 2001).
The OGC Implementation Standard for Geographic information – Simple feature access – Part 1: Common architecture (OGC, 2011) describes the common architecture for simple feature geometry. The simple feature geometry object model is Distributed Computing Platform neutral and uses unified modelling language (UML) notations. The base Geometry class has subclasses for Point, Curve, Surface and GeometryCollection. Each geometric object is associated with a Spatial Reference System, which describes the coordinate space in which the geometric object is defined. This part of OGC Simple feature access implements a profile of the spatial schema described in ISO 19107:2003, Geographic information – Spatial schema.
The OGC Implementation Standard for Geographic information – Simple feature access – Part 2: SQL option (OGC, 2010) defines a standard Structured Query Language (SQL) scheme that supports storage, retrieval, query and update of feature collections via the SQL Call-Level

Interface (SQL/CLI). A feature has both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on two-or-fewer dimensional geometric (point, curve and surface) entities in 2 or 3 spatial dimensions with linear or planar interpolation between vertices.
Kazar et al. (2008) and Verbree and Si (2008) observe that the ISO 19107 solids are not sufficient for 3D cadastral applications: the ISO 19107 solid is a simple solid whose shell is not allowed to touch (it needs to be a 2-manifold).
For proper representation of 3D cadastre, adequate 3D geometries are required. Surveying data can be acquired by the surveyors or the engineers, thus the creation and submission of 3D volumetric objects are the key phases in a 3D cadastre system. However, what are acceptable (valid) 3D cadastral object representations and how to create their 3D geometries (even the non-2-manifold geometries) are still challenges (Van Oosterom 2013). The non-manifold 3D representations (self-touching in edge or node; see Figure 1) are not well supported by current GIS, CAD, and DBMS software or by generic ISO standards such as ISO 19107 (Van Oosterom 2013).

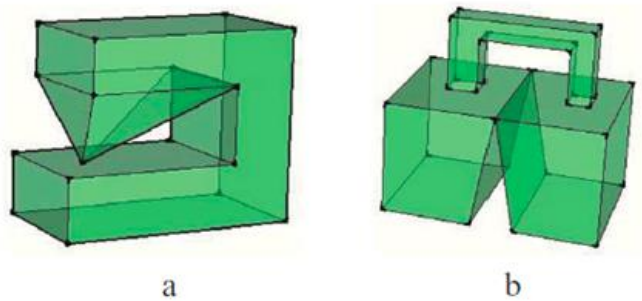

Figure 1. Solids with non-manifold conditions: (a) point non-manifold condition; and (b) edge non-manifold condition (Ying et al., 2015)

Kazar et al. (2008) and Thompson and Van Oosterom (2012) give the definition of a 3D parcel for 3D cadastre purposes. The main rule is that the volumetric object is internally connected, which means that a shell can self-touch, as long as the interior of the solid stays connected. Ying et al. (2015) follow this definition and state that a valid volumetric object is a 3D primitive that can be represented by one close polyhedron, refined by a set of connected faces. The volumetric object satisfies the following characteristics: closeness, interior connection, face-construction and proper orientation. Evidently, the volumetric object here can have through-hole/ring or cavity that allows its boundary faces to touch each other, which is not a 3-manifold in some cases.

2.1.1 Considering Land Administration Domain Model (LADM) standard

The LADM international standard ISO 19152 (ISO, 2012) represents parties (natural and non-natural persons), spatial units (survey and geometrical/ topological representations) and their relationships through rights, restrictions and responsibilities (RRRs). In this standard, the RRR, applying on a given spatial unit, or a group of spatial units, are defined in a bundle form using a basic administrative unit applied to a given spatial unit, or a group of spatial units.

Figure 2 shows a simplified database storage scheme proposed by Thompson et al. (2016) able to represent the various types of spatial units. Compared to ISO 19152, the classes LA_SpatialUnit and LA_BoundaryFaceString have been combined into a single class (LA_SpatialUnit) as there is in this context a 1-to-1 relationship between the two classes which is still conformant with ISO 19152.
There are two reasons why a polyhedron attribute of type GM_Solid for 3D spatial units is not appropriate: 1. in most cases there is an overlap between the vertical faces of polyhedron and the LA_BoundaryFaceString defined by the footprint (redundant and possible cause of inconsistency), and 2. the GM_Solid can only represent fully bound spaces. Therefore, this is not a suitable solution and the association with LA_BoundaryFace is used instead.

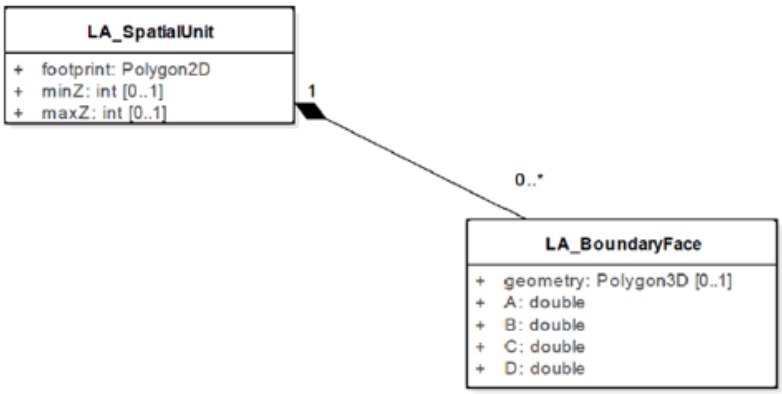

Figure 2. Simplified schema for database storage (Thompson et al., 2016)

Also note that in the simplified scheme, there is no sharing of LA_BoundaryFace's among different LA_SpatialUnit's and the association between LA_SpatialUnit and LA_BoundaryFace is also not signed (indication + or – orientation of face when used in a 3D LA_SpatialUnit). This is possible in ISO 19152 and also fits quite well in the proposed style of LandXML transport encoding - Nested Parcels Method (Thompson et al., 2016).
In a DBMS that allows in-row storage of simple geometries, this form is highly efficient. For example in PostgreSQL/PostGIS or Oracle Spatial, simple 2D spatial units (such as four sided city blocks) will be stored in-row, permitting very fast retrieval. In addition, access can be in one of three forms: 1: as a 2D footprint (this could be compared to LoD0 (level of detail) in CityGML); 2: as a "Prism" (footprint with top and/or bottom, this could be compared to LoD1 in CityGML); 3: as a complete 3D geometry (the higher LoD's in CityGML, including indoor, as one building may contain multiple spatial units) (Thompson et al., 2016).
Thompson et al. (2016) further elaborate that the down-side of this model is that there is duplication of the definition of boundaries that separate spatial units (one copy for each spatial unit involved), leading to the potential for incompatible definitions of the same boundary. The broad approach in terms of a storage scheme is that a more-or-less conventional 2D complete, non-overlapping topological coverage of the region of interest would be generated (sharing 2D boundaries), while 3D surfaces would be shared by and would separate spatial units that are

adjacent in 3D, but overlapping in 2D. A secondary advantage of this approach is that it effectively supports liminal parcels as defined in the LADM (ISO, 2012).
Another issue is that if a footprint is stored as a polygon, most DBMSs do not permit any attributes to be recorded on the individual lines - such as the nature of the line. This is an area needing consideration and in principle the LADM supports management attributes on the boundary level: both for lines (LA_BoundaryFaceString) and faces (LA_BoundaryFace) (Thompson et al., 2016).

2.2 Voxel representation

Voxel, a volumetric pixel, is a quantum unit of volume and has a numeric value (or values) associated with it that represents properties or independent variables of a real object or a value from a continuous field. Representing 3D urban scenes by voxels bring a number of advantages: calculating volumes is a matter of counting the number of voxels that constitute an object, 3D bisections become simple selection operations, storing volumetric spaces such as air, water, and underground is possible. An additional benefit of voxel storage is the atomicity of the data type; every object is represented by only one primitive (3D cube) instead of the surface representation (i.e. points, lines and polygons) (Zlatanova et al., 2016)).
Voxel storage offers a number of interesting simplification, use cases, as well as challenges. One of the major challenges is its storage and efficient handling by spatial database management systems (Gonçalves et al., 2016).
It is clear that a dense flat relational table is not ideal storage format for a large 3D grid. The Holy Grail is an architecture which allows effective compression to reduce storage footprint, and efficient data retrieval to access only the attributes of interest at a specific resolution. Such key features is what distinguishes a column-oriented architecture from a record-oriented architecture and the reason for their efficiency on analytical workloads (Abadi et al., 2008).
Despite column-oriented architectures emerging as the right candidate, their flat storage model is not yet suitable to store a large 3D city model. Gonçalves et al. (2016) extended a column-store to also support a nested column-oriented storage for 3D city models. The chosen format is Parquet[1]. It is an effective storage model for sparse data sets with a nested structure (the different LODs). Its flat columnar format fits well with the column-oriented programming model. Parquet file layout is represented in Figure 3.

[1]http://parquet.apache.org/ *(accessed on 21 August 2016)*

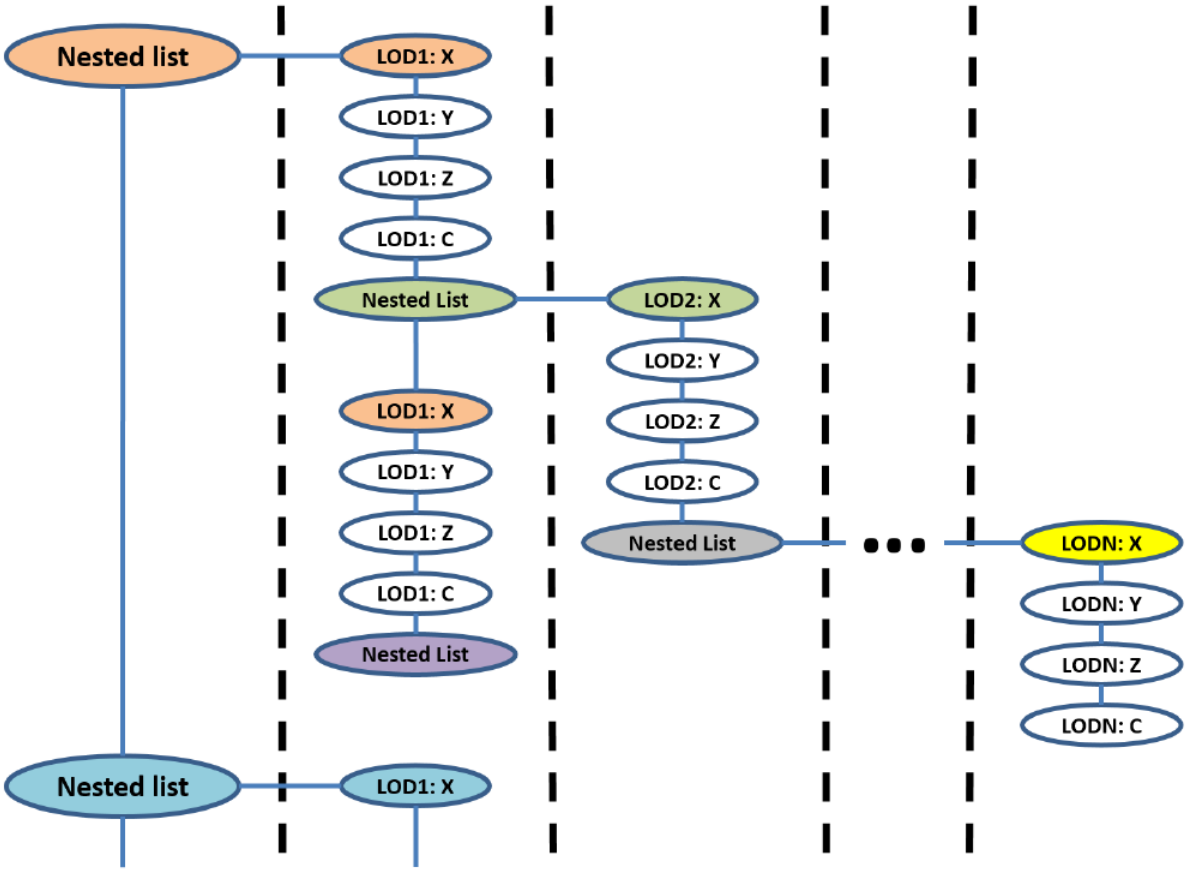

Figure 3. Parquet file layout (Gonçalves et al., 2016)

Storage challenges

For the storage and indexing of 3D voxels linked with properties, such as voxels created to simplify a point cloud, two approaches can be considered, a homogeneous voxel grid versus a heterogeneous voxel collection. The former allows for factorization of invariant properties from the data structures, while the latter is better suited to sparse models such as a 3D city model with different LODs. A homogeneous voxel grid is easy to define using a flat relational schema, i.e., real-world objects are formed by semantically grouping voxels together via foreign key relations and relational views. The schema normalization is used to reduce the storage footprint at the cost of expensive spatial joins. The schema normalization storage footprint is proportional to the size of each voxel. Hence, efficient data access becomes dependent on efficient column compression techniques and effective storage of geometric empty spaces. A heterogeneous voxel grid poses extra challenges compared to a homogeneous voxel grid due to the preservation of the geometry semantics when converting vector to raster data. The object's semantics depends on the semantic level of detail (LOD) (Gonçalves et al., 2016).

Nested column-oriented storage

For efficient storage and data retrieval at different resolutions, Gonçalves et al. (2016) embraced a column-oriented format for voxel-based 3D city models. Columnar formats have several advantages. Organization by column allows better compression, as data is more homogeneous. For large data sets the I/O is improved since it is possible to efficiently scan a subset of the columns while reading the data. Hence, to store nested data structures in flat columnar format, the schema is mapped to a list of columns in such a way that records are written and read back to its original nested data structure in an efficient way.

3D raster spatial DBMS

A voxel-based 3D city model is best managed in a spatial DBMS as each voxel has a semantic relation to a real world object and various attributes (for example color, material, porosity, reflection properties, etc.). Furthermore, a single spatial DBMS offers all functionality in one place, avoids the need for multiple software tools with associated high volume data transfer and format transformations (Gonçalves et al., 2016). They also assert that Oracle Spatial, Graphs 12c, and PostgreSQL 9.2 are developing extensions to support 3D geometries. In GIS packages, only GRASS has support for voxels, but it still stores them as flat files. The systems are still in their infancy and they offer limited functionality. Due to the complexity of their software stack, deep integration with the database engine is even further away. For their work they have extended a modern column-store, MonetDB (Idreos et al., 2012), which steps away from traditional SDBMS which are all record-oriented architectures. Through vertical partitioning of relational tables column-store significantly reduce data access. MonetDB spatial features have been matured to provide core technology components for geo-spatial big data analytics. Atomic spatial types and their operations are becoming part of the relational kernel and not an add-on. All the operations are available for spatial applications through integrated environments, such as R and Python, and a SQL front-end.

The recent development in the field of nD-array databases is described in more detail in section 7.1 nD-array Database Management Systems.

2.3 Point Cloud Database Management Systems

2.3.1 Considering LADM standard

There are two main situations in which point cloud data (see Figure 4) are of importance: one as 3D reference and two as input for the creation of 3D parcels.

The first use of point clouds in the context of 3D Cadastre, is related to the visualization of 3D parcels, were we often use reference information in order to understand the location and extent of 3D parcels. Also in the case of 2D often reference information, such as buildings, road, and waterways, are used in combination with the parcels of the cadastral map. Both in 2D and in 3D this reference information is usually in the form of vector models; for example in 3D this is BIM/IFC or CityGML. However, in 3D also point clouds could be used for this. Not only to show reference objects (building, roads), but also to give an indication of the Earth surface. It is very important to know if a 3D parcel is below, on, above the Earth surface or maybe even a mixed configuration. The benefits of using point clouds is that they are close to data acquisition, quite detailed and realistic, and do not require the often costly operations needed for the interpretation, classification and conversion to vector representation.

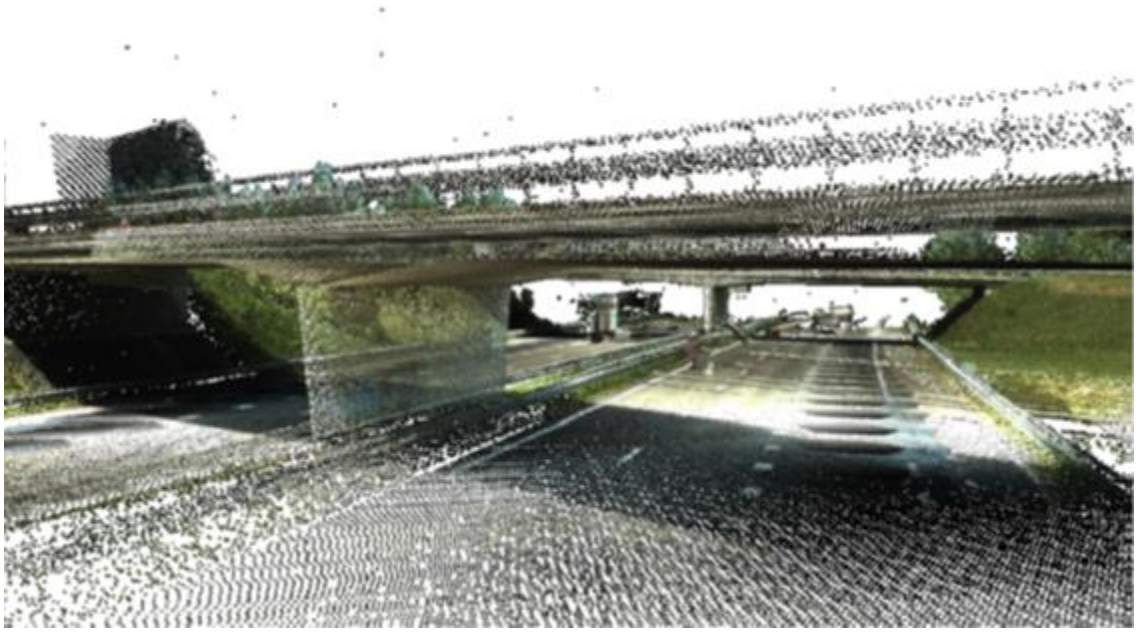

Figure 4. Example point cloud, related to a 3D construction

The second use of point clouds in the context of 3D Cadastre, is related to the creation of 3D parcels (see chapter Initial registration). 3D plans of 3D parcels, to be generated in the form of the physical objects with Rights, Restrictions and Responsibilities (RRR's), could be used when creating the 3D parcels, that is, legal spaces related to real world objects. A point cloud could be used to check if the construction is indeed realized as indicated on the plan (and therefore the local of the legal spaces is also correct). However, in case of older constructions these 3D

plans may not be available at all, and in these situations collecting reference information in the form of a point cloud may be very effective.

2.3.2 Nature of the point cloud data

State of the art spatio-temporal representations are based on either gridded (raster, voxel) or object (vector) models. Point clouds are in between: they share with the gridded model the sampling nature, and they share with the object model the ability to represent arbitrary locations (points). Today both vector models and gridded (in 3D voxel) models are quite well supported in spatial DBMSs and other software tools. After realizing the importance of the point cloud representation, there is now also an increase in support for managing the point clouds also within the spatial DBMS.

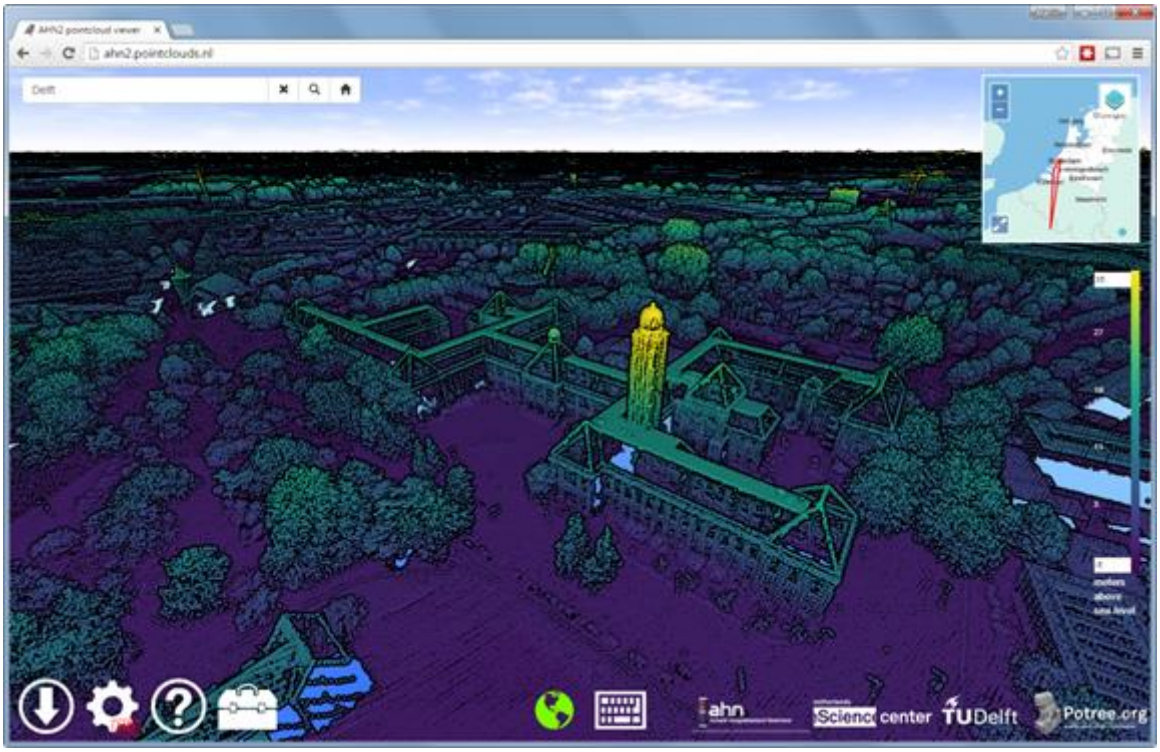

Figure 5. Point cloud data in a 3D web viewer (ahn2.pointclouds.nl)

There are various sources of indoor and outdoor point clouds, but it is fair to state that one common characteristic is they all produce rather large volumes of data. Lidar, photogrammetry, and various other survey technologies enable the collection of massive point clouds. Faced with hundreds of billions or trillions of points the traditional solutions for handling point clouds usually underperform even for classical loading and retrieving operations. To obtain insight in the features affecting performance, in earlier work (van Oosterom et al., 2015, van Oosterom et al, 2016) tests were carried out with different storage models on various systems, including Oracle Spatial and Graph, PostgreSQL-PostGIS, MonetDB and a file based solution: LAStools by Rapidlasso GmbH 2015. It should be noted that web services based on point cloud data are becoming more popular, and these could also be used very well in the context of 3D Cadastre. The requirements that these 3D web viewers pose, including level-of-detail (perspective) selections (see Figure 5), are rather difficult to meet given the huge volumes of data.

The users have a range of different datasets and types: administrative data, vector data, raster data, temporal data, etc. Therefore, a standardized and generic DBMS solution would be preferable above file based solutions when users want to combine data. Therefore, it is proposed to add a third type of spatial representation to the geographic information processing systems and standards: a point cloud data type. Based on user requirements the point cloud data type and its operators should cover (van Oosterom et al., 2015):

1. XYZ: specify the basic storage of the coordinates in a spatial reference system (SRS) using various base data types: integer, float, double, number, varchar.
2. Attributes per point: 0 or more. Example attributes are intensity I, colour RGB, classification, observation point position in addition to resulting target point, etc.
3. Data organization based on spatial coherence: blocking scheme in 2D, 3D, etc.
4. Efficient storage with compression techniques exploiting the spatial cohesion.
5. Data pyramid support: level of detail (LoD), multi-scale or vario-scale and support for perspective selections.
6. Temporal aspect: options for time per point (costly) or per block (less refined).
7. Query accuracy: for 2D, 3D or nD query ranges or geometries specify to report points or storage blocks and refine subsets of blocks with/without tolerance value.
8. Operations/ functionalities in the following categories: (a) loading, (b) selections, (c) simple analysis (not assuming 2D surfaces in 3D space), (d) conversions (some assuming 2D surfaces in 3D space), (e) towards reconstruction, (f) complex analysis (some assuming 2D surfaces in 3D space), (g) LoD use/access, and (h) updates.
9. Indicate the use of parallel processing for the operations listed in the Point 8.

2.3.3 Point Cloud Management Systems

The suitability of Database Management Systems (DBMS) for managing point cloud data is a continuous debate. File-based solutions provide efficient access to data in its original format, however, data isolation, data redundancy, and application dependency on such data format are some major drawbacks. Furthermore, file-based solutions have also poor vertical and horizontal scalability. In comparing both DBMS and File-based solutions for point clouds two storage models can be identified:

- Blocks model: nearby points are grouped in blocks which are stored in a database table, one row per block
- Flat table model: points are directly stored in a database table, one row per point, resulting in tables with many rows.

The DBMS with native point cloud support based on the blocks model are Oracle Spatial and Graph and PostgreSQL-PostGIS. For more information regarding the various point cloud storage models and their tuning (block size, compression), please refer to an earlier publication (van Oosterom et al. 2015). In addition to using native blocks solutions, it is also possible to use third-party block solutions. Point Data Abstraction Library (PDAL) is a translation library for manipulating point cloud data and it is used, in general, as an abstraction layer on management operations. Thus the same operations are available independently on which system (DBMS or File-based) actually contains the data.

3. 3D SPATIAL INDEXING AND CLUSTERING

The 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.
The review of spatial indexing is given in Breunig and Zlatanova (2011). Within the current Spatial Database Management Systems, for example PostGIS and Oracle Spatial, there are several types of indexes (Khuan et al., 2008): they are B-Tree indexes, R-Tree indexes (Guttman, 1984), 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 (it is difficult to determine which is greater, (0,0) or (0,1) or (1,0)) so B-Tree indexing is of limited use for the 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, for example Oracle Spatial implements the 2D and 3D R-Trees.
- 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 uses the R-Tree index implemented on top of GiST to index GIS data.

Several strategies have been developed for indexing of multidimensional data, although there is limited vendor support for these, and true 3D index creation is still an ongoing research problem (Schön et al., 2009).

3D R-tree
This section is based on the article of Zhu et al. (2007). R-tree is considered as one of the most promising 3D spatial indices. In an ideal case, the neighbouring objects should be in the same nodes or sibling nodes, the minimum bounding rectangle (MBR) of sibling nodes is different, and the overlap is then minimized. However, when the index expands from 2D to 3D, because of the great size and shape diversity of different objects in 3D space, the minimum bounding box (MBB) of sibling nodes will frequently overlap, and the MBBs of nodes can even contain each other. The better space proximity of R-tree is therefore the key to 3D spatial indexing in order to adequately take into consideration the principle of 3D spatial proximity. 3D spatial clustering and the corresponding 3D R-tree indices are required in order to minimize the overlap among the sibling nodes and to balance the shape and size of nodes. Proximal objects cluster together in 3D space in the same nodes or proximal sibling nodes.
For dynamic indexing as well as R-tree construction, both insertion and deletion are important basic operations. The insertion operation is more critical to the R-tree construction procedure

in complicated 3D space. The insertion of an object would result in the splitting of the R-tree node, and cluster grouping is usually used to support node splitting and node optimization. Zhu et al. (2007) propose a 3D R-tree algorithm based on 3D spatial cluster grouping. They first propose an integrative grouping criterion *W* concerned with the 3D overlap, 3D coverage and MBB shape value of nodes. Then the *k*-means algorithm is employed to improve the 3D spatial cluster grouping and inserting operation of 3D R-tree.

3D integrative grouping criterion

For a 3D spatial object set $S = \{P_1, P_2, \ldots, P_n\}$, there are clustered group sets S_i, $i = 1, \ldots, k$ and the integrative grouping criterion value *W* can be calculated using the equation:

$$W = \sum_{i=}^{k-1} \sum_{j=i+1}^{k} \text{Overlap}(S_i, S_j) + \sum_{i=1}^{k} \text{Coverage}(S_i) + \sum_{i=1}^{k} \text{Shape}(S_i).$$

The smaller the *W* value, the better the 3D spatial cluster grouping results.

3D spatial cluster grouping

The 3D spatial cluster grouping operation includes two steps: the node splitting and the optimization among nodes. Figure 6 illustrates a typical grouping result. As shown in Figure 6, the wire frame box denotes the node that needs to be split, and solid boxes denote the child nodes, and in this example it is obvious that splitting the child nodes into three groups is more rational than into two groups. For this purpose, a new 3D spatial-cluster grouping algorithm is introduced, in which the k-means clustering method of data mining is employed to partition k clusters in a set concerning the 3D spatial layout of objects. As both the spatial coverage and overlap of nodes should be minimized, as well as the shape of MBB nodes being considered, the above mentioned integrative grouping criterion value W is used as the grouping criterion.

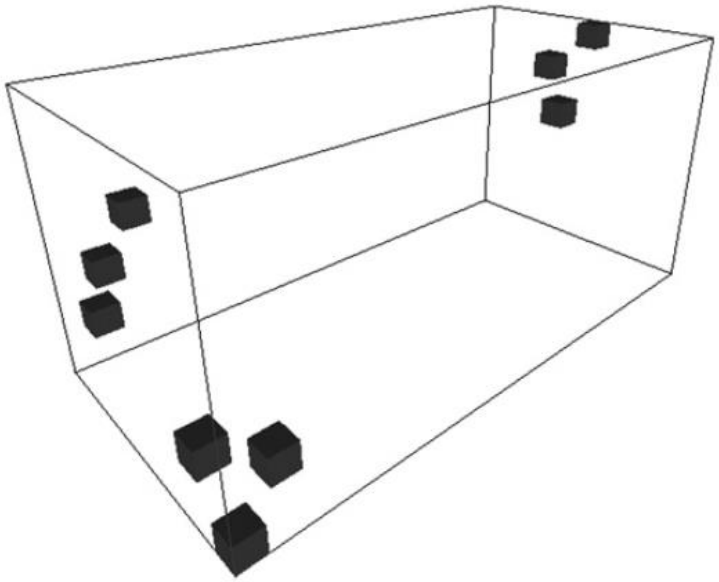

Figure 6. Spatial cluster grouping (Zhu et al., 2007)

Figure7 illustrates the flow chart of the 3D spatial cluster grouping algorithm, which includes spatial clustering and spatial grouping.

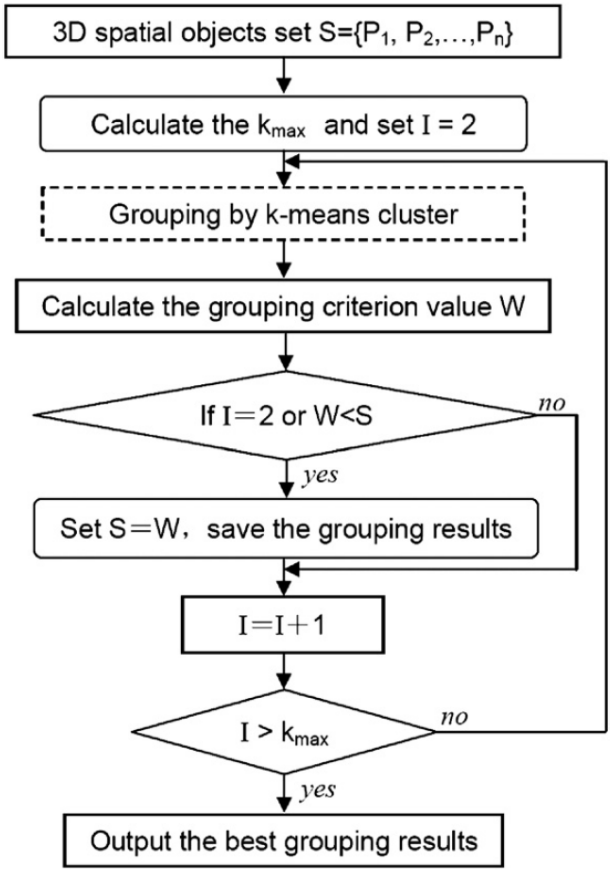

Figure 7. Flowchart of the 3D spatial cluster grouping algorithm (Zhu et al., 2007)

Spatial clustering
Step 1: Calculate the maximal group numbers k_{max}. Ensure that $n/k_{max} \geq m$, where n is the number of total spatial objects, m is the minimal number of children in a node.
Step 2: Choose different group numbers I (I = 2, …, k_{max}) as parameters; adopt the spatial grouping algorithm given below to calculate the corresponding integrative grouping criterion value W using equation above. Select the grouping strategy with the minumum value of W as the final grouping result.

Spatial grouping
Input: 3D spatial object set $S = \{P_1, P_2, \ldots, P_n\}$.
Output: k small group sets with inserted objects S_i, i = 1, …, k.
The details about the spatial grouping algorithm (and also 3D R-tree insertion) give Zhu et al. (2007).
Figure 8 illustrates an experimental result of 3D R-tree generation. The R-tree possesses inherent inefficiencies when applied to LiDAR data (Schön et al., 2013). They proposed an octree index for 3D LiDAR data atop Oracle Spatial 11g.

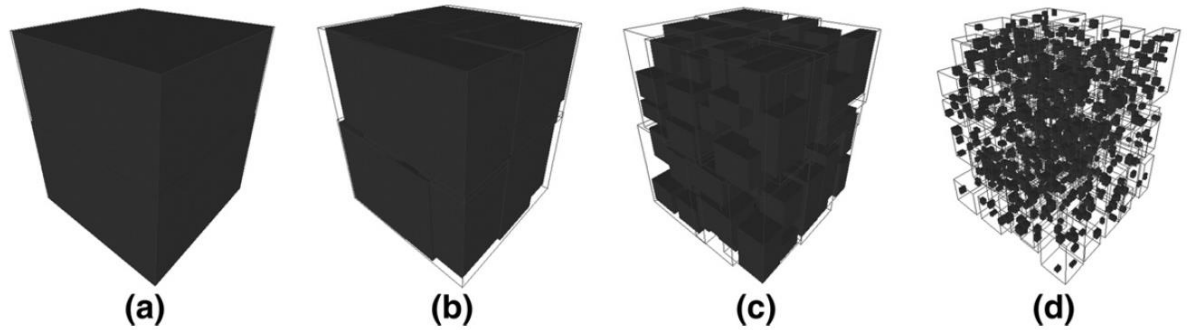

Figure 8. 3D R-Tree generation procedure. (a) Root layer. (b) 2nd middle layer. (c) 3rd middle layer. (d) Leaf layer (Zhu et al., 2007)

Octree

An octree's structure dictates that each internal node contains exactly eight child nodes regardless of its many variants. In the implementation herein a bucket point region (PR)-octree approach was adopted, where the space is decomposed into cubic blocks (or cells) through recursion, until a block is homogeneous.

By definition, an octree can result in an unbalanced hierarchical tree when the data distribution is not uniform. However, this requires the storage of the logical tree structure in the SDBMS for recursive reconstruction of the tree structure during query processing.

Therefore, the proposed implementation employs a fixed, maximum tree height (also called its tiling level), thereby resulting in a balanced tree. This improves query efficiency as neither the tree structure nor recursive cells need to be stored, only the tiling level. The selection of an appropriate tiling level is a decisive factor, involving the dataset's area and size. As such, experimentation with different levels is needed to optimize performance for a specific dataset.

The user can specify the tiling level through the parameter OCTREE_LEVEL during index creation. Each cell is associated with a unique code, here in referred to as the cell code. The cell code is obtained by using z-ordering of all cells at the specified level.

Figure 10 illustrates the 3D space decomposition through an octree, and Figure 9 illustrates cell code generation. All cells in the bottom half are assigned with the prefix '0'—zero, and all in the top half are assigned with prefix '1'—one. Cells are marked south-west (SW), south-east (SE), north-east (NE) and north-west (NW) and associated codes are 00, 01, 10, and 11 consecutively. The associated cell code is identified by traversing the octree from root node to leaf node. For example, using B to represent the bottom half and T to designate the top half, at tiling level 5, the code for the path BNW(011)–TSW(100)–TNE(110)–BSE(001)– BSW(000) is 011100110001000. Here, it only follows the tree path where the cell associated to a node in the path contains the point. The point's ROWID and associated cell code are stored in an index storage table. The metadata (for example, tilinglevel, indexname, index owner, max level, min level, etc.) for the entire index are stored as a row in an table called index metadata table (Schön et al., 2013).

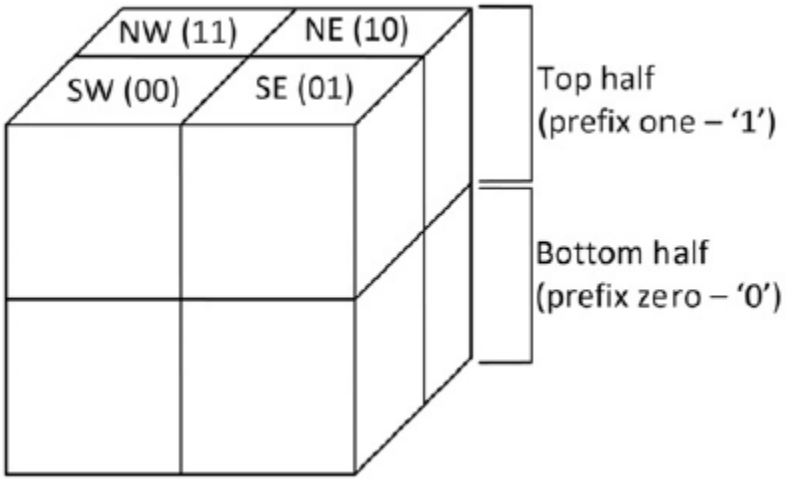

Figure 9. Quadtree sectors (Schön et al., 2013)

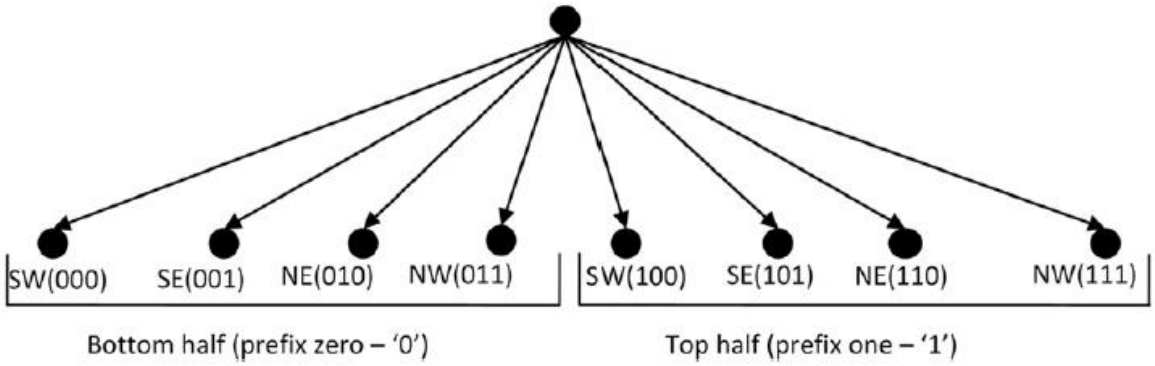

Figure 10. 3D space decomposition (Schön et al., 2013)

In the octree implementation, the index storage table has two columns: OCTREE_CODE (oracle data type RAW, in order to store the cell code, requires 3 bits for each branch), and OCTREE_R- OWID (oracle data type ROWID, 10 bytes in size, in order to store the ROWID of the 3D point geometry).
An octree offers an alternative, but currently no commercially-available SDBMSs support octree indexing (Schön et al., 2013).

4. 3D GEOMETRIES AND 3D OPERATIONS

4.1 Creation and validation

With the utilization and development of dense urban space, true 3D geometric volume primitives are needed to represent 3D parcels with the adjacency and incidence relationship. Validation is a necessary tool to guarantee the output of processing or manipulation GIS operations such as: calculation of the area of polygons; creation of buffers; conversion to other formats; Boolean operations such as intersection, touching, contain, etc. (Ledoux et al., 2009).

The definition of the polyhedron/solid given in the ISO standards (ISO, 2003) and implemented with GML (OGC, 2007) is as follows: *A GML solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces as specified in ISO 19107:2003, 6.3.18. gml:exterior specifies the outer boundary, gml:interior the inner boundary of the solid* (OGC, 2007). To be considered a valid solid, a solid must fulfil several properties or criteria. The most important criteria are: (i) it must be simple (no self-intersection of its boundary); (ii) it must be closed, or 'watertight'; (iii) its interior must be connected; (iv) its boundary surfaces must be properly oriented; (v) its surfaces are not allowed to overlap each other. It should also be noticed that since a solid is formed of 2D primitives (embedded in 3D space), these also have to be valid. For instance, if a surface has a hole (an inner ring), than this ring is not allowed to overlap with the outer boundary of the surface (Ledoux et al., 2009).
From the 3D cadastre perspective, a volumetric primitive is a complete representation of a polyhedron able to support the various calculations and analysis related to the 3D cadastral objects. The volumetric primitives in 3D space need to be mutually exclusive and they need to exhaustively partition the extent of the domain (i.e. no gaps are allowed) (Ying et al., 2015).

SQL Geometry Types
The SQL Geometry Types (OGC, 2010) extend the set of available predefined data types to include Geometry Types. A conforming implementation shall support a subset of the following set of Geometry Types: {Geometry, Point, Curve, LineString, Surface, Polygon, PolyhedralSurface, GeomCollection, MultiCurve, MultiLineString, MultiSurface, MultiPolygon, and MultiPoint}.
OGC (2010) presents a new SQL geometry type – PolyhedralSurface, which is subtyped from Surface, and implements the required constructor routines and interfaces of Surface and MultiSurface. A PolyhedralSurface is a contiguous collection of polygons, which share common boundary segments and which as a unit have the topological attributes of a surface. For each pair of polygons that "touch", the common boundary shall be expressible as a finite collection of LineStrings. Each such LineString shall be part of the boundary of at most two Polygon patches. The PolyhedralSurface could be a simple, closed polyhedron (OGC, 2011).
Kazar et al. (2008) present Oracle's data model for storing 3D geometries (following the general OGC/ISO GML3 specifications) and define more specific and refined rules for valid geometries in this model. They show that the solid representation is simpler and easier to validate than the GML model but still retains the representative power.
In Oracle, a simple solid is defined as a 'Single Volume' bounded on the exterior by one exterior composite surface and on the interior by zero or more interior composite surfaces. To demarcate the interior of the solid from the exterior, the polygons of the boundary are oriented such that their normal vector always point 'outward' from the solid. In addition, each polygon of the composite surfaces has only an outer ring but no inner ring (this is a restriction compared to the GML definitions, but without losing any expression power) (Kazar et al., 2008).
Validation rules/tests for Simple Solids (based on Kazar et al., 2008):

- Single Volume check: The volume should be contiguous.
  - Closedness test: The boundary has to be closed. Necessary condition but not sufficient (Figure 11 left, Figure 12 left, Figure 13 left are invalid)

- Connectedness test: For sufficiency, volume has to be connected. (Figure 11 right, Figure 12 right, Figure 13 right are valid). This means each component (surface, solid) of the solid should be reachable from any other component.

- Inner-outer check:
  - Every surface marked as an inner boundary should be 'inside' the solid defined by the exterior boundary.
  - Inner boundaries may never intersect, but only touch under the condition that the solid remains connected (see above)
- Orientation check: The polygons in the surfaces are always oriented such that the normals of the polygons point outward from the solid that they bound. Normal of a planar surface is defined by the right-hand thumb rule (if the fingers of the right hand curl in the direction of the sequence of the vertices, the thumb points in the direction of the normal). The volume bounded by exterior boundary is computed as positive value if every face is oriented such that each normal is pointing away from the solid due to the Green's Theorem. Similarly, the volume bounded by interior boundary is computed as negative value. If each exterior and interior boundary obeys this rule and they pass connectedness test as well, then this check is passed.
- Element-check: Every specified surface is a valid surface.
- No-inner-ring in polygons: In the composite surfaces of a solid, no inner rings are allowed.

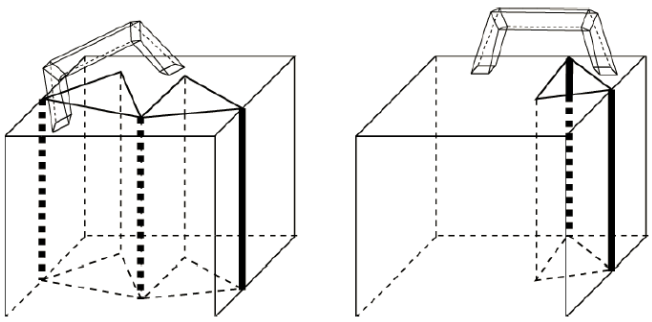

Figure 11. Invalid simple solids becoming valid via adding an additional handle making it possible to travel from one part to another part of the object (completely via the interior). Note: where handle touches the face, a part of the faces is removed (that is an interior ring is added within the exiting face to create the open connection). So, all faces have always (and everywhere) on one side the object and on the other side something else (outside, where the normal is pointing to) (Kazar et al., 2008)

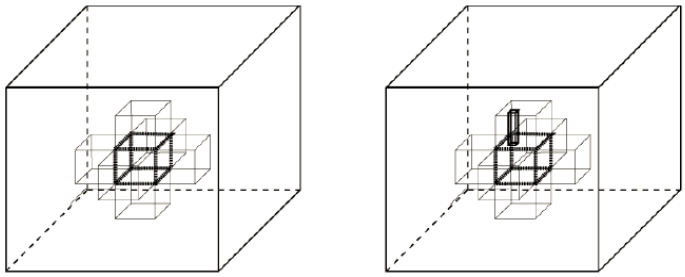

Figure 12. Left: simple solid with 6 internal (cube-shaped) boundaries separating the big cube into two parts (the internal one draw with fat lines is implied by the 6 boundaries of the 6 smaller cube-shaped holes). Therefore the left simple solid is invalid (note that removing one of the 6 holes, makes it valid again). Right: Invalid simple solids of previous figures becoming valid via adding an additional handle making it possible to travel from one part to another part of the object (completely via the interior). Right: the two parts are connected via a 'pipe' making it a valid simple solid again (Kazar et al., 2008)

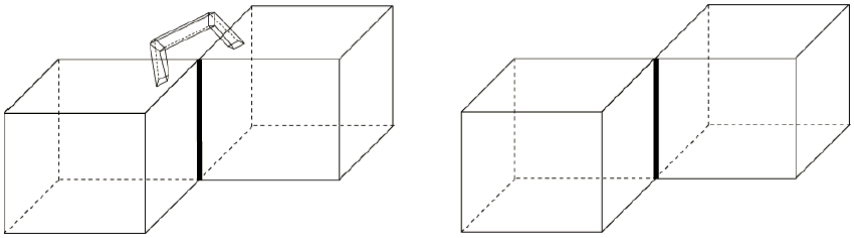

Figure 13. Left: valid simple solid (fat edge still used 4 times), but handle is added through which it is possible to travel from one part to the other part via the interior only, Right: invalid simple solid with one edge being used four times (fat line) (Kazar et al., 2008)

More on valid 3D geometries (for example, composite solids, collections) can be found in Kazar et al. (2008).

While definition exists for solids (given by the international standards for geographic information), Ledoux (2014) states that these definitions for solids are ignored by most researchers and software vendors. He states, that several different definitions are indeed used, and none is compliant with the standards: for example solids are often defined as 2-manifold objects only, while in fact they can be non-manifold objects. Exchanging and converting datasets from one format/platform to another is thus highly problematic. Ledoux (2014) presents a methodology to validate solids according to the international standards. He implemented the methodology in a prototype called *val3dity*[2]. The validator for solids in Oracle Spatial permits the validation of solids (although, as explained it is neither according to the ISO rules nor complete) but returns only one error when the solid is not valid: the first one encountered (even if a given solid contains hundreds of errors). The error comes with a code explaining its nature and, when suitable, its location (for example if a shell is not closed the

[2] https://github.com/tudelft3d/val3dity *(accessed on 20 August 2016)*

centre of the hole is given). This means that a user has to fix the solid for the error mentioned, and to run the validation function again. This step has to be followed for all the errors present, which can be a rather long and painful process for the user. Ideally, all the errors in a solid should be reported so that a user can fix them in one operation. However, cascading effects when validating should be avoided—one example is if a surface is not a valid polygon in 2D, then the validation of the shell whose boundary contains that surface should not be attempted as it will most likely not be valid. In the prototype val3dity, a "hierarchical validation" is used and efforts are made to avoid cascading errors (Ledoux, 2014).

4.2 3D operations

In the implementation specification, OGC (2011) provides the geometry functions that are not limited to any dimension.

Some of the standard functions given by OGC (Simple feature access – Part 1: Common Architecture (OGC, 2011):

- Envelope ( ): Geometry – The minimum bounding box for the Geometry, returned as a Geometry. Minimums for Z and M may be added.
- IsSimple ( ): Integer – Returns 1 (TRUE) if this geometric object has no anomalous geometric points, such as self intersection or self tangency. The description of each instantiable class will include the specific conditions that cause as instance of that class to be classified as not simple.
- Is3D ( ): Integer – Returns 1 (TRUE) if this geometric object has z coordinate values, etc.

Furthermore, OGC (2011) define methods for testing spatial relations between geometric objects:

- Equals (anotherGeometry: Geometry): Integer – Returns 1 (TRUE) if this geometric object is "spatially equal" to anotherGeometry.
- Intersects (anotherGeometry: Geometry): Integer – Returns 1 (TRUE) if this geometric object "spatially intersects" anotherGeometry.
- Touches (anotherGeometry: Geometry): Integer – Returns 1 (TRUE) if this geometric object "spatially touches" anotherGeometry, etc.

Only DBMS itself decides the implementation of the standard functions (specified by OGC) that considers the third dimension or not (Khuan, 2008).

4.3 3D Spatial Constraints

This section is based on Xu et al. (2016), who demonstrate a new methodology to conceptualise and implement geo-constraints in 3D. At first, constraints are designed and expressed using natural language. Then objects in the sentences are abstracted by geometric primitives, and their interrelationship by topological relationships. By doing so, spatial constraints become more specific and clearer to the others. Following the well-defined spatial types and operations as proposed in the ISO 19107 standard and using various tools, and attempt was made to formalise these constraints using Object Constraint Language (OCL). Finally, the constraints are translated to executable code, for example Procedural Language/Structured Query Language

(PL/SQL), and with a small modification realised in the database by trigger mechanisms. OCL is a commonly adopted method of modelling geo-constraints. It is a formal language used to describe the constraints applying to objects, and is part of UML which is preferred concept modelling scheme.
A proposed methodology of modelling 3D geo-constraints can be used as a generic approach for all spatial-related constraints specifications in four stages:

1. Natural Language
2. Geometric/Topological Abstractions
3. UML/OCL Formulations
4. Model Driven Architecture (MDA)
    a. Database PL/SQL Code
    b. Exchange Formal XML
    c. Graphic User Interface ArcGIS

An example of geo-constraint is 'a road cannot cross a building'. Then, in the spatial model, the real world objects can be described by clearly-defined geometric primitives (for example solid, surface, line, point). Here, three things in the text need to be clearly defined, 'what is a building?', 'what does cross mean?', and 'what is a road?' We can model the building using solid geometry and the road surface geometry. And the term 'cross' can be replaced by Nine-Intersection Model (9IM) 'intersect'. The situation the constraint intends to forbid can be rephrased as: 'A surface must not intersect a solid'.

UML/OCL Formulations

Various tools support automatic generation of, for example, SQL script, from UML class diagrams so that an implementation of the model can be created in the database. If OCL constraints can also be integrated to this code generation, the constraints can be integrated to database model and function through the whole lifespan of the database. A challenge of specifying a relationship constraint is that currently there is no a rigorous mechanism in UML2.2 to indicate a constraint. Only when two objects classes are related in the standard way, the constraint can be attached to the association. But when two object classes are not explicitly connected with an association link, the method to mention constraints about their relationships could lead to discussion. For example, in general a road class and a building class may not have an explicit association. However, when some spatial constraint such as minimum distance between them is to be specified, a constraint association between the road class and the building class needs to be considered. Xu et al. (2016) proposed in their research the normal association plus colour 'red' as a new type of association link in UML class diagram (see example in Figure 14).
Another difficulty which is somewhat related to the lack of constraint association in UML is that OCL itself does not support constraint expressions that have multiple classes involved. In normal OCL formula, if a constraint related to a different class than the context class needs to

be expressed, a name of association end role to navigate from one class to the other is required. For example, to specify the no-intersect rule between a building instance and a road instance, the class ‘Road‘ must be available in the context ‘Building‘. In other words, the class ‘Road‘ should have a property that is of ‘Building‘ type, or ‘Building‘ class should have a property that is of ‘Road‘ type. But in this example of no-intersect between a road and a building, neither ‘Building‘ nor ‘Road‘ has a property to typify each other. So an expression can be (Xu et al., 2016):

```
context Building
inv BldRoadNoIntersect:
   Road.allInstances() -> forAll (r | intersect(r.geometry,
geometry) = false)
```

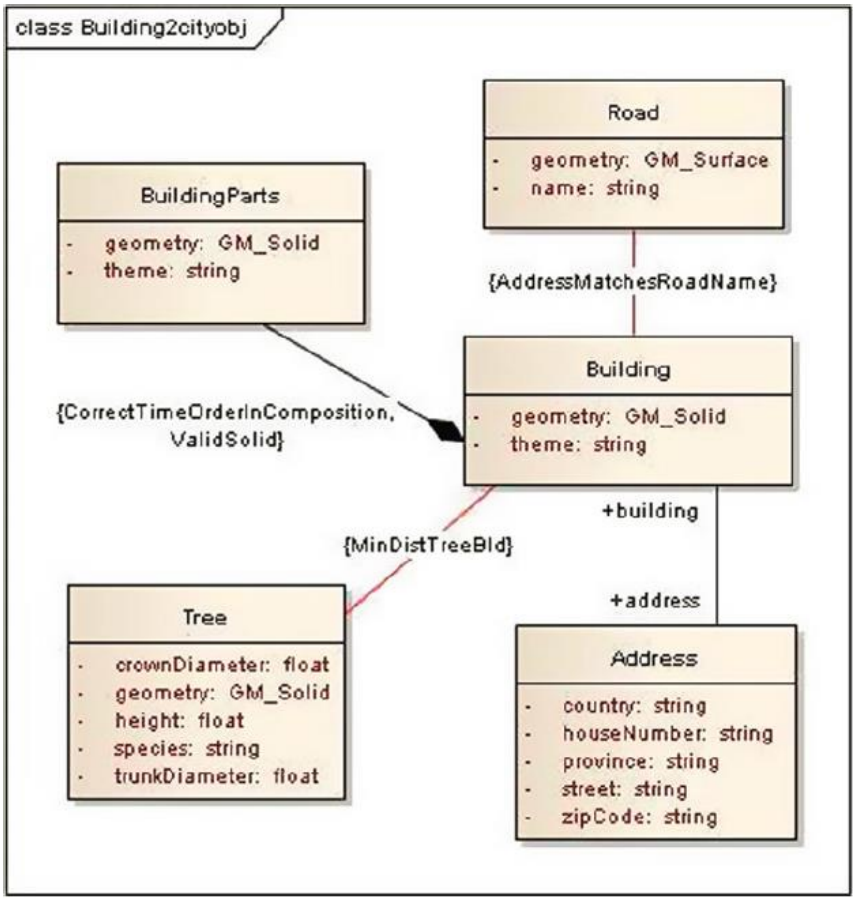

Figure 14. UML model of constraints relevant to building object class (Xu et al., 2016)

Code Generation – Focus on Database PL/SQL Code

The Model Driven Architecture principle, being supported by Object Management Group (OMG), provides a framework to define how models in one domain-specific language (for example UML, OCL) can be translated to models in the other languages. For spatial constraints,

the 3D geometries standardised in ISO19107 and 9IM topological names are not yet included in the OCL library.
When a user modifies (create/insert/update/delete) certain datasets and then tries to commit the modification to the database, the trigger will be fired. Once it detects that a constraint is not satisfied in this commitment, it will give an error message to the front-ends and reject the transaction. By this means, a trigger is able to response to the data modification at run-time and guard the database integrity. Given the trigger mechanism, if the OCL expressions are translated into SQL scripts, the spatial constraints check can be carried out by the spatial functions (for example *distance(), buffer(), intersect()*) supported by the database. In this sense, the power of data maintenance and spatial functions from database can be combined to have 3D geo-constraints integrated in database seamlessly.
However, the existing 3D functions in Oracle Spatial are relatively new and not extended. Many spatial and topological constraint checks cannot be immediately implemented yet. The most useful function in Oracle Spatial database to calculate 3D topological relationships is *SDO_AnyInteract*. It is able to detect if two 3D objects are 'disjoint' or not. But it does not disclose more details about what is happening in the 'non-disjoint' part. For example, two geometries which have 'touch' and 'intersect' do not make any difference to it. To be able to distinguish them in 3D, a new function named '3D_SurfaceRelate' is developed in this research.
Xu et al. (2016) give the examples of constraints on 3D objects. Their methodology (1. Natural language, 2. Geometry/topology, 3. UML/OCL, 4. Implementation) can be applied to many 3D topographic models, such as city models. In their research, 3DCityDB (Kolbe et al., 2009) was selected as a 3D topographic model. The necessary constraints regarding to city objects in 3D space were discovered and described in natural language first. The first attempt to formalise these constraints in UML/OCL – (pseudo 3D geo-OCL) – is explained. The well-defined 3D geometric primitives from ISO19107: 2003 standard - GM_Point, GM_Curve, GM_Surface, GM_Solid and the aggregational and compositional types of them are used as spatial types in UML class diagrams. Further, spatial operators from ISO19107 such as *distance()* and *intersect()*, as well as Oracle functions *inside()* and *validateGeometry()* are used in these formulations. The last stage was a creation of PL/SQL code. Because currently automatic model translation from OCL to SQL is not available, so PL/SQL code was written by hand. The challenges of automatic translation lie on the support of spatial types and operations in OCL standard, multiple class expression of OCL and sufficiency of spatial functions in the database.

5. 3D TOPOLOGY STRUCTURES

Topology is defined as the identification of spatial relationships between adjacent or neighbouring objects (Ellul, 2007). To model 3D topology, a number of 3D topological frameworks have been introduced (Zlatanova 2000). As Zulkifli et al. (2015) mention, these can be distinguished into two types of frameworks:

1. Classification of topological relationships between two objects (for example Egenhofer, 1995; Billen et al., 2002), and
2. Topological structures representing the structural relationship between many primitives and objects (Van Oosterom et al 2002, Zlatanova et al 2004).

In the context of the second type of framework, several 3D topological models and approaches have been developed to construct a topologically correct datasets, for example (Penninga and van Oosterom, 2008; Ledoux and Meijers, 2009; Bormann and Rank, 2009; Ghawana and Zlatanova, 2010; Brugman et al., 2011).

5.1 Considering LADM standard

These previously mentioned topological models above have not discussed LADM standard (Zulkifli et al., 2015). A comprehensive land administration model is essential to build the cadastral management system. The LADM (Land Administration Domain Model) provides a conceptual description for a land administration system, including a 3D topology spatial profile (Thompson and Van Oosterom 2012).

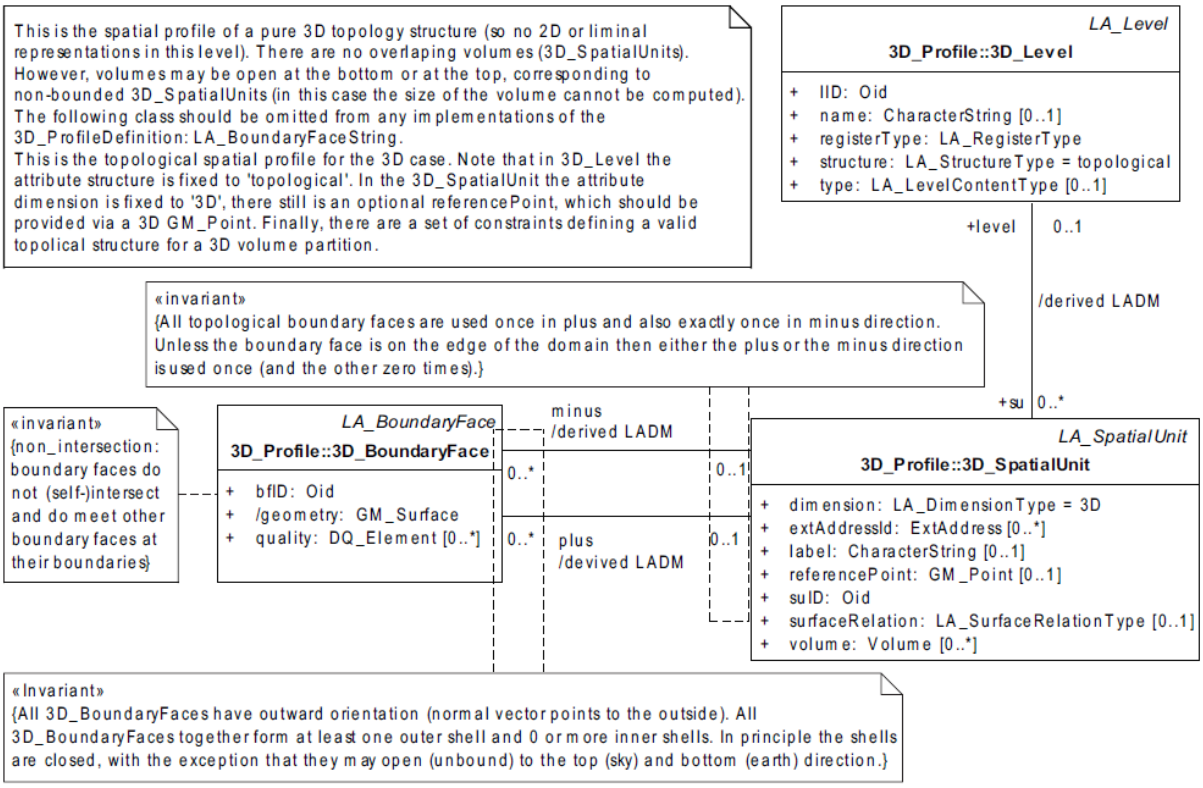

Figure 15. 3D topology based on LADM (ISO, 2012)

The LADM provides conceptual descriptions for land administration, including 3D topology. The LADM also allows for organizing land related data in a standardized and interoperable way to support different types of spatial data. According to the requirements of LADM, topological information alone is not sufficient to describe a 3D spatial unit. Geometrical information must also be associated with each topological primitive; either direct geometries, or indirect (via related topological primitives with geometries). For 3D topology model in LADM as described in Spatial profiles of Annex E7 (ISO, 2012), there are no overlapping volumes

(3D_SpatialUnit). However, volumes may be open at the bottom or at the top, corresponding to non-bounded 3D_SpatialUnits (in this case, the size of the volume cannot be computed). Note that in 3D_Level, the attribute structure is fixed to '3D', and there is still an optional referencePoint, which should be provided via 3D GM_Point. There is a set of constraints defining a valid topological structure for a 3D volume partition. In case of the 3D topology representation, a 3D boundary has plus/minus orientation information included in the association to a 3D spatial unit (see Figure 15). All topological boundary faces are used once in plus and also exactly once in minus direction. Unless the boundary face is on the edge of the domain, then either the plus or the minus direction is used once (and the other zero times). The boundary faces do not self-intersect and do meet other boundary faces at their boundaries. All 3D_BoundaryFaces have outward orientation (normal vector points to the outside). All the 3D_BoundaryFaces together form at least one outer shell and zero or more inner shells. In principle, the shells are closed, with the exception that they may open (unbound) to the top (sky) and bottom (earth) direction (Zulkifli et al., 2015).

Zulkifli et al. (2015) review 3D topology within LADM. They review characteristics of the different 3D topological models in order to choose the most suitable model for certain applications. The characteristics of the different 3D topological models are based on several main aspects (for example space or plane partition, used primitives, constructive rules, orientation and explicit or implicit relationships). The most suitable 3D topological model depends on the type of application it is used for. They conclude, that there is no single 3D topology model best suitable for all types of applications. Therefore, it is very important to define the requirements of the 3D topology model. They further conclude, that based on the reviews of the 3D topological models, a very suitable 3D topology model is the approach based on a Tetrahedral Network (TEN), proposed by Penninga and Van Oosterom (2008).

Ying et al. (2015) present an effective straightforward approach to identifying and constructing the valid volumetric cadastral object from the given faces, and build the topological relationships among 3D cadastral objects on-the-fly, based on input consisting of loose boundary 3D faces made by surveyors. These 3D faces as the cadastral boundaries with official identifications are stored in a database. The method does not change the faces themselves and faces in a given input are independently specified. Various volumetric objects, including non-manifold 3D cadastral objects (legal spaces), can be constructed correctly. They also aimed to develop a more direct method of the solid validation process, describing the steps below:

1. To build valid solids at the beginning of object generation to satisfy the validation requirements.
2. If a valid solid is built and the sets of solids directly there is no need to validate its existence afterwards.

They propose a data model oriented towards the application and storage of a 3D cadastral system. Especially, they extend the geometric-topological model in LADM, which is based on ISO 19107, and redesign the model to support non-manifold 3D objects to represent realistic 3D cadastral objects. They propose a method for creation of both 3D volumetric objects – 3D solids and non-manifold solids (shapes with self-touching or hole) along with topological relationships that are already valid. This is important to model some realistic cadastral objects.

Also the 3D volumetric objects in relation to the outer complementary space (named by Maximal Minimal Solid) can be generated. The presented approach ensures volumetric objects (polyhedral shapes) that satisfy the valid solid characteristics: face-based construction, closeness and uniqueness. Against the mainstream methods, that require one to assume that the shapes (solids) already exist in the 3D object and then test to see if this existence assumption holds, in the proposed method this assumption step is no longer required as a necessary research process. The input faces themselves are stable and they are independently specified. This direct 3D volume construction conforms to normal sequential data flow and business logic to provide valid 3D volumetric objects for 3D cadastral systems without the need for a post-production validity check.

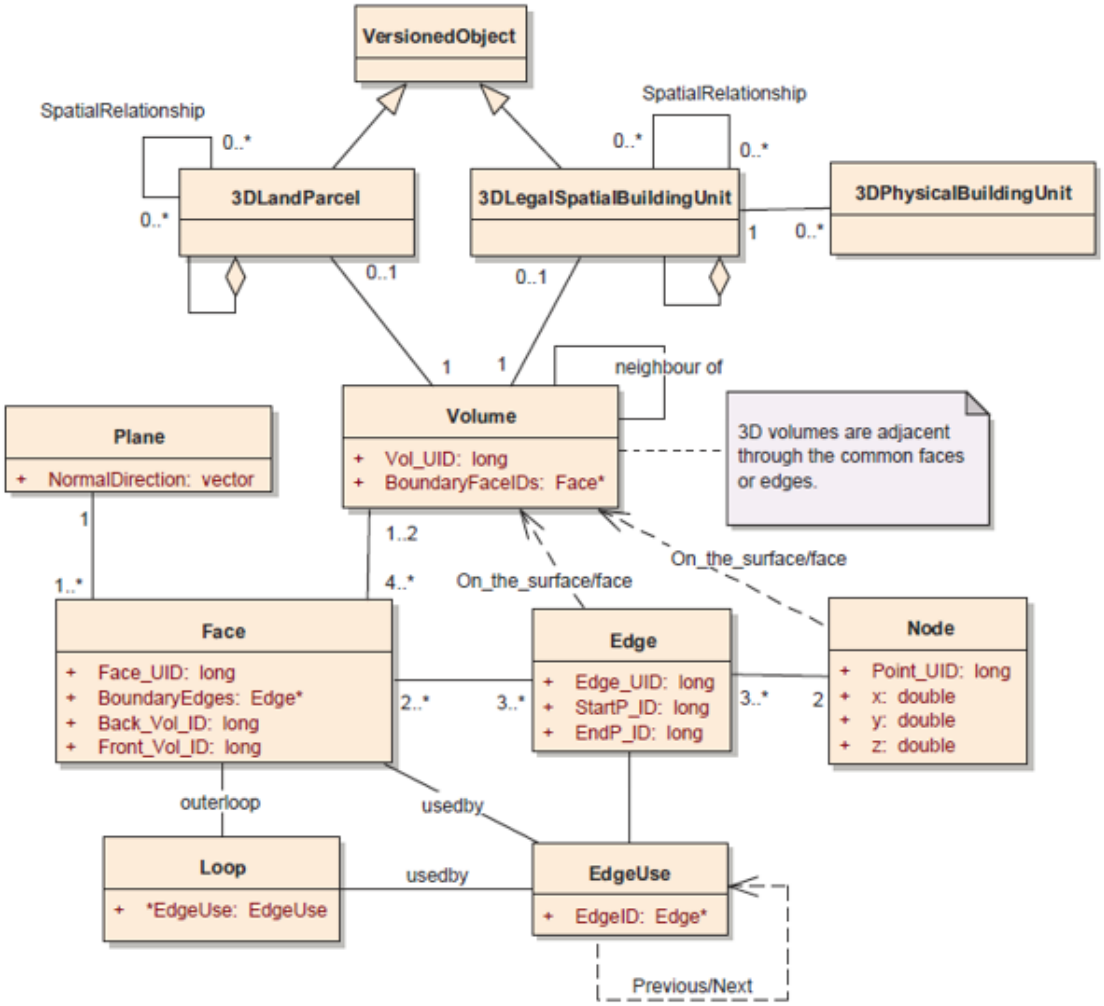

Figure 16. Data model in the prototype system (Ying et al., 2015)

The algorithm is capable of supporting various 3D shapes and non-manifold volumetric objects with holes or caves, and causes no problems with regard to the topological consistency. Real 3D volumetric objects are constructed first with the input faces, storing the references in the 3D

topological model (see Figure 16). A valid volume is made up of and closed by at least four faces with their normal directions. A class Plane is designed to emphasize the face's normal direction, which means that every face used in the body is only a half-plane face. A 3D volume is a 3D primitive to describe the volumetric object and is basically incident to faces, the lower dimensional 2D geometric primitive. The volumetric model is defined as a seamless 3D space with interior orientation, and commonly its shells which, closed and made up of the faces, together completely separate the interior and exterior of the volume; volumes cannot intersect and penetrate mutually. An important condition of Face is that its normal direction points outward or inward to the volume, which is essential for volume construction. The face's normal direction determines the interior orientation of the 3D volume, and Class Face is an oriented facet or patch with one outer loop, and zero or more inner loops. In general, the term face denotes a simple flat face that is used to define a part of the boundary.
Ding et al. (2016) propose a modelling approach for the 3D cadastral object based on extrusion. The approach does not allow overlapping among footprints which are used to construct one or more 3D objects. Based on this approach, one can extract 2D topological features from 2D footprints. Then 2D topological features and height values are used to present topological features. Using 2D feature to present 3D feature can save storage space. They used this approach in a case study and conclude that there is still need for a lot of practice to verify its availability for 3D cadastre.

6. FROM THEORY TO PRACTICE

6.1 General 3D geometry/topology capabilities

Due to the complexity of real-world spatial objects, various types of representations (for example vector, raster, constructive solid geometry, etc.) and spatial data models (topology, and geometry) have been investigated and developed. Promising developments were observed in the SDBMSs domain where more spatial data types, functions and indexing mechanism were supported. In this respect, SDBMSs are expected to become a critical component development of an operational 3D GIS. However, the native 3D support at SDBMS level has to be achieved (Khuan et al., 2008). Mostly all the main spatial database management systems (for example Oracle Spatial, PostgreSQL/PostGIS, Microsoft SQL server) support the Simple Feature Access international standard supporting 3D geometries (Janečka and Kára, 2012).

6.2 Oracle Spatial

The spatial features in Oracle Spatial consist of a set of object data types, type methods, and operators, functions, and procedures that use these types. A geometry is stored as an object, in a single row, in a column of type SDO_GEOMETRY. Spatial index creation and maintenance is done using basic DDL (CREATE, ALTER, DROP) and DML (INSERT, UPDATE, DELETE) statements. The text in this part is mostly based on the official Oracle Spatial 12g documentation[3].

[3] http://docs.oracle.com/database/121/SPATL/create-index.htm *(accessed on 19 September 2017)*

Geometry types

A geometry (in Oracle Spatial) is an ordered sequence of vertices that are connected by straight line segments or circular arcs. The semantics of the geometry are determined by its type. Oracle Spatial supports several primitive types, and geometries composed of collections of these types, including two-dimensional: points and point clusters, line string, n-point polygons, arc line strings (all arcs are generated as circular arcs), arc polygons, compound polygons, compound line string, circles, optimized rectangles. Spatial also supports the storage, indexing (R-tree) and retrieval of three-dimensional and four-dimensional geometric types, where three of four coordinates are used to define each vertex of the object being defined. The three-dimensional spatial data can include: points, point clouds (collection of points), lines, polygons, surfaces, and solids.

Table 1. SDO_GEOMETRY attributes for three-dimensional geometries (*here only Solid and Multisolid are mentioned*)

Type of 3D Data	SDO_GTYPE	Element Type, Interpretation on SDO_ELEM_INFO
Solid	3008	Simple solid formed by a single closed surface: one element type (*SDO_ETYPE, see table 2*) 1007, followed by one element type 1006 (the external surface) and optionally one or more element type 2006 (internal surfaces) Composite solid formed by multiple adjacent simple solids: one element type 1008 (holding the count of simple solids), followed by any number of element type 1007 (each describing one simple solid)
Multisolid	3009	Element definitions for one or more simple solids (element type 1007) or composite solids (element type 1008)

Table 2. Values and semantics in SDO_ELEM_INFO

SDO_ETYPE	SDO_INTERPRETATION	Meaning
1006 or 2006	$n > 1$	Surface consisting of one or more polygons, with each edge shared by no more than two polygons. A surface contains an area but not a volume. The value n in the Interpretation column specifies the number of polygons that make up the surface. The next n triplets in the SDO_ELEM_INFO array describe each of these polygon subelements. A surface must be three-dimensional.

SDO_ETYPE	SDO_INTERPRETATION	Meaning
1007	*n* = 1 or 3	Solid consisting of multiple surfaces that are completely enclosed in a three-dimensional space, so that the solid has an interior volume. A solid element can have one exterior surface defined by the 1006 elements and zero or more interior boundaries defined by the 2006 elements. The value n in the Interpretation column must be 1 or 3. Subsequent triplets in the SDO_ELEM_INFO array describe the exterior 1006 and optional interior 2006 surfaces that make up the solid element. If n is 3, the solid is an optimized box, such that only two three-dimensional points are required to define it: one with minimum values for the box in the X, Y, and Z dimensions and another with maximum values for the box in the X, Y, and Z dimensions.

Spatial Indexing

A spatial index (that is, a spatial R-tree index) must be created on each geometry column in the tables for efficient access to the data. For example, the following statement creates a spatial index named *territory_idx* using default values for all parameters:

```
CREATE INDEX territory_idx ON territories (territory_geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Spatial indexes can be built on two, three, or four dimensions of data. The default number of dimensions is two. To have any functions, procedures, or operators consider three dimensions, one must specify *PARAMETERS ('sdo_indx_dims=3')* in the CREATE INDEX statement when creating the spatial index on a spatial table containing for example geographic 3D data (longitude, latitude, ellipsoidal height). If one does not specify that parameter in the CREATE INDEX statement, a two-dimensional index is created.

The following statement creates a 3D spatial index named 3Dparcel_idx:

```
CREATE INDEX 3Dparcel_idx ON 3Dparcels (3Dparcel_geom) INDEXTYPE
IS MDSYS.SPATIAL_INDEX

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

A partitioned spatial index can be created on a partitioned table. A spatial index cannot be created on an index-organized table[4].

Extending Spatial Indexing Capabilities
Oracle Spatial enables the creation and use of spatial indexes on objects other than a geometry column. The SDO_GEOMETRY object can be embedded in a user-defined object type, and the geometry attribute of that type can be indexed. Further, one can create and use a function-based index where the function returns the SDO_GEOMETRY object.

Coordinate Reference System
The Oracle Spatial support for three-dimensional coordinate reference systems complies with the EPSG[5] model. There are two categories of three-dimensional coordinate reference systems: those based on ellipsoidal height (geographic 3D) and those based on gravity-related height (compound).

Geographic 3D Coordinate Reference Systems
A geographic three-dimensional coordinate reference system is based on longitude and latitude, plus ellipsoidal height. The ellipsoidal height is the height relative to a reference ellipsoid, which is an approximation of the real Earth. All three dimensions of the Coordinate Reference System (CRS) are based on the same ellipsoid.

[4] http://docs.oracle.com/database/121/SPATL/create-index.htm *(accessed on 19 September 2017).*
[5] The IOGP's EPSG Geodetic Parameter Dataset is a collection of definitions of coordinate reference systems and coordinate transformations which may be global, regional, national or local in application.. More on http://www.epsg.org/EPSGhome.aspx (*accessed on 19 September 2017*).

Compound 3D Coordinate Reference Systems

A compound three-dimensional coordinate reference system is based on a geographic or projected two-dimensional system, plus gravity-related height. Gravity-related height is the height as influenced by the Earth's gravitational force, where the base height (zero) is often an equipotential surface, and might be defined as above or below "sea level."

Gravity-related height is a more complex representation than ellipsoidal height, because of gravitational irregularities such as the following:

- Orthometric height - Orthometric height is also referred to as the height above the geoid. The geoid is an equipotential surface that most closely (but not exactly) matches mean sea level. An equipotential surface is a surface on which each point is at the same gravitational potential level. Such a surface tends to undulate slightly, because the Earth has regions of varying density. There are multiple equipotential surfaces, and these might not be parallel to each other due to the irregular density of the Earth.
- Height relative to mean sea level, to sea level at a specific location, or to a vertical network warped to fit multiple tidal stations. Sea level is close to, but not identical to, the geoid. The sea level at a given location is often defined based on the "average sea level" at a specific tidal gauge.

Using ellipsoidal heights enables Oracle Spatial to perform internal operations with great mathematical regularity and efficiency. Compound coordinate reference systems, on the other hand, require more complex transformations, often based on offset matrixes. Some of these matrixes have to be downloaded and configured. Furthermore, they might have a significant footprint, on disk and in main memory.

One can create a customized compound coordinate reference system, which combines a horizontal CRS with a vertical CRS (the horizontal CRS contains two dimensions, such as X and Y or longitude and latitude, and the vertical CRS contains the third dimension, such as Z or height or altitude). It means, that Oracle Spatial also supports 3D Cartesian coordinate reference systems (3D Geocentric). In this system, a point P is referred to by three real numbers (coordinates), indicating the positions of the perpendicular projections from the point to three fixed perpendicular graduated lines, called the axes which intersect at the origin.

Oracle Spatial also supports a local coordinate reference system. These refer to coordinate systems that are specific to an application. Several local coordinate systems are predefined and included with Spatial in the *SDO_COORD_REF_SYS* table. These supplied local coordinate systems, whose names start with Non-Earth, define non-Earth Cartesian coordinate systems based on different units of measurement (Meter, Millimeter, Inch, and so on).

6.3 PostGIS

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL. In addition to basic location awareness, PostGIS offers many features rarely found in other competing spatial databases such as Oracle Locator/Spatial and SQL Server. PostGIS adds extra types (geometry,

geography, raster and others) to the PostgreSQL database. The text in this part is mostly based on the official PostGIS 2.3.4 documentation[6].
It also adds functions, operators, and index enhancements that apply to these spatial types. These additional functions, operators, index bindings and types augment the power of the core PostgreSQL DBMS, making it a fast, feature-plenty, and robust spatial database management system.
The GIS objects supported by PostGIS are a superset of the "Simple Features" defined by the OGC. PostGIS supports all the objects and functions specified in the OGC "Simple Features for SQL" specification. PostGIS extends the standard with support for 3DZ, 3DM and 4D coordinates.
Some PostGIS functions related to solids:

- ST_IsSolid − Tests if the geometry is a solid. No validity check is performed.
- ST_MakeSolid − Casts the geometry into a solid. No check is performed. To obtain a valid solid, the input geometry must be a closed Polyhedral Surface or a closed TIN.
- ST_Volume − Computes the volume of a 3D solid. If applied to surface (even closed) geometries will return 0.

More information about all the spatial functions in PostGIS can be found in its official documentation.

Spatial Indexing
PostgreSQL/PostGIS supports three kinds of indexes by default: B-Tree indexes, R-Tree indexes, and GiST indexes. GiST is a generic form of indexing. In addition to GIS indexing, GiST is used to speed up searches on all kinds of irregular data structures (integer arrays, spectral data, etc.) which are not amenable to normal B-Tree indexing. The syntax for building a GiST index on a "geometry" column is as follows:

```
CREATE INDEX [indexname] ON [tablename] USING GIST ( [geometryfield] );
```

The above syntax will always build a 2D-index. To get the n-dimensional index supported in PostGIS 2.0+ for the geometry type, one can create one using this syntax:

```
CREATE INDEX [indexname] ON [tablename] U
SING GIST ([geometryfield] gist_geometry_ops_nd);
```

GiST indexes have two advantages over R-Tree indexes in PostgreSQL. Firstly, GiST indexes are "null safe", meaning they can index columns which include null values. Secondly, GiST indexes support the concept of "lossiness" which is important when dealing with GIS objects larger than the PostgreSQL 8K page size. Lossiness allows PostgreSQL to store only the

[6] http://postgis.net/docs/manual-2.3/index.html (*accessed on 19 September 2017*)

"important" part of an object in an index - 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.

BRIN Index
BRIN stands for "Block Range Index" and is a generic form of indexing that has been introduced in PostgreSQL 9.5. BRIN is a lossy kind of index, and its main usage is to provide a compromise for both read and write performance. Its primary goal is to handle very large tables for which some of the columns have some natural correlation with their physical location within the table. In addition to GIS indexing, BRIN is used to speed up searches on various kinds of regular or irregular data structures (integer, arrays etc.). Once a GIS data table exceeds a few thousand rows, one will want to build an index to speed up spatial searches of the data (unless all searches are based on attributes, in which case one will want to build a normal index on the attribute fields). GiST indexes are really performant as long as their size doesn't exceed the amount of RAM available for the database, and as long as one can afford the storage size, and the penalty in write workload. Otherwise, BRIN index can be considered as an alternative. The idea of a BRIN index is to store only the bounding box englobing all the geometries contained in all the rows in a set of table blocks, called a range. Obviously, this indexing method will only be efficient if the data is physically ordered in a way where the resulting bounding boxes for block ranges will be mutually exclusive. The resulting index will be really small, but will be less efficient than a GiST index in many cases.
Building a BRIN index is way less intensive than building a GiST index. It's quite common to build a BRIN index in less time than a GiST index would have required. As a BRIN index only store one bounding box for one to many table blocks, it is common to consume up to a thousand time less disk space for this kind of indexes[7].

Coordinate Reference Systems
The *SPATIAL_REF_SYS* table is a PostGIS included and OGC compliant database table that lists over 3000 known spatial reference systems and details needed to transform/reproject between them. The proj.4 library[8] does not contain all known projections and it is possible to define custom projections with proj.4 constructs.

6.4 3D topology
In the widely used SDBMSs such as Oracle Spatial, PostGIS, ESRI Geodatabase, 2D topology is well supported and documented. However, in most of current SDBMSs, 3D topology is not natively supported. So it is necessary to construct and store custom topology (see chapter 5 3D Topology structures).

6.4.1 Tetrahedral networks for modelling 3D topographic objects

[7] http://postgis.net/docs/manual-2.3/using_postgis_dbmanagement.html#idm2221 (*accessed on 19 September 2017*)
[8] proj.4 is a standard UNIX filter function which converts geographic longitude and latitude coordinates into cartesian coordinates (and vice versa), and it is a C API for software developers to include coordinate transformation in their own software. More on http://proj4.org/ (*accessed on 19 September 2017*)

For storing and modelling three-dimensional topographic objects (for example buildings, roads and terrain), tetrahedralisation have been proposed as an alternative to boundary representations. Penninga (2005) presented a modelling approach for 3D topography modelling based on tetrahedral network (TEN). The approach is based on two fundamental observations:

- The ISO 19101 Geographic information - Reference model defines a feature as an 'abstraction of real world phenomena'. These real world phenomena have by definition a volumetric shape. In modelling, often a lesser-dimensional representation is used in order to simplify the real world. Fundamentally there are no such things as point, line or area features; there are only features with a point, line or area representation (at a certain level of abstraction/generalization).
- The real world can be considered to be a volume partition. A volume partition can be defined (analogously to a planar partition) as a set of non-overlapping volumes that form a closed modelled space. As a consequence objects like 'air' or 'earth' are explicitly part of the real world and thus have to be modelled.

Four types of topographic features can be determined: 0D (point features), 1D (line features), 2D (area features) and 3D (volume features). For each type of feature simplexes of corresponding dimension are available to represent the features with, i.e. nodes, edges, triangles and tetrahedrons. A great advantage of using these simplexes is the well-defined character of the mutual relationships: a kD simplex is bounded by (k+1) geometrically independent simplices of (k-1) dimension (Pilouk, 1996). The important advantage of simplexes is the flatness of the faces, which enables one to describe a face using only three points. The next advantage is that every simplex, regardless its dimension, is convex, thus making convexity testing unnecessary (Penninga, 2005).
The topographic model is stored as a full TEN. The process of modelling topographic features consists of four discernible steps:

1. Start with four initial tetrahedrons, two 'air' and two 'earth' tetrahedrons;
2. Refine the earth's surface by inserting height information from a DEM;
3. Refine 'air' and 'earth' tetrahedrons in case of ill-shaped tetrahedrons by insertion of Steiner points;
4. Add real topographic features.

Triangulating or tetrahedronizing the features one-by-one before insertion in the topographic model reduces computational complexity and thus saves computer time. The results need to be inserted into the full topographic model. This requires the use of an incremental algorithm to avoid recomputing the whole model. As the complete topographic model (the TEN) will be stored in a spatial database, it is necessary to implement the incremental algorithm within the database. As a result a full DBMS approach is required, instead of using the database just to store results of the computations (Penninga, 2005).
Penninga (2008) proposed a DBMS data structure for storage of a constrained TEN. His simplicial complex-based method requires only explicit storage of tetrahedrons, while simplexes of lower dimensions (triangles, edges, and nodes), constraints and topological relationships can be derived in views. In this implementation, simplexes are encoded by their

vertices. He demonstrates, that storage requirements for 3D objects in tetrahedronised form (excluding the space in between these objects) and 3D objects stored as polyhedrons are in the same order of magnitude.
A TEN has favourable characteristics from a computational point of view. All elements of the tetrahedral network consist of flat faces (important for clear inside/outside decisions), all elements are convex and they are well defined, thus allowing relatively easy implementation of operations, such as validation of 3D objects (Penninga, 2008). A full volumetric approach contributes not only to improved analytical and validation capabilities, but also enables future integration of topography and other 3D data within the same volume partition (Penninga, 2008). Since the edit operations act as locally as possible, the resulting tetrahedronization is not necessarily of the best quality. To overcome this drawback, periodical quality improvements need to be made. Three types are distinguished: operators that add vertices, operators that remove vertices and operators that modify the TEN configuration through flips. Often, a complete TEN rebuild might be feasible to optimise TEN quality (Penninga, 2008).
Ledoux and Meijers (2013) proposed an alternative data structure for storing tetrahedralisation in a DBMS (see Figure 18). It is based on the idea of storing only the vertices and stars of edges; triangles and tetrahedra are represented implicitly. The structure permits one to store attributes for any primitives, and has the added benefit of being topological, which permits one to query it efficiently.

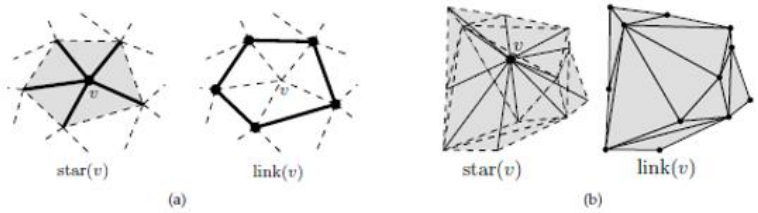

Figure 17. The star and the link of a vertex v in (a) 2D and (b) 3D (Ledoux and Meijers, 2013)

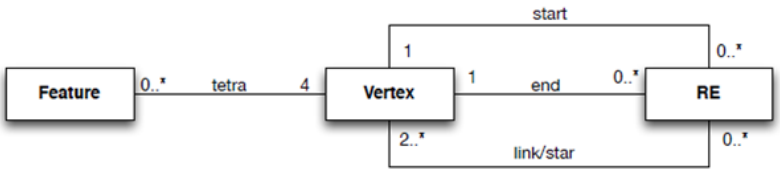

Figure 18. The UML diagram of the data model for star-based data structure (Ledoux and Meijers, 2013)

The strong point of the star-based structure is that it can be easily implemented in any DBMS supporting variable length arrays with two simple tables, and that no complex spatial index is needed (Ledoux and Meijers, 2013).

6.5 Point clouds and TINs

ESRI Geodatabase allows storing triangulated irregular network (TIN) as a planar graph where nodes are connected by edges to form triangles. Edges connect to nodes that are close to one another.
PostGIS has constructors for creating 3D geometry and has an extension and loader *pgpointcloud*[9] for storing point cloud data. It also includes extension for casting between point cloud data type and PostGIS geometry. TIN in PostGIS is modelled as a special case of polyhedral surface which is collection of adjacent triangles, which is similar to Microsoft SQL Server.
From the data structures point of view, Oracle Spatial is an example of SDBMS providing suitable data structures and mechanisms directly for TINs and point clouds. When the available specialized object types are used, a point cloud can be stored in a single row, in a single column in a user-defined table in Oracle Spatial. These object types related to point clouds and TINs are elaborated for example in (Janečka and Kára, 2012).
Martinez et al. (2014) used MonetDB and PostgreSQL with the point cloud data to understand the impact of the point cloud data on the different layers of a DBMS. It touches key issues from (adaptive) data loading to optimization of queries over point clouds. The results obtained through a micro benchmark illustrate both the capabilities to handle point cloud queries efficiently, as well as the relative merits of traditional index structures and compression techniques on the performance characteristics. They conclude, that MonetDB can be considered more modern than PostgreSQL, because it is designed from an in-memory perspective and relies on the operating system to move data between the storage hierarchies in an efficient manner. All queries are also highly parallel, using the cores available wherever possible. On the contrary, PostgreSQL represents the traditional buffer-based and iterator query engine approach. Tuning the buffer size to use all available memory by itself does not help because the logic of chasing data in buffers remains. Further they mention, that PostgreSQL does not by default support multi-core query processing.
Van Oosterom et al. (2015) designed a point cloud benchmark based on requirements from different groups of users within government, industry and academia. They analysed various data management systems: PostgreSQL, MonetDB, Oracle, and LAStools. They stated that the Oracle Exadata[10] with flat table model proved to be a very effective environment, both with respect to data loading and querying. Due to the massive parallel hardware engineered towards DBMS support, it was possible to load 23 billion points in less than 4:39 hours and storing the 12 Tb data from LAS files into a 2.2 Tb database (using 'query high' compression). In case of queries returning a very large number of points (from 10 million to over 1 billion), the system outperformed the other platforms.

7. RECENT DEVELOPMENTS OF SPATIAL DATABASES

7.1 nD-array Database Management Systems

Computer memory is inherently linear one-dimensional structure, mapping multi-dimensional data on it can be done in several ways. By far the two most common memory layouts for multi-

[9] https://github.com/pgpointcloud/pointcloud *(accessed on 21 August 2016)*
[10]http://www.oracle.com/technetwork/database/exadata/overview/index.html *(accessed on 21 August 2016)*

dimensional array data are row-major and column-major. When working with 2D arrays (matrices), row-major vs. column-major are easy to describe. The row-major layout of a matrix puts the first row in contiguous memory, then the second row right after it, then the third, and so on. Column-major layout puts the first column in contiguous memory, then the second, etc. (Bendersky, 2015).

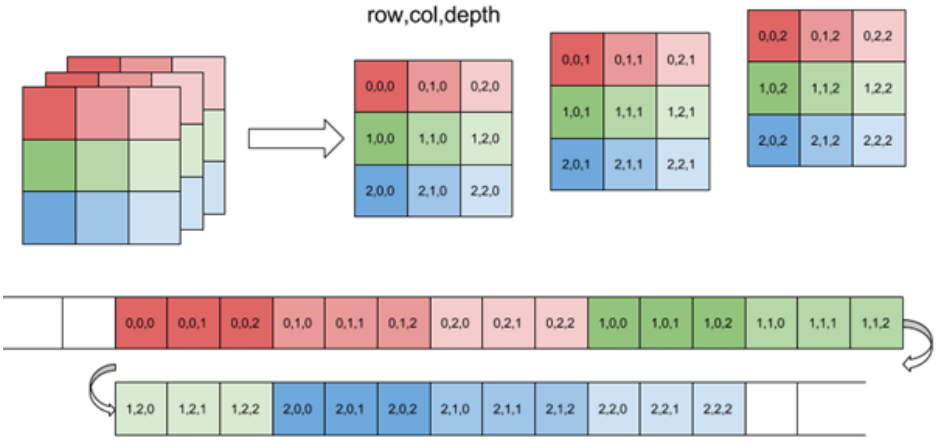

Figure 19. Mapping 3D array with $N_1 = N_2 = N_3$ in row-major (Bendersky, 2015)

The offset for a given element is:

$$offset = n_3 + N_3 * (n_2 + N_2 * n_1)$$

For example, the offset of the element with indices 2,1,1 is 22 (Bendersky, 2015).
While the database collection types set, list, and record have received in-depth attention, the fourth type, array, is still far from being integrated into database modeling. Due to this lack of attention there is only insufficient array support by today's database technology. This is surprising given that large, multi-dimensional arrays have manifold practical applications in earth sciences (such as remote sensing and climate modeling), life sciences (such as microarray data and human brain imagery), and many more areas (Bauman and Holsten, 2010).
To overcome this, large, multi-dimensional arrays as first-class database citizens have been studied by various groups worldwide. Several formalisms and languages tailored for use in array databases have been proposed and more or less completely implemented, sometimes even in operational use (Bauman and Holsten, 2010). Array Databases close a gap in the database ecosystem by adding modeling, storage, and processing support on multi-dimensional arrays (Baumann and Merticariu, 2015).
In the attempt towards a consolidation of the field Bauman and Holsten (2010) compare four important array database models: AQL, AML, ARRAY ALGEBRA, and RAM. As it turns out, ARRAY ALGEBRA is capable of expressing all other models, and additionally offers functionality not present in the other models. They show this by mapping all approaches to ARRAY ALGEBRA. This establishes a common representation suitable for comparison and allows us discussing the commonalities and differences found. Finally, a feasibility of conceptual array models for describing optimization and architecture was showed.
ARRAY ALGEBRA adopts an algebraic approach to array modeling. The targeted application domains of ARRAY ALGEBRA encompass sensor, image, and statistics data services. However, as stated in Bauman and Holsten (2010), current emphasis is on large-scale Earth Science (Gutierrez and Baumann, 2007) data.

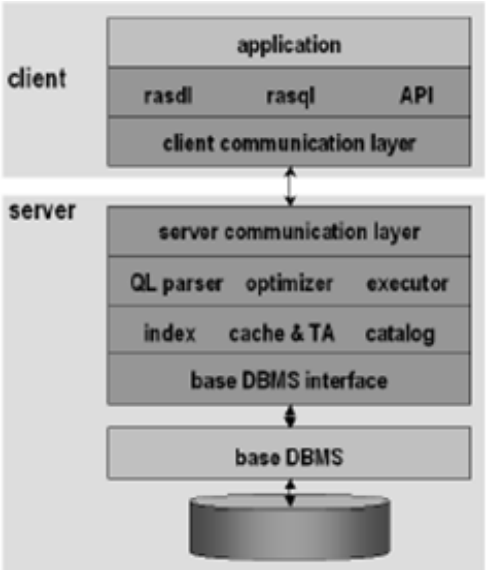

Figure 20. *RasDaMan* system architecture (dark grey) situated between application and base DBMS layers (light grey) (Baumann and Holsten, 2010)

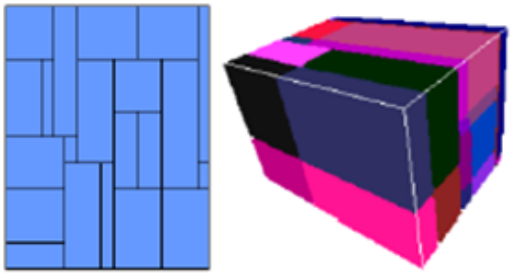

Figure 21. Sample 2-D and 3-D array tilings (Baumann and Holsten, 2010)

The *RasDaMan* array DBMS with its query language, rasql [11], implements ARRAY ALGEBRA. This system is in operational use since many years, among others as the geo raster server of the French National Geographic Institut where an airborne image map of dozen TB size is maintained. The *RasDaMan* implementation employs a middleware architecture where multidimensional arrays are partitioned into multi-dimensional sub-arrays called tiles. These tiles, which represent the units of disk access, are stored in BLOBs (binary large objects) inside some relational or object-oriented database, such as PostgreSQL or O2[12]. A spatial index helps to quickly determine the tiles affected by a query. Query processing relies on tile streaming: Physical query operators follow the open-next-close (ONC) protocol for reading their inputs tile by tile, and likewise they deliver their results in units of tiles. Based on this processing paradigm, the *RasDaMan* architecture follows a conventional multi-user DBMS approach, however, with all components crafted individually to accommodate the special needs of array

[11] http://rasdaman.org/browser/manuals_and_examples/manuals/doc-guides/ql-guide.pdf?order=name *(accessed on 16 August 2016)*
[12] http://www.sai.msu.su/sal/H/2/O2.html *(accessed on 16 August 2016)*

processing. Array definition and query languages, rasdl and rasql, are available to the application via command line tools, visual tools, and C++ and Java APIs. The client/server communication protocol connects clients to the DBMS server. A dispatcher distributes incoming queries among the *RasDaMan* server processes running. Each server process (see Figure 20) receives queries and parses, optimizes, and executes them. Auxiliary modules include catalogue manager, index manager, as well as cache and transaction manager. For example, the catalogue contains the array and collection type definitions against which semantic checks (like boundary checks for array dimensions not containing open limits) are performed during query analysis. The base DBMS interface layer abstracts from the particularities of the underlying DBMS. Adaptors exist for PostgreSQL, MySQL, Oracle, DB2, Informix, and the file system. Thereby, both array data, *RasDaMan*-internal array metadata, and non-array application data all end up in the same underlying database. As practice shows, this information integration considerable eases database administration (Baumann and Holsten, 2010).

In industrial world, for example Oracle offers the GeoRaster cartridge for 2-D geo raster imagery stored in a database. Instead of a rigorous embedding into SQL there are procedural constructs in PL/SQL which accomplish raster access as well as invocation of a set of predefined functions (Baumann and Holsten, 2010). PostGIS Raster is an extension to PostGIS which supports 2-D raster imagery through map algebra functions; unlike in *RasdaMan*, these are implemented as user-defined data types and, hence, not as tightly integrated and optimizable. PostGIS Raster generally is considered suitable for small and medium size rasters[13]. In the application domain, ARRAY ALGEBRA concepts have had much impact on the design of the Open GeoSpatial Consortium (OGC) Web Coverage Processing Service (WCPS) geo service standard (OGC, 2008) and several related OGC standards. Baumann and Holsten (2010) worked on extending the framework beyond arrays towards general meshes so as to allow retrieval on further spatiotemporal scientific data, such as Voronoi-type structures (adaptive grids can be handled already). They also investigated the seamless integration of arrays as first-class abstractions with standard SQL.

The *RasDaMan* system utilizes PostgreSQL as a backend to support point clouds through its WCS interface, thereby unifying grid and point cloud access (Baumann and Holsten, 2010).

In scaling out on point clouds, that are characterized by large numbers of points (going into the billion, and growing), relational databases possibly have a say again. For example, MonetDB[14] shows promising handling of point clouds in its column-store architecture (Martinez et al., 2014).

SciQL (Kersten et al., 2011) is a SQL-query language for science applications with arrays as first class citizens. It provides a seamless symbiosis of array-, set-, and sequence- interpretation using a clear separation of the mathematical object from its underlying storage representation. The language extends value-based grouping in SQL with structural grouping, i.e., fixed-sized and unbounded groups based on explicit relationships between its index attributes. The SciQL architecture benefits from a column store system with an adaptive storage scheme, including keeping multiple representations around for reduced impedance mismatch.

[13] http://lists.osgeo.org/pipermail/postgis-users/2014-April/039024.html *(accessed on 21 August 2016)*

[14] https://www.monetdb.org/Home *(accessed on 18 August 2016)*

SciDB is an open source data management system intended primarily for use in application domains that involve very large (petabyte) scale array data. SciDB is built to support an array data model and query language with facilities that allow users to extend system with new scalar data types and array operators (Brown, 2010).
Misev and Baumann (2014) proposed a generic model, ASQL, for modelling and querying multi-dimensional arrays in ISO SQL. The model integrates concepts from the three major array models seen today: *RasDaMan*, SciQL, and SciDB. It is declarative, optimizable, minimal, yet powerful enough for application domains in science, engineering, and beyond. ASQL has been implemented and is currently being discussed in ISO for extending standard SQL (Misev and Baumann, 2014).

7.2 File based solutions vs. nD-array database management system

Management of large datasets including storage, structuring, indexing and query is one of the crucial challenges in the era of big data. Liu et al. (2016) benchmarked NetCDF file based solutions and a multidimensional (MD) array database management system applying chunked storage to determine the best solution for storing and querying large hydrological datasets. In total nine criteria are defined to compare MD array DBMSs, as a result SciDB is selected for benchmarking.
NetCDF is notable for its simple data model, ease of use, and portability. However, according to practical experience, traditional NetCDF solutions perform inefficiently in retrieving information from large spatio-temporal datasets for certain queries. This is caused by the way it stores variable values, which is known as contiguous storage structure. Basically, for a grid full of variable values in a certain spatial area, NetCDF stores values into a one-dimensional array according to a row-majored order. When managing large numbers of MD array datasets, it is natural to adopt a DBMS solution as it could provide an easier to use and more scalable alternative (Liu et al., 2016).
Within DBMS scope, MD array DBMS is optimized further for MD array data management. It can specify metadata and supports storage of MD arrays. It employs the chunked storage structure which divides a whole dataset into separate chunks with specified chunk sizes. Based on this storage structure, MD array DBMSs then apply specific array addressing and relative offset calculation to index values, which is proved to be of high query efficiency. Hence, Liu et al. (2016) aimed at investigating whether the MD array DBMS can achieve better performance in processing queries on large MD hydrologic datasets than classic non-chunked NetCDF solution and competitive performance when compared to chunked NetCDF-4 file based solution. Their research illustrates that for big hydrological array data management, the properly chunked NetCDF-4 solution without compression is in general more efficient than the SciDB DBMS.

7.3 GPU use and massive parallel architectures for processing large-scale geospatial data

Modern Graphics Processing Units (GPUs) are now capable of general computing (Hennessy and Patterson, 2011). GPUs that are capable of general computing are facilitated with Software

Development Toolkits (SDKs) provided by hardware vendors (Zhang et al., 2015c) see Figure 22.

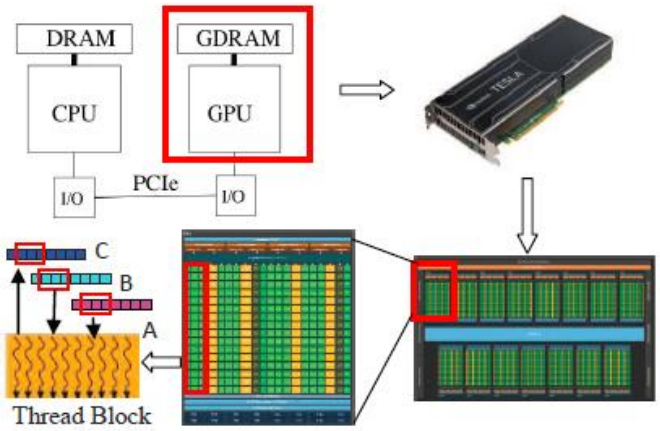

Figure 22. Illustration of GPU hardware Architecture (according to (Zhang et al., 2015c))

While geospatial data management techniques have been provided by both Spatial Databases and Geographical Information Systems (GIS), existing software is incapable of processing large-scale geospatial data for practical applications (Zhang et al., 2014). Quickly evolving processor, storage and networking technologies require new Big Data research to understand how new hardware impacts the performance of large-scale data processing.

In the past few years, the simplicity of the MapReduce computing model and its support in the open source Hadoop system have made it attractive to develop distributed geospatial computing techniques on top of MapReduce/Hadoop (Cary et al., 2009). The success of SpatialHadoop (Eldawy and Mokbel, 2013) and HadoopGIS (Aji et al., 2013) has demonstrated the effectiveness of MapReduce-based techniques for large-scale geospatial data management where parallelisms are typically identified at the spatial partition level which allows adapting traditional serial algorithms and implementations within a partition (Zhang et al., 2015c).

GPU-equipped computing nodes have much higher ratios between floating point computing power (in the order of floating point operations per second (flops), nowadays teraflops (Tflops) and fast growing) and network bandwidth (in the order of Gbps and remains stable) than regular computing nodes at which Hadoop-based systems are targeting. The gap makes efficient and scalable processing of large-scale data challenging, especially for geospatial data, whose processing is both data intensive and computing intensive (Zhang et al., 2015b).

Several techniques for processing large-scale geospatial data have been developed on both single computing nodes and clusters equipped with GPUs (You et al., 2015a; You et al., 2015b; Zhang et al., 2015a; Zhang et al., 2014).

Zhang et al. (2015c) report their work on data parallel designs for several geospatial data processing techniques. By further integrating these GPU-based techniques with distributed

computing tools, including Message Passing Interface[15] (MPI) library in the traditional High-Performance Computing (HPC) clusters and newer generation of Big Data systems (such as Impala[16] and Spark[17] for Cloud computing, it is possible to scale the data parallel geospatial processing techniques to cluster computers with good scalability.
While being aware of the complexities in developing a spatial database on GPUs, Zhang et al. (2015c) demonstrated the feasibility and efficiency of GPU-based geospatial processing, especially for large-scale data, developed modules for major geospatial data types and operations that can be directly applied to practical applications and developed a framework to integrate multiple GPU-based geospatial processing modules into an open system that can be shared by the community.

8. DISCUSSION

8.1 Modelling 3D parcels

Beside the non 2-manifold geometries (see chapter 2.1 Vector representation) for representation of 3D parcels there could be a need of further 3D Cadastre specific geometries: partly open solids and curved surfaces (boundaries).
Zlatanova et al. (2006) present design of freeform types to be considered for SQL Implementation Specifications (i.e. for an implementation in DBMS). They implemented the new geometries in Oracle Spatial as individual data types outside the SDO_GEOMETRY model. They showed that non-uniform rational basis spline (NURBS) is a very general representation of freeform shapes and demonstrated that appropriate data types for efficient management of freeform surfaces can be created at DBMS level. They argue, that many issues have to be further investigated. For example, the validation rules for freeform curves and surfaces have to be further specified, relevant functions for support at DBMS level have to be determined, spatial indexing have to be also considered.
Regarding the partly open solids, Thompson and Oosterom (2006) introduced a concept of the regular polytope. Figure 23 shows how a region ("convex polygon") can be defined as the intersection of a number of half spaces. A regular polytope is then defined as the union of a finite set of (possibly overlapping) non empty convex polytopes (Thompson and Oosterom, 2011).

[15] http://en.wikipedia.org/wiki/Message Passing Interface *(accessed on 22 August 2016)*
[16] http://impala.io *(accessed on 22 August 2016)*
[17] http://spark.apache.org *(accessed on 22 August 2016)*

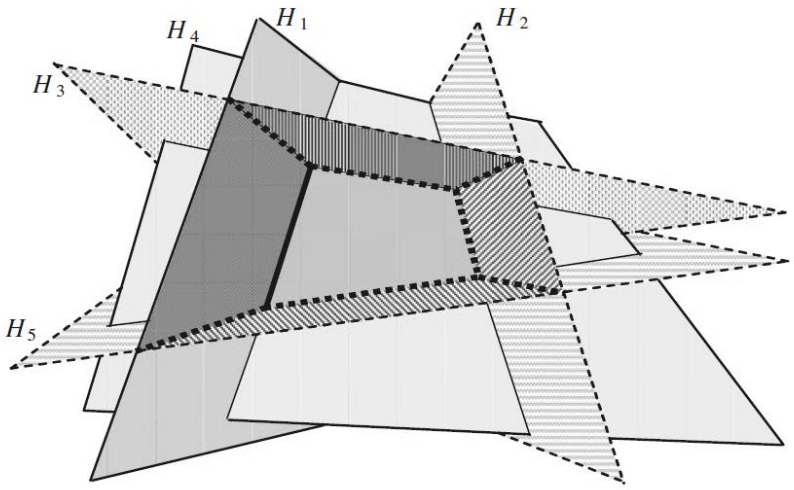

Figure 23. A convex region defined by a set of half spaces (Thompson and Oosterom, 2006)

The regular polytope, since it does not need to be bounded on all sides is a natural representation for a mix of 2D and 3D parcels (Thompson and Oosterom, 2006).

8.2 Validation of 3D solids

Spatial DBMS should enable validation of 3D solids. Ledoux (2014) mentions several possible extensions of validation of 3D solids. For the modelling of 3D buildings, the semantics information can be used. For example, if for instance one surface is labelled as the roof of the building, then an extra validation rule (over the geometry) would be to ensure that the roof is located "above" the surface labelled as the ground floor. Furthermore, the automatic repair of invalid solids could be considered.

8.3 3D Spatial Constraints

Xu et al. (2017) give suggestions regarding the future work dealing with 3D spatial constraints:

- The pseudo 3D Geo-OCL expressions need to be tested in conjunction with the UML diagrams.
- It would be useful to extend OCL code generation tools to enable automatic model translation from OCL (especially spatial constraints) to SQL.
- Further study can be conducted into detecting contradicting (spatial and non-spatial) constraints.
- Corresponding functions in database need to be developed, esp. 3D and solids related, to implement 3D spatial predicates from extended OCL.
- Test the performance of the trigger that uses 3D geometric operators.

8.4 3D topology

As previously elaborated, a suitable 3D topology model for 3D cadastre seems to be an approach based on a Tetrahedral Network (TEN), proposed by Penninga and van Oosterom

(2008): the “topological structure to organize tetrahedrons”. However, the TEN model need to be synchronized, described in a new spatial profile, with LADM specifications. As mentioned in Zulkifli et al. (2015), the future work is to develop a conceptual model of the TEN based on LADM standard. Then, the proposed conceptual models (i.e. 2D and 3D topology) should be translated into physical model to develop a prototype cadastral registration.
A full topological model for the 3D cadastre, land planning and management is needed for the following reasons: (1) to utilize the surveying boundaries to generate the 3D cadastral objects (the term “volumetric model” is used geometrically and topologically); (2) to represent the 3D volumetric objects with high quality, and consistent topology without intersection; and (3) for rapid topological queries necessary for real-time user interaction and management (Ying et al., 2015).
Another important aspect is the development of (spatial) indexes for topological models. Last but not least, operations on topological models, including conversion to geometric models, are important (Breunig and Zlatanova, 2011).
The legal and physical object proposed by (Aien 2015 et al) and the 9-intersection model by (Egenhofer and Herring, 1990) to define spatial relationships can advance the 3D topological analysis related to boundaries. To define the boundaries of a 3D RRR, the adjacency matrix for representing the relationship between legal and physical objects can be constructed. In this approach, one could analyse the 3D RRRs in relation to the physical objects and form the adjacency matrix. This will enable support of a range of common queries about the 3D RRR boundaries. This includes queries such as: “What are the 3D rights associated with this property?”, “What are the rights associated with an apartment unit?” and “what is the association of an infrastructure with the surrounding RRRs?”

8.5 Point clouds and TINs

Van Oosterom et al. (2015) state that at least two closely related level of standardization must be considered: (a) Database Structure Query Language (SQL) extension for point clouds, and (b) Web Point Cloud Services (WPCS) for progressive transfer based on multi-scale or vario-scale LoD.
Janečka and Kára (2012) suggest to extend the point cloud and TIN related data structures available in production spatial databases to enable storage of additional non-spatial attributes (semantic) related, for example to the particular point (or set of points). Such information can be then used, for example, during the update of the stored 3D geometries directly inside the spatial database.

8.6 Usage of GPU clusters for processing geospatial data

Balancing latency and throughput has profound implications in Big Data research. While traditional parallel and distributed databases are mostly targeted at reducing data processing latency for moderately sized datasets, Big Data systems need to take ownership costs and energy consumption into consideration. Using large quantities of small processors to achieve similar throughputs while reducing energy footprint is becoming an increasingly important topic in Big Data research (Zhang et al., 2015b). Motivated by the increasing gap between the computing power of GPU-equipped clusters and network bandwidth and disk I/O throughput,

Zhang et al., (2015b) proposed a low-cost prototype research cluster made of NVidia TK1 SoC[18] boards that can be interconnected with standard 1 Gbps network to facilitate Big Data research. They evaluate the performance of the tiny GPU cluster for spatial join query processing on large-scale geospatial data. Experiments on point-in-polygon test based spatial join using two real world applications with tens to hundreds of millions of points and tens of thousands of polygons have demonstrated the efficiency of the solution when compared with SpatialSpark. The future work should incorporate not only including processors, but also memory, disk and network components. Furthermore, the performance of GPU cluster should be evaluated using more real world geospatial datasets and applications, for example, distance and nearest neighbour based spatial joins (Zhan et al., 2015b).

In the age of Big Data it is not sufficient any longer that each research domain pursues its own ways of finding solutions, often reinventing the wheel or, conversely, inventing inadequate wheels. Specifically, the geoinformatics domain and core computer science domains like databases, Web services, programming languages, and supercomputing, share challenges seen from different angles. It is not too infrequent that similar ideas appear in different fields. For example, array databases offer declarative query languages on large n-D arrays which internally are partitioned for efficient access to subsets. SciHadoop is an approach independent from databases where an array-tuned query language is put on top of Hadoop. Data formats like TIFF and NetCDF also support the concept of array partitioning. It is worthwhile, therefore, to extend this small, focused survey into a larger one incorporating more domains and also implementation aspects. Fostering exchange, therefore, seems promising (Baumann, 2014).

9. CONCLUSIONS

The use of land in the vertical dimension has necessitated the creation and maintenance of 3D cadastre. The use and generation of 3D data, both cadastral and non-cadastral has increased greatly. The major technological and business drivers for the growth are sensor and hardware capabilities for capture and utilisation of large point clouds; 3D visualisation is mainstream but 3D analysis not yet; managing 3D data and bridging the gap between point cloud and GIS, CAD BIM systems; and the necessity to use 3D data to better describe the real world. Organisations are not yet in 3D because 3D modelling is more complex than 2D, converting 2D data to 3D is difficult, it requires migration from a simple to a complex data structure, economic viability, and a lack of user friendly 3D analyses tools that are yet to be developed.

Three-dimensional data models and their topological relationships are two important parts of 3D spatial data management. The expectations from a 3D spatial system are to enable data models that handle a large variety of 3D objects, automated data quality checks, search and analysis, data dissemination, 3D rendering and visualisation and close linkages to standards. Although a lot of work has been completed on defining a 2D or 3D vector geometry in standards by the OGC and the ISO, it is still insufficient to define 3D cadastral objects. 3D objects have a more rigorous definition for cadastral purposes. For a volumetric 3D cadastral object, for example, the polyhedron needs to satisfy characteristics such as closeness, interior connection,

[18] http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html *(accessed on 21 August 2016)*

face construction and proper orientation. The LADM addresses many of the issues in 3D representation and storage of 3D data in a DBMS. It allows in-row storage of 3D data in a mixed 2D-3D database allowing for fast retrievals and analysis; it allows for 3D data to be stored in different levels of detail, overlapping 2D footprint of 3D objects, and supports liminal parcels, as well as allows attribution of different boundary lines and faces. However, an identified issue is the duplication of definition of boundaries for separate spatial units.

Three-dimensional objects can be represented using voxels (volumetric pixels) as it brings advantages in object representation, object count and volume, 3D operations and simple analysis, better representation of the various levels of detail of a 3D city model, and representing 3D as a solid instead of point, line and polygon. The challenges to this are the storage and efficient handling by current spatial databases, although there are GIS systems that are working towards creating a column store structure to accommodate voxels. 3D objects can also be represented as a point cloud. LiDAR point clouds could assist to either be a reference framework of as-constructed features, or a 3D data acquisition tool for 3D physical objects, or a verification tool for pre-existing BIMs or other models. Point cloud data can be for data such as administrative, vector, raster, temporal etc. and a generic DBMS should be able to combine these data for a point cloud data type with characteristics such as xyz values, attributes per point, spatially coherent data organisation, efficient storage and compression, data pyramid support for multi-scale or vario-scale support, temporal support, query accuracy over a range of dimensions, analytical functions and parallel processing.

Spatial indexing is used by databases to improve search speeds, of the three types of indexes namely B-Tree, R-Tree and GiST, the latter two are found to be useful for GIS data. As with 2D geometry, 3D volumetric primitives would need to satisfy the adjacency and incidence (gaps and overlaps) relationship so that they are mutually exclusive and spatially exhaustive in the domain. While standards and definitions for solids such as the PolyhedralSurface in the SQL Geometry Types of OGC as well as other definitions for solids exist, they are not utilised very well currently and do not comply very well with standards. Validation of such solids and exchange of datasets between formats and platforms are highly problematic and do not usually follow any standards and error reports are usually cascading rather than in a single report making it very cumbersome to deal with errors individually.

Operations on and amongst 3D objects have been described by OGC, such as 3D architecture (Envelope(), IsSimple(), Is3D() etc.) and Spatial relationships (Equals(), Intersects(), Touches() etc.), however existing DBMS often implement them differently. 3D topological structures are an important consideration in a 3D cadastral DBMS. Topological relationships between neighbouring parcels can be between two objects or between many of the objects neighbourhood parcels. While 3D topological structures have been defined, they have not fully compliant to standards such as the LADM. The LADM not only provides a conceptual description of a land administration system, but also provides a 3D topology spatial profile. LADM also stipulates that geometrical information along with an associated topological primitive help to describe 3D spatial units.

LADM volumes can be bounded or unbounded at the top or bottom which is a reflection of real-world situations where there may be limited or unlimited rights or restrictions on the ground or skyward direction of a volumetric property. Various methods and characteristics of

constructing 3D spatial units using LADM 3D topological model have been discussed in this paper in the context of a LADM specific topological model since a single model is not suitable for all types of applications. The approach based on the Tetrahedral Network (TEN) model is a suitable 3D topological model for volumetric parcels and is proposed as an alternative to boundary representation. Two fundamental considerations are that real-world phenomena have a volumetric shape, and can be considered a volumetric partition assist in modelling of 3D space. All elements of a TEN are convex and are well-defined allowing easy validation, analytical capabilities and integration with topography and other 3D data. TEN can be stored as explicit tetrahedrons or as vertices and the star or edges. Another method is to construct and perform topological validations of 3D cadastral objects on the fly based on boundary 3D face information. This can create both manifold and non-manifold solids and can model real-world cadastral features and legal spaces. The validation requirements for volumes are reduced and rely on the algorithm to create the volume using 3D faces and stored references. Finally, another approach is to use 2D topological features with stored height values, which is then used to construct and validate 3D topological features. This approach can save storage space but is not totally viable for a 3D cadastre.

A section has been devoted to discuss the current software available to link current spatial DBMS possibilities to functional requirements and to focus on implementation and application. Developments were observed in the SDBMS domain where more spatial data types, functions and indexing mechanisms were supported. Two available SDBMS, Oracle Spatial and PostGIS were analysed in detail, while other SDBMS such as Microsoft SQL Server, MySQL have been seen to follow Simple Feature Access international standard. Most of these software including ESRI support 2D topology very well, however 3D topology is not supported natively yet. Comparison of various SDBMS for storing, and representing large point clouds was done with various software excelling in some aspects. ESRI's TIN structure, Oracle Spatial providing suitable data structure and mechanisms, MonetDB's in-memory perspective rather than a buffer perspective and ability to move data between storage hierarchies, Oracle Exadata's flat table model for data loading and querying and handling large number of points are some of the features of the current SBDMs.

A discussion on recent development of spatial databases follows with discussion on nD-array DBMS, comparison between file-based solutions vs. nD-array DBMS, and the development of modern Graphics Processing Units (GPUs) and their use in massive parallel architectures for processing large-scale geospatial data. In conclusion, the paper proposes a 3D topology model based on TEN synchronised with LADM specifications and the development of conceptual and physical model seems to be suitable for 3D cadastre and 3D registration. This topological model would utilise surveying boundaries to generate 3D cadastral objects with consistent topology and rapid query and management. Definitions for the validation of 3D solids should also consider the automatic repair of invalid solids. Point cloud and TIN related data structures available in SDBMSs should enable storage of non-spatial attributes such that database updates would store all relevant information directly inside the spatial database.

REFERENCES

Abadi, D., J., Madden, S., Hachem, N. (2008). Column-stores vs. row-stores: how different are they really? In Proceedings of the ACM SIGMOD, 2008.

Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J. (2013). Hadoop-gis: A high performance spatial data warehousing system over mapreduce. In VLDB, 6(11), pp. 1009–1020.

Baumann, P. (2014). Are Databases of Any Use in Modern Geo Services? In: Proceedings of FOSS4G-Europe 2014, Bremen. June 15-17 2014.

Baumann, P., Holsten, S. (2010). A Comparative Analysis of Array Models for Databases. In: Database Theory and Application, Bio-Science and Bio-Technology. Volume 258 of the series Communications in Computer and Information Science, Springer, pp. 80-89. doi: 10.1007/978-3-642-27157-1_9

Baumann, P., Merticariu, V. (2015) On the Efficient Evaluation of Array Joins. IEEE International Conference on Big Data. Santa Clara, CA. doi: 10.1109/BigData.2015.7363986

Bendersky, E. Memory layout of-multi-dimensional arrays. Available online: http://eli.thegreenplace.net/2015/memory-layout-of-multi-dimensional-arrays/ (accessed on 16 Aug 2016)

Billen R., Zlatanova, S., Mathonet, P., Boniver, F. (2002). The Dimensional Model: a framework to distinguish spatial relationships, in: Advances in Spatial Data handling, D. Richardson, P. van Oosterom (Eds.), Springer, pp. 285-298.

Borrmann, A. and Rank, E. (2009). Topological analysis of 3D building models using a spatial query language. Advanced Engineering Informatics 23(4), pp. 370–385.

Boss, H., Å. and Streilein, A. (2014). 3D Data Management – Relevance for a 3D Cadastre Position Paper 3. In: Proceedings of the 4th International Workshop on 3D Cadastres. 9-11 November 2014, Dubai, United Arab Emirates. ISBN 978-87-92853-28-8.

Breunig, M. and Zlatanova, S. (2011). 3D geo-database research: Retrospective and future directions. Computers & Geosciences 37, pp. 791-803. doi:10.1016/j.cageo.2010.04.016

Brown, P., G. (2010). Overview of SciDB: large scale array storage, processing and analysis. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, Indiana, USA, ACM SIGMOD Record, ACM Press, 2010, pp. 963-968.

Brugman, B., Tijssen, T. and van Oosterom, P., 2011. Validating a 3D topological structure of a 3D space partition. In: Advancing Geoinformation Science for a ChangingWorld, Springer, pp. 359–378.

Cary, A., Sun, Z., Hristidis, V., Rishe, N. (2009). Experiences on processing spatial data with mapreduce. In SSDBM, pp. 302–319.

Ding, Y., C. Wu, et al. (2016). Construction of geometric model and topology for 3D cadastre – Case study in Taizhou, Jiangsu. FIG working week 2016. Christchurch, New Zealand.

Ellul, C. (2007). Functionality and Performance – Two Important Considerations when implementing Topology in 3D. Ph.D. Thesis. University of London.

Egenhofer, M. J., (1995). Topological relations in 3-D. Technical report, National Center for Geographic Information and Analysis and Department of Spatial Information Science and Engineering Department of Computer Science university of Maine.

Eldawy, A. and Mokbel, M. (2013). A demonstration of spatialhadoop: an efficient mapreduce framework for spatial data. In VLDB, 6(2), pp. 1230–1233.
Ghawana, T. and Zlatanova, S. (2010). Data consistency checks for building a 3D model: a case study of Technical University, Delft Campus, The Netherlands. Geospatial World (4). ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-2/W1, ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, 27 – 29 November 2013, Istanbul, Turkey.
Gonçalves, R., Zlatanova, S., Kyzirakos, K., Nourian, P., Alvanaki, F., van Hage, W. (2016). A columnar architecture for modern risk management system. 2016 IEEE 12th International Conference on e-Science (e-Science), Baltimore, MD, 2016, pp. 424-429.
Gutierrez, A., G. and Baumann, P. (2007). Modeling fundamental geo-raster operations with array algebra. In Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA, pages 607–612. IEEE Computer Society.
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
Hennessy, J., L. and Patterson. D. A. (2011). Computer Architecture: A Quantitative Approach, 5th ed. Morgan Kaufmann.
Herring, J, 2001, Topic 1 Feature Geometry (Same as ISO 19107 Spatial Schema), available at http://www.iso.org
Idreos, S., Groffen, F., Nes, N., Manegold, S. and et al. (2012). Monetdb: Two decades of research in column-oriented database architectures. IEEE Data Engineering Bulletin.
ISO (2003). ISO 19107, Geographic information – Spatial Schema, ed. 1. ISO, Geneva, Switzerland
ISO (2012). ISO 19152, Geographic information – Land Administration Domain Model (LADM), ed. 1. ISO, Geneva, Switzerland.
Janečka, K. and Kára, M. (2012). Advanced Data Structures for Surface Storage. In: Proceedings of GIS Ostrava 2012 – Surface models for geosciences. VŠB-TUO, Ostrava. ISBN 978-80-248-2667-7.
Kalantari, M., Rajabifarad, A., Williamson, I., and Atazadeh, B. (2017). 3D Property Ownership Map Base for Smart Urban Land Administration. FIG working week 2017. Helsinki, Finland.
Kazar, B., M, Kothuri, R., Van Oosterom, P., Ravada, S. (2008). On valid and invalid three-dimensional geometries. In Van Oosterom P, Zlatanova S, Penninga F, and Fendel E (eds) Advances in 3D GeoInformation Systems. Berlin, Springer: 19–46.
Kersten, M., Nes, N., Zhang Y., Ivanova, M. (2011). SciQL, A Query Language for Science Applications. In: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pp. 1-12. Uppsala, Sweden.
Khuan, Ch., T., Rahman, A., A., Zlatanova, S. (2008). 3D Solids and Their Management in DBMS. In: Advances in 3D Geoinformation Systems. Lecture Notes in Geoinformation and Cartography, pp. 279-311. 10.1007/978-3-540-72135-2_16

Kolbe, P. D. T. H., König, G., Nagel, C., & Stadler, A. (2009). 3D-Geo-Database for CityGML. Institute for Geodesy and Geoinformation Science, Technische Universität Berlin, 2.0.1 edition.

Ledoux, H. (2014). On the validation of solids represented with the international standards for geographic information. Computer-Aided Civil and Infrastructure Engineering, 28(9):693–706. doi: http://dx.doi.org/10.1111/mice.12043

Ledoux, H. and Meijers, M. (2013). A star-based data structure to store efficiently 3D topography in a database. Geo-spatial Information Science, 16(4):256–266.

Ledoux, H. and Meijers, M. (2009). Extruding building footprints to create topologically consistent 3d city models. Urban and Regional Data Management, UDMS Annuals pp. 39–48.

Ledoux, H., Verbree, E., Si, H. (2009). Geometric Validation of GML Solids with the Constrained Delaunay Tetrahedralization. In: the Proceedings of the 4th International Workshop on 3D Geo-Information, 2009, pp. 143–148. Ghent, Belgium

Lee, J. and Zlatanova, S. (2008). A 3D data model and topological analyses for emergency response in urban areas. In: Geospatial information technology for emergency response, Publisher: Taylor and Francis, pp.143-168

Liu, H., van Oosterom, P., Hu, C., and Wang, W. (2016). Managing Large Multidimensional Array Hydrologic Datasets: A Case Study Comparing NetCDF and SciDB, In: Procedia Engineering, 154, pp. 207-214.

Martinez-Rubi, O., Kersten, M. L., Goncalves, R., Ivanova, M. (2014) A Column-Store Meets the Point Clouds. FOSS4G-Europe Academic Track.

Misev, D., Baumann, P. (2014) Extending the SQL Array Concept to Support Scientific Analytics. In: Proceedings of the 26th International Conference on Scientific and Statistical Database Management. ISBN: 978-1-4503-2722-0 doi:10.1145/2618243.2618255

OGC (2011). OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.

OGC (2010). OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option.

OGC (2008). Web Coverage Processing Service (WCPS) Implementation Specification. Number 08-068. Open Geospatial Consortium, 1.0.0 edition. Editor P. Baumann.

OGC (2007). Geography markup language (GML) encoding standard. Open Geospatial Consortium inc. Document 07-036, version 3.2.1.

Penninga, F. (2008). 3D Topography A Simplicial Complex-based Solution in a Spatial DBMS. Ph.D. thesis, TU Delft, Netherlands.

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. and van Oosterom, P., J., M. (2008). A Simplicial Complex-Based DBMS Approach to 3D Topographic Data Modelling. International Journal of Geographic Information Science, 22, pp. 751-779.

Pilouk, M. (1996). Integrated Modelling for 3D GIS, PhD thesis, ITC Enschede, Netherlands.

Ravada, S., Kazar, B.M., Kothuri, R. (2009). Query Processing in 3-D Spatial Databases: Experiences with Oracle Spatial 11g. 3D Geo-Information Sciences, pp.153-173. DOI: 10.1007/978-3-540-87395-2_10

Schön, B., Mosa, A., S., M., Laefer, D. F., Bertolotto, M. (2013). Octree-based indexing for 3D pointclouds within an Oracle Spatial DBMS. Computers & Geosciences 51, pp. 430-438.

Schön, B., Bertolotto, M., Laefer, D., F. (2009). Storage, manipulation, and visualization of LiDAR data. In: Remondino, F., El-Hakim, S., Gonzo, L. (Eds.) Proceedings of 3rd International Workshop, 3D-ARCH'2009: 3D Virtual Reconstruction and Visualization of Complex Architectures, Trento, Italy, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-5/W1, ISSN:1682-1777.

Stoter, J., Ledoux, H., Zlatanova, S., Biljecki, F. (2016). Towards sustainable and clean 3D Geoinformation. In Thomas H. Kolbe, Ralf Bill and Andreas Donaubauer (eds.), Geoinformationssysteme 2016: Beiträge zur 3. Münchner GI-Runde, Wichmann Herbert, Munich, Germany, February 2016, pp. 100–113.

Thompson, R. and Van Oosterom, P. (2011). Connectivity in the regular polytope representation. Geoinformatica 15, pp. 223-246.

Thompson, R. and Van Oosterom, P. (2012). Modelling and validation of 3D cadastral objects. Urban and Regional Data Management. S. Zlatanova, H. Ledoux, E. Fendel and M. Rumor. Leiden, Taylor & Francis. UDMS Annual 2011.

Thompson, R. and Van Oosterom, P. (2006). Implementation issues in the storage of spatial data as regular polytopes. In: Information Systems for Sustainable Development – Part I. UDMS 06, Aalborg.

Thompson, R., Van Oosterom, P., Soon, K.H., Priebbenow, R. (2016). A Conceptual Model Supporting a Range of 3D Parcel Representations Through all Stages: Data Capture, Transfer and Storage. FIG Working Week 2016. Christchurch, New Zealand.

Van Oosterom, P. (2013). Research and development in 3D cadastres. Computers, Environment and Urban Systems 40, pp. 1–6.

Van Oosterom, P., Martinez-Rubi, O., Tijssen, T., Gonçalves, R. (2016). Realistic Benchmarks for Point Cloud Data Management Systems, Chapter in: Advances in 3D Geoinformation (Alias Abdul-Rahman, ed.), pp. 1-30.

Van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., Gonçalves, R. (2015). Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. Computers & Graphics. Volume 49, pp. 92-125. http://dx.doi.org/10.1016/j.cag.2015.01.007

Van Oosterom, P., Stoter, J., Quak, W., Zlatanova, S. (2002). The balance between geometry and topology, In: Advances in Spatial Data Handling, 10th International Symposium on Spatial Data Handling, D. Richardson and P. van Oosterom (Eds.), Springer-Verlag, Berlin, pp. 209-224

Verbree, E. and Si, H. (2008). Validation and storage of polyhedra through constrained Delaunay tetrahedralization. In Cova T J, Miller H J, Beard K, Frank A U, and Goodchild M F (eds) GIScience 2008: Proceedings of the Fifth International Conference. Berlin, Springer-Verlag: 354–69

Xu, D., van Oosterom, P., Zlatanova, S. (2017). A Methodology for Modelling of 3D Spatial Constraints. In: Abdul-Rahman A. (eds) Advances in 3D Geoinformation. Lecture Notes in Geoinformation and Cartography. Springer, Cham.

Ying, S., Guo, R., Li, L., van Oosterom, P., Stoter, J. (2015) Construction of 3D Volumetric Objects for a 3D Cadastral System. Transactions in GIS. Vol. 19 Issue 5, pp. 758-779. 10.1111/tgis.12129

Ying, S., Guo, R., Li, L., van Oosterom, P., Ledoux, H., Stoter, J. (2011). Design and Development of a 3D Cadastral System Prototype based on the LADM and 3D Topology. In 2nd International Workshop on 3D Cadastres. Delft, the Netherlands.

You, S., Zhang, J., Gruenwald, L. (2015a). Scalable and Efficient Spatial Data Management on Multi-Core CPU and GPU Clusters: A Preliminary Implementation based on Impala. In: Proceedings of International Workshop on Big Data Management on Emerging Hardware (HardBD'15), Seoul, Korea.

You, S., Zhang, J., Gruenwald, L. (2015b). Large-Scale Spatial Join Query Processing in Cloud. In: Proceedings of International Workshop on Cloud Data Management (CloudDM'15), Seoul, Korea.

Zhang, J., You, S., Gruenwald, L. (2015a). A Lightweight Distributed Execution Engine for Large-Scale Spatial Join Query Processing. Technical report,

Available online: http://wwwcs.engr.ccny.cuny.edu/~jzhang/papers/lde_spatial_tr.pdf (accessed on 15 August 2016)

Zhang, J., You, S., Gruenwald, L. (2015b). Tiny GPU Cluster for Big Spatial Data: A Preliminary Performance Evaluation. In: 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops.

Zhang, J., You, S., Gruenwald, L. (2015c) Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters. SIGSPATIAL Special. Vol. 6 Issue 3, pp. 27-34. doi: 10.1145/2766196.2766201

Zhang, J., You, S., Gruenwald, L. (2014). Parallel Online Spatial and Temporal Aggregations on Multi-core CPUs and Many-Core GPUs. Information Systems, vol. 44, pp. 134–154.

Zhu, Q., Gong, J., Zhang, Y. (2007). An efficient 3D R-tree spatial index method for virtual geographic environments. ISPRS Journal of Photogrammetry & Remote Sensing 62, pp. 217-224.

Zlatanova, S. (2000). On 3D topological relationships, In: Proceedings of the 11th International Workshop on Database and Expert System Applications (DEXA 2000), 6-8 September, Greenwich, London, UK, pp. 913-919

Zlatanova, S., Nourian, P., Gonçalves, R., Vo, A., V. (2016). Towards 3D raster GIS: On developing a raster engine for spatial DBMS. ISPRS WG IV/2 Workshop "Global Geospatial Information and High Resolution Global Land Cover/Land Use Mapping", April 21, 2016, Novosibirsk, Russian Federation.

Zlatanova, S., Pu, S., Bronsvoort, W., F. (2006). Freeform curves and surfaces in DBMS: a step forward in spatial data integration. In: ISPRS Archives – Volume XXXVI Part 4.

Zlatanova, S., Abdul Rahman, A., Shi, W. (2004). Topological models and frameworks for 3D spatial objects, In: Journal of Computers & Geosciences, May, Vol. 30, No. 4, pp. 419-428

Zulkifli, N. A., Abdul Rahman, A., van Oosterom, P. (2015). An overview of 3D topology for LADM-based objects. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-2/W4, 2015. doi: 10.5194/isprsarchives-XL-2-W4-71-2015

ACKNOWLEDGEMENTS

The first author of the publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports.

BIOGRAPHICAL NOTES AND CONTACT DETAILS

Karel JANEČKA has a Ph.D. (2009) Geomatics, University of West Bohemia in Pilsen. He had been working as a database programmer at the Czech Office for Surveying, Mapping and Cadastre in Section of cadastral central database between 2006 and 2008. Since 2009 he is a researcher at University of West Bohemia, Department of Geomatics. His research activities are spatial data infrastructures (SDI), geographical information systems (GIS), spatial databases, spatial data mining, and 3D cadastre. He has experience with coordination of several EU projects and is also reviewer of several international scientific journals. Since 2012 he is the president of the Czech Association for Geoinformation and member of National Mirror Committee 122 Geographic information/Geomatics.
University of West Bohemia
Technická 8, Pilsen, CZECH REPUBLIC
Phone: + 420 607982581
E-mail: kjanecka@kgm.zcu.cz
Website: http://gis.zcu.cz

Sudarshan KARKI is a Senior Spatial Information Officer in the Department of Natural Resource and Mines, Queensland Government, Australia. He is a surveyor and has completed a Master of Spatial Science by Research at the University of Southern Queensland (USQ) in 2013 and a professional Master's Degree in Geo-informatics from ITC, The Netherlands in 2003. He has continued his research interest in 3D cadastre and is currently undertaking his PhD research at USQ.
Queensland Government, Department of Natural Resources and Mines
Landcentre, Cnr Main and Vulture Streets, Woolloongabba, Brisbane, Queensland 4102 AUSTRALIA
Tel.: +61 7 3330 4720
Email: Sudarshan.Karki@dnrm.qld.gov.au

Peter van Oosterom obtained an MSc in Technical Computer Science in 1985 from Delft University of Technology, the Netherlands. In 1990 he received a PhD from Leiden University. From 1985 until 1995 he worked at the TNO-FEL laboratory in The Hague. From 1995 until 2000 he was senior information manager at the Dutch Cadastre, where he was

involved in the renewal of the Cadastral (Geographic) database. Since 2000, he is professor at the Delft University of Technology, and head of the 'GIS Technology' Section, Department OTB, Faculty of Architecture and the Built Environment, Delft University of Technology, the Netherlands. He is the current chair of the FIG Working Group on '3D Cadastres'.
Delft University of Technology
Faculty of Architecture and the Built Environment Department OTB
GIS Technology Section Julianalaan 134
2628 BL Delft THE NETHERLANDS
Phone: +31 15 2786950, Fax +31 15 2784422
E-mail: P.J.M.vanOosterom@tudelft.nl

Sisi Zlatanova is an associate professor at Faculty of Architecture and the Built Environment, Delft University of Technology, the Netherlands. She has graduated as a surveyor at the University of Architecture, Civil Engineering and Geodesy, Sofia, Bulgaria in 1983 and has obtained her PhD degree on '3D GIS for urban modelling' at the Graz University of Technology, Graz, Austria in 2000. She is teaching several courses related to 3D GIS, 3D databases and their application for disaster management within TU Delft, the University of Venice (2007, 2008). Her research interests are in 3D geo-information and their applications for emergency management (especially flood management). She is chair and cochair of several conferences among which Gi4DM, 3Dgeoinfo and UDMS.
Delft University of Technology
Faculty of Architecture and the Built Environment
Julianalaan 134
2628 BL Delft
THE NETHERLANDS
E-mail: s.zlatanova@tudelft.nl
Website: http://staff.tudelft.nl/S.Zlatanova

Mohsen KALANTARI is a Senior Lecturer in Geomatics Engineering and Associate Director at the Centre for SDIs and Land Administration (CSDILA) in the Department of Infrastructure Engineering at The University of Melbourne. He teaches Land Administration Systems (LAS) and his area of research involves the use of technologies in LAS and SDI. He has also worked as a technical manager at the Department of Sustainability and Environment (DSE), Victoria, Australia.
Department of Infrastructure Engineering, University of Melbourne
VIC 3010 AUSTRALIA
Phone: +61 3 8344 0274
E-mail: mohsen.kalantari@unimelb.edu.au
Website: http://www.csdila.unimelb.edu.au/people/saeid-kalantari-soltanieh.html

Tarun Ghawana is currently a Visiting Faculty and Dissertation Coordinator at Centre for Disaster Management Studies at a Delhi State University for MBA (Disaster Management) Programme. He is associated with Integrated Spatial Analytics Consultants Pvt. Ltd., India as

an external researcher since 2009. He is an MSc (GIS) from ITC, Netherlands and has international research publications on various topics related to land administration, 3D Cadastre, GIS and disaster management. He has worked with academia, private consultants and government departments in India, Netherlands, Germany and Kenya on SDI and GIS based natural resource management.
Integrated Spatial Analytics Consultants Pvt. Ltd.
Dwarka, New Delhi, India
Phone: +9958117758
Email: tarungh@gmail.com