

Validating a 3D topological structure of a 3D space partition

Bregje Brugman, Theo Tijssen, Peter van Oosterom

Delft University of Technology, OTB, Section GIS-technology,
Delft, The Netherlands.

bregjebrugman@gmail.com, {T.P.M.Tijssen, P.J.M.vanOosterom}
@tudelft.nl

Abstract. The goal of this research is to develop a 3D topological structure to represent a 3D space partition with validation functionality and support for conversions from topological to geometrical primitives. Several 3D topological structures have been presented in the past, mainly by researchers. The technical (implementation) model developed in this paper is based on the conceptual model of the ISO 19107 ‘spatial schema’ standard and consists of four topological primitives: node, edge, face, and volume, which are related to each other via their (co)boundary relationships. In our setting, only linear primitives (no curves) are supported and no isolated and dangling primitives are allowed. In our model, the rings, the shells, and the orientation play key roles within the topological structure and the functions that implement the geometrical realization.

There was no formal definition of a valid 3D topological structure available and this paper presents such a definition, which is the main novel contribution. This definition is presented in three levels, where at every next level the definition is further refined such that finally a set of rules is proposed, which can be implemented unambiguously. In order to validate a 3D topological structure, the involved volumes must be valid as well as the whole structure, which means the relationships between the volumes. The rules for a valid structure have been implemented on top of Oracle Spatial and tested with artificial and real-world test data.

1 Introduction

The first use of topology has been attributed to Euler in 1736; since then, topology has evolved in mathematics and, more recently, also in GIS. Since the second half of the 20th century, 2D topological data structures are well established with structures like TIGER (Topologically Integrated Geographic Encoding and Referencing system) (Boudriault 1987) and GBF/DIME (Geographic Base File/Dual Independent Map Encoding), both from the US Census Bureau. Several 3D topological structures have been developed as well, most of them by researchers; for example 3D FDS, the 3D Formal Vector Data Structure (Molenaar 1990) and SSS, the Simplified Spatial Schema (Zlatanova 2000). No commercial geo-DBMS has implemented a 3D topological structure yet. ISpatial is currently developing a 3D topology model as an extension for Oracle Spatial.

A geo-DBMS (Database Management System), where geometric data is stored together with administrative data in one DBMS, is very useful for spatial data management purposes. The main advantages of using an integrated architecture are the capability of a DBMS to handle large volumes of data, the ability to ensure the logical consistency and integrity of data and the ability of multi-user control. Most mainstream DBMSs currently support spatial data types and spatial functions built on these spatial data types. Most spatial data types within a DBMS are defined as single geometries, which describes the geometric primitive in a spatial reference system. For some purposes, like managing data structured in a partition, a topological structure will be more suitable. A topological structure describes the relationships between the primitives (node, edge, face, and/or volume).

Geo-DBMSs are well-developed for 2D spatial data management, but underdeveloped for the third dimension, while 3D spatial data management is becoming more and more important within the 'geo-industry'. For sectors like urban planning, emergency services, hydrology and telecommunication, 3D data management is required. The availability of 3D data is also growing due to new data acquisition techniques. For the past 30 years a lot of research has been carried out and a lot of progress has been made in the field of 3D spatial data management. GEO++ is an early example of a 3D GIS, based on the geo-DBMS Postgres (van Oosterom, Vertegaal and van Hekken 1994). In the commercial sector, Oracle Spatial has implemented a 3D single geometry data type, but most commercial geo-DBMSs have only included 3D coordinates within their single geometries. This means usually that each x,y coordinate has (only) one z-value. This is often referred to as 2,5D. This option of storing 3D coordinates (x,y,z) in the geo-DBMS makes it possible to model 3D with 2D primi-

tives, for example by combining several polygons. Modeling in this way is restricted; some spatial functions do not work, for example the validation of a volumetric object as a whole (Khuan et al. 2008). 3D topological structures are less developed than 3D single geometries.

Before performing any spatial analysis on a topological structure or single geometry, the representation needs to be valid. When data is added or updated the topological rules need to be checked. Therefore a validation function is needed. Current geo-information standards are underdeveloped and not unambiguous with respect to defining a valid 3D topological structure, which leads to different interpretations. Implementation specifications for 3D primitives (both geometrical and topological) are not set yet.

2D validation functions exist for single geometries, but also for topological structures (like Oracle's 2D topological validation function). 3D validation functions also exist but are rather rare and mainly for single geometries, for example Borrmann (2008). No real 3D validation functions for topological structure exist. Only ISpatial is currently developing a 3D topological structure including a validation function (Watson et al. 2008).

In this article, the implementation of a 3D topological structure with validation functionality will be described. As far as the authors know, this is the first time a validation function is implemented for a 3D topological structure. This article is divided into four sections. First, an overview of relevant work is given in section 2, which is followed by a discussion in section 3 of the formal definition and validation rules which are needed for a valid 3D topological structure. These validation rules have been translated into tests which are implemented as a prototype in Oracle Spatial as described in section 4. Section 5 presents the tests with artificial and real data. Finally, the conclusion in section 6 contains the discussion about the main contributions of this work and suggestions for future research.

2 Review of related work

Validation functions are characterized by the moment of validation, the user influence, and the validation rules. The validation rules are the most important aspect of validation and are based on the definition of a valid primitive/structure. In the case of the 3D primitives, no geo-information standards are yet available. 2D single geometries are standardized, including implementation specifications, in the Simple Feature Specification of the OGC and ISO (OGC 05-126/134:2005 and ISO 19125:2004). The OGC has published a candidate version in 2006 with a corrigendum, correcting editorial and minor technical issues in 2010. In this version, still

only 2D geometrical primitives are discussed, but possibly within a 3D context (x,y,z coordinates). The Complex Feature Specification, the OGC standard (at implementation specification level), for topological primitives and structure is not finished yet. For 3D primitives (topological and geometrical), there is an abstract ISO specification (ISO 19107:2003 Geographic information — Spatial schema), but this is not the needed implementation specification. Even the available 2D geo-information standards, however, are ambiguous and incomplete as Van Oosterom et al. (2004) pointed out for simple polygons. Also the interpretations of different GIS and DBMS vendors differ from each other and from the standards. As an introduction to the proposed 3D topological structure, Oracle's 3D single geometry (subsection 2.1) and 1Spatial's 3D topology (subsection 2.2) will be described briefly below. In the context of 3D city models, Gröger and Plümer (2009) present a set of axioms for valid models. A drawback is that they do not allow non-2-manifold objects; see [Figure 1](#) top-left object. Further, they do not consider a space partition, but a city model, that is, a collection of 3D volume objects related to the Earth surface (with a lot of unmodeled empty space).

2.1 3D single geometry (Oracle)

The geometrical equivalents of the topological primitives are the point, line, polygon and solid. Oracle has implemented a geometrical 3D primitive called the 'simple solid' (single geometry) since the release of Oracle Database 11g. The definition of this simple solid: 'a 'single volume' bounded on the exterior by one outer bounding surface and on the interior by zero or more inner bounding surfaces. To demarcate the interior of the solid from the exterior, the 3D-polygons of the outer bounding surface are oriented such that their normal vector always point 'outward' from the solid. In addition, each 3D-polygon of the bounding surfaces has only an outer boundary and no inner boundary (Kazar et al. 2008). Oracle also included a validation function for these simple solids with a set of predefined rules. The function checks for type consistency and geometry consistency (Murray et al. 2010).

Determining the validity of a solid is not always easy. A solid can be connected and closed but still not bound a single volume. This is related to the number of times an edge is used in a solid. An edge in a solid can be used more than two times (as long as it is an even number) and still remain valid; see [Figure 1](#). Furthermore it is not enough to test the inner and outer bounding surfaces only for intersections. An outer bounding surface could

be completely covered by an inner bounding surface without touching each other.

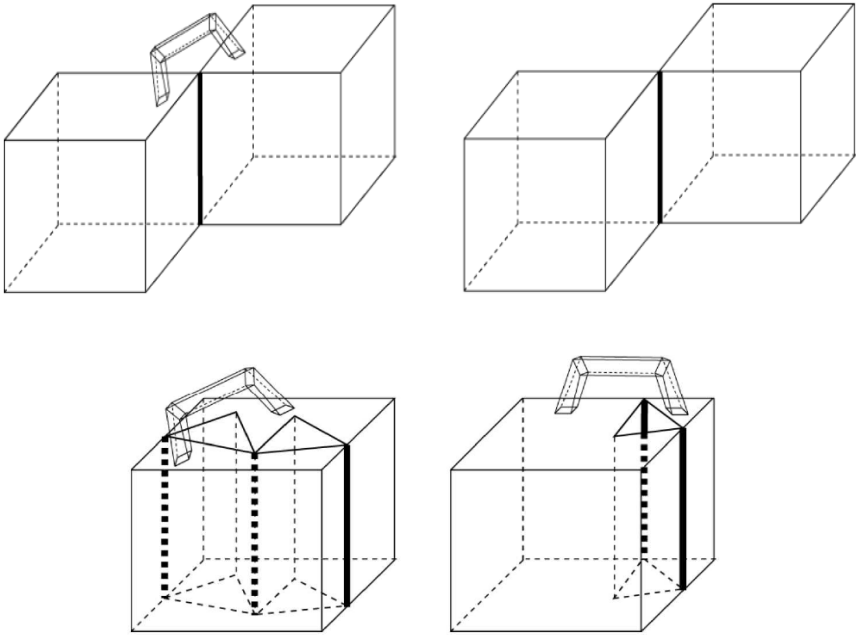


Fig. 1. Top left a valid simple solid; top right an invalid simple solid; bottom left a valid simple solid; and bottom right a valid simple solid; Note the thick edge is used 4 times (source: Kazar et al. 2008)

2.2 3D topology (1Spatial)

At this moment, Radius Topology is only available in 2D, but 1Spatial is currently performing research driven development for a 3D variant. An implementation method describing how to implement a user defined geometry into the topological structure of Radius Topology is available (Watson et al. 2008). The implementation of the topological structure in Oracle Spatial consists of four primitive tables (one for each primitive) including explicit storage of the geometry of the nodes, edges, and faces and three linking tables storing the relationships between the primitives including their orientation.

The 1Spatial validation rules are not enough to test for a valid 3D topological partition. First, the validation rules apply to single primitives and not to the whole structure; intersecting volumes will not be detected. Sec-

ond, these rules are not enough to guarantee valid 3D primitives (e.g. solids are not checked for a contiguous volume or proper orientation).

3 Validation rules for 3D topology structures

The main difference between validating single geometries and validating topological structures is the relationship between the 3D primitives. Topological primitives (volumes) are part of a whole structure, while the single geometries ‘stand alone’; they are validated as one object, only dealing with their internal geometrical characteristics. A topological primitive must be valid on its own and valid within the structure. The explicit relationship between neighbouring primitives is one of the advantages and characteristics of a topological structure. A topological structure is less appropriate when there are no (direct) neighbours around.

The topological structure will be validated according to a predefined set of rules. Before defining a valid structure, some initial conditions will be set. The designed 3D topological structure represents a full space partition of volumes. The volumes are represented by their boundaries, the structure will be linear, and finally the structure will be ISO 19107 compliant; see [Figure 2](#). The top level definition of a valid structure: ‘*a topological space which is divided into a set of non-overlapping valid linear volumes without any gaps*’. In the second level of detail of this definition, four aspects can be distinguished:

- a. The *topological space* is defined by 1 or more inner shells of the universal volume, which has no outer shell containing the inner shell(s). These inner shells must be valid shells and are not allowed to intersect (touch is allowed).
- b. A *set of non-overlapping volumes without any gaps* means that every face is on the boundary of exactly two volumes, one on each side. Furthermore volumes are not allowed to intersect and no isolated and dangling primitives are allowed.
- c. *Linear volumes* can be established by planar faces and straight edges.
- d. The definition of a *valid volume* is based on existing definitions of valid solids. Each volume must consist of one outer shell and zero or more inner shells. Each shell consists of 4 or more valid faces, which are non-overlapping, properly oriented, and connected in a topological cycle. Shells are allowed to touch each other or themselves by node or edge (not by face). The geometric realization of each volume bounds a single volume.

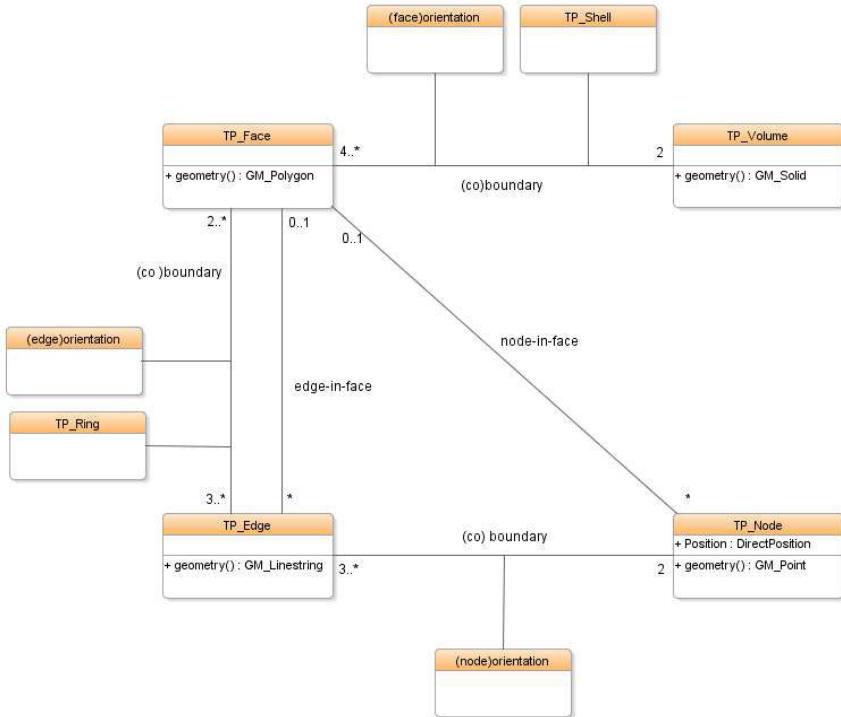


Fig. 2. The conceptual schema of our 3D topological structure (based on ISO 19107)

To meet the above definition, the structure needs to meet the following rules, which form the third and most detailed level of our definition of a valid 3D topological structure. Some rules deal with topology, others are more geometrical in nature. All rules are explained below.

1. *Unique primitives*: Each node has unique xyz values (within the geometrical tolerance distance), which hold the coordinates of a node. When each node is unique, each (unique) combination of primitives forming another primitive is unique as well.
2. *Each primitive is a volume or part of the boundary of a volume*: Because the structure consists of only volumes without isolated primitives, each primitive must be part of a volume. Shells and rings can be added to this rule: each node is part of an edge, each edge is part of a ring, each ring is part of a face, each face is part of a shell, and each shell is part of a volume.
3. *Each undirected primitive is associated with two opposite directed primitives*: Since the structure is a full space partition and no isolated

and dangling primitives are allowed (rules 2 and 5), every primitive is associated with two opposite directed primitives of the same dimension. Each volume has a neighbour on each side; this means each face is associated with two directed faces. Since each primitive is part of a volume (rule 2), it is sufficient to only test the faces.

4. *Proper orientation:* Rule 3 requires that every primitive has a positive and a negative direction, but the topological structure does not only require an orientation but also requires a proper orientation. This means that every face part of a shell is oriented outward from the volume it bounds. In this way the interior of a volume can be distinguished from the exterior. The orientation of an inner ring of a face must be opposite from the orientation of the outer ring.
5. *Each boundary is closed:* When each boundary is closed and each primitive is part of a volume (rule 2), it means no dangling primitives (edges and faces) are present in the structure. The boundary of a volume is specified by its shells, the boundary of a face by its rings, the boundary of an edge by its nodes. Each volume must consist of one outer shell and zero or more inner shells, where the inner shells are directly inside the outer shell. Each shell must be closed, which is the case when each edge in the shell is on the boundary of two or more (even number) faces (of the same shell). Each face must contain one outer ring and zero or more inner rings, where the inner rings are directly inside the outer ring. Each ring consists of 3 or more edges and must be closed. A closed ring is established when all edges are connected in a topological cycle. This means each node within the ring is part of two directed edges (within the ring), once as start node and once as end node. Each edge has a start node and an end node.
6. *Valid extent:* The 3D objects that together make up a data set are located in the universal volume, a volume without an outer shell. The extent of the data set is comprised of one or more inner shells of the universal volume. A 'hole' in the data set could be modeled as a small outer shell of the universal volume (contained within a larger inner shell of the universal volume).
7. *The structure is linear:* A linear structure (no curves) is one of the preset conditions; therefore it needs to be checked. When all faces are planar and all edges are straight, the structure is linear. When a face is planar, it means all nodes of that face are on the same plane (within a geometrical tolerance value). This topological structure only uses straight edges, defined by a start and end node (so the edges do not include intermediate vertices).

8. *Inner boundaries must be inside outer boundaries:* Every inner shell must be inside the outer shell of the same volume. Inner shells are not allowed to be inside other inner shells of the same volume. The same applies for inner rings. Inner rings must be inside the outer ring of the same face and are not allowed to be inside other inner rings of the same face.
9. *No intersections:* No intersections between and within shells and rings are allowed. No intersections between and within shells means no intersection of faces (within one shell or between different shells), but faces are allowed to touch (when a node or edge is present at the intersection or when it is about a node-in-face or edge-in-face singularity; see [Figure 3](#)). No intersection between rings is allowed, although they are allowed to touch when a node or edge is present. An inner ring is allowed to touch the outer ring in one point (of the same face). Outer and inner rings are not allowed to touch themselves.
10. *Bounding single volumes/areas:* Every volume should have a contiguous interior and every face should have a contiguous interior. The boundaries are already checked for being closed (rule 5). All primitives involved in the boundary are connected to each other (rule 2), but this does not guarantee that the interior is contiguous; see [Figure 1](#) top-right. A more specific test is needed for this purpose.

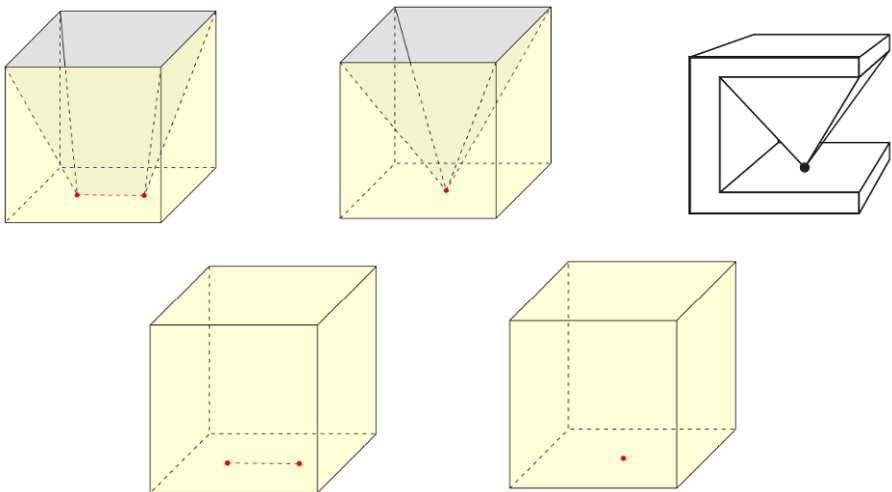


Fig. 3. Top row: allowed singularities; top-right example from Borrmann (2008; see page 218, fig 9.6 right); bottom row: invalid singularities

Some additional notes can be made related to rule 9 and the singularities. That is, lower dimensional primitives completely inside the interior of a higher dimensional one; e.g. node-in-edge, node-in-face, node-in-volume, edge-in-face, edge-in-volume, and face-in-volume. These singularities are only allowed if they are needed to define a volume in the context of modeling space partitions. Therefore, the face-in-volume, edge-in-volume, and node-in-volume are per definition invalid. Further, the node-in-edge singularity should be treated (remodeled) as a split of the edge as this has no drawbacks. Therefore, it is not allowed to have the node-in-edge singularity. In the UML model (Figure 2), these singularities are not included as associations and only the remaining allowed singularities, edge-in-face and node-in-face, are included.

4 Implementation

Different approaches exist in designing a structure. A balance has to be found between little redundancy,(relatively) easy validation tests and geometry operations. If a lot of redundant information is stored, the structure is large and slow and more chances for storing contradictions are present. On the other hand, it will be easier to extract information from the structure (less derivations needed). If, on the other hand, little redundant information is stored, it will require more complex algorithms to retrieve information and it will be difficult to check if the structure is still valid after an edit. This topological structure will consist of manageable tables with little redundancy; a balance is aimed for between simplicity and usability; see Figure 4.

4.1 Conversion functions

For various reasons, it is useful to have conversion functions that ‘materialize’ the topological primitives into geometrical objects. An obvious one is visualization; virtually all display engines require geometry as input, and they are not designed to make ‘direct’ use of topological structures. Although topology certainly has advantages for storage and maintenance of large data sets, many applications also need geometry as input for their operations. Certain GIS operations benefit from being executed in the topology domain (e.g. finding the neighborhood of an object); others perform better or simply can only be executed on geometry objects (e.g. length, area and volume calculations). Conversion functions in the other direction, from ‘raw’ (geometry) data to topological structure, would be very helpful

to fill the structure, but that was not part of the current research. The efforts of ISpatial as described before can be considered complementary to this research; their system concentrates on generating clean topology from raw data.

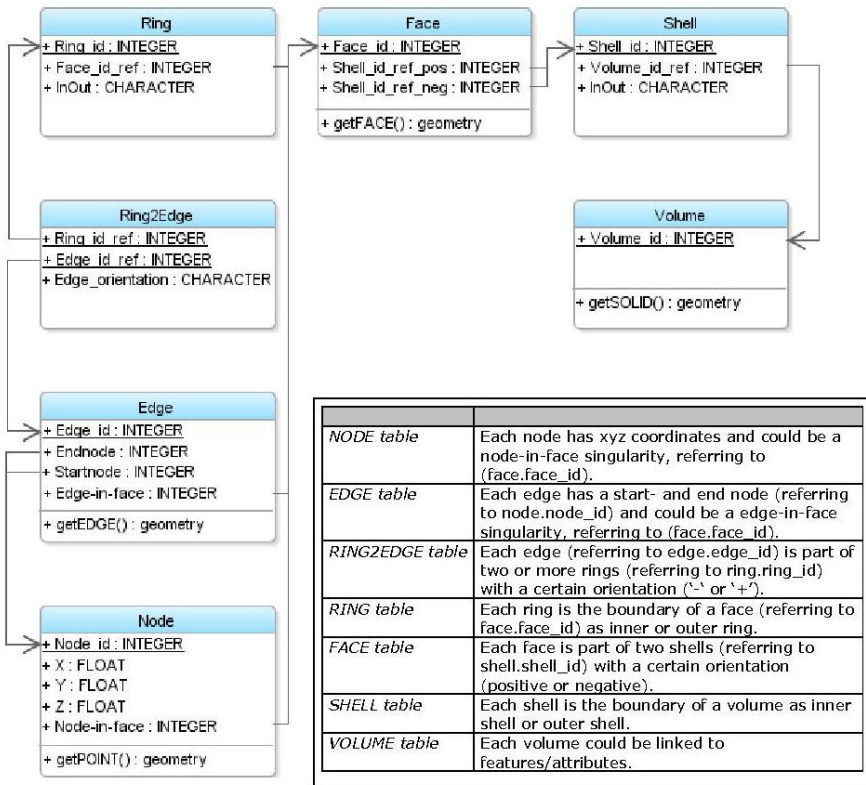


Fig. 4. The logical (near technical) schema of our 3D topological structure

The to-geometry conversion functions developed coincide with the 3D primitives: getPOINT(id), getLINE(id), getPOLYGON(id), and getSOLID(id). The difficulty of implementation of these functions ranges from trivial, the first three, to rather complex, the last one. The structure, as presented in this paper, makes clear that the geometry of points, lines, and polygons can simply be ‘looked up’ in the relevant tables. The getSOLID function is more difficult because Oracle geometry is the target for the geometry object. The 3D polygons that make up an Oracle solid cannot contain inner rings, but in our model inner rings are allowed. So each face that contains inner rings has to be split up into a set of planar polygons

without inner rings. As an example below a PL/SQL snippet from the get-POLYGON function:

```

--select outer ring and inner rings
SELECT ring_id INTO v_Oring FROM ring
  WHERE face_id_ref=i_face AND InOut='O';
SELECT ringordinates INTO v_ordinate FROM
  ringordinates WHERE ring_id=v_Oring;
SELECT ring_id BULK COLLECT INTO tab_Iring FROM
  ring WHERE face_id_ref=i_face AND InOut='I';
v_info_array:=sdo_elem_info_array(1,1003,1);
-- if no inner rings present
IF tab_Iring.count=0 THEN
  FACE:=SDO_GEOMETRY(3003,NULL,NULL,
    v_info_array, v_ordinate);
  RETURN FACE;
-- if inner rings are present:
ELSE
  FOR i IN tab_Iring.FIRST..tab_Iring.LAST LOOP
    v_Iring:=tab_Iring(i);
    SELECT ringordinates INTO v_Iordinate FROM
      ringordinates WHERE ring_id=v_Iring;
    . . .
  END LOOP;

```

4.2 Validation checks

Orientation and boundaries (especially the groupings in rings and shells) play an important role in topology and validation. Therefore orientation and boundaries are stored explicitly, although it means extra storage and extra validation tests, but this is in no proportion to the advantages. The orientation of edges is stored explicitly by references to a start and end node. The implementation is done in Oracle Spatial, therefore Oracle (proprietary) data types are used. A topological structure has its strengths (and weaknesses) compared to a geometrical model. This means that validation tests based on geometrical characteristics, which is inevitable, will be reduced to a minimum and the validation tests will be based, as much as possible, on the topological relationships between the nodes, edges, faces, and volumes.

Simple (fast) tests will be done on the whole structure and reduce the hard (time-consuming) tests as much as possible. Some constraints are enforced by the use of primary and foreign keys and ‘non-nullable’ columns. Tolerance values play an important role in validation. In all tests, which

require a tolerance value, tolerance values are applied. Table 1 gives an overview of the relationship between the ten validation rules and the eight validation tests.

Table 1. Validation rules linked to the validation tests

Rule	Test
1. Unique primitives	1
2. Each primitive is a volume or part of the boundary of a volume	2
3. Each undirected primitive is associated with two opposite directed primitives	n.a.
4. Proper orientation	5/4
5. Each boundary is closed	$\frac{3}{4}$
6. Valid extent	3
7. The structure is linear	6
8. Inner boundaries must be inside outer boundaries	7
9. No intersections	7
10. Bounding single volumes/areas	2/8

Rule 3 is automatically implemented by the columns in the face table (and the primary key on the face_id). Therefore, no test is needed for testing ‘each face is associated with two (opposite) directed faces.’ The other tests are organized in blocks: first, block I (uniqueness and references) is performed; if successful then block II (structure and orientation) is performed; and finally block III (geometry) is performed, only if I and II are both successful.

Block I, test 1) unique primitives: In this test, nodes, edges, and faces are tested for their uniqueness. Unique volumes do not need to be tested because volumes cannot refer to the same shell (by the primary-key on the shell) and all shells are unique (if all faces are unique). Although all primitives have unique id's (enforced by primary keys), they still need to be tested for their uniqueness. To test if nodes are unique they are stored as point geometries in a temporary table and a spatial index is created. Then each point is checked whether other points can be found inside the tolerance distance from the point using the spatial index to speed up the procedure.

Block I, test 2) primitive references: In this test, the references between the primitives (one dimension higher/lower) will be tested; node-edge, edge-face, and face-volume, in order to avoid isolated (no reference to a

primitive in a higher dimension) and nonexistent (no reference to a primitive in a lower dimension) primitives. This check does not exclude dangling edges and faces, which will be checked in test 4. Checking the references is partly taken care of by not allowing NULL values in the tables and applying foreign keys, but it is necessary to perform some additional tests for isolated nodes and edges and for nonexistent faces, rings, and shells. The primitives are also tested for their minimum number of boundary primitives: a volume must consist of 4 or more faces and a face must consist of 3 or more edges. An edge must consist of exactly two nodes, but this is taken care of by the columns start node and end node in the EDGE table and therefore does not need to be tested. The third aspect in this test deals with primitives used more than two times in one boundary. A node could be on the boundary of many edges, but an edge can only be used once in the boundary of a face (irrespectively of being an inner or outer ring) and a face can only be used once on the boundary of volume.

Block II, test 3) each face/volume consists of one outer boundary: Each face is tested for exactly 1 outer ring (no more and no less than 1 outer ring) and each volume for 1 outer shell. The universal volume is an exception.

Block II, test 4) closed boundaries: Each boundary has to be closed. The boundaries of faces and volumes are checked by testing for closed rings and shells (by the orientation of the edges and faces within the rings and shells). A closed ring is established when the last node equals the first node. A closed shell is established when each edge in the shell is on the boundary of two or more (even number of) faces (of the same shell). This means that each edge, involved in the shell, must have an equal distribution of negative and positive references.

Block II, test 5) proper orientation: Every face in a shell is oriented outward (positive). For the orientation of rings within a face, the outer ring should be counter clockwise when observed from outside the volume (normal of vector will point outward) and the orientation of an inner ring must be opposite from the orientation of the outer ring (of the same face).

Block III, test 6) planar faces: A face is planar when all nodes of that face are on the same plane (within a planarity tolerance value). In this test, an optimal plane is fitted through the points of the face and then the distance of each point to the optimal plane is calculated. The test fails if the distance is bigger than the tolerance value. This is currently implemented by taking the plane defined by two arbitrary edges of the outer ring and checking if the 'average' of nodes is on the plane (first the outer ring is tested, next the inner rings). This is not a very good test as with a bit of luck one could pass this test as a point far above the plane is averaged out by a point far below the plane. A better test would be to first find the 'best

fitting' plane of a face by the least-squares error fitting method on all boundary nodes (of outer and inner rings) and nodes related to proper singularities in this face (node-in-face and via edge-in-face). Then check if all nodes are within the tolerance distance of the plane. One remaining issue is: how is the tolerance defined? Some possible options are: a. fixed value (but this would mean that a very large face has to be relatively flat compared to a small face); b. relative value, that is, expressed as the amount of curvature allowed (fraction of the diameter of the face; with diameter face defined as the distance between the two points most far apart).

Block III, test 7) no intersections: Primitives are not allowed to intersect but are allowed to touch. This can be summarized in 'volumes are not allowed to intersect and self-intersect (but touch)', which can be translated into 'faces are not allowed to intersect and self-intersect (but touch)'. When all volumes are not self-intersecting, no intersecting volumes will be present because the structure is a full space partition (and each face is part of exactly two different shells). Therefore, volumes do not need to be tested on intersections between each other but each volume has to be tested on self-intersection (including the inner shells of the universal volume). This will be done by testing each volume for intersecting faces. In four particular cases, touching faces are valid; at an edge-sharing or node-sharing touch and at an edge-in-face or a node-in-face singularity. Self-intersecting faces will always lead to a face intersection somewhere in the volume, therefore no separate test for self-intersecting faces is needed. Inner rings are allowed to touch their outer ring (in one node only). In addition, inner rings and inner shells need to be tested for not being completely outside the accompanying outer ring or outer shell (except for the universal volume) and outer rings and outer shells need to be tested for not being completely covered by their accompanying inner ring or inner shell (except for the universal volume with a 'hole', which is modeled as a small outer shell). This test makes use of Oracle's SDO_ANYINTERACT (which is fitted with a tolerance value).

Block III, test 8) bounding single volumes/areas: Each volume and each face must have a single, contiguous interior. Most volumes will have a single interior if they have passed tests 1 through 7. However, some complicated situations make this last test difficult, see [Figure 1](#) for examples. A solution for this problem is suggested by Kazar et al. (2008). They suggest the tetrahedronization of a volume. Next, all connected tetrahedra will be marked via shared triangles (by a Boolean value) starting at a random tetrahedron. At the end, the number of marked tetrahedra must be equal to the total number of tetrahedra, otherwise the volume is not contiguous. This test has not been implemented in the current research.

All tests are written as PL/SQL scripts. At the moment the validation tests run on the whole dataset (global testing), but internally they test each object separately therefore it could be very useful for a local test as well (for example after an edit/update).

5 Test with real data

For testing the prototype, a topological correct and clean data set of the TU Delft Campus has been used: 370 buildings, from the 2D Large Scale Base map of Delft (Grootschalige Basiskaart in Dutch), which were extruded to 3D; see [Figure 5](#) (Ledoux and Meijers 2011). The original data was converted to the 3D topological structure (Brugman 2010). First, the data has been analyzed and inserted into the prototype, then the validation tests and geometry operations have been tested on the data set. The data consist of 370 (unique) volumes and 8152 (unique) faces. Furthermore 13467 (unique) edges and 5841 (unique) nodes could be derived. When analyzing the data in more detail, the following information could be extracted: the volumes (buildings) are scattered around the area (campus) and the air or the underground are not explicitly modeled in this case. When clustering the volumes, 169 clusters can be distinguished. A cluster consists of buildings that are connected to each other; connected buildings share at least one face. Different clusters are separated from each other by ‘air’.

The data was inserted into the tables of the prototype. In order to present a full space partition, a ‘mini’ universe has been created for each cluster. This means the universal volume has 169 inner shells. The validation tests and geometry operations are tested on the whole campus data set and on the largest cluster (cluster 381) with 33 buildings; see [Figure 6](#). This cluster consists of 33 volumes and the universal volume with one inner shell, 893 faces, 1455 edges, and 594 nodes. At the moment, test 7 (no intersections) is the most time consuming, but with the help of spatial index structures this can be improved (but performance was not in the scope of the current research).

Bentley’s MicroStation, the software used for the visualization, does not know of or ‘understand’ the 3D topological structure. Visualization was achieved through the to-geometry conversion functions, `getPOLYGON` in particular, that produces geometries that can be visualized with a CAD/GIS-like MicroStation.

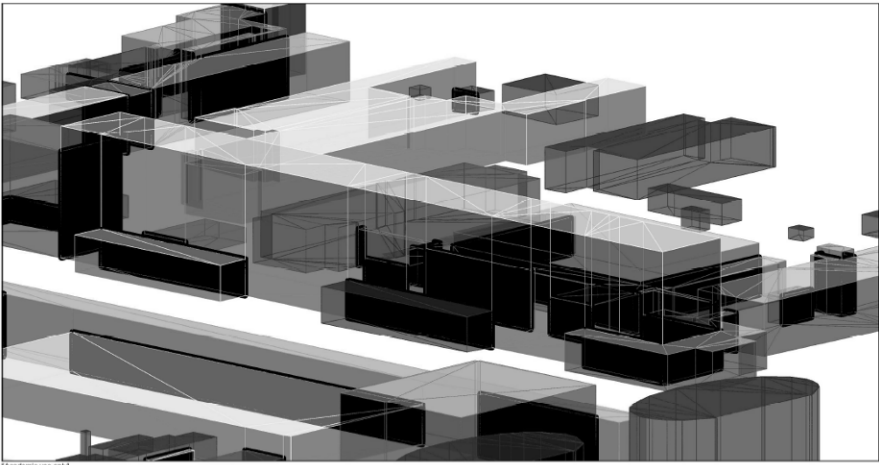


Fig. 5. Several buildings with shared faces (shown in black; visualized with MicroStation)

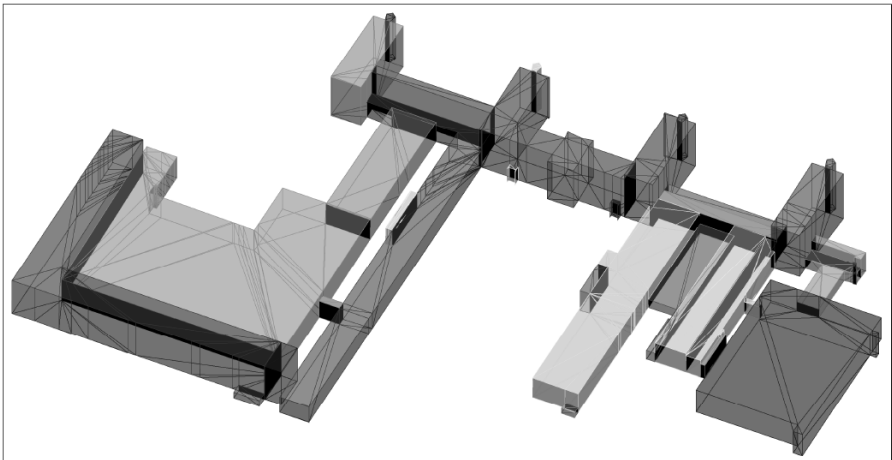


Fig. 6. Cluster 381 of the Campus data set (visualized with MicroStation)

Some tests could not be carried out because the data set contained no inner shells and no inner rings. These were tested on a hand-made, artificial mini data set, which consists of only 10 volumes (plus one universal volume), 58 faces, 110 edges, and 67 nodes; see [Figure 7](#).

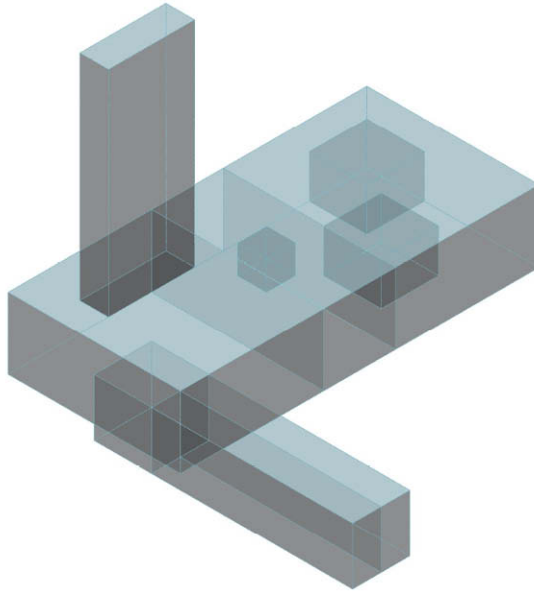


Fig. 7. Artificial mini data set (visualized with MicroStation)

6 Conclusion

The most important novel contributions of this paper are all included in the developed prototype (developed in Oracle Spatial using PL/SQL):

- Table structure for a balanced implementation of a 3D topological structure based on ISO 19107;
- Validation tests, based on a 3-level formal definition of a valid structure; and
- Conversion functions to `sdo_geometry`, i.e. the geometrical realization with four `TO_GEOMETRY` operations: ‘Node to Point’, ‘Edge to Line’, ‘Face to Polygon’, and ‘Volume to Solid’.

The current implementation is limited to linear primitives (no curves) and the following topics are outside the scope of the current research (but within the scope of future research): updating/editing, modeling features, and efficient validation tests/queries. Also, in the future test 8 ‘contiguous volume’ should be implemented. The important open question is: are the 10 validation rules sufficient and are they all needed?

Other future work could include automatic repair of invalid situations, e.g. non-flat faces. Assume a non-flat face is detected during validation,

but that it is further well-connected in the structure. This can be repaired by splitting it into two (or more) faces. Normally, splitting a face should be a local operation and keeps the topology structure intact as it does not create new intersections. After the split check both faces again for flatness and repeat splitting until the faces are flat enough. For sure this process will terminate, because if you continue until you get triangles, then these will always fulfill the criterion of flatness. However, when trying to minimize the number of resulting faces, the procedure will be non-trivial. What is an optimal split, i.e. removing most of the 'tension' in the non-flat face? Also the proper handling of holes (inner rings) must be taken care of. Instead of this top-down approach, perhaps a bottom-up alternative approach would be easier: 1. triangulate face (including inner rings), 2. merge neighbor triangles if 'merger' is flat enough, and 3. stop when no more 'mergers' are possible. However, it is again not clear in which way a minimum number of flat faces can be obtained.

Acknowledgments

The authors of this paper would like to express their sincere gratitude to our colleagues Rod Thompson, Hugo Ledoux, and to the three anonymous AGILE reviewers for their constructive remarks and suggestions.

References

- Borrmann A. (2008) Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken. PhD thesis Technische Universität München.
- Brugman, B. (2010) 3D topological structure management within a DBMS - validating a topological volume. GIMA Master's Thesis
- Gröger, G. and Plümer, L. (2009) How to achieve consistency for 3D city model. *GeoInformatica*, DOI: 10.1007/s10707-009-0091-6.
- OGC (2005), OpenGIS Implementation Specification for Geographic information – Simple feature access – Part 1: Common architecture (OGC 05-126) / Part 2: SQL option (OGC 05-134), version 1.1.0.
- ISO/TC211 (2003) International standard ISO 19107, Geographic information – spatial schema. ISO: Geneva, Switzerland
- Kazar, B.M., R. Kothuri, P. van Oosterom & S. Ravada (2008) On valid and invalid three dimensional geometries. *Advances in 3D Geoinformation Systems*, Springer, pp. 19-46

- Khuan, C.T., A. Abdul Rahman & S. Zlatanova (2008) 3D solids and their management in DBMS. *Advances in 3D Geoinformation Systems*, Springer, 2008, pp. 279-311
- Ledoux, H. and Meijers, M. (2011) Topologically consistent 3D city models obtained by extrusion. *International Journal of Geographical Information Science*. In Press.
- Molenaar, M. (1990) A formal data structure for 3D vector maps. *Proceedings of EGIS'90*, Vol. 2. Amsterdam, The Netherlands, pp. 770–781
- Murray, C., D. Abugov, N. Alexander, B. Blackwell, R. Chatterjee, D. Geringer, M. Horhammer, Y. Hu, B. Kazar, R. Kothuri, S. Ravada, J. Wang, J. Yang. (2010) *Oracle Spatial Developer's Guide 11g Release 2 (11.2)*
- Oosterom, P. van, W. Quak & T. Tijssen (2004) About invalid, valid and clean polygons. Peter F. Fisher(Ed.); *Developments in Spatial Data Handling*, 11th International Symposium on Spatial Data Handling, 2004, pp. 1-16
- Oosterom, P. van, W. Vertegaal & M. van Hekken (1994) *Integrated 3D modelling within a GIS*. AGDM'94: Delft, Netherlands
- Watson, P.J., M.J. Martin & T.D.c. Bevan (2008) WO 2008/138002 A1 *Three-dimensional topology building method and system*. World Intellectual Property Organization: Geneva, Switzerland
- Zlatanova, S. (2000) *3D GIS for urban development*. PhD thesis, ITC, The Netherlands, 222pp