Oracle Spatial 11g Beta testing

initial draft - confidential

K.L Emgard, M. Sc
drs. T.P.M. Tijssen
dr. S. Zlatanova
ir. S. Oude Elbrink (ITC Enschede)
prof. dr.ir. P.J.M. van Oosterom

TU Delft, GISt Report
Delft, April 2007

Table of contents

1. Introduction

This report describes the test results from the testing of 3D spatial data types in the Beta version of Oracle Database. Three test datasets are introduced to test the intended functionality of loading and retrieval of data as well as computational functions: 1. A dataset of simple, planar polygons; 2. A dataset of simple, complex and corrupt solids; 3. A dataset of a building block model from extruded polygons. The testing focuses on validation, volumetric computation and distance calculations between different types of solid objects. Furthermore, the main focus was on functionality and no performance testing was carried out.

1.1. Test configuration

All testing has been done on a simple PC: one CPU (Intel Pentium 4, 2.8 GHz), 1 GB RAM and one 120 GB disk (ATA). So especially the I/O capabilities of this machine are limited. The operating system used is Red Hat Enterprise Linux 4, Update 4.

The most important software used is of course the Oracle DBMS, the versions used for testing are:

- 11g Enterprise Edition Release 11.1.0.3.0 (Linux Beta 4, 32-bit)
- 11g Enterprise Edition Release 11.1.0.4.0 (Linux Beta 4 Drop 3, 32-bit)

Test results described in the report are based on the latest version. Before the Linux versions were used some work, like developing the test scripts, was done in a Windows Beta 2 version. But that version was, for 3D Spatial functionality, not mature enough to allow serious testing. In fact only the latest version (11.1.0.4.0) made really possible the testing of 3D functionality. The first thing required for 3D data is robust validation, version 11.1.0.3.0 still produced 'strange' results (to put it mildly) for validation of 3D objects. This this improve with the latest version (see section 5).

As client software mainly SQL*Plus (various versions) and sometimes SQL Developer was used. A number of objects were visualized by means of MicroStation / GeoGraphics (Bentley).

2. Documentation and examples

Each new version of the Beta software and documentation has shown clear improvements. The descriptions and parameters of 3D Spatial data types and functions are relatively complete in the latest version of the documentation (Spatial Developer's Guide of March 16 2007). Only the examples of the use of point cloud and TIN subprograms in chapters 27 and 29 of the Developer's Guide are still missing (but these are important to inform the user about these subprograms). Only in the most recent software distribution (11.1.0.4.0) the point cloud and TIN sample programs, both PL/SQL and Java, are included.

What is apparently missing from the documentation is a description of 3D objects with a SDO_ETYPE of 1008 (should be included in Table 2-2). In the geometry examples in paragraph 2.7.9 (Example 2-16) the last example of the 'polygons3d' table with ID=15 has a SDO_GEOMETRY that starts with:

sdo_geometry(3008,null,null,sdo_elem_info_array(1,1008,2,1,1007,1,1,1006,6,1,1003,…..

This type of 3D object, from the description a "composite solid", with a SDO_ETYPE of 1008, is not included in other relevant parts of the documentation.

Another characteristic of Example 2-16, probably not intended as such, is that not all geometries of the 'polygons3d' table validate. The validation result of the object with ID=7 (3D surface with 3 3D polygons) is:

```
54516, 0000, "adjacent outer rings of composite surface cannot be on same plane"
// *Cause:   The conditional flag was set, and a composite surface had
//           at least two outer rings sharing a common edge on the same plane.
// *Action:  Change those outer rings into one larger outer ring.
```

The validation result of the object with ID=11 (Simple solid with 6 polygons) is:

```
54535, 0000, "incorrect box surface because it is on arbitrary plane"
// *Cause:   The axis aligned box surface was not
//           on the yz, xz, or xy plane.
// *Action:  Ensure that the first and fourth coordinates, or the second and
//           fifth coordinates, or the third and sixth coordinates are the
//           same. This means that the surface is on the yz, xz or xy plane,
//           respectively.
```

Other examples of error codes and descriptions (of validation errors occurring later in this report) are:

```
13349, 00000, "polygon boundary crosses itself"
// *Cause: The boundary of a polygon intersects itself.
// *Action: Correct the geometric definition of the object.
```

```
54502, 0000, "solid not closed"
// *Cause:   The solid geometry was not closed i.e.,
//           faces of solid are not 2-manifold due to incorrectly
//           defined, oriented, or traversed line segment
//           because each edge of a solid must
//           be traversed exactly twice, once in one direction and
//           once in the reverse direction.
// *Action:  Correct the orientation of the edges of the neighboring polygons.

54503, 0000, "incorrect solid orientation"
// *Cause:   The orientation of the solid was not correct.
// *Action:  Correct the orientation or specification of the outer or inner
//           solid geometry according to the geometry rules for such a solid.

54505, 0000, "ring does not lie on a plane"
// *Cause:   The ring was not flat.
// *Action:  Make sure all of the vertices of the ring are on the same plane.
```

The `545xx` error codes are not yet included in the most recent Error Messages manual (Beta draft of January 2007).

Most of the error descriptions are clear and help in correcting the errors. But what is still missing from the documentation is an overall description what constitutes a valid 3D surface and a valid 3D solid according to the Oracle specification. For 2D geometries is listed in detail which checks are applied while validating a geometry. This description has to be modified and extended to include 3D surfaces and 3D solids.

3. Overview of datasets

This section describes the data sets that are used in the project. An overview of the data and the way it is organized is given.

3.1. Single polygon dataset

A planar polygon and three modifications of the polygon (2-4) were created to check the validation of the geometry.

1. A planar polygon
2. A non-planar polygon
3. A planar polygon intersecting itself
4. A non-planar polygon that is not closed

The geometry script of the four polygon objects is described further in appendix A.

3.2. Single solid dataset

1. Tetrahedron
2. Tetrahedron twice as large as tetrahedron 1
3. Larger tetrahedron
4. Block
5. Block with one of the polygons defined twice
6. Non-closed block with one dangling polygon
7. Non-closed block with one polygon missing
8. Block with one of the normal vectors pointing inwards
9. Block with all normal vectors pointing inwards
10. Block with a hole through the block
11. Block with a prismatic hole separating the block into two pieces.
12. Block with a cavity
13. Block with a prismatic cavity
14. Tetrahedron with a tetrahedron hole completely inside of it
15. Two blocks with a shared edge
16. Two disjoint blocks
17. Two blocks overlapping each other

The objects 15-17 are not included in figures 1 and 2. the geometry script of the solids are described in appendix A.

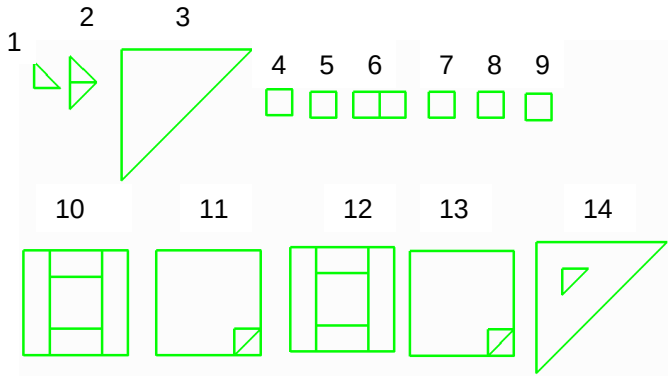


Figure 1: Top view of the solids 1-14 used in the test (converted by hand to multi-polygons and visualized in MicroStation/GeoGraphics). Most objects were moved from their original position for visualization purposes.

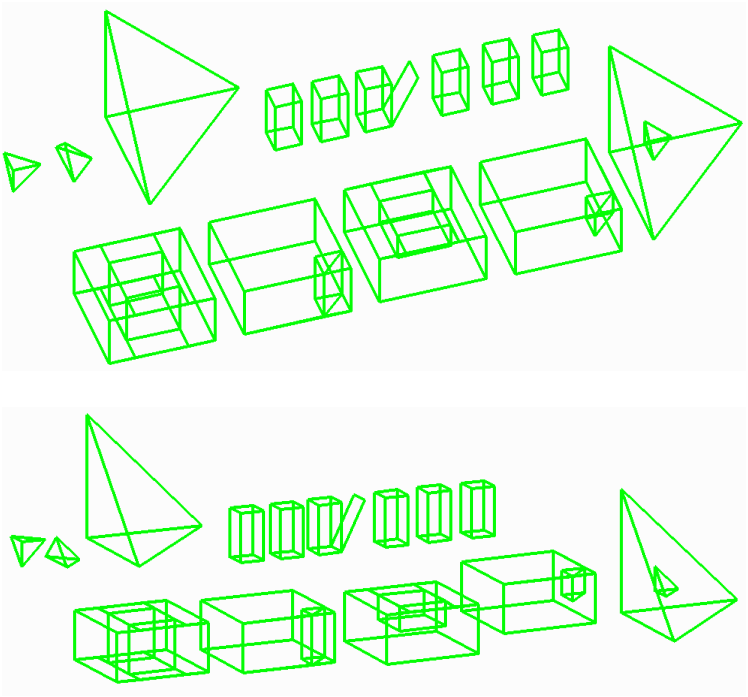


Figure 2: Isometric views of the test solids 1-14.

Objects 1-4 are simple tetrahedrons and a block located within a distance from each other for testing of distance and interaction between the geometries. These objects have one face in the same plane.

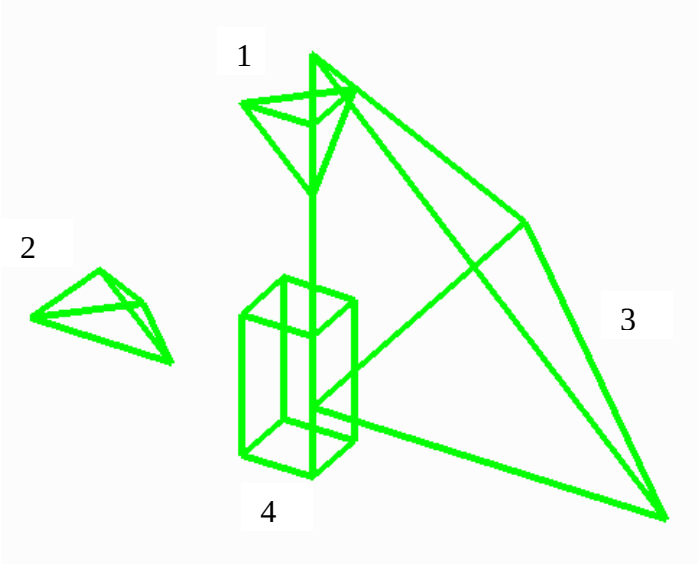


Figure 3: View of solids 1-4 (with correct relative position).

The closest distance between solid 1 and 3 as well as 4 and 3 is zero units. The closest distance between 1 and 4 as well as 3 and 2 is two units. The closest distance between 4 and 2 is one unit and the closest distance between 1 and 2 is the square root of 12.5 (approx. 3.535).

The solids 5-9 and 14 are modified examples of solids 4 and 3. Solids 10-13 are different versions of a larger box with hole formations. Since these objects are overlapping the distances between them are zero units.

3.3. Dataset of building solids

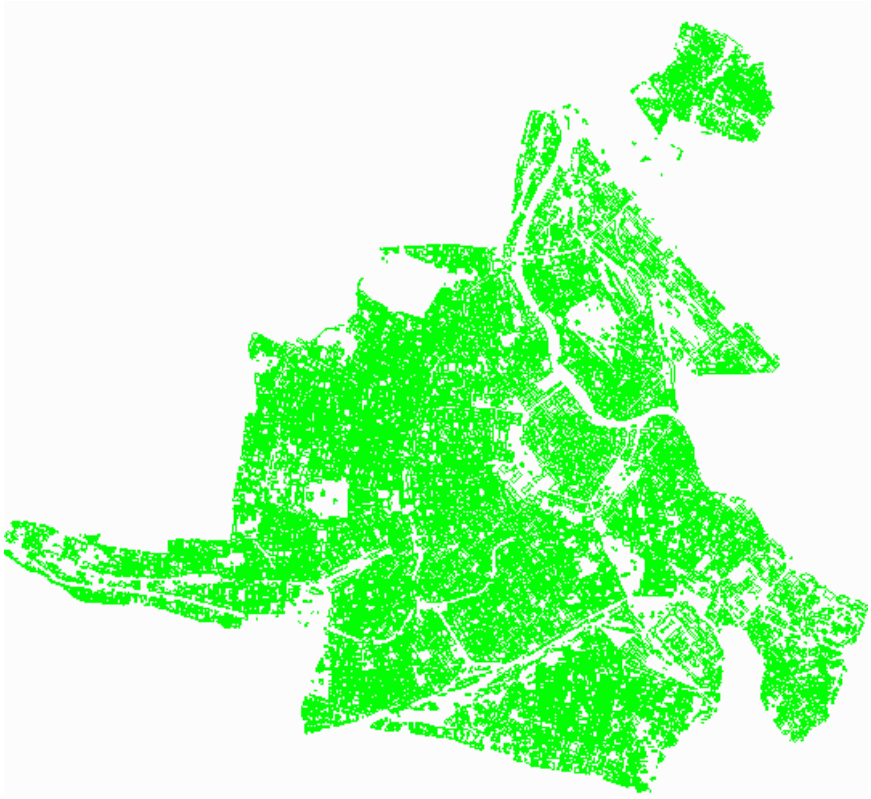


Figure 4: 20021 building blocks.

A dataset containing 20021 extruded building solids, created as extrusions from building footprints with one z for footprint and one z for roof. The original dataset consist of 2D footprints of buildings and two z-values (one on the ground and one on the roof of the object) per footprint. The roof and the wall polygons are created with a in-house program in a uniform way: footprints and roofs are assigned z-coordinates which corresponds to the two given z-values. Using the two polygons, all the walls are created. As a result, all the walls are vertical, all the polygons are planar and have the normal vectors pointing outside the buildings. The only exception is the normal vector of the footprint, which is toward inside the building. Due to limited time for testing this issue was not solved before creating solids from the input data. The 3D data set is stored in a topology schema maintained in three files, i.e. *nodes*, *faces* and *body*. Node contains node_id, x,y,z. Face contains face_id, seq_nodes, node_id. Body contains body_id, seq_faces, face_id.

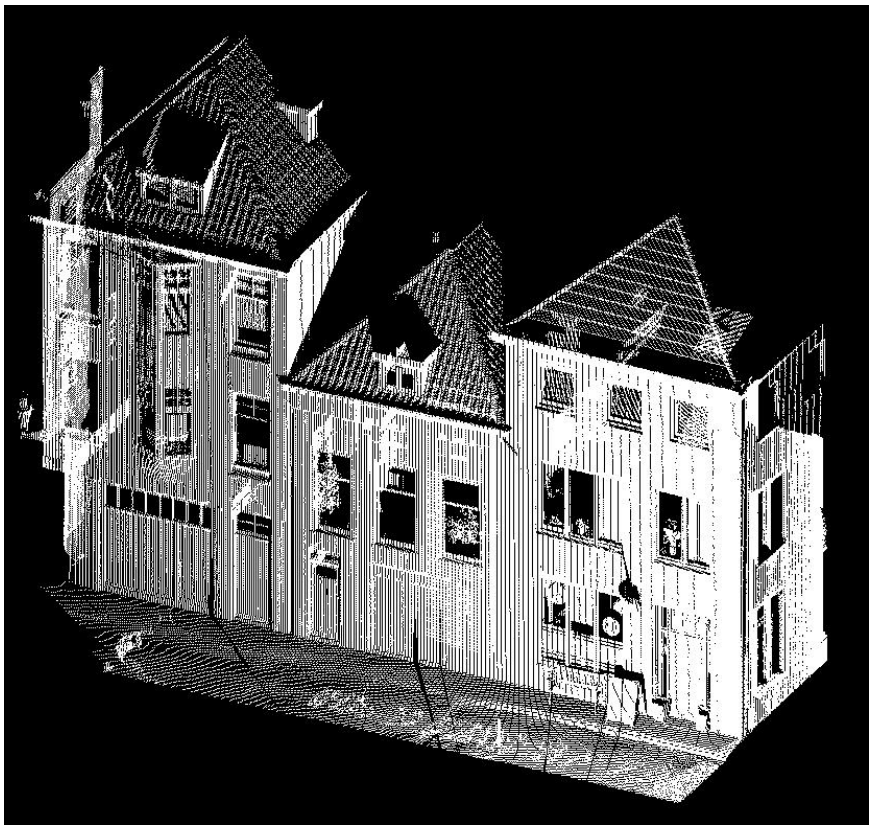
3.4. Generation of building dataset

The solids are created from those files in two steps. Firstly, the dataset is loaded from node, edge and face topology in Oracle relational tables with the same structure as of the

three files. Secondly, a script extracts the geometry from the three relational tables and creates the oracle solid datatype containing the geometry description in SDO_GEOMETRY format (see appendix C for the full geometry description).

3.5. Point cloud dataset

The dataset used is a point cloud with 20,353,414 points acquired by a terrestrial laser scanner. The point density is around 50-200 points per square meter. Each laser point contains x, y, z coordinate and color (r,g,b) information. However, only x y and z coordinates are loaded to Oracle 11g for testing. The scanned scene is the city center of a Dutch town called Vlaardingen. See image of an example scanning.



4. Metadata

The metadata is added after loading, with the following dimensions, extents and tolerances (SRID of all data used is NULL):

Single polygon dataset

```
INSERT INTO user_sdo_geom_metadata VALUES ('polygons3d', 'geometry', sdo_dim_array(
   sdo_dim_element('X', -100,100, 0.005),
   sdo_dim_element('Y', -100,100, 0.005),
   sdo_dim_element('Z', -100,100, 0.005)), NULL);
```

Single solid dataset

```
INSERT INTO user_sdo_geom_metadata VALUES ('solid_tst', 'geom',
  sdo_dim_array(
    sdo_dim_element('X', -100,100, 0.005),
    sdo_dim_element('Y', -100,100, 0.005),
    sdo_dim_element('Z', -100,100, 0.005)), NULL);
```

Buildings dataset

```
INSERT INTO user_sdo_geom_metadata VALUES('sdo_solid_w', 'GEOM',
  sdo_dim_array(
    sdo_dim_element('X', -1000,20000, 0.005),
    sdo_dim_element('Y', -1000,20000, 0.005),
    sdo_dim_element('Z', -100,100, 0.005)), NULL);
```

5. Validation

The single polygon dataset is validated as expected. The first polygon is valid and polygons 2, 3 and 4 are classified invalid:

```
1  TRUE
2  54505 Point:0,Edge:3,Ring:1,
3  13349 Point:0,Edge:2,Ring:1,
4  54505 Point:0,Edge:4,Ring:1,
```

The following objects in the single solid dataset are considered invalid by Oracle:

```
 5  54502 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
 6  54502 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
 7  54502 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
 8  54502 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
 9  54503 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
15  54502 Point:0,Edge:0,Ring:0,Polygon:0,Comp-Surf:1,
```

Objects 5-9 are intentionally defined with errors to test validation, the validation result is correct. Solid 11 is separated into two volumes that are connected by two edges but the object is considered valid by Oracle. It is not completely clear whether this is the correct result, to determine that the rules for valid solids must be known (but if you extend the 2D case to 3D this should be invalid: in 2D an interior boundary is not allowed to touch the exterior boundary in more than one point). Solid 10, 12 and 13 are containing holes and cavities but in this case the volume of the object is coherent in one piece. They are also validated as expected. Solid 14 test the inside cavity functionality (2006) and the validation was successful. Object 15, 16 and 17 are indented to test weather touching, overlapping or disjoint volume parts could be valid in a single solid object (3008). Solid 15 is correctly classified as invalid due to one edge being defined four times. Anyhow, both object 16 and 17 are surprisingly classified as valid. Solid 16 contain two cubes that are covering each other (one of the cubes are partly inside of the other). In solid 17 the two cubes are completely separated from each other (disjoint). Both 16 and 17 would better be represented by multisolids, but was also valid as solids.

None of the building solids are validated because of the 54502 error (while traversing edges one or more edges are defined more or less than two times, or the direction is wrong). This is correct because the footprints of the buildings have the wrong orientation. The building dataset was not corrected or examined further.

6. Querying data

Before querying data, all invalid solids (5-9 and 15) are removed from the script and the testing continues without the corrupt objects. First functions that do not require spatial indexing are tested (scripts in Appendix B).

ID	SDO_GEOM.SDO_LENGTH(P.GEOM,0.005)
1	10.8284271
2	15.6568542
3	54.1421356
4	16
10	112
11	77.6568542
12	104
13	77.6568542

ID	SDO_GEOM.SDO_AREA(P.GEOM,0.005)
1	2.3660254
2	3.73205081
3	59.1506351
4	10
10	72
11	69.8284271
12	72
13	67.4142136

ID	SDO_GEOM.SDO_VOLUME(P.GEOM,0.005)
1	.166666667
2	.333333333
3	20.8333333
4	2
10	24
11	31
12	28
13	31.5

The length of the small tetrahedron (object 1) does not seam to be correct: it has 6 edges: 3 times of length 1 and 3 times of sqrt(2): 7.2426 (and not 10.8284 as stated above). The same type of error for the length of object 2: 5 * sqrt(2) + 2 = 9.0716 (and not 15.6568 as above) and again the same of object 3. However, the length of object 4 is correct again. The area and volume calculations of objects 10-13 are manually verified to be correct. The distance function is tested on solids 1-4 (see figure 3) and is manually verified to be correct:

ID	ID	SDO_GEOM.SDO_DISTANCE(P.GEOM,Q.GEOM,0.005)
4	3	0
4	2	1
4	1	2
3	2	2
3	1	0
2	1	3,53553391

The SDO_ANYINTERACT operator also shows the correct results, in line with the distance function. Solid 3 interacts with 1 and solid 4 with solid 3 due to the zero distance. A spatial index is created before using the ANY_INTERACT operator.

ID	ID
3	1
4	3

7. Point cloud generation

No problem found for the small example dataset. 10 pc-lobs created of 100.000 points in 5 seconds. These points are regularly spaced, because they are created like this (no real measured points).

The testing of our real point cloud dataset (containing 20,353,414 points with x,y,z coordinates) resulted in the following error:

```
Point cloud object creation failed when setting blk capacity =
1,000,000.
The error message is:

ORA-54608: CREATE_PC: error writing Point Cloud LOB
ORA-13234: R-tree-index [LOB Write failure: Block capacity too high]
```

When trying with a lower blk capacity of 100,000 the creation was not finished and interrupted after 4.5 hours. Appendix D contains the script to load the point cloud data.

8. Conclusions

In general the functionality tested produces the results that can be expected. Some tests were done with polygons in 3D space (all tested cases were correctly treated by the validation function), point clouds and TINs, but our current tests did focus on solids. In some cases the documentation and examples must be completed or improved. The most serious problem with the latest version (11.1.0.4.0) is the lack of a good description of the rules that determine whether solids are valid or not and validation that enforces these rules. In our opinion some invalid solids were validated; e.g. in case of one outer shell and a specific inner shell, and in case of multiple outer shell. The last case is easy as a solid can have only one outer shell (only multisolids can have multiple outer shells, which should be non-overlapping). The first case may be more difficult to validate correctly (some serious topology analysis must be used).

The length computation of solids does not always seam to work correctly (tetrahedrons produce wrong results, while a block does produce the right result).

Our wish list for additional functionality with respect to solids does include:

- At the moment no tool is readily available to inspect the geometries graphically. With a conversion function from solid to mutlipolygon (3008 to 3007) the geometries could more easily be inspected with a standard tool. This would be useful until there are viewers that can read the solid data types.
- Functions to support transformations, such as rotation, scaling, translation of objects (polygons, solids) in 3D space.
- More functions and operations, e.g. the unary operations buffer, convex hull, centroid, etc. (supported orthogonally), as today testing showed that they produce 2D resulting geometries.
- Further, especially the binary operations such as union, intersection, difference would also be useful in 3D (in case one would like to support a constructive solid geometry, CSG approach)
- It would be nice if there was a function to correct orientation of normal vectors (orientation of the faces). Perhaps also other correction functions (e.g. flatten faces within given tolerances, remove gaps or overlaps between ‘neighbor’ faces within a solid).

A remark could be done about the encoding of the solid feature that can be experienced as quite redundant since each vertex is described several times. The simple node, face topology (faceset) used in the prototype of Calin Arens (Computers & Geosciences, Volume 31, 2, pp. 165-177) which is also used in VRML could be an alternative. It is more user friendly (less typing or generating) and less error prone.

Our wish list for additional functionality with respect to point clouds and TINs does include:

- More transparency in handling point clouds and TINs, just specify the relevant parameters (block size, 'thematic attributes', source of data, levels of detail, type of clustering,..) instead of doing several steps by hand.
- Conversion functions; e.g. from point could to TIN and vice versa, from TIN to polygonal surface patches (with no more that X surface patches per object), etc.
- Also, methods to influence the clustering of the input data (in case the source data would have a strange ordering; e.g. very long 'scan' lines covering the whole domain).

The initial testing of the point cloud data type showed some problems with dataset that was larger than the provided test dataset (100,000 points) that was inserted without problems. This problem has to be investigated further (also at our side, the test configuration, size of various storages, etc.).
Our other future testing plans involve performance testing of the new 3D solids, further testing with the building dataset and extended testing of the point clouds (also adding r, g, b values) and TIN data types. Functionality testing of the point cloud and TIN data types, does also cover testing different types queries/selections: 'sampling/generalization' and box/viewshed based.

Appendix A

Insertion of data and validation (all polygons and solids)

```
spool solid_L2.log
set lines 110
set pages 500
set serveroutput on
column validate_result format a80
set trim on
set trimspool on
set timing on
set echo on

drop table solid_tst;
drop table polygons3d;
drop INDEX polygons3d_sidx;
drop INDEX solid_tst_sidx;

create table solid_tst (id integer, geom sdo_geometry);
create table polygons3d(id number, geometry sdo_geometry);
--
delete from user_sdo_geom_metadata;
delete from user_sdo_geom_metadata where table_name='SOLID_TST';
delete from user_sdo_geom_metadata where table_name='polygons3d';


---------------------------------------------------------------

/*
CREATE INDEX polygons3d_sidx on polygons3d(geometry)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('sdo_indx_dims=3');

CREATE INDEX solid_tst_sidx on solid_tst(geom)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('sdo_indx_dims=3');
*/


-- delete from mscatalog where tablename = 'POLYGONS3D';
-- insert into mscatalog (tablename, entitynum) values ('POLYGONS3D',2);

-- delete from mscatalog where tablename = 'SOLID_TST';
-- insert into mscatalog (tablename, entitynum) values ('SOLID_TST',3);
```

```
----------------------------------------------------------------

-- 1 planar polygon
insert into polygons3d values(1,
SDO_Geometry (3003,NULL,NULL ,
SDO_Elem_Info_Array(1,1003,1),
SDO_Ordinate_Array(0.5,0.0,0.0,
0.5,1.0,0.0,
0.0,1.0,1.0,
0.0,0.0,1.0,
0.5,0.0,0.0
)));

-- 2 non-planar polygon
insert into polygons3d values(2,
SDO_Geometry (3003,NULL,NULL ,
SDO_Elem_Info_Array(1,1003,1),
SDO_Ordinate_Array(0.5,0.0,0.0,
1.5,1.0,0.0,
0.0,1.0,1.0,
0.0,0.0,1.0,
0.5,0.0,0.0
)));

-- 3 planar polygon intersecting itself
insert into polygons3d values(3,
SDO_Geometry (3003,NULL,NULL ,
SDO_Elem_Info_Array(1,1003,1),
SDO_Ordinate_Array(0.5,0.0,0.0,
0.5,1.0,0.0,
0.0,0.0,1.0,
0.0,1.0,1.0,
0.5,0.0,0.0
)));

-- 4 non-planar polygon not closed
insert into polygons3d values(4,
SDO_Geometry (3003,NULL,NULL ,
SDO_Elem_Info_Array(1,1003,1),
SDO_Ordinate_Array(0.5,0.0,0.0,
0.5,1.0,0.0,
0.0,1.0,1.0,
0.0,0.0,1.0,
0.5,0.0,1.0
)));
```

```
-- SOLIDS Start here.

-----------------------------------------------------
-- 1: SOLID with 4 polygons (tetrahedron 1)
-----------------------------------------------
insert into solid_tst values(1,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,4,1,1003,1,13,1003,1,25,1003,1,37,1003,1),
SDO_Ordinate_Array(
0,0,4, 0,1,4, 0,0,3, 0,0,4,
0,0,4, 0,0,3, 1,0,4, 0,0,4,
0,0,4, 1,0,4, 0,1,4, 0,0,4,
0,0,3, 0,1,4, 1,0,4, 0,0,3
)));
-----------------------------------------------------
-- 2: SOLID with 4 polygons (tetrahedron2, double size of tetrahedron1)
-----------------------------------------------
insert into solid_tst values(2,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,4,1,1003,1,13,1003,1,25,1003,1,37,1003,1),
SDO_Ordinate_Array(
0,2,0, 0,3,1, 0,4,0, 0,2,0,
0,2,0, 0,4,0, 1,3,0, 0,2,0,
0,2,0, 1,3,0, 0,3,1, 0,2,0,
1,3,0, 0,4,0, 0,3,1, 1,3,0
)));
-----------------------------------------------------
-- 3: SOLID Large tetrahedron 3
-----------------------------------------------------
insert into solid_tst values(3,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,4,1,1003,1,13,1003,1,25,1003,1,37,1003,1),
SDO_Ordinate_Array(
0,0,0, 0,0,5, 5,0,0, 0,0,0,
0,0,0, 5,0,0, 0,-5,0, 0,0,0,
0,0,0, 0,-5,0, 0,0,5, 0,0,0,
5,0,0, 0,0,5, 0,-5,0, 5,0,0
)));


-----------------------------------------------------------
--4:  block with 6 polygons
-----------------------------------------------------------
insert into solid_tst values(4,
SDO_Geometry (3008,NULL,NULL ,
```

```
SDO_Elem_Info_Array(1,1007,1,1,1006,6,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1,76,1003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,1.0,-1.0, 1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,-1.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,-1.0, 0.0,1.0,-1.0, 0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,1.0,-1.0, 1.0,1.0,-1.0
)));

-- Here starts modified solids.
----------------------------------------------------------------
-- 5: closed block with 6 polygons and one polygon defined twice
----------------------------------------------------------
insert into solid_tst values(5,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,7,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1,76,1003,1,91,1003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,1.0,-1.0, 1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,-1.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,-1.0, 0.0,1.0,-1.0, 0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,1.0,-1.0, 1.0,1.0,-1.0
)));

----------------------------------------------------------

----------------------------------------------------------------
-- 6: non closed block with 6 polygons and one dangling polygon
----------------------------------------------------------
insert into solid_tst values(6,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,6,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1,76,1003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,1.0,-1.0, 2.0,1.0,1.0,  2.0,0.0,1.0,  1.0,0.0,-1.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,-1.0, 0.0,1.0,-1.0, 0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,1.0,-1.0, 1.0,1.0,-1.0
)));
```

```
-------------------------------------------------------------


-------------------------------------------------------------------------
-- 7: non closed block with only 5 polygons (1 polygon missing)
-------------------------------------------------------------
insert into solid_tst values (7,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,5,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,1.0,-1.0, 1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,-1.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,1.0,-1.0, 1.0,1.0,-1.0
)));

-------------------------------------------------------------------




-------------------------------------------------------------
-- 8: closed block with 6 polygons. The first face got the normal pointing inwards.
-------------------------------------------------------------
insert into solid_tst values(8,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,6,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1,76,1003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,0.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0,  1.0,0.0,-1.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,-1.0, 0.0,1.0,-1.0, 0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,1.0,-1.0, 1.0,1.0,-1.0
)));


-------------------------------------------------------------




-------------------------------------------------------------
-- 9: closed block with 6 polygons. All faces got the normals pointing inwards.
-------------------------------------------------------------
```

```
insert into solid_tst values(9,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,6,1,1003,1,16,1003,1,31,1003,1,46,1003,1,61,1
003,1,76,1003,1),
SDO_Ordinate_Array(
1.0,0.0,-1.0, 1.0,0.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0, 1.0,0.0,-1.0,
1.0,0.0,1.0,  1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 0.0,1.0,-1.0, 0.0,1.0,1.0,
1.0,1.0,-1.0, 1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,1.0,-1.0, 1.0,1.0,-1.0,
1.0,1.0,1.0,  1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,1.0,1.0,  1.0,1.0,1.0,
1.0,1.0,-1.0, 0.0,1.0,-1.0, 0.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,1.0,-1.0
)));

-- Here starts complex solids.

------------------------------------------------------------
-- 10: block with 6 polygons and a block hole trough the block
------------------------------------------------------------
insert into solid_tst values(10,
SDO_Geometry (3008,NULL,NULL,
SDO_Elem_Info_Array(
1,1007,1,1,1006,16,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
121,1003,1,
136,1003,1,
151,1003,1,
166,1003,1,
181,1003,1,
196,1003,1,
211,1003,1,
226,1003,1
),
SDO_Ordinate_Array(
4.0,0.0,-1.0, 4.0,4.0,-1.0, 4.0,4.0,1.0,  4.0,0.0,1.0,  4.0,0.0,-1.0,
4.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 4.0,0.0,-1.0, 4.0,0.0,1.0,
0.0,4.0,1.0,  0.0,4.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,4.0,1.0,
4.0,4.0,-1.0, 0.0,4.0,-1.0, 0.0,4.0,1.0,  4.0,4.0,1.0,  4.0,4.0,-1.0,
1.0,3.0,1.0,  1.0,1.0,1.0,  1.0,1.0,-1.0, 1.0,3.0,-1.0, 1.0,3.0,1.0,
1.0,1.0,1.0,  3.0,1.0,1.0,  3.0,1.0,-1.0, 1.0,1.0,-1.0, 1.0,1.0,1.0,
```

3.0,1.0,1.0, 3.0,3.0,1.0, 3.0,3.0,-1.0, 3.0,1.0,-1.0, 3.0,1.0,1.0,
3.0,3.0,1.0, 1.0,3.0,1.0, 1.0,3.0,-1.0, 3.0,3.0,-1.0, 3.0,3.0,1.0,
0.0,0.0,1.0, 1.0,0.0,1.0, 1.0,4.0,1.0, 0.0,4.0,1.0, 0.0,0.0,1.0,
1.0,0.0,1.0, 3.0,0.0,1.0, 3.0,1.0,1.0, 1.0,1.0,1.0, 1.0,0.0,1.0,
1.0,3.0,1.0, 3.0,3.0,1.0, 3.0,4.0,1.0, 1.0,4.0,1.0, 1.0,3.0,1.0,
3.0,0.0,1.0, 4.0,0.0,1.0, 4.0,4.0,1.0, 3.0,4.0,1.0, 3.0,0.0,1.0,
0.0,4.0,-1.0, 1.0,4.0,-1.0, 1.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,4.0,-1.0,
1.0,4.0,-1.0, 3.0,4.0,-1.0, 3.0,3.0,-1.0, 1.0,3.0,-1.0, 1.0,4.0,-1.0,
1.0,1.0,-1.0, 3.0,1.0,-1.0, 3.0,0.0,-1.0, 1.0,0.0,-1.0, 1.0,1.0,-1.0,
3.0,4.0,-1.0, 4.0,4.0,-1.0, 4.0,0.0,-1.0, 3.0,0.0,-1.0, 3.0,4.0,-1.0
)));

-- 11: block with 6 polygons and a prismatic hole trough the block that is separating the block into two parts connected only by edges
---
insert into solid_tst values(11,
SDO_Geometry (3008,NULL,NULL,
SDO_Elem_Info_Array(
1,1007,1,1,1006,11,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
127,1003,1,
139,1003,1,
160,1003,1
),
SDO_Ordinate_Array(
4.0,0.0,-1.0, 4.0,4.0,-1.0, 4.0,4.0,1.0, 4.0,0.0,1.0, 4.0,0.0,-1.0,
4.0,0.0,1.0, 0.0,0.0,1.0, 0.0,0.0,-1.0, 4.0,0.0,-1.0, 4.0,0.0,1.0,
0.0,4.0,1.0, 0.0,4.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0, 0.0,4.0,1.0,
4.0,4.0,-1.0, 0.0,4.0,-1.0, 0.0,4.0,1.0, 4.0,4.0,1.0, 4.0,4.0,-1.0,
3.0,1.0,1.0, 3.0,0.0,1.0, 3.0,0.0,-1.0, 3.0,1.0,-1.0, 3.0,1.0,1.0,
3.0,0.0,1.0, 4.0,1.0,1.0, 4.0,1.0,-1.0, 3.0,0.0,-1.0, 3.0,0.0,1.0,
4.0,1.0,1.0, 3.0,1.0,1.0, 3.0,1.0,-1.0, 4.0,1.0,-1.0, 4.0,1.0,1.0,
0.0,4.0,1.0, 0.0,0.0,1.0, 3.0,0.0,1.0, 3.0,1.0,1.0, 4.0,1.0,1.0, 4.0,4.0,1.0, 0.0,4.0,1.0,
3.0,0.0,1.0, 4.0,0.0,1.0, 4.0,1.0,1.0, 3.0,0.0,1.0,
0.0,4.0,-1.0, 4.0,4.0,-1.0, 4.0,1.0,-1.0, 3.0,1.0,-1.0, 3.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,4.0,-1.0,
4.0,1.0,-1.0, 4.0,0.0,-1.0, 3.0,0.0,-1.0, 4.0,1.0,-1.0

```
)));


------------------------------------------------------------
-- 12: block with 6 polygons and a block hole PARTLY trough the block
------------------------------------------------------------
insert into solid_tst values(12,
SDO_Geometry (3008,NULL,NULL,
SDO_Elem_Info_Array(
1,1007,1,1,1006,14,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
121,1003,1,
136,1003,1,
151,1003,1,
166,1003,1,
181,1003,1,
196,1003,1
),
SDO_Ordinate_Array(
4.0,0.0,-1.0, 4.0,4.0,-1.0, 4.0,4.0,1.0,  4.0,0.0,1.0,  4.0,0.0,-1.0,
4.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 4.0,0.0,-1.0, 4.0,0.0,1.0,
0.0,4.0,1.0,  0.0,4.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,4.0,1.0,
4.0,4.0,-1.0, 0.0,4.0,-1.0, 0.0,4.0,1.0,  4.0,4.0,1.0,  4.0,4.0,-1.0,
1.0,3.0,1.0,  1.0,1.0,1.0,  1.0,1.0,0.0, 1.0,3.0,0.0, 1.0,3.0,1.0,
1.0,1.0,1.0,  3.0,1.0,1.0,  3.0,1.0,0.0, 1.0,1.0,0.0, 1.0,1.0,1.0,
3.0,1.0,1.0,  3.0,3.0,1.0,  3.0,3.0,0.0, 3.0,1.0,0.0, 3.0,1.0,1.0,
3.0,3.0,1.0,  1.0,3.0,1.0,  1.0,3.0,0.0, 3.0,3.0,0.0, 3.0,3.0,1.0,
0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,4.0,1.0,  0.0,4.0,1.0,  0.0,0.0,1.0,
1.0,0.0,1.0,  3.0,0.0,1.0,  3.0,1.0,1.0,  1.0,1.0,1.0,  1.0,0.0,1.0,
1.0,3.0,1.0,  3.0,3.0,1.0,  3.0,4.0,1.0,  1.0,4.0,1.0,  1.0,3.0,1.0,
3.0,0.0,1.0,  4.0,0.0,1.0,  4.0,4.0,1.0,  3.0,4.0,1.0,  3.0,0.0,1.0,
0.0,4.0,-1.0, 4.0,4.0,-1.0, 4.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,4.0,-1.0,
1.0,3.0,0.0,  1.0,1.0,0.0,  3.0,1.0,0.0,  3.0,3.0,0.0,  1.0,3.0,0.0
)));


------------------------------------------------------------
-- 13: block with 6 polygons and a prismatic hole PARTLY trough the block.
------------------------------------------------------------
```

```
insert into solid_tst values(13,
SDO_Geometry (3008,NULL,NULL,
SDO_Elem_Info_Array(
1,1007,1,1,1006,11,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
127,1003,1,
139,1003,1,
154,1003,1
),
SDO_Ordinate_Array(
4.0,0.0,-1.0, 4.0,4.0,-1.0, 4.0,4.0,1.0,  4.0,0.0,1.0,  4.0,0.0,-1.0,
4.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,-1.0, 4.0,0.0,-1.0, 4.0,0.0,1.0,
0.0,4.0,1.0,  0.0,4.0,-1.0, 0.0,0.0,-1.0, 0.0,0.0,1.0,  0.0,4.0,1.0,
4.0,4.0,-1.0, 0.0,4.0,-1.0, 0.0,4.0,1.0,  4.0,4.0,1.0,  4.0,4.0,-1.0,
3.0,1.0,1.0,  3.0,0.0,1.0,  3.0,0.0,0.0, 3.0,1.0,0.0, 3.0,1.0,1.0,
3.0,0.0,1.0,  4.0,1.0,1.0,  4.0,1.0,0.0, 3.0,0.0,0.0, 3.0,0.0,1.0,
4.0,1.0,1.0,  3.0,1.0,1.0,  3.0,1.0,0.0, 4.0,1.0,0.0, 4.0,1.0,1.0,
0.0,4.0,1.0,  0.0,0.0,1.0,  3.0,0.0,1.0,  3.0,1.0,1.0,  4.0,1.0,1.0,  4.0,4.0,1.0,  0.0,4.0,1.0,
3.0,0.0,1.0,  4.0,0.0,1.0,  4.0,1.0,1.0,  3.0,0.0,1.0,
0.0,4.0,-1.0, 4.0,4.0,-1.0, 4.0,0.0,-1.0, 0.0,0.0,-1.0, 0.0,4.0,-1.0,
3.0,1.0,0.0,  3.0,0.0,0.0,  4.0,1.0,0.0,  3.0,1.0,0.0
)));



--------------------------------------------------------------------------------
-- 14: SOLID with hole (small tetrahedron completley inside the Large one)
--------------------------------------------------------------------------------

insert into solid_tst values(14,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,
1,1006,4,   1,1003,1,13,1003,1,25,1003,1,37,1003,1,
49,2006,4, 49,2003,1,61,2003,1,73,2003,1,85,2003,1),
SDO_Ordinate_Array(
0,0,0, 0,0,5, 5,0,0, 0,0,0,
0,0,0, 5,0,0, 0,-5,0, 0,0,0,
0,0,0, 0,-5,0, 0,0,5, 0,0,0,
5,0,0, 0,0,5, 0,-5,0, 5,0,0,
```

```
1,-1,1, 2,-1,1, 1,-1,2, 1,-1,1,
1,-1,1, 1,-2,1, 2,-1,1, 1,-1,1,
1,-1,1, 1,-1,2, 1,-2,1, 1,-1,1,
2,-1,1, 1,-2,1, 1,-1,2, 2,-1,1
)));


------------------------------------------------------------
--15:  two blocks with a shared edge
------------------------------------------------------------
insert into solid_tst values(15,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,12,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
121,1003,1,
136,1003,1,
151,1003,1,
166,1003,1
),
SDO_Ordinate_Array(
1.0,0.0,0.0,  1.0,1.0,0.0,  1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,0.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,0.0,  1.0,0.0,0.0,  1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,0.0,  0.0,0.0,0.0,  0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,0.0,  0.0,1.0,0.0,  0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,0.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,0.0,  1.0,0.0,0.0,  0.0,0.0,0.0,  0.0,1.0,0.0,  1.0,1.0,0.0,
1.0,1.0,1.0,  1.0,2.0,1.0,  1.0,2.0,2.0,  1.0,1.0,2.0,  1.0,1.0,1.0,
1.0,1.0,2.0,  0.0,1.0,2.0,  0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,2.0,
0.0,2.0,2.0,  0.0,2.0,1.0,  0.0,1.0,1.0,  0.0,1.0,2.0,  0.0,2.0,2.0,
1.0,2.0,1.0,  0.0,2.0,1.0,  0.0,2.0,2.0,  1.0,2.0,2.0,  1.0,2.0,1.0,
1.0,2.0,2.0,  0.0,2.0,2.0,  0.0,1.0,2.0,  1.0,1.0,2.0,  1.0,2.0,2.0,
1.0,2.0,1.0,  1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,2.0,1.0,  1.0,2.0,1.0
)));


------------------------------------------------------------
--16:  two disjoint blocks
------------------------------------------------------------
```

```
insert into solid_tst values(16,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,12,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
106,1003,1,
121,1003,1,
136,1003,1,
151,1003,1,
166,1003,1
),
SDO_Ordinate_Array(
1.0,0.0,0.0,  1.0,1.0,0.0,  1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,0.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,0.0,  1.0,0.0,0.0,  1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,0.0,  0.0,0.0,0.0,  0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,0.0,  0.0,1.0,0.0,  0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,0.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,0.0,  1.0,0.0,0.0,  0.0,0.0,0.0,  0.0,1.0,0.0,  1.0,1.0,0.0,
1.0,2.0,2.0,  1.0,3.0,2.0,  1.0,3.0,3.0,  1.0,2.0,3.0,  1.0,2.0,2.0,
1.0,2.0,3.0,  0.0,2.0,3.0,  0.0,2.0,2.0,  1.0,2.0,2.0,  1.0,2.0,3.0,
0.0,3.0,3.0,  0.0,3.0,2.0,  0.0,2.0,2.0,  0.0,2.0,3.0,  0.0,3.0,3.0,
1.0,3.0,2.0,  0.0,3.0,2.0,  0.0,3.0,3.0,  1.0,3.0,3.0,  1.0,3.0,2.0,
1.0,3.0,3.0,  0.0,3.0,3.0,  0.0,2.0,3.0,  1.0,2.0,3.0,  1.0,3.0,3.0,
1.0,3.0,2.0,  1.0,2.0,2.0,  0.0,2.0,2.0,  0.0,3.0,2.0,  1.0,3.0,2.0
)));



------------------------------------------------------------
--17:  two blocks that are overlapping
------------------------------------------------------------
insert into solid_tst values(17,
SDO_Geometry (3008,NULL,NULL ,
SDO_Elem_Info_Array(1,1007,1,1,1006,12,
1,1003,1,
16,1003,1,
31,1003,1,
46,1003,1,
61,1003,1,
76,1003,1,
91,1003,1,
```

```
106,1003,1,
121,1003,1,
136,1003,1,
151,1003,1,
166,1003,1
),
SDO_Ordinate_Array(
1.0,0.0,0.0,  1.0,1.0,0.0,  1.0,1.0,1.0,  1.0,0.0,1.0,  1.0,0.0,0.0,
1.0,0.0,1.0,  0.0,0.0,1.0,  0.0,0.0,0.0,  1.0,0.0,0.0,  1.0,0.0,1.0,
0.0,1.0,1.0,  0.0,1.0,0.0,  0.0,0.0,0.0,  0.0,0.0,1.0,  0.0,1.0,1.0,
1.0,1.0,0.0,  0.0,1.0,0.0,  0.0,1.0,1.0,  1.0,1.0,1.0,  1.0,1.0,0.0,
1.0,1.0,1.0,  0.0,1.0,1.0,  0.0,0.0,1.0,  1.0,0.0,1.0,  1.0,1.0,1.0,
1.0,1.0,0.0,  1.0,0.0,0.0,  0.0,0.0,0.0,  0.0,1.0,0.0,  1.0,1.0,0.0,
1.0,0.5,0.5,  1.0,1.5,0.5,  1.0,1.5,1.5,  1.0,0.5,1.5,  1.0,0.5,0.5,
1.0,0.5,1.5,  0.0,0.5,1.5,  0.0,0.5,0.5,  1.0,0.5,0.5,  1.0,0.5,1.5,
0.0,1.5,1.5,  0.0,1.5,0.5,  0.0,0.5,0.5,  0.0,0.5,1.5,  0.0,1.5,1.5,
1.0,1.5,0.5,  0.0,1.5,0.5,  0.0,1.5,1.5,  1.0,1.5,1.5,  1.0,1.5,0.5,
1.0,1.5,1.5,  0.0,1.5,1.5,  0.0,0.5,1.5,  1.0,0.5,1.5,  1.0,1.5,1.5,
1.0,1.5,0.5,  1.0,0.5,0.5,  0.0,0.5,0.5,  0.0,1.5,0.5,  1.0,1.5,0.5
)));



---------------------------------------------------------------------------

-- Metadata insertion

INSERT INTO user_sdo_geom_metadata VALUES('solid_tst', 'geom',
sdo_dim_array( sdo_dim_element('X', -100,100, 0.005),
sdo_dim_element('Y', -100,100, 0.005),
sdo_dim_element('Z', -100,100, 0.005)), NULL);

INSERT INTO user_sdo_geom_metadata VALUES('POLYGONS3D', 'GEOMETRY',
sdo_dim_array( sdo_dim_element('X', -100,100, 0.005),
sdo_dim_element('Y', -100,100, 0.005),
sdo_dim_element('Z', -100,100, 0.005)), NULL);

---------------------------------------------------------------------------

commit;

select id,validate_result from (
  select id, sdo_geom.validate_geometry_with_context (GEOMETRY,
    (select diminfo from user_sdo_geom_metadata where table_name='POLYGONS3D'
and column_name='GEOMETRY')) validate_result
from POLYGONS3D) where validate_result <> 'TRUE';
```

```
select id,validate_result from (
  select id, sdo_geom.validate_geometry_with_context (GEOM,
    (select diminfo from user_sdo_geom_metadata where table_name='SOLID_TST' and
column_name='GEOM')) validate_result
from SOLID_TST) where validate_result <> 'TRUE';

spool off
exit
```

Appendix B

Queries on objects that are expected to be valid and also validated (polygon 1 and solids 1-4 and 10-15)

```
set linesize 1000
set pagesize 50
set echo on
set timing on

drop table solid_tst;
drop table polygons3d;
drop INDEX polygons3d_sidx;
drop INDEX solid_tst_sidx;

create table solid_tst (id integer, geom sdo_geometry);
create table polygons3d(id number, geometry sdo_geometry);
--
delete from user_sdo_geom_metadata;
delete from user_sdo_geom_metadata where table_name='SOLID_TST';
delete from user_sdo_geom_metadata where table_name='polygons3d';


INSERTIONS NOT INCLUDED – SEE APENDIX A.

-- Metadata insertion

INSERT INTO user_sdo_geom_metadata VALUES('solid_tst', 'geom',
sdo_dim_array( sdo_dim_element('X', -100,100, 0.005),
sdo_dim_element('Y', -100,100, 0.005),
sdo_dim_element('Z', -100,100, 0.005)), NULL);

INSERT INTO user_sdo_geom_metadata VALUES('POLYGONS3D', 'GEOMETRY',
sdo_dim_array( sdo_dim_element('X', -100,100, 0.005),
sdo_dim_element('Y', -100,100, 0.005),
sdo_dim_element('Z', -100,100, 0.005)), NULL);

------------------------------------------------------------------------

commit;

SELECT c.id,
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(c.geometry, 0.005)
FROM polygons3d c;
```

```
SELECT p.id, SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(p.geom,
0.005) FROM solid_tst p;



SELECT p.id, SDO_GEOM.SDO_LENGTH(p.geom, 0.005) FROM solid_tst p;
SELECT p.id, SDO_GEOM.SDO_AREA(p.geom, 0.005) FROM solid_tst p WHERE
p.id>9;
SELECT p.id, SDO_GEOM.SDO_VOLUME(p.geom, 0.005) FROM solid_tst p
WHERE p.id>9;

SELECT p.id, q.id, SDO_GEOM.SDO_DISTANCE(p.geom, q.geom, 0.005) FROM
solid_tst p, solid_tst q WHERE p.id > q.id AND p.id<5 AND q.id<5;


CREATE INDEX solid_tst_sidx on solid_tst(geom) INDEXTYPE IS
mdsys.spatial_index PARAMETERS ('sdo_indx_dims=3');

SELECT p.id, q.id FROM solid_tst p, solid_tst q WHERE
SDO_ANYINTERACT(p.geom, q.geom)='TRUE' AND p.id > q.id AND p.id<5 AND
q.id<5;


-- some additional queries (note function s not yet supported in 3D)

SELECT id,
  SDO_GEOM.SDO_CENTROID(geom, 0.005),
  SDO_GEOM.SDO_CONVEXHULL(geom, 0.005),
  SDO_GEOM.SDO_BUFFER(geom, 1, 0.005),
  SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(geom, 0.005)
FROM solid_tst;

SELECT p.id, q.id,
   SDO_GEOM.SDO_UNION(p.geom, q.geom, 0.005)
FROM solid_tst p, solid_tst q
WHERE p.id > q.id;

SELECT p.id, q.id,
  SDO_GEOM.SDO_INTERSECTION(p.geom, q.geom, 0.005)
FROM solid_tst p, solid_tst q
WHERE p.id > q.id;

exit
```

Appendix C

Scripts for creation of building dataset from the topologic structure (sss).

Doitall.sql

```
--converts sss to sdo_geometry multipolygon
set serveroutput on;
set timing on;
@create_sdo_body_col; --creates tables
@return_3Dcollection; -- function to return a polygon from sss
@populate_sdo_body_col; -- creates multipolygon data type (collection of polygons)

select * from sdo_3Dcol_w;

delete from user_sdo_geom_metadata where table_name = 'SDO_3DCOL_W';


insert into user_sdo_geom_metadata values
('SDO_3DCOL_W','GEOM', mdsys.SDO_DIM_ARRAY
(
mdsys.SDO_DIM_ELEMENT ('X',2800, 16000, .00001),
mdsys.SDO_DIM_ELEMENT ('Y',6000, 18000, .00001),
mdsys.SDO_DIM_ELEMENT ('Z',150, 350, .00001)
), NULL
);

create index sdo_3DCOL_w_idx ON sdo_3DCOL_w (geom) INDEXTYPE IS
MDSYS.SPATIAL_INDEX
parameters ('sdo_indx_dims=3');

delete from mscatalog where tablename = 'SDO_3DCOL_W';
insert into mscatalog (tablename, entitynum) values ('SDO_3DCOL_W',1);

commit;
```

return_3dcollection.sql

```
DROP FUNCTION return_3Dcollection;

CREATE OR REPLACE FUNCTION return_3Dcollection (BodyID Rbodyg_w.body_id%type)
RETURN mdsys.sdo_geometry IS

collection mdsys.sdo_geometry:=mdsys.sdo_geometry (3007,NULL,NULL,
      mdsys.sdo_ELEM_INFO_ARRAY(0),mdsys.sdo_ORDINATE_ARRAY(0));
```

```
TYPE xx is varray(2000) of number;
x xx := xx (0);
y xx := xx(0);
z xx := xx(0);
FaceID xx := xx(0);

ordinate_array mdsys.sdo_ordinate_array := mdsys.sdo_ordinate_array (2000);
elem_info_array mdsys.sdo_elem_info_array :=mdsys.sdo_elem_info_array (2000);


ii number;
i number;
l number;
jj number;
ifc number;
numf number;
iface number;

ll number :=0;  -- number values in ordinate_array
ff number :=0;  -- number values in elem_info_array

offset number := 1;
etype number := 1003;
interp number := 1;


cursor crd is
  select x,y,z
    from Rface_w, Rnode_w
    where
      Rface_w.face_id=iface and
      Rface_w.node_id=Rnode_w.node_id
    order by seq;

cursor fac is
  select face_id
    from Rbodyg_w
    where body_id=BodyID
    order by seq;


BEGIN

elem_info_array.delete;
ordinate_array.delete;
FaceID.delete;

OPEN fac;
FETCH fac BULK COLLECT INTO FaceID;
CLOSE fac;

 numf := FaceID.count;  --number faces in BodyID

FOR ifc IN 1..numf LOOP

  ff:=ff+1;
```

```
 elem_info_array.extend;
 elem_info_array (ff):=offset;

 ff:=ff+1;
 elem_info_array.extend;
 elem_info_array (ff):=etype;

 ff:=ff+1;
 elem_info_array.extend;
 elem_info_array(ff):=interp;


 iface:=FaceID (ifc);

 x.delete;
 y.delete;
 z.delete;


 OPEN crd;
 FETCH crd BULK COLLECT INTO x, y, z;
 CLOSE crd;

 jj:=0;

 l:=x.count;

--dbms_output.put_line ('number nodes in faces '|| l);

 FOR ii IN 1..l LOOP

   jj:=jj+1;
   ordinate_array.extend;
   ordinate_array(ll+jj):=x(ii);

   jj:=jj+1;
   ordinate_array.extend;
   ordinate_array(ll+jj):= y(ii);

   jj:=jj+1;
   ordinate_array.extend;
   ordinate_array(ll+jj):= z(ii);

 END LOOP;

 ll :=ordinate_array.count;       --total number coordinates

 --dbms_output.put_line  ('ordinate_array '|| ordinate_array.count);


   ordinate_array.extend;
   ordinate_array(ll+1):=x(1);

   ordinate_array.extend;
   ordinate_array(ll+2):= y(1);

   ordinate_array.extend;
```

```
    ordinate_array(ll+3):= z(1);


  ll:= ordinate_array.count;

  offset :=ordinate_array.count +1;
  --dbms_output.put_line  ('offset '|| offset);

END LOOP;

  --dbms_output.put_line ('elem_info_array '|| elem_info_array.count);
  --dbms_output.put_line  ('ordinate_array '|| ordinate_array.count);

  collection := mdsys.sdo_geometry (3007, NULL, NULL,
    elem_info_array, ordinate_array);

RETURN collection;
END return_3Dcollection;
```

Populate_sdo_body.sql

```
DECLARE

 col3D mdsys.sdo_geometry;

 ii number;


BEGIN

col3D:=mdsys.sdo_geometry (3007,NULL,NULL,
      mdsys.SDO_ELEM_INFO_ARRAY(0),mdsys.sdo_ordinate_array(0));


FOR ii in 1..20021 LOOP

     col3D:=return_3Dcollection (ii);
     insert into sdo_3Dcol_w values (ii,col3D);


END LOOP;


END;
```

Create_sdo_body.sql

```
DROP TABLE sdo_3dcol_w;

CREATE TABLE sdo_3dcol_w (body_id number, geom mdsys.sdo_geometry);
```

Appendix D

Point cloud generation script

```
declare
  pc sdo_pc;
begin
  -- Initialize the Point Cloud object.

pc := sdo_pc_pkg.init(
          'BASE', -- Table that has the SDO_POINT_CLOUD column defined
          'PC',   -- Column name of the SDO_POINT_CLOUD object
          'BLKTAB', -- Table to store blocks of the point cloud
           'blk_capacity=1000000', -- max # of points per block
           mdsys.sdo_geometry(2003, 8307, null,
              mdsys.sdo_elem_info_array(1,1003,3),
              mdsys.sdo_ordinate_array(82840, 435757, 82416, 436292)),
-- Extent
    0.05, -- Tolerance for point cloud
           3, -- Total number of dimensions
           null);



  -- Insert the Point Cloud object  into the "base" table.
  insert into base values (pc);

  -- Create the blocks for the point cloud.
  sdo_pc_pkg.create_pc(
    pc,       -- Initialized PointCloud object
    'INPTAB2', -- Name of input table to ingest into the pointcloud
    'RES'     -- Name of output table that stores the points (with
ptn_id,pt_id)
  );
end;
/
```