

**Scripts create structure**

```
1 //CREATE STRUCTURE//
2
3
4 CREATE TABLE NODE (NODE_ID NUMBER NOT NULL, X NUMBER NOT NULL, Y NUMBER
   NOT NULL, Z NUMBER NOT NULL, NODE_IN_FACE_ID_REF NUMBER
5 CONSTRAINT NODE_PK PRIMARY KEY (NODE_ID)ENABLE);
6
7 CREATE TABLE EDGE (EDGE_ID NUMBER NOT NULL, STARTNODE NUMBER NOT NULL,
   ENDNODE NUMBER NOT NULL, EDGE_IN_FACE_ID_REF NUMBER
8 CONSTRAINT EDGE_PK PRIMARY KEY (EDGE_ID) ENABLE);
9
10 CREATE TABLE RING2EDGE (EDGE_ID_REF NUMBER NOT NULL, RING_ID_REF NUMBER
   NOT NULL, ORIENTATION VARCHAR2(1) NOT NULL,
11 CONSTRAINT RING2EDGE_PK PRIMARY KEY(EDGE_ID_REF, RING_ID_REF)ENABLE );
12
13 CREATE TABLE RING (RING_ID NUMBER NOT NULL, FACE_ID_REF NUMBER NOT NULL,
   INOUT VARCHAR2(1) NOT NULL, RING_ORDINATES MDSYS.SDO_ORDINATE_ARRAY,
14 CONSTRAINT RING_PK PRIMARY KEY (RING_ID) ENABLE);
15
16 CREATE TABLE FACE (FACE_ID NUMBER NOT NULL, SHELL_ID_REF_POS NUMBER NOT
   NULL, SHELL_ID_REF_NEG NUMBER NOT NULL,
17 CONSTRAINT FACE_PK PRIMARY KEY (FACE_ID) ENABLE);
18
19 CREATE TABLE SHELL (SHELL_ID NUMBER NOT NULL, VOLUME_ID_REF NUMBER NOT
   NULL, INOUT VARCHAR2(1) NOT NULL,
20 CONSTRAINT SHELL_PK PRIMARY KEY (SHELL_ID)ENABLE);
21
22 CREATE TABLE VOLUME (VOLUME_ID NUMBER NOT NULL,
23 CONSTRAINT VOLUME_PK PRIMARY KEY (VOLUME_ID)ENABLE);
24
25 ALTER TABLE EDGE ADD CONSTRAINT ENDNODE_FK FOREIGN KEY (ENDNODE)
   REFERENCES NODE (NODE_ID) ENABLE;
26 ALTER TABLE EDGE ADD CONSTRAINT STARTNODE_FK FOREIGN KEY (STARTNODE)
   REFERENCES NODE (NODE_ID) ENABLE;
27 ALTER TABLE RING2EDGE ADD CONSTRAINT EDGE_FK FOREIGN KEY (EDGE_ID_REF)
   REFERENCES EDGE (EDGE_ID) ENABLE;
28 ALTER TABLE RING2EDGE ADD CONSTRAINT RING_FK FOREIGN KEY (RING_ID_REF)
   REFERENCES RING (RING_ID) ENABLE;
29 ALTER TABLE RING ADD CONSTRAINT FACE_FK FOREIGN KEY (FACE_ID_REF)
   REFERENCES FACE (FACE_ID) ENABLE;
30 ALTER TABLE FACE ADD CONSTRAINT SHELLPOS_FK FOREIGN KEY
   (SHELL_ID_REF_POS) REFERENCES SHELL (SHELL_ID) ENABLE;
31 ALTER TABLE FACE ADD CONSTRAINT SHELLNEG_FK FOREIGN KEY
   (SHELL_ID_REF_NEG) REFERENCES SHELL (SHELL_ID) ENABLE;
32 ALTER TABLE SHELL ADD CONSTRAINT VOLUME_FK FOREIGN KEY (VOLUME_ID_REF)
   REFERENCES VOLUME (VOLUME_ID) ENABLE;
```

```
1 //CREATE VIEW NODE2SHELL//
2
3
4 --DROP VIEW node2shell;
5 CREATE VIEW node2shell AS
6
7 SELECT FACE.shell_id_ref_pos AS shell, RING.face_id_ref AS
   face,RING.ring_id AS ring,RING2EDGE.edge_id_ref AS edge, EDGE.startnode
   AS node FROM EDGE,RING2EDGE,RING,FACE
8 WHERE FACE.face_id=RING.face_id_ref AND RING.ring_id=
   RING2EDGE.ring_id_ref AND RING2EDGE.edge_id_ref=EDGE.edge_id
9 UNION ALL
10 SELECT FACE.shell_id_ref_pos AS shell, RING.face_id_ref AS
   face,RING.ring_id AS ring,RING2EDGE.edge_id_ref AS edge, EDGE.endnode AS
   node FROM EDGE,RING2EDGE,RING,FACE
11 WHERE FACE.face_id=RING.face_id_ref AND RING.ring_id=
   RING2EDGE.ring_id_ref AND RING2EDGE.edge_id_ref=EDGE.edge_id
12
13 UNION ALL
14
15 SELECT FACE.shell_id_ref_neg AS shell, RING.face_id_ref AS
   face,RING.ring_id AS ring,RING2EDGE.edge_id_ref AS edge,EDGE.startnode AS
   node FROM EDGE,RING2EDGE,RING,FACE
16 WHERE FACE.face_id=RING.face_id_ref AND RING.ring_id=
   RING2EDGE.ring_id_ref AND RING2EDGE.edge_id_ref=EDGE.edge_id
17 UNION ALL
18 SELECT FACE.shell_id_ref_neg AS shell, RING.face_id_ref AS
   face,RING.ring_id AS ring,RING2EDGE.edge_id_ref AS edge,EDGE.endnode AS
   node FROM EDGE,RING2EDGE,RING,FACE
19 WHERE FACE.face_id=RING.face_id_ref AND RING.ring_id=
   RING2EDGE.ring_id_ref AND RING2EDGE.edge_id_ref=EDGE.edge_id
20
21 ORDER BY shell,face,ring,edge,node;
```

---

```
1 //CREATE VIEW RINGORDINATES//
2
3
4 --DROP VIEW ringordinates;
5 CREATE VIEW ringordinates AS
6 SELECT ring_id, getRINGORDINATES(ring_id) as ringordinates FROM ring;
```

```
1 //CREATE INTERACTION TABLE FOR THE BENEFIT OF SDO_ANYINTERACT FUNCTION,  
  WHICH IS USED IN VALIDATION TESTS//  
2  
3  
4 --DROP TABLE chk_interaction CASCADE CONSTRAINTS;  
5 CREATE TABLE chk_interaction (id NUMBER, geom SDO_GEOMETRY);  
6  
7 --DELETE FROM user_sdo_geom_metadata WHERE table_name = 'chk_interaction';  
8 INSERT INTO user_sdo_geom_metadata (TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)  
9 VALUES ('chk_interaction','geom',  
10 SDO_DIM_ARRAY(  
11 SDO_DIM_ELEMENT('X', 84935, 86085, 0.005),  
12 SDO_DIM_ELEMENT('Y', 444960, 446801, 0.005),  
13 SDO_DIM_ELEMENT('Z', -5, 95, 0.005)  
14 ),NULL);  
15  
16 --DROP INDEX SDX_interaction;  
17 CREATE INDEX SDX_interaction ON chk_interaction(geom)  
18 INDEXTYPE IS MDSYS.SPATIAL_INDEX  
19 PARAMETERS ('SDO_INDX_DIMS=3');
```



## Scripts validation tests

```
1 //VALIDATION TEST 1) UNIQUE PRIMITIVES//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE xyz_rec IS RECORD (x NUMBER, y NUMBER, z NUMBER);
8 TYPE xyz_type IS TABLE OF xyz_rec;
9 TYPE edge_rec1 IS RECORD (edge NUMBER, startnode NUMBER, endnode NUMBER);
10 TYPE edge_rec2 IS RECORD (startnode NUMBER, endnode NUMBER);
11 TYPE edge_type1 IS TABLE OF edge_rec1;
12 TYPE edge_type2 IS TABLE OF edge_rec2;
13 TYPE face_rec IS RECORD (rings NUMBER, min NUMBER, max NUMBER, edges
    NUMBER);
14 TYPE face_type IS TABLE OF face_rec;
15 TYPE element_type IS TABLE OF NUMBER;
16 tab_xyz xyz_type;
17 tab_node element_type;
18 tab_edge1 edge_type1;
19 tab_edge2 edge_type2;
20 tab_edge element_type;
21 tab_face face_type;
22 tab_face2 element_type;
23 tab_testface1 element_type;
24 tab_testface2 element_type;
25 vx NUMBER;
26 vy NUMBER;
27 vz NUMBER;
28 v_node NUMBER;
29 v_edge NUMBER;
30 v_startnode NUMBER;
31 v_endnode NUMBER;
32 v_result NUMBER;
33 result1 NUMBER:=0;
34 result2 NUMBER:=0;
35 result3 NUMBER:=0;
36 v_testface1 NUMBER;
37 v_testface2 NUMBER;
38 v_rings NUMBER;
39 v_min NUMBER;
40 v_max NUMBER;
41 v_edges NUMBER;
42
43
44 BEGIN
45
46 --check unique xyz-values of nodes
47 SELECT x,y,z BULK COLLECT INTO tab_xyz FROM node GROUP BY x,y,z HAVING
    COUNT(*)>1;
48 IF tab_xyz.count=0 THEN dbms_output.put_line('unique nodes');
49 ELSE
50 FOR i IN tab_xyz.FIRST..tab_xyz.LAST LOOP
51 vx:=tab_xyz(i).x;
52 vy:=tab_xyz(i).y;
53 vz:=tab_xyz(i).z;
54 dbms_output.put_line('nodes with equal coordinates
    ('||vx||','||vy||','||vz||')');
55 SELECT node_id BULK COLLECT INTO tab_node FROM node WHERE x=vx AND y=vy
    AND z=vz;
56 FOR i IN tab_node.FIRST..tab_node.LAST LOOP
57 v_node:=tab_node(i);
58 dbms_output.put_line(v_node);
59 END LOOP;
```

```
60 END LOOP;
61 END IF;
62
63 --search for edges with same start- and end node (start node equals end
node)
64 SELECT edge_id,startnode,endnode BULK COLLECT INTO tab_edgel FROM edge
where startnode= endnode;
65 IF tab_edgel.count=0 THEN result1:=1;
66 ELSE
67 FOR i IN tab_edgel.FIRST..tab_edgel.LAST LOOP
68 v_edge:=tab_edgel(i).edge;
69 dbms_output.put_line('edge '||v_edge||' (start node equals end node)');
70 END LOOP;
71 END IF;
72
73 --check edges with same begin- and end node
74 SELECT startnode,endnode BULK COLLECT INTO tab_edge2 FROM edge GROUP BY
startnode,endnode HAVING COUNT(*)>1;
75 IF tab_edge2.count=0 THEN result2:=1;
76 ELSE FOR i IN tab_edge2.FIRST..tab_edge2.LAST LOOP
77 v_startnode:=tab_edge2(i).startnode;
78 v_endnode:=tab_edge2(i).endnode;
79 SELECT edge_id BULK COLLECT INTO tab_edge FROM edge WHERE
startnode=v_startnode AND endnode=v_endnode;
80 FOR i IN tab_edge.FIRST..tab_edge.LAST LOOP
81 v_edge:=tab_edge(i);
82 dbms_output.put_line('edges with same start- and end node: '||v_edge);
83 END LOOP;
84 END LOOP;
85 END IF;
86
87 --search for edges where end node is begin node and begin node is end node
88 SELECT a.edge_id, a.startnode, a.endnode BULK COLLECT INTO tab_edgel FROM
edge a, edge b WHERE a.endnode= b.startnode AND a.startnode= b.endnode
AND a.startnode<> b.startnode;
89 IF tab_edgel.count=0 THEN result3:=1;
90 ELSE
91 FOR i IN tab_edgel.FIRST..tab_edgel.LAST LOOP
92 v_edge:=tab_edgel(i).edge;
93 dbms_output.put_line('edge '||v_edge||' (end node = start node and start
node = end node)');
94 END LOOP;
95 END IF;
96
97 --conclusion edges
98 v_result:=result1+result2+result3;
99 IF v_result=3 THEN dbms_output.put_line('unique edges'); END IF;
100
101 --check faces: first eliminate potentially equal faces based on some main
characteristics
102 SELECT rings,min,max,edges BULK COLLECT INTO tab_face FROM (SELECT
ring.face_id_ref AS face, count(distinct ring.ring_id) AS rings,
min(ring2edge.edge_id_ref) AS min ,max(ring2edge.edge_id_ref) AS max,
count (ring2edge.edge_id_ref) AS edges FROM ring2edge, ring WHERE
ring.ring_id= ring2edge.ring_id_ref GROUP BY ring.face_id_ref) GROUP BY
rings,min,max,edges HAVING COUNT(*)=2;
103 IF tab_face.count=0 THEN dbms_output.put_line('unique faces');
104
105 --then compare collections of edges of two faces with the same main
characteristics
106 ELSE
107 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
108 v_rings:=tab_face(i).rings;
```

```
109 v_min:=tab_face(i).min;
110 v_max:=tab_face(i).max;
111 v_edges:=tab_face(i).edges;
112 SELECT min(face) INTO v_testface1 FROM (SELECT ring.face_id_ref AS face,
count(distinct ring.ring_id) AS rings, min(ring2edge.edge_id_ref) AS min
,max(ring2edge.edge_id_ref) AS max, count(ring2edge.edge_id_ref) AS edges
FROM ring2edge, ring WHERE ring.ring_id= ring2edge.ring_id_ref GROUP BY
ring.face_id_ref) WHERE rings=v_rings AND min=v_min AND max=v_max AND
edges=v_edges;
113 SELECT max(face) INTO v_testface2 FROM (SELECT ring.face_id_ref AS face,
count(distinct ring.ring_id) AS rings, min(ring2edge.edge_id_ref) AS min
,max(ring2edge.edge_id_ref) AS max, count(ring2edge.edge_id_ref) AS edges
FROM ring2edge, ring WHERE ring.ring_id= ring2edge.ring_id_ref GROUP BY
ring.face_id_ref) WHERE rings=v_rings AND min=v_min AND max=v_max AND
edges=v_edges;
114 SELECT edge_id_ref BULK COLLECT INTO tab_testface1 FROM ring2edge,ring
WHERE ring2edge.ring_id_ref=ring.ring_id AND ring.face_id_ref=v_testface1;
115 SELECT edge_id_ref BULK COLLECT INTO tab_testface2 FROM ring2edge,ring
WHERE ring2edge.ring_id_ref=ring.ring_id AND ring.face_id_ref=v_testface2;
116 IF tab_testface1=tab_testface2 THEN dbms_output.put_line('equal faces:
'||v_testface1||','||v_testface2); END IF;
117 END LOOP;
118 END IF;
119
120 --select not tested faces
121 SELECT rings,min,max,edges BULK COLLECT INTO tab_face FROM (SELECT
ring.face_id_ref AS face, count(distinct ring.ring_id) AS rings,
min(ring2edge.edge_id_ref) AS min ,max(ring2edge.edge_id_ref) AS max,
count(ring2edge.edge_id_ref) AS edges FROM ring2edge, ring WHERE
ring.ring_id= ring2edge.ring_id_ref GROUP BY ring.face_id_ref) GROUP BY
rings,min,max,edges HAVING COUNT(*)>2;
122 IF tab_face.count>0 THEN dbms_output.put_line('faces not tested:');
123 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
124 v_rings:=tab_face(i).rings;
125 v_min:=tab_face(i).min;
126 v_max:=tab_face(i).max;
127 v_edges:=tab_face(i).edges;
128 SELECT face BULK COLLECT INTO tab_face2 FROM (SELECT ring.face_id_ref AS
face, count(distinct ring.ring_id) AS rings, min(ring2edge.edge_id_ref)
AS min ,max(ring2edge.edge_id_ref) AS max, count(ring2edge.edge_id_ref)
AS edges FROM ring2edge, ring WHERE ring.ring_id= ring2edge.ring_id_ref
GROUP BY ring.face_id_ref) WHERE rings=v_rings AND min=v_min AND
max=v_max AND edges=v_edges;
129 FOR i IN tab_face2.FIRST..tab_face2.LAST LOOP
130 dbms_output.put_line(tab_face2(i));
131 END LOOP;
132 END LOOP;
133 END IF;
134
135
136 END;
```

```
1 //VALIDATIONTEST 2) PRIMITIVE REFERENCES//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE volume_face_rec IS RECORD (volume NUMBER, face NUMBER);
8 TYPE ring_edge_rec IS RECORD (ring NUMBER, edges NUMBER);
9 TYPE face_edge_rec IS RECORD (face NUMBER, edge NUMBER);
10 TYPE volume_face_type IS TABLE OF volume_face_rec;
11 TYPE ring_edge_type IS TABLE OF ring_edge_rec;
12 TYPE face_edge_type IS TABLE OF face_edge_rec;
13 TYPE element_type IS TABLE OF NUMBER;
14 tab_face_edge FACE_EDGE_TYPE;
15 tab_ring_edge RING_EDGE_TYPE;
16 tab_volume_face VOLUME_FACE_TYPE;
17 tab_node ELEMENT_TYPE;
18 tab_edge ELEMENT_TYPE;
19 tab_ring ELEMENT_TYPE;
20 tab_face ELEMENT_TYPE;
21 tab_shell ELEMENT_TYPE;
22 v_node NUMBER;
23 v_edge NUMBER;
24 v_ring NUMBER;
25 v_face NUMBER;
26 v_shell NUMBER;
27 v_volume NUMBER;
28
29
30 BEGIN
31
32 --chk for isolated nodes
33 SELECT node_id BULK COLLECT INTO tab_node FROM node WHERE node_id NOT IN
34 (SELECT startnode FROM edge) AND node_id NOT IN (SELECT endnode FROM edge);
35 IF tab_node.count=0 THEN dbms_output.put_line('no isolated nodes');
36 ELSE
37 dbms_output.put_line('isolated nodes:');
38 FOR i IN tab_node.FIRST..tab_node.LAST LOOP
39 v_node:=tab_node(i);
40 dbms_output.put_line(v_node);
41 END LOOP;
42 END IF;
43
44 --chk for isolated edges
45 SELECT edge_id BULK COLLECT INTO tab_edge FROM edge WHERE edge_id NOT IN
46 (SELECT edge_id_ref FROM ring2edge);
47 IF tab_edge.count=0 THEN dbms_output.put_line('no isolated edges');
48 ELSE
49 dbms_output.put_line('isolated edges:');
50 FOR i IN tab_edge.FIRST..tab_edge.LAST LOOP
51 v_edge:=tab_edge(i);
52 dbms_output.put_line(v_edge);
53 END LOOP;
54 END IF;
55
56 --chk for nonexistent rings
57 SELECT DISTINCT ring_id BULK COLLECT INTO tab_ring FROM ring WHERE
58 ring_id NOT IN (SELECT ring_id_ref FROM ring2edge);
59 IF tab_ring.count=0 THEN dbms_output.put_line('no nonexistent rings');
60 ELSE
61 dbms_output.put_line('nonexistent rings:');
62 FOR i IN tab_ring.FIRST..tab_ring.LAST LOOP
63 v_ring:=tab_ring(i);
```

```
61 dbms_output.put_line(v_ring);
62 END LOOP;
63 END IF;
64
65 --chk for nonexistent faces
66 SELECT DISTINCT face_id BULK COLLECT INTO tab_face FROM FACE WHERE
   face_id NOT IN (SELECT face_id_ref FROM ring);
67 IF tab_face.count=0 THEN dbms_output.put_line('no nonexistent faces');
68 ELSE
69 dbms_output.put_line('nonexistent faces:');
70 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
71 v_face:=tab_face(i);
72 dbms_output.put_line(v_face);
73 END LOOP;
74 END IF;
75
76 --chk for nonexistent shells
77 SELECT DISTINCT shell_id BULK COLLECT INTO tab_shell FROM SHELL WHERE
   shell_id NOT IN (SELECT shell_id_ref_pos FROM FACE) AND shell_id NOT IN
   (SELECT shell_id_ref_neg FROM FACE);
78 IF tab_shell.count=0 THEN dbms_output.put_line('no nonexistent shells');
79 ELSE
80 dbms_output.put_line('nonexistent shells:');
81 FOR i IN tab_shell.FIRST..tab_shell.LAST LOOP
82 v_shell:=tab_shell(i);
83 dbms_output.put_line(v_shell);
84 END LOOP;
85 END IF;
86
87 --check for rings existing of less than 3 edges
88 SELECT ring_id_ref, count(DISTINCT edge_id_ref) BULK COLLECT INTO
   tab_ring_edge FROM ring2edge GROUP BY ring_id_ref HAVING COUNT(*) <3;
89 IF tab_ring_edge.count=0 THEN dbms_output.put_line('no rings with less
   than 3 edges');
90 ELSE
91 dbms_output.put_line('rings with less than 3 edges:');
92 FOR i IN tab_ring_edge.FIRST..tab_ring_edge.LAST LOOP
93 v_ring:=tab_ring_edge(i).ring;
94 dbms_output.put_line(v_ring);
95 END LOOP;
96 END IF;
97
98 --check for faces with rings (inner and outer rings) referring to the
   same edge
99 SELECT r.face_id_ref, r2e.edge_id_ref BULK COLLECT INTO tab_face_edge
   FROM ring2edge r2e,ring r WHERE r2e.ring_id_ref=r.ring_id GROUP BY
   r.face_id_ref, r2e.edge_id_ref HAVING COUNT(*)>1;
100 IF tab_face_edge.count=0 THEN dbms_output.put_line('no faces with inner
   and outer rings referring to the same edge');
101 ELSE
102 dbms_output.put_line('faces with rings referring to the same edges');
103 FOR i IN tab_face_edge.FIRST..tab_face_edge.LAST LOOP
104 v_face:=tab_face_edge(i).face;
105 v_edge:=tab_face_edge(i).edge;
106 dbms_output.put_line(v_face||','||v_edge);
107 END LOOP;
108 END IF;
109
110 --check for more than 1 node in a face which is used more than one time
111 SELECT face_id BULK COLLECT INTO tab_face FROM FACE;
112 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
113 v_face:=tab_face(i);
114 select node BULK COLLECT INTO tab_node from (select startnode as node
```

```
from edge where edge_id in (select edge from edge2shell where face=v_face)
115 union all
116 select endnode as node from edge where edge_id in (select edge from
edge2shell where face=1)) group by node having count(*)>2;
117 IF tab_node.count>1 THEN dbms_output.put_line('more than one node is used
two times in a face: '||v_face); END IF;
118 END LOOP;
119
120 --check for shells existing of less than 4 faces
121 SELECT shell BULK COLLECT INTO tab_shell FROM (SELECT shell_id_ref_pos AS
shell, count(*) AS cnt FROM FACE GROUP BY shell_id_ref_pos
122 UNION ALL
123 SELECT shell_id_ref_neg AS shell, count(*) AS cnt FROM FACE GROUP BY
shell_id_ref_neg) GROUP BY shell HAVING SUM(cnt)<4;
124 IF tab_shell.count=0 THEN dbms_output.put_line('no shells with less than
4 faces');
125 ELSE
126 dbms_output.put_line('shells with less than 4 faces:');
127 FOR i IN tab_shell.FIRST..tab_shell.LAST LOOP
128 v_shell:=tab_shell(i);
129 dbms_output.put_line(v_shell);
130 END LOOP;
131 END IF;
132
133 --check for volumes with shells (inner and outer shells) referring to the
same face
134 SELECT s2v.volume_id_ref, f2s.face_id BULK COLLECT INTO tab_volume_face
FROM FACE f2s, SHELL s2v WHERE f2s.shell_id_ref_pos= s2v.shell_id OR
f2s.shell_id_ref_neg= s2v.shell_id GROUP BY f2s.face_id,
s2v.volume_id_ref HAVING COUNT(*)>1;
135 IF tab_volume_face.count=0 THEN dbms_output.put_line('no volumes with
inner and outer shells referring to the same face');
136 ELSE
137 dbms_output.put_line('volumes with shells referring to the same face:');
138 FOR i IN tab_volume_face.FIRST..tab_volume_face.LAST LOOP
139 v_volume:=tab_volume_face(i).volume;
140 v_face:=tab_volume_face(i).face;
141 dbms_output.put_line(v_volume||','||v_face);
142 END LOOP;
143 END IF;
144
145 END;
```

```
1 //VALIDATIONTEST 3) ONE OUTER BOUNDARY//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_face ELEMENT_TYPE;
9 tab_volume ELEMENT_TYPE;
10 v_face NUMBER;
11 v_volume NUMBER;
12
13
14 BEGIN
15
16 --chk for faces with too many (more than 1) outer rings
17 SELECT face_id_ref BULK COLLECT INTO tab_face FROM ring WHERE InOut='O'
   GROUP BY face_id_ref, InOut HAVING COUNT(*)>1;
18 IF tab_face.count=0 THEN dbms_output.put_line('no faces with more than 1
   outer ring');
19 ELSE
20 dbms_output.put_line('faces with more than 1 outer ring:');
21 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
22 v_face:=tab_face(i);
23 dbms_output.put_line(v_face);
24 END LOOP;
25 END IF;
26
27 --chk for faces with only inner rings (no outer ring)
28 SELECT DISTINCT face_id_ref BULK COLLECT INTO tab_face FROM ring WHERE
   InOut='I' AND face_id_ref NOT IN (SELECT face_id_ref FROM ring WHERE
   InOut='O');
29 IF tab_face.count=0 THEN dbms_output.put_line('no faces with only inner
   rings (no outer ring)');
30 ELSE
31 dbms_output.put_line('faces with only inner rings (no outer ring):');
32 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
33 v_face:=tab_face(i);
34 dbms_output.put_line(v_face);
35 END LOOP;
36 END IF;
37
38 --chk for faces with more than one inner rings
39 SELECT count(*) BULK COLLECT INTO tab_face FROM ring WHERE InOut='O'
   group by face_id_ref having count(*)>1;
40 IF tab_face.count=0 THEN dbms_output.put_line('no faces with more than
   one inner ring');
41 ELSE
42 dbms_output.put_line('faces with more than one inner ring:');
43 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
44 v_face:=tab_face(i);
45 dbms_output.put_line(v_face);
46 END LOOP;
47 END IF;
48
49
50 --chk for volumes with too many (more than 1) outer shells
51 SELECT volume_id_ref BULK COLLECT INTO tab_volume FROM SHELL WHERE
   InOut='O' GROUP BY volume_id_ref, InOut HAVING COUNT(*)>1;
52 IF tab_volume.count=0 THEN dbms_output.put_line('no volumes with more
   than 1 outer shell');
53 ELSE
54 dbms_output.put_line('volumes with more than 1 outer shell:');
```

```
55 FOR i IN tab_volume.FIRST..tab_volume.LAST LOOP
56 v_volume:=tab_volume(i);
57 dbms_output.put_line(v_volume);
58 END LOOP;
59 END IF;
60
61 --chk for volumes with only inner shells (no outer shell), except for the
universal volume
62 SELECT DISTINCT volume_id_ref BULK COLLECT INTO tab_volume FROM SHELL
WHERE InOut='I' AND volume_id_ref NOT IN (SELECT volume_id_ref FROM SHELL
WHERE InOut='O') AND volume_id_ref <>0;
63 IF tab_volume.COUNT=0 THEN dbms_output.put_line('no volumes with only
inner shells (no outer shell)');
64 ELSE
65 dbms_output.put_line('volumes with only innner shells (no outer shell):');
66 FOR i IN tab_volume.FIRST..tab_volume.LAST LOOP
67 v_volume:=tab_volume(i);
68 dbms_output.put_line(v_volume);
69 END LOOP;
70 END IF;
71
72 --does the universal volume exist?
73 select volume_id bulk collect into tab_volume from volume where
volume_id=0;
74 IF tab_volume.count=0 THEN dbms_output.put_line('no universal volume
present'); end if;
75 select count(*) bulk collect into tab_volume from SHELL where inout='I'
and volume_id_ref=0 group by volume_id_ref;
76 IF tab_volume.count=0 THEN dbms_output.put_line('the universal volume has
not inner shells');end if;
77 select count(*) bulk collect into tab_volume from SHELL where inout='O'
and volume_id_ref=0 group by volume_id_ref;
78 IF tab_volume.count>0 THEN dbms_output.put_line('the universal volume has
a outer shell');end if;
79 END;
```

```
1 //VALIDATIONTEST 4) CLOSED RING//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_ring ELEMENT_TYPE;
9 v_ring NUMBER;
10 v_ringordinates SDO_ORDINATE_ARRAY;
11
12
13 BEGIN
14
15 --select rings without valid ringordinates
16 SELECT ring_id BULK COLLECT INTO tab_ring FROM ringordinates WHERE
ringordinates IS NULL;
17 IF tab_ring.count=0 THEN dbms_output.put_line('all rings are closed');
18 ELSE
19 FOR i IN tab_ring.FIRST..tab_ring.LAST LOOP
20 v_ring:=tab_ring(i);
21 dbms_output.put_line('ring without a closed boundary: '||v_ring);
22 SELECT getRINGORDINATES(ring_id) INTO v_ringordinates FROM ring WHERE
ring_id=v_ring;
23 END LOOP;
24 END IF;
25
26 END;
```

```
1 //VALIDATIONTEST 4) CLOSED SHELL//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_edge element_type;
9 tab_shell element_type;
10 v_shell NUMBER;
11 v_neg1 NUMBER;
12 v_neg2 NUMBER;
13 v_pos1 NUMBER;
14 v_pos2 NUMBER;
15 v_edge NUMBER;
16
17 BEGIN
18
19 --select all shells
20 SELECT shell_id BULK COLLECT INTO tab_shell FROM SHELL;
21 IF tab_shell.COUNT=0 THEN dbms_output.put_line('no shells in structure');
22 ELSE FOR i IN tab_shell.FIRST..tab_shell.LAST LOOP
23 v_shell:=tab_shell(i);
24
25 --select all edges from a shell
26 SELECT DISTINCT edge BULK COLLECT INTO tab_edge FROM node2shell WHERE
shell=v_shell;
27
28 --check per collected edge for an even distribution of negative and
positive references
29 IF tab_edge.count=0 THEN dbms_output.put_line('no edges in shell
'||v_shell);
30 ELSE FOR i IN tab_edge.FIRST..tab_edge.LAST LOOP
31 v_edge:=tab_edge(i);
32
33 --select pos. oriented edge in neg. oriented face
34 SELECT count(*) INTO v_neg1 FROM ring2edge WHERE orientation='+' AND
ring_id_ref IN (SELECT ring_id FROM ring WHERE face_id_ref IN (SELECT
face_id FROM FACE WHERE shell_id_ref_neg=v_shell)) AND edge_id_ref=v_edge;
35 --dbms_output.put_line ('shell: '||v_shell||' edge: '||v_edge||' v_neg1:
'||v_neg1);
36
37 --select pos. oriented edge in pos. oriented face
38 SELECT count(*) INTO v_pos1 FROM ring2edge WHERE orientation='+' AND
ring_id_ref IN (SELECT ring_id FROM ring WHERE face_id_ref IN (SELECT
face_id FROM FACE WHERE shell_id_ref_pos=v_shell)) AND edge_id_ref=v_edge;
39 --dbms_output.put_line ('shell: '||v_shell||' edge: '||v_edge||' v_pos1:
'||v_pos1);
40
41 --select neg. oriented edge in pos. oriented face
42 SELECT count(*) INTO v_neg2 FROM ring2edge WHERE orientation='-' AND
ring_id_ref IN (SELECT ring_id FROM ring WHERE face_id_ref IN (SELECT
face_id FROM FACE WHERE shell_id_ref_pos=v_shell)) AND edge_id_ref=v_edge;
43 --dbms_output.put_line ('shell: '||v_shell||' edge: '||v_edge||' v_neg2:
'||v_neg2);
44
45 --select neg. oriented edge in neg. oriented face
46 SELECT count(*) INTO v_pos2 FROM ring2edge WHERE orientation='-' AND
ring_id_ref IN (SELECT ring_id FROM ring WHERE face_id_ref IN (SELECT
face_id FROM FACE WHERE shell_id_ref_neg=v_shell)) AND edge_id_ref=v_edge;
47 --dbms_output.put_line ('shell: '||v_shell||' edge: '||v_edge||' v_pos2:
'||v_pos2);
48
```

```
49 --conclusion
50 IF (v_neg1+v_neg2)<>(v_pos1+v_pos2) THEN dbms_output.put_line('no even
distribution of edges in shell '||v_shell||', at edge '||v_edge); END IF;
51 END LOOP;
52 END IF;
53 END LOOP;
54 END IF;
55
56 END;
```

```
1 //VALIDATIONTEST 5) A PROPER FACE ORIENTATION//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_rec IS RECORD (face NUMBER, Iring NUMBER);
8 TYPE element_type IS TABLE OF element_rec;
9 tab_Irings element_type;
10 v_normalInner normal_array;
11 v_normalOuter normal_array;
12 v_face NUMBER;
13 v_Iring NUMBER;
14 v_Oring NUMBER;
15 v_resultX NUMBER;
16 v_resultY NUMBER;
17 v_resultZ NUMBER;
18
19 BEGIN
20
21 --select all inner rings
22 SELECT face_id_ref,ring_id BULK COLLECT INTO tab_Irings FROM ring WHERE
    InOut='I';
23
24 --compare normal of inner ring with normal of accompanying outer ring
25 IF tab_Irings.count=0 THEN dbms_output.put_line('no inner rings');
26 ELSE
27 dbms_output.put_line('inner rings in the following faces are not oriented
    properly (or not flat):');
28 FOR i IN tab_Irings.FIRST..tab_Irings.LAST LOOP
29 v_face:=tab_Irings(i).face;
30 v_Iring:=tab_Irings(i).Iring;
31 SELECT ring_id INTO v_Oring FROM ring WHERE face_id_ref=v_face AND
    InOut='O';
32 v_normalInner:=getNORMAL(v_Iring);
33 v_normalOuter:=getNORMAL(v_Oring);
34 v_resultX:=v_normalInner(1)+v_normalOuter(1);
35 v_resultX:=v_normalInner(2)+v_normalOuter(2);
36 v_resultX:=v_normalInner(3)+v_normalOuter(3);
37 IF v_resultX>0.01 OR v_resultY>0.01 OR v_resultZ>0.01 THEN
    dbms_output.put_line(v_face);END IF;
38 END LOOP;
39 END IF;
40
41 END;
```

```
1 //VALIDATIONTEST 5) A PROPER VOLUME ORIENTATION//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_Ishell ELEMENT_TYPE;
9 tab_face ELEMENT_TYPE;
10 v_Ishell NUMBER;
11 v_face NUMBER;
12 v_Oring NUMBER;
13 v_minZ NUMBER;
14 v_shellNEG NUMBER;
15 v_shellPOS NUMBER;
16 v_normal normal_array;
17
18 BEGIN
19
20 --select all inner shells
21 SELECT shell_id BULK COLLECT INTO tab_Ishell FROM SHELL WHERE InOut='I';
22 FOR i IN tab_Ishell.FIRST..tab_Ishell.LAST LOOP
23 v_Ishell:=tab_Ishell(i);
24 --dbms_output.put_line('inner shell: '||v_Ishell);
25
26 SELECT MIN(z) INTO v_minZ FROM node WHERE node_id IN (SELECT node FROM
node2shell WHERE shell=v_Ishell);
27 SELECT DISTINCT face BULK COLLECT INTO tab_face FROM node2shell,node
WHERE shell=v_Ishell AND node.z=v_minZ AND node.node_id= node2shell.node;
28 --dbms_output.put_line(tab_face.count);
29 FOR i IN tab_face.FIRST..tab_face.LAST LOOP
30 v_face:=tab_face(i);
31 --dbms_output.put_line('face '||v_face);
32 SELECT ring_id INTO v_Oring FROM ring WHERE face_id_ref=v_face AND
InOut='O';
33 --dbms_output.put_line ('outer ring '||v_Oring);
34 v_normal:=getNORMAL(v_Oring);
35 EXIT WHEN v_normal(3)<>0;
36 END LOOP;
37 --dbms_output.put_line('face '||v_face);
38 if v_normal(3)=0 THEN dbms_output.put_line('normalZ '||v_normal(3)); END
if;
39 SELECT shell_id_ref_pos,shell_id_ref_neg INTO v_shellPOS, v_shellNEG FROM
FACE WHERE face_id=v_face;
40 IF v_shellPOS=v_Ishell THEN IF v_normal(3)>=0 THEN
dbms_output.put_line('1.not oriented properly'); END IF;
41 ELSIF v_shellNEG=v_Ishell THEN IF v_normal(3)>=0 THEN
dbms_output.put_line('2.not oriented properly'); END IF;
42 END IF;
43
44 END LOOP;
45
46
47 END;
```

```
1 //VALIDATIONTEST 6) PLANAR FACES//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 v_tolerance NUMBER:=0.005;
8 TYPE element_type IS TABLE OF NUMBER;
9 tab_face element_type;
10 tab_Iring element_type;
11 v_face NUMBER;
12 v_Oring NUMBER;
13 v_Iring NUMBER;
14 x1 NUMBER;
15 y1 NUMBER;
16 z1 NUMBER;
17 x2 NUMBER;
18 y2 NUMBER;
19 z2 NUMBER;
20 x3 NUMBER;
21 y3 NUMBER;
22 z3 NUMBER;
23 x4 NUMBER;
24 y4 NUMBER;
25 z4 NUMBER;
26 a NUMBER:=4;
27 b NUMBER:=5;
28 c NUMBER:=6;
29 d NUMBER:=7;
30 e NUMBER:=8;
31 f NUMBER:=9;
32 vec1x NUMBER;
33 vec1y NUMBER;
34 vec1z NUMBER;
35 vec2x NUMBER;
36 vec2y NUMBER;
37 vec2z NUMBER;
38 normalX NUMBER;
39 normalY NUMBER;
40 normalZ NUMBER;
41 normalXouter NUMBER;
42 normalYouter NUMBER;
43 normalZouter NUMBER;
44 centroidX NUMBER;
45 centroidY NUMBER;
46 centroidZ NUMBER;
47 v_number NUMBER;
48 v_count NUMBER;
49 v_ordinate SDO_ORDINATE_ARRAY;
50 v_Nfactor NUMBER;
51 v_NfactorOuter NUMBER;
52 v_d NUMBER;
53 vx NUMBER;
54 vx_totaal NUMBER;
55 vy_totaal NUMBER;
56 vz_totaal NUMBER;
57 vy NUMBER;
58 vz NUMBER;
59 v_dOuter NUMBER;
60 Var_A NUMBER;
61 distance NUMBER;
62
63 BEGIN
```

```
64
65 --getting started
66 SELECT face_id BULK COLLECT INTO tab_face FROM FACE;
67 IF tab_face.count=0 THEN dbms_output.put_line('no faces');
68 ELSE FOR i IN tab_face.FIRST..tab_face.LAST LOOP
69 v_face:=tab_face(i);
70 dbms_output.put_line ('face: '||v_face);
71 SELECT ring_id INTO v_Oring FROM ring WHERE face_id_ref=v_face AND
    InOut='O';
72
73 --get centroid of outer ring
74 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
    ring_id=v_Oring;
75 v_count:=v_ordinate.count;
76 v_count:=(v_count-3)/3;
77 --dbms_output.put_line('v_count '||v_count);
78 v_number:=1;
79 vx_totaal:=0;
80 vy_totaal:=0;
81 vz_totaal:=0;
82 FOR v_counter IN 1..v_count LOOP
83 --dbms_output.put_line('v_number '||v_number);
84 vx:=v_ordinate(v_number);
85 --dbms_output.put_line('vx '||vx);
86 vx_totaal:=(vx_totaal+vx);
87 --dbms_output.put_line('vx-totaal '||vx_totaal);
88 v_number:=v_number+1;
89 vy:=v_ordinate(v_number);
90 vy_totaal:=vy_totaal+vy;
91 v_number:=v_number+1;
92 vz:=v_ordinate(v_number);
93 vz_totaal:=vz+vz_totaal;
94 v_number:=v_number+1;
95 END LOOP;
96 CentroidX:=vx_totaal/v_count;
97 CentroidY:=vy_totaal/v_count;
98 CentroidZ:=vz_totaal/v_count;
99 --dbms_output.put_line('centroidX: '||CentroidX);
100 --dbms_output.put_line('centroidY: '||CentroidY);
101 --dbms_output.put_line('centroidZ: '||CentroidZ);
102
103 --select x,y,z values of four nodes (representing 2 edges)
104 x1:= v_ordinate(1);
105 y1:= v_ordinate(2);
106 z1:= v_ordinate(3);
107 x2:= v_ordinate(4);
108 y2:= v_ordinate(5);
109 z2:= v_ordinate(6);
110 x3:= v_ordinate(a);
111 y3:= v_ordinate(b);
112 z3:= v_ordinate(c);
113 x4:= v_ordinate(d);
114 y4:= v_ordinate(e);
115 z4:= v_ordinate(f);
116
117 --calculate the normal of the ring
118 vec1x:=x2-x1;
119 vec1y:=y2-y1;
120 vec1z:=z2-z1;
121 vec2x:=x4-x3;
122 vec2y:=y4-y3;
123 vec2z:=z4-z3;
124 normalX:= (vec1y * vec2z) - (vec1z * vec2y);
```

```
125 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
126 normalZ:= (vec1x * vec2y) - (vec1y * vec2x);
127
128 --in case the edges are parallel
129 IF normalX=0 AND normalY=0 AND normalZ=0 THEN
130 --select x,y,z values of two nodes representing another second edge
131 LOOP
132 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
ring_id=v_Oring;
133 dbms_output.put_line(v_Oring);
134 a:=a+1;
135 b:=b+1;
136 c:=c+1;
137 d:=d+1;
138 e:=e+1;
139 f:=f+1;
140 x3:= v_ordinate(a);
141 y3:= v_ordinate(b);
142 z3:= v_ordinate(c);
143 x4:= v_ordinate(d);
144 y4:= v_ordinate(e);
145 z4:= v_ordinate(f);
146 --calculate the normal of the ring
147 vec1x:=x2-x1;
148 vec1y:=y2-y1;
149 vec1z:=z2-z1;
150 vec2x:=x4-x3;
151 vec2y:=y4-y3;
152 vec2z:=z4-z3;
153 normalX:= (vec1y * vec2z) - (vec1z * vec2y);
154 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
155 normalZ:= (vec1x * vec2y) - (vec1y * vec2x);
156 --dbms_output.put_line(normalX||'|'||normalY||'|'||normalZ);
157 --dbms_output.put_line(normalX+normalY+normalZ);
158 Exit WHEN (normalX+normalY+normalZ)<>0;
159 END LOOP;
160 END IF;
161
162 --calculate distance to centroid
163 v_Nfactor:= sqrt((normalX*normalX)+(normalY*normalY)+(normalZ*normalZ));
164 --dbms_output.put_line('1) '|v_Nfactor);
165 v_d:=(normalX*-(x1)+(normalY*-(y1)+(normalZ*-(z1));
166 --dbms_output.put_line('2) '|v_d);
167 Var_A:=normalX*CentroidX+normalY*CentroidY+normalZ*CentroidZ+v_d;
168 --dbms_output.put_line('3) '|var_A);
169 distance:=Var_A/v_Nfactor;
170 --dbms_output.put_line('4) '|distance);
171 IF distance between -(v_tolerance) AND v_tolerance THEN
dbms_output.enable; dbms_output.put_line('distance centroid to outer ring
within tolerance value IS flat');
172 ELSE dbms_output.enable; dbms_output.put_line('distance centroid to outer
ring greater than tolerance value IS NOT flat');
173 END IF;
174 --IF distance >-(v_tolerance) THEN dbms_output.put_line('distance
centroid to outer ring of face('||v_face||')': '|distance); END IF;
175
176 --check for inner rings
177 SELECT ring_id BULK COLLECT INTO tab_Iring FROM ring WHERE
face_id_ref=v_face AND InOut='I';
178 IF tab_Iring.count>0
179 THEN FOR i IN tab_Iring.FIRST..tab_Iring.LAST LOOP
180 v_Iring:=tab_Iring(i);
181 normalXouter:=normalX;
```

```
182 normalYouter:=normalY;
183 normalZouter:=normalZ;
184 v_dOuter:=v_d;
185 v_NfactorOuter:=v_Nfactor;
186
187
188 --chk centroid within inner ring
189 --get centroid of inner ring
190 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
ring_id=v_Iring;
191 v_count:=v_ordinate.count;
192 v_count:=(v_count-3)/3;
193 v_number:=1;
194 vx_totaal:=0;
195 vy_totaal:=0;
196 vz_totaal:=0;
197 FOR v_counter IN 1..v_count LOOP
198 vx:=v_ordinate(v_number);
199 vx_totaal:=(vx_totaal+vx);
200 v_number:=v_number+1;
201 vy:=v_ordinate(v_number);
202 vy_totaal:=(vy_totaal+vy);
203 v_number:=v_number+1;
204 vz:=v_ordinate(v_number);
205 vz_totaal:=(vz_totaal+vz);
206 v_number:=v_number+1;
207 END LOOP;
208 CentroidX:=vx_totaal/v_count;
209 CentroidY:=vy_totaal/v_count;
210 CentroidZ:=vz_totaal/v_count;
211
212 --select x,y,z values of four nodes (representing 2 edges)
213 x1:= v_ordinate(1);
214 y1:= v_ordinate(2);
215 z1:= v_ordinate(3);
216 x2:= v_ordinate(4);
217 y2:= v_ordinate(5);
218 z2:= v_ordinate(6);
219 x3:= v_ordinate(a);
220 y3:= v_ordinate(b);
221 z3:= v_ordinate(c);
222 x4:= v_ordinate(d);
223 y4:= v_ordinate(e);
224 z4:= v_ordinate(f);
225
226 --calculate the normal of the ring
227 vec1x:=x2-x1;
228 vec1y:=y2-y1;
229 vec1z:=z2-z1;
230 vec2x:=x4-x3;
231 vec2y:=y4-y3;
232 vec2z:=z4-z3;
233 normalX:= (vec1y * vec2z) - (vec1z * vec2y);
234 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
235 normalZ:= (vec1x * vec2y) - (vec1y * vec2x);
236
237 --in case the edges are parallel
238 IF normalX=0 AND normalY=0 AND normalZ=0 THEN
239 --select x,y,z values of two nodes representing another second edge
240 LOOP
241 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
ring_id=v_Iring;
242 a:=a+1;
```

```
243 b:=b+1;
244 c:=c+1;
245 d:=d+1;
246 e:=e+1;
247 f:=f+1;
248 x3:= v_ordinate(a);
249 y3:= v_ordinate(b);
250 z3:= v_ordinate(c);
251 x4:= v_ordinate(d);
252 y4:= v_ordinate(e);
253 z4:= v_ordinate(f);
254 --calculate the normal of the ring
255 vec1x:=x2-x1;
256 vec1y:=y2-y1;
257 vec1z:=z2-z1;
258 vec2x:=x4-x3;
259 vec2y:=y4-y3;
260 vec2z:=z4-z3;
261 normalX:= (vec1y * vec2z) - (vec1z * vec2y);
262 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
263 normalZ:= (vec1x * vec2y) - (vec1y * vec2x);
264 CONTINUE WHEN normalX=0 AND normalY=0 AND normalZ=0;
265 END LOOP;
266 END IF;
267
268 --calculate distance to centroid
269 v_Nfactor:= sqrt((normalX*normalX)+(normalY*normalY)+(normalZ*normalZ));
270 v_d:=(normalX*-(x1)+(normalY*-(y1)+(normalZ*-(z1)));
271 Var_A:=normalX*CentroidX+normalY*CentroidY+normalZ*CentroidZ+v_d;
272 distance:=Var_A/v_Nfactor;
273 IF distance >v_tolerance THEN dbms_output.put_line('distance centroid to
inner ring('||v_Iring||') of face('||v_face||'): '||distance); END IF;
274 IF distance <-(v_tolerance) THEN dbms_output.put_line('distance centroid
to inner ring('||v_Iring||') of face('||v_face||'): '||distance); END IF;
275
276 --chk centroid with outer ring
277 --calculate distance of centroid (inner ring) to outer ring
278 Var_A:=normalXouter*CentroidX+normalYouter*CentroidY+normalZouter*CentroidZ
+v_dOuter;
279 distance:=Var_A/v_NfactorOuter;
280
281 IF distance >v_tolerance THEN dbms_output.put_line('distance centroid of
inner ring('||v_Iring||') to outer ring of face('||v_face||'):
'||distance); END IF;
282 IF distance <-(v_tolerance) THEN dbms_output.put_line('distance centroid
of inner ring('||v_Iring||') to outer ring of face('||v_face||'):
'||distance); END IF;
283 END LOOP;
284 END IF;
285 END LOOP;
286 END IF;
287 END;
```

```
1 //VALIDATIONTEST 7) SELF INTERSECTING VOLUMES//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_volume element_type;
9 tab_Ishell element_type;
10 tab_faces element_type;
11 tab_edges element_type;
12 tab_chkedges element_type;
13 tab_nodes1 element_type;
14 tab_nodes2 element_type;
15 tab_nodes3 element_type;
16 tab_check element_type;
17 v_volume NUMBER;
18 v_Ishell NUMBER;
19 v_Oshell NUMBER;
20 v_face NUMBER;
21 v_edge NUMBER;
22 v_startnode NUMBER;
23 v_endnode NUMBER;
24 v_test NUMBER;
25 vx NUMBER;
26 vy NUMBER;
27 vz NUMBER;
28 nr NUMBER;
29 pointtype SDO_POINT_TYPE;
30 v_ordinates SDO_ORDINATE_ARRAY;
31 v_point SDO_GEOMETRY;
32 v_shellgeom SDO_GEOMETRY;
33 v_fgeom SDO_GEOMETRY;
34 sql_truncate VARCHAR(100):='truncate table chk_interaction';
35
36
37 BEGIN
38
39
40 --getting started
41 execute immediate sql_truncate;
42 SELECT DISTINCT volume_id_ref BULK COLLECT INTO tab_volume FROM SHELL;
43 IF tab_volume.count=0 THEN dbms_output.put_line('no volumes in structure');
44 ELSE
45 FOR i IN tab_volume.FIRST..tab_volume.LAST LOOP
46 v_volume:=tab_volume(i);
47 dbms_output.put_line('volume: '||v_volume);
48
49 --select edges per volume
50 SELECT DISTINCT edge BULK COLLECT INTO tab_edges FROM node2shell WHERE
shell IN (SELECT shell_id FROM SHELL WHERE volume_id_ref=v_volume);
51
52 --insert all edges in chk_interaction table and add geometry
53 nr:=tab_edges.count;
54 FORALL i IN 1..nr
55 INSERT INTO chk_interaction (id) VALUES (tab_edges(i));
56 UPDATE chk_interaction SET geom= getLINE(id);
57
58 --select all faces per volume
59 SELECT DISTINCT face_id BULK COLLECT INTO tab_faces FROM FACE WHERE
shell_id_ref_pos IN (SELECT shell_id FROM SHELL WHERE
volume_id_ref=v_volume) OR shell_id_ref_neg IN (SELECT shell_id FROM
SHELL WHERE volume_id_ref=v_volume);
```

```
60 IF tab_faces.count=0 THEN dbms_output.put_line('no faces selected');
61
62 --chk interaction edges per face (where edge is not part of that face)
63 ELSE FOR i IN tab_faces.FIRST..tab_faces.LAST LOOP
64 v_face:=tab_faces(i);
65 --dbms_output.put_line('face: '||v_face);
66 v_fgeom:=getPOLYGON(v_face);
67 SELECT id BULK COLLECT INTO tab_chkedges FROM chk_interaction WHERE
      (SDO_ANYINTERACT(geom,v_fgeom)='TRUE') AND (id NOT IN (SELECT
      r2e.edge_id_ref FROM ring2edge r2e, ring r WHERE
      r2e.ring_id_ref=r.ring_id AND face_id_ref=v_face));
68
69 --chk edges which interact with the specified face for sharing nodes
70 IF tab_chkedges.count>0 THEN --dbms_output.put_line('closer check needed
      for face '||v_face);
71 FOR i IN tab_chkedges.FIRST..tab_chkedges.LAST LOOP
72 v_edge:=tab_chkedges(i);
73 --dbms_output.put_line(v_edge);
74 SELECT node BULK COLLECT INTO tab_nodes1 FROM (SELECT startnode AS node
      FROM edge WHERE edge_id=v_edge
75 UNION ALL SELECT endnode AS node FROM edge WHERE edge_id=v_edge);
76 SELECT DISTINCT node BULK COLLECT INTO tab_nodes2 FROM (
77 SELECT startnode AS node FROM edge WHERE edge_id IN (SELECT
      r2e.edge_id_ref FROM ring2edge r2e, ring r WHERE
      r2e.ring_id_ref=r.ring_id AND face_id_ref=v_face)
78 UNION ALL
79 SELECT endnode AS node FROM edge WHERE edge_id IN (SELECT r2e.edge_id_ref
      FROM ring2edge r2e, ring r WHERE r2e.ring_id_ref=r.ring_id AND
      face_id_ref=v_face));
80 tab_nodes3:=tab_nodes1 MULTISSET INTERSECT tab_nodes2;
81 IF tab_nodes3.count=0 THEN dbms_output.put_line ('face '||v_face||' has a
      non-valid intersection with another face of the same volume
      ('||v_volume||'));
82 END IF;
83 END LOOP;
84 END IF;
85 END LOOP;
86 END IF;
87
88
89 --closure
90 execute immediate sql_truncate;
91 END LOOP;
92 END IF;
93
94 END;
```

```
1 //VALIDATIONTEST 7) EVERY INNER RING MUST BE INSIDE ITS ACCOMPANYING
  OUTER RING//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_Iring element_type;
9 tab_edges element_type;
10 tab_edgesOUTER element_type;
11 tab_chkedges element_type;
12 tab_check element_type;
13 v_Iring NUMBER;
14 v_Oring NUMBER;
15 v_edge NUMBER;
16 nr NUMBER;
17 vx NUMBER;
18 vy NUMBER;
19 vz NUMBER;
20 v_linegeom SDO_GEOMETRY;
21 pointtype SDO_POINT_TYPE;
22 v_point SDO_GEOMETRY;
23 v_ringgeom SDO_GEOMETRY;
24 v_ordinates SDO_ORDINATE_ARRAY;
25 sql_truncate VARCHAR(100):='truncate table chk_interaction';
26
27
28 BEGIN
29
30 --getting started, select inner rings
31 execute immediate sql_truncate;
32 SELECT DISTINCT ring_id BULK COLLECT INTO tab_Iring FROM ring WHERE
  InOut='I';
33 IF tab_Iring.count=0 THEN dbms_output.put_line('no inner rings in
  structure');
34 ELSE
35 FOR i IN tab_Iring.FIRST..tab_Iring.LAST LOOP
36 v_Iring:=tab_Iring(i);
37 dbms_output.put_line('inner ring: '||v_Iring);
38
39
40 --select all edges of inner ring and insert them in the chk_interaction
  table and add geometry
41 SELECT DISTINCT edge BULK COLLECT INTO tab_edges FROM edge2shell WHERE
  ring=v_Iring;
42 nr:=tab_edges.count;
43 FORALL i IN 1..nr
44 INSERT INTO chk_interaction (id) VALUES (tab_edges(i));
45 UPDATE chk_interaction SET geom= getLINE(id);
46
47 --select accompanying outer ring
48 SELECT ring_id INTO v_Oring FROM ring WHERE InOut='O' AND face_id_ref IN
  (SELECT face_id_ref FROM ring WHERE InOut='I' AND ring_id=v_Iring);
49 SELECT ringordinates INTO v_ordinates FROM ringordinates WHERE
  ring_id=v_Oring;
50 v_ringgeom:=SDO_GEOMETRY(3003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),v_ord
  inates);
51 SELECT id BULK COLLECT INTO tab_check FROM chk_interaction WHERE
  (SDO_ANYINTERACT(geom,v_ringgeom)='TRUE');
52 IF tab_check.count<>nr THEN dbms_output.put_line('inner ring not
  (completely) inside outer ring'); END IF;
53 END LOOP;
```

```
54
55
56 --closure
57 execute immediate sql_truncate;
58 END IF;
59
60
61 END;
```

```
1 //VALIDATIONTEST 7) EVERY INNER SHELL MUST BE INSIDE ITS ACCOMPANYING
  OUTER SHELL//
2
3
4 SET SERVEROUTPUT ON
5
6 DECLARE
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_volume element_type;
9 tab_Ishell element_type;
10 tab_faces element_type;
11 tab_edges element_type;
12 tab_chkedges element_type;
13 tab_nodes1 element_type;
14 tab_nodes2 element_type;
15 tab_nodes3 element_type;
16 tab_check element_type;
17 v_volume NUMBER;
18 v_Ishell NUMBER;
19 v_Oshell NUMBER;
20 v_face NUMBER;
21 v_edge NUMBER;
22 v_startnode NUMBER;
23 v_endnode NUMBER;
24 v_test NUMBER;
25 vx NUMBER;
26 vy NUMBER;
27 vz NUMBER;
28 nr NUMBER;
29 pointtype SDO_POINT_TYPE;
30 v_ordinates SDO_ORDINATE_ARRAY;
31 v_point SDO_GEOMETRY;
32 v_shellgeom SDO_GEOMETRY;
33 v_sgeom SDO_GEOMETRY;
34 sql_truncate VARCHAR(100):='truncate table chk_interaction';
35
36
37 BEGIN
38
39
40 --getting started
41 execute immediate sql_truncate;
42
43
44 --select all inner shells
45 SELECT shell_id BULK COLLECT INTO tab_Ishell FROM SHELL WHERE InOut='I'
  AND volume_id_ref<>0;
46 IF tab_Ishell.COUNT=0 THEN dbms_output.put_line('no inner shells'); ELSE
47 FOR i IN tab_Ishell.FIRST..tab_Ishell.LAST LOOP
48 v_Ishell:=tab_Ishell(i);
49 dbms_output.put_line('inner shell: '||v_Ishell);
50
51 --select all faces of the inner shell and insert them in the
  chk_interaction table and add geometry
52 SELECT DISTINCT face_id BULK COLLECT INTO tab_faces FROM FACE WHERE
  shell_id_ref_pos=v_Ishell OR shell_id_ref_neg = v_Ishell;
53 --dbms_output.put_line(tab_faces.count);
54 IF tab_faces.count=0 THEN dbms_output.put_line('no faces selected');
55 ELSE
56 nr:=tab_faces.count;
57 FOR i IN tab_faces.FIRST..tab_faces.LAST LOOP
58 v_face:=tab_faces(i);
59 --dbms_output.put_line(v_face);
```

```
60 INSERT INTO chk_interaction (id) VALUES (v_face);
61 END LOOP;
62 UPDATE chk_interaction SET geom= getPOLYGON(id);
63
64 --select accompanying outer shell and chk interaction of the faces of the
inner shell with the outer shell
65 SELECT shell_id INTO v_Oshell FROM SHELL WHERE InOut='O' AND
volume_id_ref IN (SELECT volume_id_ref FROM SHELL WHERE InOut='I' AND
shell_id=v_Ishell);
66 v_sgeom:=getSOLID(v_Oshell);
67 --dbms_output.put_line(v_Oshell);
68 SELECT id BULK COLLECT INTO tab_check FROM chk_interaction WHERE
(SDO_ANYINTERACT(geom,v_sgeom)='TRUE');
69 IF tab_check.count<>nr THEN dbms_output.put_line('inner shell not
(completely) inside outer shell');
70 --dbms_output.put_line(tab_check.count||';'||nr);
71 END IF;
72
73
74 --closure
75 execute immediate sql_truncate;
76
77 END IF;
78 END LOOP;
79 END IF;
80
81 END;
```



## Scripts geometry operations

```
1 //OPERATION GETPOINT//
2
3
4 CREATE OR REPLACE FUNCTION getpoint(i_node IN NUMBER)
5 RETURN SDO_GEOMETRY IS
6
7 pointtype sdo_point_type;
8 vx NUMBER;
9 vy NUMBER;
10 vz NUMBER;
11 v_point SDO_GEOMETRY;
12 v_node NUMBER;
13
14 BEGIN
15
16 v_node:=i_node;
17 SELECT x INTO vx FROM node WHERE node_id=v_node;
18 SELECT y INTO vy FROM node WHERE node_id=v_node;
19 SELECT z INTO vz FROM node WHERE node_id=v_node;
20 pointtype:=sdo_point_type(vx,vy,vz);
21 v_point:=SDO_GEOMETRY (3001, NULL, pointtype, NULL,NULL);
22 RETURN v_point;
23
24 END;
```

```
1 //OPERATION GETLINE//
2
3
4 CREATE OR REPLACE FUNCTION getLINE(i_edge IN NUMBER)
5 RETURN SDO_GEOMETRY AS
6
7 v_startnode NUMBER;
8 v_endnode NUMBER;
9 v_edge NUMBER;
10 v_xs NUMBER;
11 v_ys NUMBER;
12 v_zs NUMBER;
13 v_xe NUMBER;
14 v_ye NUMBER;
15 v_ze NUMBER;
16 v_ordinate SDO_ORDINATE_ARRAY;
17 LINE SDO_GEOMETRY;
18
19 BEGIN
20
21 --getting started
22 v_edge:=i_edge;
23
24 --select begin and end node
25 SELECT startnode INTO v_startnode FROM edge WHERE edge_id=v_edge;
26 SELECT endnode INTO v_endnode FROM edge WHERE edge_id=v_edge;
27 SELECT x INTO v_xs FROM node WHERE node_id= v_startnode;
28 SELECT y INTO v_ys FROM node WHERE node_id= v_startnode;
29 SELECT z INTO v_zs FROM node WHERE node_id= v_startnode;
30 SELECT x INTO v_xe FROM node WHERE node_id= v_endnode;
31 SELECT y INTO v_ye FROM node WHERE node_id= v_endnode;
32 SELECT z INTO v_ze FROM node WHERE node_id= v_endnode;
33 v_ordinate:=sdo_ordinate_array(v_xs,v_ys,v_zs,v_xe,v_ye,v_ze);
34
35 --closure
36 IF v_ordinate.count=6
37 THEN
38     LINE:=SDO_GEOMETRY(3002,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,2,1),v_ordinate);
39 ELSE LINE:=NULL; END IF;
40 RETURN LINE;
41 EXCEPTION WHEN NO_DATA_FOUND THEN LINE:=NULL;
42 RETURN LINE;
43 END;
```

```
1 //OPERATION GETPOLYGON//
2
3
4 CREATE OR REPLACE FUNCTION getPOLYGON(i_face IN NUMBER)
5 RETURN SDO_GEOMETRY AS
6
7 TYPE ring_type IS TABLE OF NUMBER;
8 tab_Iring ring_type;
9 v_Oring NUMBER;
10 v_Iring NUMBER;
11 v_x NUMBER;
12 v_y NUMBER;
13 v_z NUMBER;
14 x NUMBER;
15 v_offset NUMBER;
16 v_count NUMBER;
17 v_ordinate SDO_ORDINATE_ARRAY;
18 v_Iordinate SDO_ORDINATE_ARRAY;
19 v_info_array SDO_ELEM_INFO_ARRAY;
20 FACE SDO_GEOMETRY;
21
22 BEGIN
23
24 --select outer ring and inner rings
25 SELECT ring_id INTO v_Oring FROM ring WHERE face_id_ref=i_face AND
    InOut='O';
26 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
    ring_id=v_Oring;
27 SELECT ring_id BULK COLLECT INTO tab_Iring FROM ring WHERE
    face_id_ref=i_face AND InOut='I';
28 v_info_array:=sdo_elem_info_array(1,1003,1);
29
30 --when no innerrings present
31 IF tab_Iring.count=0 THEN
32 FACE:=SDO_GEOMETRY(3003,NULL,NULL,v_info_array, v_ordinate); RETURN FACE;
33
34 --when inner rings are present:
35 ELSE FOR i IN tab_Iring.FIRST..tab_Iring.LAST LOOP
36 v_Iring:=tab_Iring(i);
37 SELECT ringordinates INTO v_Iordinate FROM ringordinates WHERE
    ring_id=v_Iring;
38
39 --adjust array-info
40 v_offset:=v_ordinate.count+1;
41 v_count:=v_info_array.count+1;
42 v_info_array.extend;
43 v_info_array(v_count):=(v_offset);
44 v_count:=v_count+1;
45 v_info_array.extend;
46 v_info_array(v_count):=(2003);
47 v_count:=v_count+1;
48 v_info_array.extend;
49 v_info_array(v_count):=(1);
50
51 --extend ordinate-array
52 x:=v_Iordinate.count;
53 FOR v_counter IN 1..x LOOP
54 v_ordinate.extend;
55 v_ordinate(v_offset):=v_Iordinate(v_counter);
56 v_offset:=v_offset+1;
57 END LOOP;
58 END LOOP;
59
```

```
60 FACE:=SDO_GEOMETRY(3003,NULL,NULL,v_info_array,v_ordinate);
61 RETURN FACE;
62 END IF;
63 EXCEPTION WHEN NO_DATA_FOUND THEN FACE:=NULL;RETURN FACE;
64 RETURN FACE;
65
66 END;
```

```
1 //OPERATION GETSOLID//
2
3
4 CREATE OR REPLACE FUNCTION getSOLID(i_volume IN NUMBER)
5 RETURN SDO_GEOMETRY AS
6
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_Ishell element_type;
9 tab_POSface element_type;
10 tab_NEGface element_type;
11 v_volume NUMBER;
12 v_Oshell NUMBER;
13 v_Ishell NUMBER;
14 v_face NUMBER;
15 v_nrfaces NUMBER;
16 v_offset NUMBER;
17 v_count NUMBER;
18 v_loop NUMBER;
19 x NUMBER;
20 v_innerrings NUMBER;
21 v_facelordinate SDO_ORDINATE_ARRAY;
22 v_face2ordinate SDO_ORDINATE_ARRAY;
23 v_ordinate SDO_ORDINATE_ARRAY;
24 v_Fordinate SDO_ORDINATE_ARRAY;
25 v_info_array SDO_ELEM_INFO_ARRAY;
26 VOLUME SDO_GEOMETRY;
27
28 BEGIN
29
30 --getting started
31 v_volume:=i_volume;
32 v_ordinate:=sdo_ordinate_array();
33 v_info_array:=sdo_elem_info_array(1,1007,1);
34
35 --select shells (outer and inner)
36 SELECT shell_id INTO v_Oshell FROM SHELL WHERE volume_id_ref=v_volume AND
  InOut='O';
37 SELECT shell_id BULK COLLECT INTO tab_Ishell FROM SHELL WHERE
  volume_id_ref=v_volume AND InOut='I';
38
39
40 --get faces of outer shell
41 SELECT face_id BULK COLLECT INTO tab_POSface FROM FACE WHERE
  shell_id_ref_pos=v_Oshell;
42 SELECT face_id BULK COLLECT INTO tab_NEGface FROM FACE WHERE
  shell_id_ref_neg=v_Oshell;
43 v_nrfaces:=tab_POSface.count+tab_NEGface.count;
44 --dbms_output.put_line('nr of faces: '||v_nrfaces);
45
46 --adjust array-info for outer shell
47 v_info_array.extend;
48 v_info_array(4):=1;
49 v_info_array.extend;
50 v_info_array(5):=1006;
51 v_info_array.extend;
52 v_info_array(6):=(v_nrfaces);
53
54 --get pos. oriented faces first (of outer shell)
55 IF tab_POSface.count>0 THEN
56 FOR i IN tab_POSface.FIRST..tab_POSface.LAST LOOP
57 v_face:=tab_POSface(i);
58 --dbms_output.put_line('(pos) face: '||v_face);
59
```

```
60 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
   face_id_ref=v_face;
61 --dbms_output.put_line('count inner rings: '||v_innerrings);
62 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
   face: '||v_face||' (geen solid geconstrueerd)'); VOLUME:=NULL;
63 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
   '||v_face);
64 v_nrfaces:=v_nrfaces+1;
65 --dbms_output.put_line ('number of faces: '||v_nrfaces);
66 v_info_array(6):=(v_nrfaces);
67 deleteINNERRINGS(v_face,v_facelordinate,v_face2ordinate);
68
69 --adjust array-info for facel
70 v_offset:=v_ordinate.count+1;
71 v_count:=v_info_array.count+1;
72 v_info_array.extend;
73 v_info_array(v_count):=(v_offset);
74 v_count:=v_count+1;
75 v_info_array.extend;
76 v_info_array(v_count):=(1003);
77 v_count:=v_count+1;
78 v_info_array.extend;
79 v_info_array(v_count):=(1);
80 --extend ordinate-array for facel
81 x:=v_facelordinate.count;
82 FOR v_counter IN 1..x LOOP
83 v_ordinate.extend;
84 v_ordinate(v_offset):=v_facelordinate(v_counter);
85 v_offset:=v_offset+1;
86 END LOOP;
87
88 --adjust array-info for face2
89 v_offset:=v_ordinate.count+1;
90 v_count:=v_info_array.count+1;
91 v_info_array.extend;
92 v_info_array(v_count):=(v_offset);
93 v_count:=v_count+1;
94 v_info_array.extend;
95 v_info_array(v_count):=(1003);
96 v_count:=v_count+1;
97 v_info_array.extend;
98 v_info_array(v_count):=(1);
99 --extend ordinate-array for face2
100 x:=v_face2ordinate.count;
101 FOR v_counter IN 1..x LOOP
102 v_ordinate.extend;
103 v_ordinate(v_offset):=v_face2ordinate(v_counter);
104 v_offset:=v_offset+1;
105 END LOOP;
106
107 ELSE SELECT ringordinates INTO v_Fordinate FROM ringordinates WHERE
   ring_id IN (SELECT ring_id FROM ring WHERE face_id_ref=v_face AND
   InOut='O');
108 --dbms_output.put_line('geen inner ring');
109
110 --adjust array-info
111 v_offset:=v_ordinate.count+1;
112 v_count:=v_info_array.count+1;
113 v_info_array.extend;
114 v_info_array(v_count):=(v_offset);
115 v_count:=v_count+1;
116 v_info_array.extend;
117 v_info_array(v_count):=(1003);
```

```
118 v_count:=v_count+1;
119 v_info_array.extend;
120 v_info_array(v_count):=(1);
121
122 --extend ordinate-array
123 x:=v_Fordinate.count;
124 FOR v_counter IN 1..x LOOP
125 v_ordinate.extend;
126 v_ordinate(v_offset):=v_Fordinate(v_counter);
127 v_offset:=v_offset+1;
128 END LOOP;
129 END IF;
130 END LOOP;
131 END IF;
132
133
134 --get neg. oriented faces of outer shell
135 IF tab_NEGface.count>0 THEN
136 FOR i IN tab_NEGface.FIRST..tab_NEGface.LAST LOOP
137 v_face:=tab_NEGface(i);
138 --dbms_output.put_line('(neg) face: '||v_face);
139
140 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
face_id_ref=v_face;
141 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
face: '||v_face||' (geen solid geconstrueerd)'); VOLUME:=NULL;
142 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
' ||v_face);
143 v_nrfaces:=v_nrfaces+1;
144 --dbms_output.put_line('number of faces '||v_nrfaces);
145 v_info_array(6):=(v_nrfaces);
146 deleteINNERRINGS(v_face,v_facelordinate,v_face2ordinate);
147
148 --adjust array-info for facel
149 v_offset:=v_ordinate.count+1;
150 v_count:=v_info_array.count+1;
151 v_info_array.extend;
152 v_info_array(v_count):=(v_offset);
153 v_count:=v_count+1;
154 v_info_array.extend;
155 v_info_array(v_count):=(1003);
156 v_count:=v_count+1;
157 v_info_array.extend;
158 v_info_array(v_count):=(1);
159 --extend ordinate-array for facel
160 x:=v_facelordinate.count;
161 v_loop:=(x/3);
162 FOR v_counter IN 1..v_loop LOOP
163 v_count:=x-2;
164 --dbms_output.put_line ('offset: '||v_offset);
165 --dbms_output.put_line ('count: '||v_count);
166 v_ordinate.extend;
167 v_ordinate(v_offset):=v_facelordinate(v_count);
168 v_offset:=v_offset+1;
169 v_count:=x-1;
170 --dbms_output.put_line ('offset: '||v_offset);
171 --dbms_output.put_line ('count: '||v_count);
172 v_ordinate.extend;
173 v_ordinate(v_offset):=v_facelordinate(v_count);
174 v_offset:=v_offset+1;
175 v_count:=x;
176 --dbms_output.put_line ('offset: '||v_offset);
177 --dbms_output.put_line ('count: '||v_count);
```

```
178 v_ordinate.extend;
179 v_ordinate(v_offset):=v_facelordinate(v_count);
180 v_offset:=v_offset+1;
181 x:=x-3;
182 END LOOP;
183
184 --adjust array-info for face2
185 v_offset:=v_ordinate.count+1;
186 v_count:=v_info_array.count+1;
187 v_info_array.extend;
188 v_info_array(v_count):=(v_offset);
189 v_count:=v_count+1;
190 v_info_array.extend;
191 v_info_array(v_count):=(1003);
192 v_count:=v_count+1;
193 v_info_array.extend;
194 v_info_array(v_count):=(1);
195 --extend ordinate-array for face2
196 x:=v_facelordinate.count;
197 v_loop:=(x/3);
198 FOR v_counter IN 1..v_loop LOOP
199 v_count:=x-2;
200 --dbms_output.put_line ('offset: '||v_offset);
201 --dbms_output.put_line ('count: '||v_count);
202 v_ordinate.extend;
203 v_ordinate(v_offset):=v_face2ordinate(v_count);
204 v_offset:=v_offset+1;
205 v_count:=x-1;
206 --dbms_output.put_line ('offset: '||v_offset);
207 --dbms_output.put_line ('count: '||v_count);
208 v_ordinate.extend;
209 v_ordinate(v_offset):=v_face2ordinate(v_count);
210 v_offset:=v_offset+1;
211 v_count:=x;
212 --dbms_output.put_line ('offset: '||v_offset);
213 --dbms_output.put_line ('count: '||v_count);
214 v_ordinate.extend;
215 v_ordinate(v_offset):=v_face2ordinate(v_count);
216 v_offset:=v_offset+1;
217 x:=x-3;
218 END LOOP;
219
220
221 ELSE SELECT ringordinates INTO v_Fordinate FROM ringordinates WHERE
ring_id IN (SELECT ring_id FROM ring WHERE face_id_ref=v_face AND
InOut='O');
222 --dbms_output.put_line('geen inner rings');
223
224 --adjust array-info
225 v_offset:=v_ordinate.count+1;
226 v_count:=v_info_array.count+1;
227 v_info_array.extend;
228 v_info_array(v_count):=(v_offset);
229 v_count:=v_count+1;
230 v_info_array.extend;
231 v_info_array(v_count):=(1003);
232 v_count:=v_count+1;
233 v_info_array.extend;
234 v_info_array(v_count):=(1);
235 --dbms_output.put_line ('face: '||v_face);
236
237 --extend ordinate-array
238 x:=v_Fordinate.count;
```

```
239 --dbms_output.put_line ('number of ordinates: '||x);
240 v_loop:=(x/3);
241 FOR v_counter IN 1..v_loop LOOP
242 v_count:=x-2;
243 --dbms_output.put_line ('offset: '||v_offset);
244 --dbms_output.put_line ('count: '||v_count);
245 v_ordinate.extend;
246 v_ordinate(v_offset):=v_Fordinate(v_count);
247 v_offset:=v_offset+1;
248 v_count:=x-1;
249 --dbms_output.put_line ('offset: '||v_offset);
250 --dbms_output.put_line ('count: '||v_count);
251 v_ordinate.extend;
252 v_ordinate(v_offset):=v_Fordinate(v_count);
253 v_offset:=v_offset+1;
254 v_count:=x;
255 --dbms_output.put_line ('offset: '||v_offset);
256 --dbms_output.put_line ('count: '||v_count);
257 v_ordinate.extend;
258 v_ordinate(v_offset):=v_Fordinate(v_count);
259 v_offset:=v_offset+1;
260 x:=x-3;
261 END LOOP;
262 END IF;
263 END LOOP;
264 END IF;
265
266
267 --check for inner shells
268 IF tab_Ishell.count>0 THEN
269 FOR i IN tab_Ishell.FIRST..tab_Ishell.LAST LOOP
270 v_Ishell:=tab_Ishell(i);
271 --dbms_output.put_line ('inner shell: '||v_Ishell);
272 --get faces of inner shell
273 SELECT face_id BULK COLLECT INTO tab_POSface FROM FACE WHERE
    shell_id_ref_pos=v_Ishell;
274 SELECT face_id BULK COLLECT INTO tab_NEGface FROM FACE WHERE
    shell_id_ref_neg=v_Ishell;
275 v_nrfaces:=tab_POSface.count+tab_NEGface.count;
276 --dbms_output.put_line ('number of faces: '||v_nrfaces);
277 --dbms_output.put_line ('pos oriented faces: '||tab_POSface.count);
278 --dbms_output.put_line ('neg oriented faces: '||tab_NEGface.count);
279
280
281 --adjust array-info for inner shell
282 v_offset:=v_ordinate.count+1;
283 --dbms_output.put_line ('offset: '||v_offset);
284 v_count:=v_info_array.count+1;
285 --dbms_output.put_line ('count: '||v_count);
286 v_info_array.extend;
287 v_info_array(v_count):=(v_offset);
288 v_count:=v_count+1;
289 v_info_array.extend;
290 v_info_array(v_count):=(2006);
291 v_count:=v_count+1;
292 v_info_array.extend;
293 v_info_array(v_count):=(v_nrfaces);
294
295 --get pos. oriented faces first (of inner shell)
296 IF tab_POSface.count>0 THEN
297 FOR i IN tab_POSface.FIRST..tab_POSface.LAST LOOP
298 v_face:=tab_POSface(i);
299 SELECT ringordinates INTO v_Fordinate FROM ringordinates WHERE ring_id IN
```

```
(SELECT ring_id FROM ring WHERE face_id_ref=v_face AND InOut='0');
300
301 --adjust array-info
302 v_offset:=v_ordinate.count+1;
303 v_count:=v_info_array.count+1;
304 v_info_array.extend;
305 v_info_array(v_count):=(v_offset);
306 v_count:=v_count+1;
307 v_info_array.extend;
308 v_info_array(v_count):=(2003);
309 v_count:=v_count+1;
310 v_info_array.extend;
311 v_info_array(v_count):=(1);
312
313 --extend ordinate-array
314 x:=v_Fordinate.count;
315 FOR v_counter IN 1..x LOOP
316 v_ordinate.extend;
317 v_ordinate(v_offset):=v_Fordinate(v_counter);
318 v_offset:=v_offset+1;
319 END LOOP;
320 END LOOP;
321 END IF;
322
323 --get neg. oriented faces of inner shell
324 IF tab_NEGface.count>0 THEN
325 FOR i IN tab_NEGface.FIRST..tab_NEGface.LAST LOOP
326 v_face:=tab_NEGface(i);
327 SELECT ringordinates INTO v_Fordinate FROM ringordinates WHERE ring_id IN
    (SELECT ring_id FROM ring WHERE face_id_ref=v_face AND InOut='0');
328
329 --adjust array-info
330 v_offset:=v_ordinate.count+1;
331 v_count:=v_info_array.count+1;
332 v_info_array.extend;
333 v_info_array(v_count):=(v_offset);
334 v_count:=v_count+1;
335 v_info_array.extend;
336 v_info_array(v_count):=(2003);
337 v_count:=v_count+1;
338 v_info_array.extend;
339 v_info_array(v_count):=(1);
340
341 --extend ordinate-array
342 x:=v_Fordinate.count;
343 --dbms_output.put_line ('number of ordinates: '||x);
344 v_loop:=(x/3);
345 FOR v_counter IN 1..v_loop LOOP
346 v_count:=x-2;
347 --dbms_output.put_line ('offset: '||v_offset);
348 --dbms_output.put_line ('count: '||v_count);
349 v_ordinate.extend;
350 v_ordinate(v_offset):=v_Fordinate(v_count);
351 v_offset:=v_offset+1;
352 v_count:=x-1;
353 --dbms_output.put_line ('offset: '||v_offset);
354 --dbms_output.put_line ('count: '||v_count);
355 v_ordinate.extend;
356 v_ordinate(v_offset):=v_Fordinate(v_count);
357 v_offset:=v_offset+1;
358 v_count:=x;
359 --dbms_output.put_line ('offset: '||v_offset);
360 --dbms_output.put_line ('count: '||v_count);
```

```
361 v_ordinate.extend;
362 v_ordinate(v_offset):=v_Fordinate(v_count);
363 v_offset:=v_offset+1;
364 x:=x-3;
365 END LOOP;
366 END LOOP;
367 END IF;
368 END LOOP;
369 END IF;
370
371 --result
372 --dbms_output.put_line ('info-array: '||v_info_array(1));
373 --dbms_output.put_line ('ordinate-array: '||v_ordinate(1));
374 VOLUME:=SDO_GEOMETRY(3008,NULL,NULL,v_info_array,v_ordinate);
375 RETURN VOLUME;
376 RETURN VOLUME;
377 EXCEPTION WHEN NO_DATA_FOUND THEN dbms_output.put_line('no data
found');VOLUME:=NULL;
378 RETURN VOLUME;
379
380 END;
```

## Scripts assistant functions

```
1 //FUNCTION RINGORDINATES//
2
3
4 CREATE OR REPLACE FUNCTION getRINGordinates(i_ring IN NUMBER) RETURN
  SDO_ORDINATE_ARRAY AS
5 TYPE element_type IS TABLE OF NUMBER;
6 tab_node element_type;
7 v_ring NUMBER;
8 v_edge NUMBER;
9 v_node NUMBER;
10 v_orientation VARCHAR2(1);
11 v_startnode NUMBER;
12 v_chkedge NUMBER;
13 v_chknode NUMBER;
14 v_counter NUMBER;
15 v_x NUMBER;
16 v_y NUMBER;
17 v_z NUMBER;
18 v_nredges NUMBER;
19 v_nrordinates NUMBER;
20 v_ordinate SDO_ORDINATE_ARRAY;
21 RING SDO_ORDINATE_ARRAY;
22
23 BEGIN
24
25 --getting started
26 v_ring:=i_ring;
27
28 --set up check: count edges to check with number of ordinates
29 SELECT count(edge_id_ref) INTO v_nredges FROM ring2edge WHERE
  ring_id_ref=v_ring;
30 v_nrordinates:=(v_nredges*3)+3;
31
32 --select a startedge and startnode
33 SELECT min(edge_id_ref) INTO v_edge FROM ring2edge WHERE
  ring_id_ref=v_ring;
34 --dbms_output.put_line('startedge: '||v_edge);
35 SELECT orientation INTO v_orientation FROM ring2edge WHERE
  ring_id_ref=v_ring AND edge_id_ref=v_edge;
36 --dbms_output.put_line('orientation startedge: '||v_orientation);
37 IF v_orientation='- '
38 THEN SELECT endnode, startnode INTO v_startnode, v_node FROM edge WHERE
  edge_id=v_edge;
39 ELSE SELECT startnode, endnode INTO v_startnode, v_node FROM edge WHERE
  edge_id=v_edge;
40 END IF;
41 --dbms_output.put_line('startnode: '||v_startnode);
42
43 --insert startnode in ordinate-array
44 SELECT x INTO v_x FROM node WHERE node_id= v_startnode;
45 SELECT y INTO v_y FROM node WHERE node_id= v_startnode;
46 SELECT z INTO v_z FROM node WHERE node_id= v_startnode;
47 v_ordinate:=sdo_ordinate_array(v_x,v_y,v_z);
48 v_counter:=3;
49
50 --insert the second node in ordinate-array
51 SELECT x INTO v_x FROM node WHERE node_id= v_node;
52 SELECT y INTO v_y FROM node WHERE node_id= v_node;
53 SELECT z INTO v_z FROM node WHERE node_id= v_node;
54 v_counter:=v_counter+1;
55 v_ordinate.extend;
56 v_ordinate(v_counter):=(v_x);
57 v_counter:=v_counter+1;
```

```
58 v_ordinate.extend;
59 v_ordinate(v_counter):=(v_y);
60 v_counter:=v_counter+1;
61 v_ordinate.extend;
62 v_ordinate(v_counter):=(v_z);
63
64 --continue collecting and inserting nodes (collect next node and check
orientation edge)
65 LOOP
66
67 --search for negative oriented edge
68 BEGIN
69 SELECT startnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
endnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_ring) AND edge_id <>v_edge;
70 v_node:=v_chknode;
71 v_edge:=v_chkedge;
72 --chk (neg.) orientation
73 SELECT orientation INTO v_orientation FROM ring2edge WHERE
edge_id_ref=v_chkedge AND ring_id_ref=v_ring;
74 IF v_orientation='+' THEN dbms_output.put_line('1.edge not oriented
properly: '||v_edge); END IF;
75 EXCEPTION WHEN NO_DATA_FOUND THEN v_chknode:=0;
76 END;
77
78 --search for positive oriented edge (when no negative oriented edge is
found)
79 IF v_chknode=0 THEN
80 SELECT endnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
startnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_ring) AND edge_id <>v_edge;
81 v_node:=v_chknode;
82 v_edge:=v_chkedge;
83 --chk (pos.) orientation
84 SELECT orientation INTO v_orientation FROM ring2edge WHERE
edge_id_ref=v_chkedge AND ring_id_ref=v_ring;
85 IF v_orientation='-' THEN dbms_output.put_line('2.edge not oriented
properly: '||v_edge); END IF;
86 END IF;
87
88 --insert v_node in ordinate-array
89 SELECT x INTO v_x FROM node WHERE node_id= v_node;
90 SELECT y INTO v_y FROM node WHERE node_id= v_node;
91 SELECT z INTO v_z FROM node WHERE node_id= v_node;
92 v_counter:=v_counter+1;
93 v_ordinate.extend;
94 v_ordinate(v_counter):=(v_x);
95 v_counter:=v_counter+1;
96 v_ordinate.extend;
97 v_ordinate(v_counter):=(v_y);
98 v_counter:=v_counter+1;
99 v_ordinate.extend;
100 v_ordinate(v_counter):=(v_z);
101 EXIT WHEN v_node= v_startnode OR v_ordinate.count=v_nrordinates;
102 END LOOP;
103
104 --check end-startnode
105 IF v_node<>v_startnode THEN dbms_output.put_line('startnode does not
equal endnode (possibly more than 2-manifold or a wrong orientation of
the first edge'); END IF;
106 --check number of nodes/ordinates
107 IF v_ordinate.count=v_nrordinates THEN
108 RING:=v_ordinate;
```

```
109 RETURN RING;  
110 ELSE RING:=NULL; RETURN RING; END IF;  
111 EXCEPTION WHEN NO_DATA_FOUND THEN RING:=NULL;  
112 RETURN RING;  
113  
114 END;
```

```
1 //FUNCTION GETNORMAL//
2
3
4 CREATE OR REPLACE FUNCTION getNORMAL(i_ring IN NUMBER) RETURN
  NORMAL_ARRAY AS
5
6 NORMAL NORMAL_ARRAY;
7 v_ring NUMBER;
8 x1 NUMBER;
9 y1 NUMBER;
10 z1 NUMBER;
11 x2 NUMBER;
12 y2 NUMBER;
13 z2 NUMBER;
14 x3 NUMBER;
15 y3 NUMBER;
16 z3 NUMBER;
17 x4 NUMBER;
18 y4 NUMBER;
19 z4 NUMBER;
20 a NUMBER:=4;
21 b NUMBER:=5;
22 c NUMBER:=6;
23 d NUMBER:=7;
24 e NUMBER:=8;
25 f NUMBER:=9;
26 vec1x NUMBER;
27 vec1y NUMBER;
28 vec1z NUMBER;
29 vec2x NUMBER;
30 vec2y NUMBER;
31 vec2z NUMBER;
32 normalX NUMBER;
33 normalY NUMBER;
34 normalZ NUMBER;
35 v_Nfactor NUMBER;
36 v_ordinate SDO_ORDINATE_ARRAY;
37
38 BEGIN
39
40 --getting started
41 v_ring:=i_ring;
42
43 --select x,y,z values of four nodes (representing 2 edges)
44 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
  ring_id=v_ring;
45 x1:= v_ordinate(1);
46 y1:= v_ordinate(2);
47 z1:= v_ordinate(3);
48 x2:= v_ordinate(4);
49 y2:= v_ordinate(5);
50 z2:= v_ordinate(6);
51 x3:= v_ordinate(a);
52 y3:= v_ordinate(b);
53 z3:= v_ordinate(c);
54 x4:= v_ordinate(d);
55 y4:= v_ordinate(e);
56 z4:= v_ordinate(f);
57
58 --calculate the normal of the ring
59 vec1x:=x2-x1;
60 vec1y:=y2-y1;
61 vec1z:=z2-z1;
```

```
62 vec2x:=x4-x3;
63 vec2y:=y4-y3;
64 vec2z:=z4-z3;
65 normalX:= (veclx * vec2z) - (veclz * vec2y);
66 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
67 normalZ:= (vec1x * vec2y) - (veclx * vec2x);
68
69 --in case the edges are parallel
70 IF (normalX=0 AND normalY=0 AND normalZ=0) THEN
71 --select x,y,z values of two nodes representing another second edge
72 LOOP
73 SELECT ringordinates INTO v_ordinate FROM ringordinates WHERE
ring_id=v_ring;
74 a:=a+1;
75 b:=b+1;
76 c:=c+1;
77 d:=d+1;
78 e:=e+1;
79 f:=f+1;
80 x3:= v_ordinate(a);
81 y3:= v_ordinate(b);
82 z3:= v_ordinate(c);
83 x4:= v_ordinate(d);
84 y4:= v_ordinate(e);
85 z4:= v_ordinate(f);
86 --calculate the normal of the ring
87 vec1x:=x2-x1;
88 vec1y:=y2-y1;
89 vec1z:=z2-z1;
90 vec2x:=x4-x3;
91 vec2y:=y4-y3;
92 vec2z:=z4-z3;
93 normalX:= (veclx * vec2z) - (veclz * vec2y);
94 normalY:= -((vec2z * vec1x) - (vec2x * vec1z));
95 normalZ:= (vec1x * vec2y) - (veclx * vec2x);
96 CONTINUE WHEN (normalX=0 AND normalY=0 AND normalZ=0);
97 END LOOP;
98 END IF;
99
100 --normalize normal (to delete the influence of the lenght of a normal)
101 v_Nfactor:= sqrt((normalX*normalX)+(normalY*normalY)+(normalZ*normalZ));
102 IF v_Nfactor=0 THEN NORMAL:=normal_array(0,0,0); RETURN NORMAL;
103 ELSE
104 normalX:=normalX/v_Nfactor;
105 normalY:=normalY/v_Nfactor;
106 normalZ:=normalZ/v_Nfactor;
107
108 --result
109 NORMAL:=normal_array(normalX,normalY,normalZ);
110 RETURN NORMAL;
111 END IF;
112
113 END;
```

```
1 //FUNCTION GETSHELL//
2
3
4 CREATE OR REPLACE FUNCTION getSHELL(i_shell IN NUMBER)
5 RETURN SDO_GEOMETRY AS
6
7 TYPE element_type IS TABLE OF NUMBER;
8 tab_POSface element_type;
9 tab_NEGface element_type;
10 v_shell NUMBER;
11 v_InOut VARCHAR(1);
12 v_face NUMBER;
13 v_nrfaces NUMBER;
14 v_offset NUMBER;
15 v_count NUMBER;
16 v_loop NUMBER;
17 x NUMBER;
18 v_innerrings NUMBER;
19 v_ordinate SDO_ORDINATE_ARRAY;
20 v_facelordinate SDO_ORDINATE_ARRAY;
21 v_face2ordinate SDO_ORDINATE_ARRAY;
22 v_Fordinate SDO_ORDINATE_ARRAY;
23 v_info_array SDO_ELEM_INFO_ARRAY;
24 SHELL SDO_GEOMETRY;
25
26 BEGIN
27
28 --GETTING STARTED
29 v_shell:=i_shell;
30 --dbms_output.put_line('shell: '||v_shell);
31 v_ordinate:=sdo_ordinate_array();
32 v_info_array:=sdo_elem_info_array(1,1007,1);
33 SELECT InOut INTO v_InOut FROM SHELL WHERE shell_id=v_shell;
34 --dbms_output.put_line('InOut: '||v_InOut);
35
36 --ORIGINALLY AN OUTER SHELL
37 IF v_InOut='O' THEN
38 --dbms_output.put_line('outershell');
39 --get faces of shell
40 SELECT face_id BULK COLLECT INTO tab_POSface FROM FACE WHERE
shell_id_ref_pos=v_shell;
41 SELECT face_id BULK COLLECT INTO tab_NEGface FROM FACE WHERE
shell_id_ref_neg=v_shell;
42 v_nrfaces:=tab_POSface.count+tab_NEGface.count;
43 --dbms_output.put_line('nr of faces: '||v_nrfaces);
44
45 --adjust array-info for shell
46 v_info_array.extend;
47 v_info_array(4):=1;
48 v_info_array.extend;
49 v_info_array(5):=1006;
50 v_info_array.extend;
51 v_info_array(6):=(v_nrfaces);
52
53 --get pos. oriented faces first
54 IF tab_POSface.count=0 THEN dbms_output.put_line('no pos. oriented faces
in shell');
55 ELSE FOR i IN tab_POSface.FIRST..tab_POSface.LAST LOOP
56 v_face:=tab_POSface(i);
57 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
face_id_ref=v_face;
58 --dbms_output.put_line('count inner rings: '||v_innerrings);
59 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
```

```
face: '||v_face||' (geen solid geconstrueerd'); SHELL:=NULL;
60 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
    ||v_face);
61 v_nrfaces:=v_nrfaces+1;
62 --dbms_output.put_line ('number of faces: '||v_nrfaces);
63 v_info_array(6):=(v_nrfaces);
64 deleteINNERINGS(v_face,v_facelordinate,v_face2ordinate);
65
66 --adjust array-info for face1
67 v_offset:=v_ordinate.count+1;
68 v_count:=v_info_array.count+1;
69 v_info_array.extend;
70 v_info_array(v_count):=(v_offset);
71 v_count:=v_count+1;
72 v_info_array.extend;
73 v_info_array(v_count):=(1003);
74 v_count:=v_count+1;
75 v_info_array.extend;
76 v_info_array(v_count):=(1);
77 --extend ordinate-array for face1
78 x:=v_facelordinate.count;
79 FOR v_counter IN 1..x LOOP
80 v_ordinate.extend;
81 v_ordinate(v_offset):=v_facelordinate(v_counter);
82 v_offset:=v_offset+1;
83 END LOOP;
84
85 --adjust array-info for face2
86 v_offset:=v_ordinate.count+1;
87 v_count:=v_info_array.count+1;
88 v_info_array.extend;
89 v_info_array(v_count):=(v_offset);
90 v_count:=v_count+1;
91 v_info_array.extend;
92 v_info_array(v_count):=(1003);
93 v_count:=v_count+1;
94 v_info_array.extend;
95 v_info_array(v_count):=(1);
96 --extend ordinate-array for face2
97 x:=v_face2ordinate.count;
98 FOR v_counter IN 1..x LOOP
99 v_ordinate.extend;
100 v_ordinate(v_offset):=v_face2ordinate(v_counter);
101 v_offset:=v_offset+1;
102 END LOOP;
103
104 ELSE SELECT ring_ordinates INTO v_Fordinate FROM ring WHERE
    face_id_ref=v_face AND InOut='O';
105
106 --adjust array-info
107 v_offset:=v_ordinate.count+1;
108 v_count:=v_info_array.count+1;
109 v_info_array.extend;
110 v_info_array(v_count):=(v_offset);
111 v_count:=v_count+1;
112 v_info_array.extend;
113 v_info_array(v_count):=(1003);
114 v_count:=v_count+1;
115 v_info_array.extend;
116 v_info_array(v_count):=(1);
117
118 --extend ordinate-array
119 x:=v_Fordinate.count;
```

```
120 FOR v_counter IN 1..x LOOP
121 v_ordinate.extend;
122 v_ordinate(v_offset):=v_Fordinate(v_counter);
123 v_offset:=v_offset+1;
124 END LOOP;
125 END IF;
126 END LOOP;
127 END IF;
128
129 --get neg. oriented faces of shell
130 IF tab_NEGface.count<>0 THEN
131 FOR i IN tab_NEGface.FIRST..tab_NEGface.LAST LOOP
132 v_face:=tab_NEGface(i);
133
134 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
face_id_ref=v_face;
135 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
face: '||v_face||' (geen solid geconstrueerd)'); SHELL:=NULL;
136 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
' ||v_face);
137 v_nrfaces:=v_nrfaces+1;
138 --dbms_output.put_line('number of faces ' ||v_nrfaces);
139 v_info_array(6):=(v_nrfaces);
140 deleteINNERRINGS(v_face,v_facelordinate,v_face2ordinate);
141
142 --adjust array-info for facel
143 v_offset:=v_ordinate.count+1;
144 v_count:=v_info_array.count+1;
145 v_info_array.extend;
146 v_info_array(v_count):=(v_offset);
147 v_count:=v_count+1;
148 v_info_array.extend;
149 v_info_array(v_count):=(1003);
150 v_count:=v_count+1;
151 v_info_array.extend;
152 v_info_array(v_count):=(1);
153 --extend ordinate-array for facel
154 x:=v_facelordinate.count;
155 v_loop:=(x/3);
156 FOR v_counter IN 1..v_loop LOOP
157 v_count:=x-2;
158 --dbms_output.put_line ('offset: ' ||v_offset);
159 --dbms_output.put_line ('count: ' ||v_count);
160 v_ordinate.extend;
161 v_ordinate(v_offset):=v_facelordinate(v_count);
162 v_offset:=v_offset+1;
163 v_count:=x-1;
164 --dbms_output.put_line ('offset: ' ||v_offset);
165 --dbms_output.put_line ('count: ' ||v_count);
166 v_ordinate.extend;
167 v_ordinate(v_offset):=v_facelordinate(v_count);
168 v_offset:=v_offset+1;
169 v_count:=x;
170 --dbms_output.put_line ('offset: ' ||v_offset);
171 --dbms_output.put_line ('count: ' ||v_count);
172 v_ordinate.extend;
173 v_ordinate(v_offset):=v_facelordinate(v_count);
174 v_offset:=v_offset+1;
175 x:=x-3;
176 END LOOP;
177
178 --adjust array-info for face2
179 v_offset:=v_ordinate.count+1;
```

```
180 v_count:=v_info_array.count+1;
181 v_info_array.extend;
182 v_info_array(v_count):=(v_offset);
183 v_count:=v_count+1;
184 v_info_array.extend;
185 v_info_array(v_count):=(1003);
186 v_count:=v_count+1;
187 v_info_array.extend;
188 v_info_array(v_count):=(1);
189 --extend ordinate-array for face2
190 x:=v_facelordinate.count;
191 v_loop:=(x/3);
192 FOR v_counter IN 1..v_loop LOOP
193 v_count:=x-2;
194 --dbms_output.put_line ('offset: '||v_offset);
195 --dbms_output.put_line ('count: '||v_count);
196 v_ordinate.extend;
197 v_ordinate(v_offset):=v_face2ordinate(v_count);
198 v_offset:=v_offset+1;
199 v_count:=x-1;
200 --dbms_output.put_line ('offset: '||v_offset);
201 --dbms_output.put_line ('count: '||v_count);
202 v_ordinate.extend;
203 v_ordinate(v_offset):=v_face2ordinate(v_count);
204 v_offset:=v_offset+1;
205 v_count:=x;
206 --dbms_output.put_line ('offset: '||v_offset);
207 --dbms_output.put_line ('count: '||v_count);
208 v_ordinate.extend;
209 v_ordinate(v_offset):=v_face2ordinate(v_count);
210 v_offset:=v_offset+1;
211 x:=x-3;
212 END LOOP;
213
214
215 ELSE SELECT ring_ordinates INTO v_Fordinate FROM ring WHERE
face_id_ref=v_face AND InOut='O';
216
217 --adjust array-info
218 v_offset:=v_ordinate.count+1;
219 v_count:=v_info_array.count+1;
220 v_info_array.extend;
221 v_info_array(v_count):=(v_offset);
222 v_count:=v_count+1;
223 v_info_array.extend;
224 v_info_array(v_count):=(1003);
225 v_count:=v_count+1;
226 v_info_array.extend;
227 v_info_array(v_count):=(1);
228
229 --extend ordinate-array
230 x:=v_Fordinate.count;
231 v_loop:=(x/3);
232 FOR v_counter IN 1..v_loop LOOP
233 v_count:=x-2;
234 v_ordinate.extend;
235 v_ordinate(v_offset):=v_Fordinate(v_count);
236 v_offset:=v_offset+1;
237 v_count:=x-1;
238 v_ordinate.extend;
239 v_ordinate(v_offset):=v_Fordinate(v_count);
240 v_offset:=v_offset+1;
241 v_count:=x;
```

```
242 v_ordinate.extend;
243 v_ordinate(v_offset):=v_Fordinate(v_count);
244 v_offset:=v_offset+1;
245 x:=x-3;
246 END LOOP;
247 END IF;
248 END LOOP;
249 END IF;
250
251
252 --ORIGINALLY AN INNER SHELL
253 ELSIF v_InOut='I' THEN
254 --dbms_output.put_line('innershell');
255 --get faces of shell
256 SELECT face_id BULK COLLECT INTO tab_POSface FROM FACE WHERE
shell_id_ref_pos=v_shell;
257 SELECT face_id BULK COLLECT INTO tab_NEGface FROM FACE WHERE
shell_id_ref_neg=v_shell;
258 v_nrfaces:=tab_POSface.count+tab_NEGface.count;
259 --dbms_output.put_line(v_nrfaces);
260
261 --adjust array-info for shell
262 v_info_array.extend;
263 v_info_array(4):=1;
264 v_info_array.extend;
265 v_info_array(5):=1006;
266 v_info_array.extend;
267 v_info_array(6):=(v_nrfaces);
268
269 --get pos. oriented faces first (because originally an inner shell, the
orientation is the other way around)
270 IF tab_NEGface.count=0 THEN dbms_output.put_line('no pos. oriented faces
in shell');
271 ELSE FOR i IN tab_NEGface.FIRST..tab_NEGface.LAST LOOP
272 v_face:=tab_NEGface(i);
273 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
face_id_ref=v_face;
274 --dbms_output.put_line('count inner rings: '||v_innerrings);
275 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
face: '||v_face||' (geen solid geconstrueerd)'); SHELL:=NULL;
276 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
' ||v_face);
277 v_nrfaces:=v_nrfaces+1;
278 --dbms_output.put_line ('number of faces: '||v_nrfaces);
279 v_info_array(6):=(v_nrfaces);
280 deleteINNERRINGS(v_face,v_facelordinate,v_face2ordinate);
281
282 --adjust array-info for facel
283 v_offset:=v_ordinate.count+1;
284 v_count:=v_info_array.count+1;
285 v_info_array.extend;
286 v_info_array(v_count):=(v_offset);
287 v_count:=v_count+1;
288 v_info_array.extend;
289 v_info_array(v_count):=(1003);
290 v_count:=v_count+1;
291 v_info_array.extend;
292 v_info_array(v_count):=(1);
293 --extend ordinate-array for facel
294 x:=v_facelordinate.count;
295 FOR v_counter IN 1..x LOOP
296 v_ordinate.extend;
297 v_ordinate(v_offset):=v_facelordinate(v_counter);
```

```
298 v_offset:=v_offset+1;
299 END LOOP;
300
301 --adjust array-info for face2
302 v_offset:=v_ordinate.count+1;
303 v_count:=v_info_array.count+1;
304 v_info_array.extend;
305 v_info_array(v_count):=(v_offset);
306 v_count:=v_count+1;
307 v_info_array.extend;
308 v_info_array(v_count):=(1003);
309 v_count:=v_count+1;
310 v_info_array.extend;
311 v_info_array(v_count):=(1);
312 --extend ordinate-array for face2
313 x:=v_face2ordinate.count;
314 FOR v_counter IN 1..x LOOP
315 v_ordinate.extend;
316 v_ordinate(v_offset):=v_face2ordinate(v_counter);
317 v_offset:=v_offset+1;
318 END LOOP;
319
320 ELSE SELECT ring_ordinates INTO v_Fordinate FROM ring WHERE
    face_id_ref=v_face AND InOut='O';
321
322 --adjust array-info
323 v_offset:=v_ordinate.count+1;
324 v_count:=v_info_array.count+1;
325 v_info_array.extend;
326 v_info_array(v_count):=(v_offset);
327 v_count:=v_count+1;
328 v_info_array.extend;
329 v_info_array(v_count):=(1003);
330 v_count:=v_count+1;
331 v_info_array.extend;
332 v_info_array(v_count):=(1);
333
334 --extend ordinate-array
335 x:=v_Fordinate.count;
336 FOR v_counter IN 1..x LOOP
337 v_ordinate.extend;
338 v_ordinate(v_offset):=v_Fordinate(v_counter);
339 v_offset:=v_offset+1;
340 END LOOP;
341 END IF;
342 END LOOP;
343 END IF;
344
345 --get neg. oriented faces of shell (because originally an inner shell,
    the orientation is the other way around)
346 IF tab_POSface.count<>0 THEN
347 FOR i IN tab_POSface.FIRST..tab_POSface.LAST LOOP
348 v_face:=tab_POSface(i);
349
350 SELECT count(*) INTO v_innerrings FROM ring WHERE InOut='I' AND
    face_id_ref=v_face;
351 IF v_innerrings>1 THEN dbms_output.put_line ('more than 1 inner ring in
    face: '||v_face||' (geen solid geconstrueerd)'); SHELL:=NULL;
352 ELSIF v_innerrings=1 THEN dbms_output.put_line ('1 inner ring, face:
    '||v_face);
353 v_nrfaces:=v_nrfaces+1;
354 --dbms_output.put_line('number of faces '||v_nrfaces);
355 v_info_array(6):=(v_nrfaces);
```

```
356 deleteINNERRINGS(v_face,v_facelordinate,v_face2ordinate);
357
358 --adjust array-info for facel
359 v_offset:=v_ordinate.count+1;
360 v_count:=v_info_array.count+1;
361 v_info_array.extend;
362 v_info_array(v_count):=(v_offset);
363 v_count:=v_count+1;
364 v_info_array.extend;
365 v_info_array(v_count):=(1003);
366 v_count:=v_count+1;
367 v_info_array.extend;
368 v_info_array(v_count):=(1);
369 --extend ordinate-array for facel
370 x:=v_facelordinate.count;
371 v_loop:=(x/3);
372 FOR v_counter IN 1..v_loop LOOP
373 v_count:=x-2;
374 --dbms_output.put_line ('offset: '||v_offset);
375 --dbms_output.put_line ('count: '||v_count);
376 v_ordinate.extend;
377 v_ordinate(v_offset):=v_facelordinate(v_count);
378 v_offset:=v_offset+1;
379 v_count:=x-1;
380 --dbms_output.put_line ('offset: '||v_offset);
381 --dbms_output.put_line ('count: '||v_count);
382 v_ordinate.extend;
383 v_ordinate(v_offset):=v_facelordinate(v_count);
384 v_offset:=v_offset+1;
385 v_count:=x;
386 --dbms_output.put_line ('offset: '||v_offset);
387 --dbms_output.put_line ('count: '||v_count);
388 v_ordinate.extend;
389 v_ordinate(v_offset):=v_facelordinate(v_count);
390 v_offset:=v_offset+1;
391 x:=x-3;
392 END LOOP;
393
394 --adjust array-info for face2
395 v_offset:=v_ordinate.count+1;
396 v_count:=v_info_array.count+1;
397 v_info_array.extend;
398 v_info_array(v_count):=(v_offset);
399 v_count:=v_count+1;
400 v_info_array.extend;
401 v_info_array(v_count):=(1003);
402 v_count:=v_count+1;
403 v_info_array.extend;
404 v_info_array(v_count):=(1);
405 --extend ordinate-array for face2
406 x:=v_facelordinate.count;
407 v_loop:=(x/3);
408 FOR v_counter IN 1..v_loop LOOP
409 v_count:=x-2;
410 --dbms_output.put_line ('offset: '||v_offset);
411 --dbms_output.put_line ('count: '||v_count);
412 v_ordinate.extend;
413 v_ordinate(v_offset):=v_face2ordinate(v_count);
414 v_offset:=v_offset+1;
415 v_count:=x-1;
416 --dbms_output.put_line ('offset: '||v_offset);
417 --dbms_output.put_line ('count: '||v_count);
418 v_ordinate.extend;
```

```
419 v_ordinate(v_offset):=v_face2ordinate(v_count);
420 v_offset:=v_offset+1;
421 v_count:=x;
422 --dbms_output.put_line ('offset: '||v_offset);
423 --dbms_output.put_line ('count: '||v_count);
424 v_ordinate.extend;
425 v_ordinate(v_offset):=v_face2ordinate(v_count);
426 v_offset:=v_offset+1;
427 x:=x-3;
428 END LOOP;
429
430 ELSE SELECT ring_ordinates INTO v_Fordinate FROM ring WHERE
      face_id_ref=v_face AND InOut='O';
431
432 --adjust array-info
433 v_offset:=v_ordinate.count+1;
434 v_count:=v_info_array.count+1;
435 v_info_array.extend;
436 v_info_array(v_count):=(v_offset);
437 v_count:=v_count+1;
438 v_info_array.extend;
439 v_info_array(v_count):=(1003);
440 v_count:=v_count+1;
441 v_info_array.extend;
442 v_info_array(v_count):=(1);
443
444 --extend ordinate-array
445 x:=v_Fordinate.count;
446 v_loop:=(x/3);
447 FOR v_counter IN 1..v_loop LOOP
448 v_count:=x-2;
449 v_ordinate.extend;
450 v_ordinate(v_offset):=v_Fordinate(v_count);
451 v_offset:=v_offset+1;
452 v_count:=x-1;
453 v_ordinate.extend;
454 v_ordinate(v_offset):=v_Fordinate(v_count);
455 v_offset:=v_offset+1;
456 v_count:=x;
457 v_ordinate.extend;
458 v_ordinate(v_offset):=v_Fordinate(v_count);
459 v_offset:=v_offset+1;
460 x:=x-3;
461 END LOOP;
462 END IF;
463 END LOOP;
464 END IF;
465 END IF;
466
467
468 --RESULT
469 SHELL:=SDO_GEOMETRY(3008,NULL,NULL,v_info_array,v_ordinate);
470 RETURN SHELL;
471 RETURN SHELL;
472 EXCEPTION WHEN NO_DATA_FOUND THEN SHELL:=NULL;
473 RETURN SHELL;
474
475 END;
```

```
1 //FUNCTION DISSOLVEINNERRINGS//
2
3
4 SET SERVEROUTPUT ON
5
6 CREATE OR REPLACE PROCEDURE deleteINNERRINGS(i_face IN NUMBER,o_ring1 OUT
  SDO_ORDINATE_ARRAY, o_ring2 OUT SDO_ORDINATE_ARRAY) AS
7
8 TYPE element_type IS TABLE OF NUMBER;
9 tab_face element_type;
10 tab_node1 element_type;
11 tab_node2 element_type;
12 v_face NUMBER;
13 v_Iring NUMBER;
14 v_Oring NUMBER;
15 v_edge NUMBER;
16 v_node NUMBER;
17 v_test NUMBER;
18 v_orientation VARCHAR(1);
19 v_max1 NUMBER;
20 v_max2 NUMBER;
21 v_max NUMBER;
22 v_min1 NUMBER;
23 v_min2 NUMBER;
24 v_min NUMBER;
25 v_NodeOmax NUMBER;
26 v_NodeOmin NUMBER;
27 v_NodeImax NUMBER;
28 v_NodeImin NUMBER;
29 v_NodeIO NUMBER;
30 v_ordinate SDO_ORDINATE_ARRAY;
31 v_counter NUMBER;
32 vx NUMBER;
33 vy NUMBER;
34 vz NUMBER;
35 v_chkedge NUMBER;
36 v_chknode NUMBER;
37 v_ringid NUMBER;
38 v_shellpos NUMBER;
39 v_shellneg NUMBER;
40 v_faceid NUMBER;
41
42
43 BEGIN
44
45
46
47
48
49 v_face:=i_face;
50 SELECT ring_id INTO v_Oring FROM ring WHERE face_id_ref=v_face AND
  InOUT='O';
51 SELECT ring_id INTO v_Iring FROM ring WHERE face_id_ref=v_face AND
  InOUT='I';
52 --dbms_output.put_line('face: '||v_face);
53 --dbms_output.put_line('Inner ring: '||v_Iring);
54 --dbms_output.put_line('Outer ring: '||v_Oring);
55
56 --check for nodes used in both inner and outer ring
57 SELECT startnode BULK COLLECT INTO tab_node1 FROM edge WHERE (startnode
  IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge FROM
  edge2shell WHERE shell=11)) OR startnode IN (SELECT endnode FROM edge
  WHERE edge_id IN (SELECT edge FROM edge2shell WHERE shell=11))) AND
```

```

(startnode IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge
FROM edge2shell WHERE shell=10)) OR startnode IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge FROM edge2shell WHERE shell=10)));
58 SELECT endnode BULK COLLECT INTO tab_node2 FROM edge WHERE (endnode IN
(SELECT startnode FROM edge WHERE edge_id IN (SELECT edge FROM edge2shell
WHERE shell=11)) OR endnode IN (SELECT endnode FROM edge WHERE edge_id IN
(SELECT edge FROM edge2shell WHERE shell=11))) AND (endnode IN (SELECT
startnode FROM edge WHERE edge_id IN (SELECT edge FROM edge2shell WHERE
shell=10)) OR endnode IN (SELECT endnode FROM edge WHERE edge_id IN
(SELECT edge FROM edge2shell WHERE shell=10)));
59 v_test:=(tab_nodel.count+tab_node2.count);
60 IF v_test>1 THEN dbms_output.put_line('more than 1 node is used in both
inner and outer ring');
61 ELSIF v_test=1 THEN dbms_output.put_line('1 node is used in both inner
and outer ring');
62 IF tab_nodel.count=1 THEN v_nodeIO:=tab_nodel(1); ELSE
v_nodeIO:=tab_node2(1); END IF;
63 ELSE v_nodeIO:=-99;
64 END IF;
65
66
67 --select min/max nodes of outer ring, first by X-ordinate
68 SELECT max(x) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
69 SELECT max(x) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
70 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
71 --dbms_output.put_line('max X outer ring: '||v_max);
72 SELECT min(x) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
73 SELECT min(x) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
74 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
75 --dbms_output.put_line('min X outer ring: '||v_min);
76
77 IF v_min=v_max THEN --chk Y
78 SELECT max(y) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
79 SELECT max(y) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
80 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
81 --dbms_output.put_line('max Y outer ring: '||v_max);
82 SELECT min(y) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
83 SELECT min(y) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
84 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
85 --dbms_output.put_line('min Y outer ring: '||v_min);
86
87 IF v_min=v_max THEN --chk Z
88 SELECT max(z) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
89 SELECT max(z) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE

```

```
ring_id_ref=v_Oring));
90 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
91 --dbms_output.put_line('max Z outer ring: '||v_max);
92 SELECT min(z) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
93 SELECT min(z) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring));
94 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
95 --dbms_output.put_line('min Z outer ring: '||v_min);
96 SELECT min(node_id) INTO v_nodeOmax FROM node WHERE z=v_max AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring)));
97 SELECT min(node_id) INTO v_nodeOmin FROM node WHERE z=v_min AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring)));
98 --dbms_output.put_line('max node Outer ring: '||v_nodeOmax);
99 --dbms_output.put_line('min node Outer ring: '||v_nodeOmin);
100 --select min/max nodes of inner ring
101 SELECT max(z) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
102 SELECT max(z) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
103 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
104 --dbms_output.put_line('max Z inner ring: '||v_max);
105 SELECT min(z) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
106 SELECT min(z) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
107 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
108 --dbms_output.put_line('min Z inner ring: '||v_min);
109 SELECT min(node_id) INTO v_nodeImax FROM node WHERE z=v_max AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring)));
110 SELECT min(node_id) INTO v_nodeImin FROM node WHERE z=v_min AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring)));
111 --dbms_output.put_line('max node inner ring: '||v_nodeImax);
112 --dbms_output.put_line('min node inner ring: '||v_nodeImin);
113
114 ELSE --- continue with Y
115 SELECT min(node_id) INTO v_nodeOmax FROM node WHERE y=v_max AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring)));
116 SELECT min(node_id) INTO v_nodeOmin FROM node WHERE y=v_min AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
```

```
ring_id_ref=v_Oring));
117 --dbms_output.put_line('max node Outer ring: '||v_nodeOmax);
118 --dbms_output.put_line('min node Outer ring: '||v_nodeOmin);
119 --select min/max nodes of inner ring
120 SELECT max(y) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
121 SELECT max(y) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
122 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
123 --dbms_output.put_line('max Y inner ring: '||v_max);
124 SELECT min(y) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
125 SELECT min(y) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
126 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
127 --dbms_output.put_line('min Y inner ring: '||v_min);
128 SELECT min(node_id) INTO v_nodeImax FROM node WHERE y=v_max AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring)));
129 SELECT min(node_id) INTO v_nodeImin FROM node WHERE y=v_min AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring)));
130 --dbms_output.put_line('max node inner ring: '||v_nodeImax);
131 --dbms_output.put_line('min node inner ring: '||v_nodeImin);
132 END IF;
133
134 ELSE --- continue with X
135 SELECT min(node_id) INTO v_nodeOmax FROM node WHERE x=v_max AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring)));
136 SELECT min(node_id) INTO v_nodeOmin FROM node WHERE x=v_min AND (node_id
IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
ring2edge WHERE ring_id_ref=v_Oring)) OR node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Oring)));
137 --dbms_output.put_line('max node Outer ring: '||v_nodeOmax);
138 --dbms_output.put_line('min node Outer ring: '||v_nodeOmin);
139 --select min/max nodes of inner ring
140 SELECT max(x) INTO v_max1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
141 SELECT max(x) INTO v_max2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
142 IF v_max1>=v_max2 THEN v_max:=v_max1; ELSE v_max:=v_max2; END IF;
143 --dbms_output.put_line('max x inner ring: '||v_max);
144 SELECT min(x) INTO v_min1 FROM node WHERE node_id IN (SELECT startnode
FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
145 SELECT min(x) INTO v_min2 FROM node WHERE node_id IN (SELECT endnode FROM
edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring));
146 IF v_min1<=v_min2 THEN v_min:=v_min1; ELSE v_min:=v_min2; END IF;
```

```
147 --dbms_output.put_line('min x inner ring: '||v_min);
148 SELECT min(node_id) INTO v_nodeImax FROM node WHERE x=v_max AND (node_id
    IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
    ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
    edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Iring)));
149 SELECT min(node_id) INTO v_nodeImin FROM node WHERE x=v_min AND (node_id
    IN (SELECT startnode FROM edge WHERE edge_id IN (SELECT edge_id_ref FROM
    ring2edge WHERE ring_id_ref=v_Iring)) OR node_id IN (SELECT endnode FROM
    edge WHERE edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Iring)));
150 --dbms_output.put_line('max node inner ring: '||v_nodeImax);
151 --dbms_output.put_line('min node inner ring: '||v_nodeImin);
152 END IF;
153
154 IF v_nodeIO<>-99 THEN
155 IF v_nodeIO=v_nodeImax THEN v_nodeImin:=v_nodeIO; v_nodeOmin:=v_nodeIO;
    ELSE v_nodeImax:=v_nodeIO; v_nodeOmax:=v_nodeIO; END IF;
156 END IF;
157
158
159 --RING 1
160 --start at max of Outer ring
161 BEGIN
162 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Oring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE startnode=v_nodeOmax)
    AND orientation = '+';
163 v_orientation:='+';
164 SELECT endnode INTO v_node FROM edge WHERE edge_id=v_edge;
165 EXCEPTION WHEN NO_DATA_FOUND THEN v_edge:=0;
166 END;
167 IF v_edge=0 THEN
168 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Oring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE endnode=v_nodeOmax)
    AND orientation = '-';
169 v_orientation:='-';
170 SELECT startnode INTO v_node FROM edge WHERE edge_id=v_edge; END IF;
171 --dbms_output.put_line('startnode outer ring: '||v_nodeOmax);
172 --dbms_output.put_line('startedge outer ring: '||v_edge);
173 --dbms_output.put_line('orientation startedge outer ring:
    '||v_orientation);
174 --dbms_output.put_line('node 2 outer ring: '||v_node);
175
176 --insert startnode in ordinate-array
177 SELECT x INTO vx FROM node WHERE node_id= v_nodeOmax;
178 SELECT y INTO vy FROM node WHERE node_id= v_nodeOmax;
179 SELECT z INTO vz FROM node WHERE node_id= v_nodeOmax;
180 v_ordinate:=sdo_ordinate_array(vx,vy,vz);
181 v_counter:=3;
182 --insert the second node in ordinate-array
183 SELECT x INTO vx FROM node WHERE node_id= v_node;
184 SELECT y INTO vy FROM node WHERE node_id= v_node;
185 SELECT z INTO vz FROM node WHERE node_id= v_node;
186 v_counter:=v_counter+1;
187 v_ordinate.extend;
188 v_ordinate(v_counter):=(vx);
189 v_counter:=v_counter+1;
190 v_ordinate.extend;
191 v_ordinate(v_counter):=(vy);
192 v_counter:=v_counter+1;
193 v_ordinate.extend;
194 v_ordinate(v_counter):=(vz);
195
```

```

196 --continue collecting and inserting nodes
197 IF v_node<>v_nodeOmin THEN LOOP
198 --search for negative oriented edge
199 BEGIN
200 SELECT startnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
    endnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Oring) AND edge_id <>v_edge;
201 v_node:=v_chknode;
202 v_edge:=v_chkedge;
203 EXCEPTION WHEN NO_DATA_FOUND THEN v_chknode:=0;
204 END;
205 --search for positive oriented edge (when no negative oriented edge is
    found)
206 IF v_chknode=0 THEN
207 SELECT endnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
    startnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Oring) AND edge_id <>v_edge;
208 v_node:=v_chknode;
209 v_edge:=v_chkedge;
210 END IF;
211
212 --insert v_node in ordinate-array
213 --dbms_output.put_line('v_node: '||v_node);
214 SELECT x INTO vx FROM node WHERE node_id= v_node;
215 SELECT y INTO vy FROM node WHERE node_id= v_node;
216 SELECT z INTO vz FROM node WHERE node_id= v_node;
217 --dbms_output.put_line('v_counter: '||v_counter);
218 v_counter:=v_counter+1;
219 v_ordinate.extend;
220 v_ordinate(v_counter):=(vx);
221 v_counter:=v_counter+1;
222 v_ordinate.extend;
223 v_ordinate(v_counter):=(vy);
224 v_counter:=v_counter+1;
225 v_ordinate.extend;
226 v_ordinate(v_counter):=(vz);
227 EXIT WHEN v_node=v_nodeOmin;
228 END LOOP;
229 END IF;
230
231 --continue at min of inner ring
232 BEGIN
233 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Iring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE startnode=v_nodeImin)
    AND orientation = '+';
234 v_orientation:= '+';
235 SELECT endnode INTO v_node FROM edge WHERE edge_id=v_edge;
236 EXCEPTION WHEN NO_DATA_FOUND THEN v_edge:=0;
237 END;
238 IF v_edge=0 THEN
239 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Iring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE endnode=v_nodeImin)
    AND orientation = '-';
240 v_orientation:= '-';
241 SELECT startnode INTO v_node FROM edge WHERE edge_id=v_edge; END IF;
242 --dbms_output.put_line('startnode inner ring: '||v_nodeImin);
243 --dbms_output.put_line('startedge inner ring: '||v_edge);
244 --dbms_output.put_line('orientation startedge inner ring:
    '||v_orientation);
245 --dbms_output.put_line('node 2 inner ring: '||v_node);
246
247 --insert startnode in ordinate-array
248 SELECT x INTO vx FROM node WHERE node_id= v_nodeImin;

```

```
249 SELECT y INTO vy FROM node WHERE node_id= v_nodeImin;
250 SELECT z INTO vz FROM node WHERE node_id= v_nodeImin;
251 v_counter:=v_counter+1;
252 v_ordinate.extend;
253 v_ordinate(v_counter):=(vx);
254 v_counter:=v_counter+1;
255 v_ordinate.extend;
256 v_ordinate(v_counter):=(vy);
257 v_counter:=v_counter+1;
258 v_ordinate.extend;
259 v_ordinate(v_counter):=(vz);
260 --insert the second node in ordinate-array
261 SELECT x INTO vx FROM node WHERE node_id= v_node;
262 SELECT y INTO vy FROM node WHERE node_id= v_node;
263 SELECT z INTO vz FROM node WHERE node_id= v_node;
264 v_counter:=v_counter+1;
265 v_ordinate.extend;
266 v_ordinate(v_counter):=(vx);
267 v_counter:=v_counter+1;
268 v_ordinate.extend;
269 v_ordinate(v_counter):=(vy);
270 v_counter:=v_counter+1;
271 v_ordinate.extend;
272 v_ordinate(v_counter):=(vz);
273
274 --continue collecting and inserting nodes
275 IF v_node<>v_nodeImax THEN LOOP
276 --search for negative oriented edge
277 BEGIN
278 SELECT startnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
endnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring) AND edge_id <>v_edge;
279 v_node:=v_chknode;
280 v_edge:=v_chkedge;
281 EXCEPTION WHEN NO_DATA_FOUND THEN v_chknode:=0;
282 END;
283 --search for positive oriented edge (when no negative oriented edge is
found)
284 IF v_chknode=0 THEN
285 SELECT endnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
startnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
ring_id_ref=v_Iring) AND edge_id <>v_edge;
286 v_node:=v_chknode;
287 v_edge:=v_chkedge;
288 END IF;
289
290 --insert v_node in ordinate-array
291 SELECT x INTO vx FROM node WHERE node_id= v_node;
292 SELECT y INTO vy FROM node WHERE node_id= v_node;
293 SELECT z INTO vz FROM node WHERE node_id= v_node;
294 v_counter:=v_counter+1;
295 v_ordinate.extend;
296 v_ordinate(v_counter):=(vx);
297 v_counter:=v_counter+1;
298 v_ordinate.extend;
299 v_ordinate(v_counter):=(vy);
300 v_counter:=v_counter+1;
301 v_ordinate.extend;
302 v_ordinate(v_counter):=(vz);
303 EXIT WHEN v_node=v_nodeImax;
304 END LOOP;
305 END IF;
306
```

```
307 --insert end node (equalling the start node = max node of outer ring) in
    ordinate array
308 SELECT x INTO vx FROM node WHERE node_id= v_nodeOmax;
309 SELECT y INTO vy FROM node WHERE node_id= v_nodeOmax;
310 SELECT z INTO vz FROM node WHERE node_id= v_nodeOmax;
311 v_counter:=v_counter+1;
312 v_ordinate.extend;
313 v_ordinate(v_counter):=(vx);
314 v_counter:=v_counter+1;
315 v_ordinate.extend;
316 v_ordinate(v_counter):=(vy);
317 v_counter:=v_counter+1;
318 v_ordinate.extend;
319 v_ordinate(v_counter):=(vz);
320 /*
321 SELECT max(ring_id) INTO v_ringid FROM ring;
322 v_ringid:=v_ringid+1;
323 SELECT shell_id_ref_pos, shell_id_ref_neg INTO v_shellpos, v_shellneg
    FROM FACE WHERE face_id=v_face;
324 SELECT max(face_id) INTO v_faceid FROM FACE;
325 v_faceid:=v_faceid+1;
326 INSERT INTO FACE(face_id,shell_id_ref_pos,shell_id_ref_neg) VALUES
    (v_faceid,v_shellpos,v_shellneg);
327 INSERT INTO ring (ring_id, face_id_ref,InOut,ring_ordinates) VALUES
    (v_ringid,v_faceid,'0',v_ordinate);
328 */
329 o_ringl:=v_ordinate;
330
331
332
333 --RING 2
334 --start at min of Outer ring
335 BEGIN
336 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Oring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE startnode=v_nodeOmin)
    AND orientation = '+' ;
337 v_orientation:='+' ;
338 SELECT endnode INTO v_node FROM edge WHERE edge_id=v_edge;
339 EXCEPTION WHEN NO_DATA_FOUND THEN v_edge:=0;
340 END;
341 IF v_edge=0 THEN
342 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Oring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE endnode=v_nodeOmin)
    AND orientation = '-' ;
343 v_orientation:='-';
344 SELECT startnode INTO v_node FROM edge WHERE edge_id=v_edge; END IF;
345 --dbms_output.put_line('startnode outer ring: '||v_nodeOmin);
346 --dbms_output.put_line('startedge outer ring: '||v_edge);
347 --dbms_output.put_line('orientation startedge outer ring:
    '||v_orientation);
348 --dbms_output.put_line('node 2 outer ring: '||v_node);
349
350 --insert startnode in ordinate-array
351 SELECT x INTO vx FROM node WHERE node_id= v_nodeOmin;
352 SELECT y INTO vy FROM node WHERE node_id= v_nodeOmin;
353 SELECT z INTO vz FROM node WHERE node_id= v_nodeOmin;
354 v_ordinate:=sdo_ordinate_array(vx,vy,vz);
355 v_counter:=3;
356 --insert the second node in ordinate-array
357 SELECT x INTO vx FROM node WHERE node_id= v_node;
358 SELECT y INTO vy FROM node WHERE node_id= v_node;
359 SELECT z INTO vz FROM node WHERE node_id= v_node;
360 v_counter:=v_counter+1;
```

```
361 v_ordinate.extend;
362 v_ordinate(v_counter):=(vx);
363 v_counter:=v_counter+1;
364 v_ordinate.extend;
365 v_ordinate(v_counter):=(vy);
366 v_counter:=v_counter+1;
367 v_ordinate.extend;
368 v_ordinate(v_counter):=(vz);
369
370 --continue collecting and inserting nodes
371 IF v_node<>v_nodeOmax THEN LOOP
372 --search for negative oriented edge
373 BEGIN
374 SELECT startnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
    endnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Oring) AND edge_id <>v_edge;
375 v_node:=v_chknode;
376 v_edge:=v_chkedge;
377 EXCEPTION WHEN NO_DATA_FOUND THEN v_chknode:=0;
378 END;
379 --search for positive oriented edge (when no negative oriented edge is
    found)
380 IF v_chknode=0 THEN
381 SELECT endnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
    startnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
    ring_id_ref=v_Oring) AND edge_id <>v_edge;
382 v_node:=v_chknode;
383 v_edge:=v_chkedge;
384 END IF;
385
386 --insert v_node in ordinate-array
387 --dbms_output.put_line('v_node: '||v_node);
388 SELECT x INTO vx FROM node WHERE node_id= v_node;
389 SELECT y INTO vy FROM node WHERE node_id= v_node;
390 SELECT z INTO vz FROM node WHERE node_id= v_node;
391 --dbms_output.put_line('v_counter: '||v_counter);
392 v_counter:=v_counter+1;
393 v_ordinate.extend;
394 v_ordinate(v_counter):=(vx);
395 v_counter:=v_counter+1;
396 v_ordinate.extend;
397 v_ordinate(v_counter):=(vy);
398 v_counter:=v_counter+1;
399 v_ordinate.extend;
400 v_ordinate(v_counter):=(vz);
401 EXIT WHEN v_node=v_nodeOmax;
402 END LOOP;
403 END IF;
404
405
406 --continue at max of inner ring
407 BEGIN
408 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Iring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE startnode=v_nodeImax)
    AND orientation = '+' ;
409 v_orientation:='+' ;
410 SELECT endnode INTO v_node FROM edge WHERE edge_id=v_edge;
411 EXCEPTION WHEN NO_DATA_FOUND THEN v_edge:=0;
412 END;
413 IF v_edge=0 THEN
414 SELECT edge_id_ref INTO v_edge FROM ring2edge WHERE ring_id_ref=v_Iring
    AND edge_id_ref IN (SELECT edge_id FROM edge WHERE endnode=v_nodeImax)
    AND orientation = '-' ;
```

```
415 v_orientation:='-';
416 SELECT startnode INTO v_node FROM edge WHERE edge_id=v_edge; END IF;
417 --dbms_output.put_line('startnode inner ring: '||v_nodeImax);
418 --dbms_output.put_line('startedge inner ring: '||v_edge);
419 --dbms_output.put_line('orientation startedge inner ring:
  ||v_orientation);
420 --dbms_output.put_line('node 2 inner ring: '||v_node);
421
422 --insert startnode in ordinate-array
423 SELECT x INTO vx FROM node WHERE node_id= v_nodeImax;
424 SELECT y INTO vy FROM node WHERE node_id= v_nodeImax;
425 SELECT z INTO vz FROM node WHERE node_id= v_nodeImax;
426 v_counter:=v_counter+1;
427 v_ordinate.extend;
428 v_ordinate(v_counter):=(vx);
429 v_counter:=v_counter+1;
430 v_ordinate.extend;
431 v_ordinate(v_counter):=(vy);
432 v_counter:=v_counter+1;
433 v_ordinate.extend;
434 v_ordinate(v_counter):=(vz);
435 --insert the second node in ordinate-array
436 SELECT x INTO vx FROM node WHERE node_id= v_node;
437 SELECT y INTO vy FROM node WHERE node_id= v_node;
438 SELECT z INTO vz FROM node WHERE node_id= v_node;
439 v_counter:=v_counter+1;
440 v_ordinate.extend;
441 v_ordinate(v_counter):=(vx);
442 v_counter:=v_counter+1;
443 v_ordinate.extend;
444 v_ordinate(v_counter):=(vy);
445 v_counter:=v_counter+1;
446 v_ordinate.extend;
447 v_ordinate(v_counter):=(vz);
448
449 --continue collecting and inserting nodes
450 IF v_node<>v_nodeImin THEN LOOP
451 --search for negative oriented edge
452 BEGIN
453 SELECT startnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
  endnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
  ring_id_ref=v_Iring) AND edge_id <>v_edge;
454 v_node:=v_chknode;
455 v_edge:=v_chkedge;
456 EXCEPTION WHEN NO_DATA_FOUND THEN v_chknode:=0;
457 END;
458 --search for positive oriented edge (when no negative oriented edge is
  found)
459 IF v_chknode=0 THEN
460 SELECT endnode,edge_id INTO v_chknode,v_chkedge FROM edge WHERE
  startnode=v_node AND edge_id IN (SELECT edge_id_ref FROM ring2edge WHERE
  ring_id_ref=v_Iring) AND edge_id <>v_edge;
461 v_node:=v_chknode;
462 v_edge:=v_chkedge;
463 END IF;
464
465 --insert v_node in ordinate-array
466 SELECT x INTO vx FROM node WHERE node_id= v_node;
467 SELECT y INTO vy FROM node WHERE node_id= v_node;
468 SELECT z INTO vz FROM node WHERE node_id= v_node;
469 v_counter:=v_counter+1;
470 v_ordinate.extend;
471 v_ordinate(v_counter):=(vx);
```

```
472 v_counter:=v_counter+1;
473 v_ordinate.extend;
474 v_ordinate(v_counter):=(vy);
475 v_counter:=v_counter+1;
476 v_ordinate.extend;
477 v_ordinate(v_counter):=(vz);
478 EXIT WHEN v_node=v_nodeImin;
479 END LOOP;
480 END IF;
481
482 --insert end node (equalling the start node = min node of outer ring) in
    ordinate array
483 SELECT x INTO vx FROM node WHERE node_id= v_nodeOmin;
484 SELECT y INTO vy FROM node WHERE node_id= v_nodeOmin;
485 SELECT z INTO vz FROM node WHERE node_id= v_nodeOmin;
486 v_counter:=v_counter+1;
487 v_ordinate.extend;
488 v_ordinate(v_counter):=(vx);
489 v_counter:=v_counter+1;
490 v_ordinate.extend;
491 v_ordinate(v_counter):=(vy);
492 v_counter:=v_counter+1;
493 v_ordinate.extend;
494 v_ordinate(v_counter):=(vz);
495 /*
496 SELECT max(ring_id) INTO v_ringid FROM ring;
497 v_ringid:=v_ringid+1;
498 SELECT shell_id_ref_pos, shell_id_ref_neg INTO v_shellpos, v_shellneg
    FROM FACE WHERE face_id=v_face;
499 SELECT max(face_id) INTO v_faceid FROM FACE;
500 v_faceid:=v_faceid+1;
501 INSERT INTO FACE(face_id,shell_id_ref_pos,shell_id_ref_neg) VALUES
    (v_faceid,v_shellpos,v_shellneg);
502 INSERT INTO ring (ring_id, face_id_ref,InOut,ring_ordinates) VALUES
    (v_ringid,v_faceid,'O',v_ordinate);
503 */
504 o_ring2:=v_ordinate;
505
506
507
508 END;
```