Geo1006 Lab4: PostGIS and valid polygons in the DBMS

Segher ter Braak, Benthe Kock, Ming Chieh Hu

Task 1

	id integer	st_astext text
1	2	POLYGON((33300 19200,19200 30000,8300 15000,20000 4200,33300 19200),(25000 13300,17500 13300,17500 19200,25000 19200,25000 133…

Task 2

Table - Building

```
CREATE TABLE building (id integer, the_geom geometry(POLYGON,0));

CREATE INDEX buildings_geom_idx
    ON building
    USING GIST (the_geom);
```

Table - Residential unit

```
CREATE TABLE residential_unit (id integer, the_geom geometry(POINT,0));

CREATE INDEX residential_geom_idx
    ON residential_unit
    USING GIST (the_geom);
```

Task 3

Table - Building

```
ALTER TABLE building ADD CONSTRAINT building_geom_valid_check
    CHECK (ST_IsValid(the_geom));

INSERT INTO building (id, the_geom) VALUES
    (1, ST_GeometryFromText('POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))'));
INSERT INTO building (id, the_geom) VALUES
    (2, ST_GeometryFromText('POLYGON((20 20, 20 30, 30 30, 30 20, 20 20), (22 22,
22 28, 28 28, 28 22, 22 22))'));
INSERT INTO building (id, the_geom) VALUES
    (3, ST_GeometryFromText('POLYGON((40 40, 40 50, 50 50, 50 40, 40 40), (45 40,
45 45, 49 45, 45 40))'));
INSERT INTO building (id, the_geom) VALUES
    (4, ST_GeometryFromText('POLYGON((60 60, 60 80, 80 80, 80 60, 60 60), (65 65,
65 75, 75 75, 75 65, 65 65))'));
```

Polygon cases where the inner-ring touches the outer-ring are rejected. For example:

```
insert into building (id, the_geom) values
    (3, ST_GeometryFromText('POLYGON((40 40, 40 50, 50 50, 50 40, 40 40), (45 40,
45 45, 50 45, 45 40))'));
```

Gives the following error:

```
NOTICE:  Interior is disconnected at or near point 50 45

ERROR:  Failing row contains (3, 0103000000020000000500000000000000000044400000000000004440000000...).new row
for relation "building" violates check constraint "buildings_geom_valid_check"

ERROR:  new row for relation "building" violates check constraint "buildings_geom_valid_check"
SQL state: 23514
Detail: Failing row contains (3, 0103000000020000000500000000000000000044400000000000004440000000...).
```

Table - Residential unit

```
ALTER TABLE residential_unit ADD CONSTRAINT residential_geom_valid_check
CHECK (ST_IsValid(the_geom));

INSERT INTO residential_unit (id, the_geom)
VALUES
    (1, ST_GeometryFromText('POINT(2 2)')),
    (2, ST_GeometryFromText('POINT(3 3)')),
    (3, ST_GeometryFromText('POINT(8 8)')),
    (4, ST_GeometryFromText('POINT(22 22)')),
    (5, ST_GeometryFromText('POINT(25 25)')),
    (6, ST_GeometryFromText('POINT(45 45)')),
    (7, ST_GeometryFromText('POINT(50 50)')),
    (8, ST_GeometryFromText('POINT(65 65)')),
    (9, ST_GeometryFromText('POINT(70 70)')),
    (10, ST_GeometryFromText('POINT(30 30)')),
    (11, ST_GeometryFromText('POINT(60 60)')),
    (12, ST_GeometryFromText('POINT(80 80)'));
```

Task 4

1) **The main difference between (valid) polygon definitions from computational geometry and the definition from OpenGIS.**

   Computational geometry: Self intersecting polygons are invalid (or not clear if valid by definition).
   OpenGIS: Rings can not cross but may touch tangentially (the same goes for one point to touch)

   Computational geometry: Open surfaces are allowed, if the surface has a clear orientation (no unoriëntable surfaces like a Möbius Band).
   OpenGIS: Surfaces are topologically closed

   Computational geometry: A polygon may have cut lines, spikes or punctures
   OpenGIS: A polygon may not have cut lines, spikes or punctures.

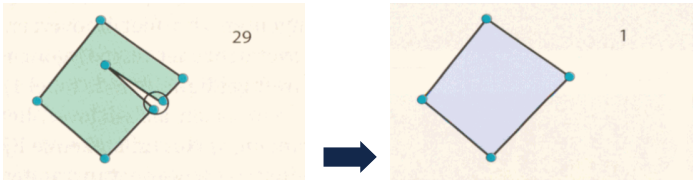
   Computational geometry: Exterior boundaries should be CCW and interior boundaries should be CW.
   OpenGIS: Exterior boundaries and interior boundaries may be used regardless of orientation.

Give 2 examples which are valid according to computational geometry and not according to OGC.
Computational geometry disallows: dangling segments, multi-polygon, polygon with holes, ring touches itself, self-intersection polygons, (partial) overlapping edges. In OGC, non-closed polygons and spikes are disallowed, but they're not specified in computational geometry definitions.
Examples:
1. Non-closed polygon: `POLYGON((0 0, 10 0, 10 10, 0 10))`
2. Polygon with a spike: `POLYGON((0 0, 20 0, 20 40, 19.9 20, 0 20, 0 0))`

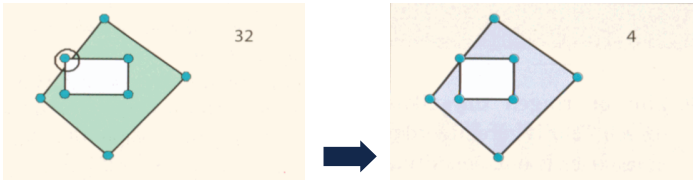
Give one example for which the reverse is true (valid in OGC, not in computational geometry)
Computational geometry disallows polygons with holes. Whereas OpenGIS does not.
Valid in OCG but not in computational geometry:
`POLYGON((0 0, 10 0, 10 10, 0 10, 0 0), (2 2, 2 8, 8 8, 8 2, 2 2))`

2) **Describe at least 5 aspects which are relevant when deciding if a polygon is valid or not according to the definition of van Oosterom, Quak and Tijssen.**
   - The polygon must not have any self-intersections or overlapping edges.
   - All boundary rings must be properly closed; all nodes must be connected to two line segments.
   - Rings are not allowed to cross each other. They may however touch at single points or overlap partially, as long as the overall interior forms one connected area.
   - Outer boundaries must be oriented counterclockwise and inner boundaries must be oriented clockwise.
   - A small numerical tolerance is used to handle small differences in coordinate values to compensate for computational limitations, ensuring practical validity.

3) **What is the difference between a valid and a clean polygon? Give two examples of polygons which are valid (according to vOQT), but not clean.**
   a) Polygon example 29, it resembles example 12 or example 1 after cleaning.



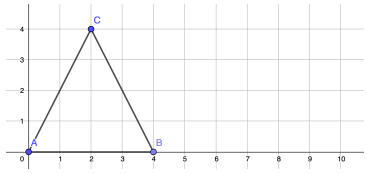
   b) Polygon example 32, it resembles example 4 after cleaning.



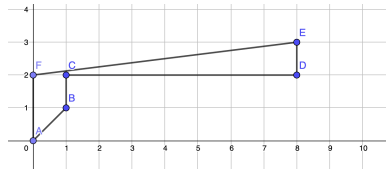
4) Why is the concept of robustness of a polygon useful (e.g. in an application)?

Robustness is the ability of a polygon to remain valid under small changes in its coordinates, ensuring resistance to become invalid for example due to computational or precision issues. This is important for application because high robustness provides confidence that polygons will remain valid and reliable during operations like geometric computation, or data transfer.

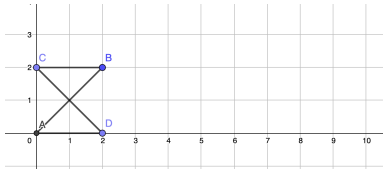
5) Compute the robustness value of the following polygons:



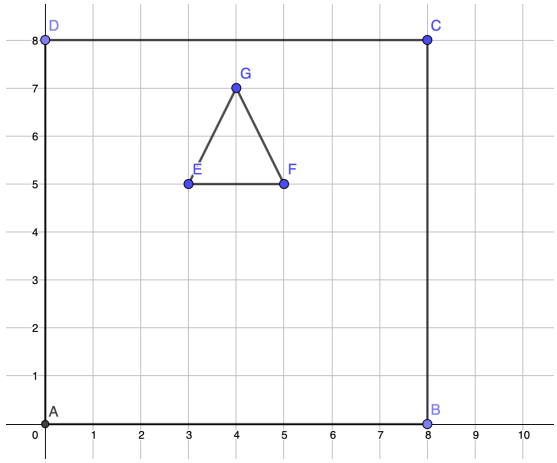
a. (0 0, 4 0, 2 4, 0 0)



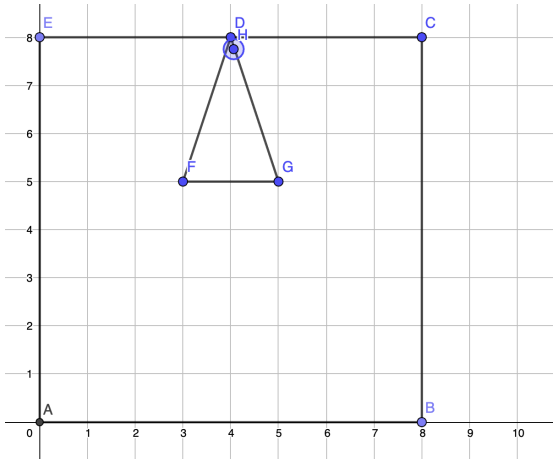
b. (0 0, 1 1, 1 2, 8 2, 8 3, 0 2, 0 0)



c. (0 0, 2 2, 0 2, 2 0, 0 0)



d. (0 0, 8 0, 8 8, 0 8, 0 0) (3 5, 5 5, 4 7, 3 5)



e. (0 0, 8 0, 8 8, 4 8, 0 8, 0 0) (3 5, 5 5, 4 8, 3 5)

Query - Calculate robustness using ST_MinimumClearance()

```
-- Insertion is omitted here
create or replace view valid_polygon as
select id, ST_IsValid(pgn), ST_IsValidReason(pgn), ST_MinimumClearance(pgn)
from test_polygon;

select id, ST_IsValid, ST_IsValidReason, (
      case
            when ST_IsValid = false then 0
            else ST_MinimumClearance
      end
)as robustness
from valid_polygon vp;
```

Query - Result

	123 id	st_isvalid	ABC st_isvalidreason	123 robustness
1	1	[v]	Valid Geometry	3.577708764
2	2	[v]	Valid Geometry	0.1240347346
3	3	[ ]	Self-intersection[1 1]	0
4	4	[v]	Valid Geometry	1
5	5	[v]	Valid Geometry	1.8973665961

6) Test the validation capabilities of the current version of PostGIS by loading the examples from the paper 'About Invalid, Valid and Clean Polygons' and comparing the results to the results in 2004 (in the paper).

Results

	id character varying (5)	st_isvalid boolean	st_isvalidreason text
1	1a	true	Valid Geometry
2	1b	true	Valid Geometry
3	2	true	Valid Geometry
4	3	true	Valid Geometry
5	4a	true	Valid Geometry
6	4b	false	Ring Self-intersection[13750 22500]
7	5	true	Valid Geometry
8	6	true	Valid Geometry
9	7	false	Ring Self-intersection[13300 21700]
10	8	false	Ring Self-intersection[13300 21700]
11	9	false	Self-intersection[10698 18300]
12	11	false	Self-intersection[38300 27500]
13	12	false	Ring Self-intersection[24200 25800]
14	13	false	Self-intersection[30000 21727.6595744681]
15	14a	false	Hole lies outside shell[36700 8300]
16	14b	false	Hole lies outside shell[36700 8300]
17	15	false	Ring Self-intersection[36700 8300]
18	16	false	Self-intersection[10698 18300]
19	17	false	Ring Self-intersection[14840 24000]
20	18	false	Interior is disconnected[29775 21900]
21	19	false	Self-intersection[22576.1674909331 16700]
22	20	false	Interior is disconnected[20800 15800]
23	21	false	Interior is disconnected[24200 16700]
24	22	false	Self-intersection[19606.2187673515 16899.444752915]
25	23	false	Ring Self-intersection[19600 16900]
26	24	false	Holes are nested[21700 18300]
27	25	false	Ring Self-intersection[20000 21700]
28	26	true	Valid Geometry
29	27	true	Valid Geometry
30	28	true	Valid Geometry
31	29	true	Valid Geometry
32	30	false	Self-intersection[14895.928611266 24076.9659788065]
33	31	true	Valid Geometry
34	32	false	Self-intersection[13750 22500]
35	33	true	Valid Geometry
36	34	true	Valid Geometry
37	35	false	Self-intersection[15419.1572468486 24797.005385571…
38	36	true	Valid Geometry
39	37	false	Self-intersection[26000 24791.4893617021]

Comparing the results - Polygons marked as true in the ST_IsValid column align with the results of PostGIS in the paper, indicating consistency in handling valid geometries. For the polygon with ID 10, we weren't able to create it and an error came up, since the polygon was not closed. This is in line with the results from the paper which also marked it as invalid. All the other invalid cases in our results differ from the results from the paper, since in the paper they were considered as valid. This indicates that the current PostGIS is more selective in (in)validity.

7) **Try to create your own 'critical' polygons (if possible different types than in the paper): one that you think should be valid and one that you think that should be invalid. Load these two polygons in PostGIS and report the result.**

```
insert into own_polygon values ('1', ST_GeomFromText(  'POLYGON((33300
19200, 19200 30000, 8300 15000, 3800 15001, 20000 4200, 33300 19200))',0));
```

Valid

```
insert into own_polygon values ('2', ST_GeomFromText(  'POLYGON((33300
19200, 19200 30000, 8300 15000, 8300 15001, 8300 15002, 20000 4200, 33300
19200))',0));
```

Invalid

These polygons are both the same shape but one has two consecutive points with a small distance to each other on the same line, the latter is the same but with three points of the same distance on a line. It is curious that in the first polygon the small difference is accepted, but in the other not. This has probably something to do with the precision.

8) **Give a definition of what you would call a valid polyhedron (i.e. a volume in 3D space with flat faces as boundaries) in a style similar to the valid 2D polygon definition by vOQT**

a) A polyhedron should have exactly 4 faces and 4 nodes in 3D.
b) It's a volume in 3D, so no nodes are allowed to touch any face or edge except of which it's connected to.
c) All nodes are at least connected to 3 edges and 3 unique points.
d) It's described by 4 polygons with exactly 3 points in 3D, example usage:
```
ST_GeomFromText('POLYHEDRON((0 0 0, 1 0 0, 0 1 0), (1 0 0, 2 0 0, 1 1
0), (0 1 0, 1 1 0, 0 2 0), (0 0 0, 0 1 0, 0 0 1))')
```