## 1) Определить рейтинг отрезков по сумме расстояний до них от точек на рисунке?

CREATE OR REPLACE VIEW segments AS (

SELECT public.object.idobject AS segment\_id, point.x AS x, point.y AS y

FROM public.object

INNER JOIN objpoints ON public.object.idobject = objpoints.idobject

INNER JOIN point ON objpoints.idpoint = point.idpoint

WHERE objtype = 1

);

-- DROP FUNCTION IF EXISTS calc\_distance;

CREATE OR REPLACE FUNCTION calc\_distance(segment\_pk INT, point\_id INT) RETURNS FLOAT

LANGUAGE plpgsql

AS

$$

DECLARE \_a FLOAT;

DECLARE \_b FLOAT;

DECLARE \_c FLOAT;

DECLARE x0 FLOAT;

DECLARE y0 FLOAT;

DECLARE x1 FLOAT;

DECLARE y1 FLOAT;

DECLARE x2 FLOAT;

DECLARE y2 FLOAT;

BEGIN

SELECT x, y INTO x0, y0

FROM point

WHERE point\_id = point.idpoint

LIMIT 1;

SELECT x, y INTO x1, y1

FROM segments

WHERE segment\_pk = segments.segment\_id

LIMIT 1;

SELECT x, y INTO x2, y2

FROM segments

WHERE segment\_pk = segments.segment\_id

LIMIT 1 OFFSET 1;

\_a = y1 - y2;

\_b = x2 - x1;

\_c = (x1 \* y2 - x2 \* y1);

RETURN ABS(\_a \* x0 + \_b \* y0 + \_c) / SQRT(\_a \* \_a + \_b \* \_b);

END;

$$;

SELECT segments.segment\_id AS segment\_id,

SUM(calc\_distance(segments.segment\_id, point.idpoint)) AS rating

FROM segments, point

GROUP BY segments.segment\_id ORDER BY rating DESC;

Вариант 2

DROP VIEW IF EXISTS segments;

DROP VIEW IF EXISTS layer\_point;

CREATE OR REPLACE VIEW segments AS (

SELECT public.object.idobject AS segment\_id, layer.idlayer AS layer\_id, point.x AS x, point.y AS y

FROM public.object

INNER JOIN objpoints ON public.object.idobject = objpoints.idobject

INNER JOIN point ON objpoints.idpoint = point.idpoint

INNER JOIN entity ON entity.identity = public.object.idobject

INNER JOIN layer ON layer.idlayer = entity.layer

WHERE objtype = 1

);

CREATE OR REPLACE VIEW layer\_point AS(

SELECT layer.idlayer AS layer, point.idpoint AS point\_id

FROM entity

JOIN layer ON layer.idlayer = entity.layer

JOIN public.object AS o ON o.idobject = entity.identity

JOIN objpoints ON objpoints.idobject = o.idobject

JOIN point ON point.idpoint = objpoints.idpoint

);

— DROP FUNCTION IF EXISTS calc\_distance;

CREATE OR REPLACE FUNCTION calc\_distance(segment\_pk INT, point\_id INT) RETURNS FLOAT

LANGUAGE plpgsql

AS

$$

DECLARE \_a FLOAT;

DECLARE \_b FLOAT;

DECLARE \_c FLOAT;

DECLARE x0 FLOAT;

DECLARE y0 FLOAT;

DECLARE x1 FLOAT;

DECLARE y1 FLOAT;

DECLARE x2 FLOAT;

DECLARE y2 FLOAT;

BEGIN

SELECT x, y INTO x0, y0

FROM point

WHERE point\_id = point.idpoint

LIMIT 1;

SELECT x, y INTO x1, y1

FROM segments

WHERE segment\_pk = segments.segment\_id

LIMIT 1;

SELECT x, y INTO x2, y2

FROM segments

WHERE segment\_pk = segments.segment\_id

LIMIT 1 OFFSET 1;

\_a = y1 - y2;

\_b = x2 - x1;

\_c = (x1 \* y2 - x2 \* y1);

RETURN ABS(\_a \* x0 + \_b \* y0 + \_c) / SQRT(\_a \* \_a + \_b \* \_b);

END;

$$;

— SELECT \* FROM segments;

SELECT segments.segment\_id AS segment\_id, segments.layer\_id AS layer\_id, SUM(calc\_distance(segments.segment\_id, layer\_point.point\_id)) AS rating

FROM segments

JOIN layer\_point ON layer\_point.layer = segments.layer\_id

GROUP BY segment\_id, layer\_id

ORDER BY rating DESC;

— SELECT segments.segment\_id AS segment\_id,

— SUM(calc\_distance(segments.segment\_id, point.idpoint)) AS rating

— FROM segments, point

— GROUP BY segments.segment\_id ORDER BY rating DESC;

## 2) Определить рейтинг типов графических объектов по их количеству на каждом слое.

WITH points AS

(SELECT count(p.idpoint), 'Point' as name, layer

FROM layer as l

JOIN entity as e ON l.idlayer = e.layer

JOIN point as p ON p.idpoint = e.identity

GROUP BY layer

ORDER BY layer), segments AS

(SELECT layer, 'Segment' as name, count(name)

FROM layer as l

JOIN entity as e ON l.idlayer = e.layer

JOIN object as o ON o.idobject = e.identity

JOIN objtype as ot ON ot.idobjtype = o.objtype

WHERE name = 'Segment'

GROUP BY layer ORDER BY layer),

circles AS

(SELECT layer, 'Circle' as name, count(name)

FROM layer as l

JOIN entity as e ON l.idlayer = e.layer

JOIN object as o ON o.idobject = e.identity

JOIN objtype as ot ON ot.idobjtype = o.objtype

WHERE name = 'Circle'

GROUP BY layer

ORDER BY layer),

arcs AS

(SELECT layer, 'Arc' as name, count(name)

FROM layer as l

JOIN entity as e ON l.idlayer = e.layer

JOIN object as o ON o.idobject = e.identity

JOIN objtype as ot ON ot.idobjtype = o.objtype

WHERE name = 'Arc'

GROUP BY layer

ORDER BY layer),

counted as

(SELECT layer, name, count

FROM (

SELECT \*

FROM points

UNION

SELECT \*

FROM segments

UNION

SELECT \*

FROM circles

UNION

SELECT \*

FROM arcs) as T

GROUP BY layer, name, count),

rating as

(SELECT points.layer as lyr, GREATEST(points.count, segments.count,circles.count, arcs.count), points.name as pname, points.count as pc, segments.name as sname, segments.count as sc, circles.name as cname, circles.count as cc, arcs.name as aname, arcs.count as ac

FROM points

FULL JOIN segments ON points.layer = segments.layer

FULL JOIN circles ON circles.layer = points.layer

FULL JOIN arcs ON arcs.layer = points.layer)

SELECT

CASE

WHEN GREATEST = pc THEN 'Point'

WHEN GREATEST = sc THEN 'Segment'

WHEN GREATEST = cc THEN 'Circle'

WHEN GREATEST = ac THEN 'Arc'

END AS most, lyr, GREATEST

FROM rating;

## 3) Для каждой пары перпендикулярных отрезков определить тот из них, чья длина наибольшая

SELECT hor\_id, hor\_len, vert\_id, vert\_len, hor\_layer AS layer,

CASE

WHEN hor\_len >= vert\_len THEN hor\_id

WHEN vert\_len > hor\_len THEN vert\_id

END AS longest

FROM (

(

SELECT DISTINCT entity.identity AS hor\_id, SUM(point.x)\*SUM(point.x) + SUM(point.y)\*SUM(point.y) AS hor\_len, constrtype.name AS hor\_type, entity.layer AS hor\_layer

FROM entity

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint ON public.constraint.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = public.constraint.constrtype

JOIN public.object ON object.idobject = entity.identity

JOIN objtype ON objtype.idobjtype = public.object.objtype

JOIN objpoints ON objpoints.idobject = public.object.idobject

JOIN point ON point.idpoint = objpoints.idpoint

WHERE objtype.name = 'Segment' AND constrtype.name = 'Horizontal'

GROUP BY entity.identity, hor\_type

) AS hor

JOIN (

SELECT DISTINCT entity.identity AS vert\_id, SUM(point.x)\*SUM(point.x) + SUM(point.y)\*SUM(point.y) AS vert\_len, constrtype.name AS vert\_type, entity.layer AS vert\_layer

FROM entity

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint ON public.constraint.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = public.constraint.constrtype

JOIN public.object ON public.object.idobject = entity.identity

JOIN objtype ON objtype.idobjtype = public.object.objtype

JOIN objpoints ON objpoints.idobject = public.object.idobject

JOIN point ON point.idpoint = objpoints.idpoint

WHERE objtype.name = 'Segment' AND constrtype.name = 'Vertical'

GROUP BY entity.identity, vert\_type

) AS vert

ON hor.hor\_layer = vert.vert\_layer

)

//

CREATE VIEW seglen AS

SELECT DISTINCT a.idobject, (a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y) AS length, i.idconstraint FROM

(SELECT o.idobject, objtype, p.idpoint,num,x,y FROM object as o

JOIN objpoints as op ON o.idobject=op.idobject

JOIN point as p ON p.idpoint=op.idpoint

WHERE o.objtype=1) AS a

JOIN

(SELECT o.idobject, objtype, p.idpoint,num,x,y FROM object as o

JOIN objpoints as op ON o.idobject=op.idobject

JOIN point as p ON p.idpoint=op.idpoint

WHERE o.objtype=1) AS b

ON a.idpoint != b.idpoint AND a.idobject=b.idobject

JOIN constrinfo as i ON i.identity=a.idobject

JOIN public.constraint AS c ON c.idconstraint=i.idconstraint

WHERE c.constrtype=7;

SELECT x.id1 as segment1, x.id2 AS segment2, x.s1l AS segment1len, x.s2l AS segment2len,

CASE WHEN x.s1l>x.s2l THEN x.id1

ELSE x.id2 END AS longest

FROM (

SELECT s1.idobject AS id1, s2.idobject AS id2, s1.length AS s1l, s2.length AS s2l

FROM seglen s1

JOIN seglen s2 ON s1.idconstraint=s2.idconstraint AND s1.idobject != s2.idobject

) as x;

## 4) Определить количество окружностей, связанных ограничениями с равными отрезками (скорее всего неправильно)

SELECT COUNT(id2) AS number\_of\_circles FROM

(

SELECT e.identity AS id, \_c.idconstraint AS idcon

FROM entity AS e

JOIN constrinfo AS \_c ON e.identity = \_c.identity

JOIN public.object AS o ON o.idobject = e.identity

JOIN public.constraint AS c1 ON c1.idconstraint = \_c.idconstraint

WHERE constrtype = 1 AND objtype = 1

) AS t1

JOIN

(

SELECT e2.identity AS id2, idconstraint AS idcon2

FROM entity AS e2

JOIN constrinfo AS c2 ON e2.identity = c2.identity

JOIN public.object AS o2 ON o2.idobject = e2.identity

WHERE objtype = 5

) AS t2

ON t1.idcon = t2.idcon2;

## 5) Определить дуги, имеющие общие точки с отрезками

SELECT arc\_table.arc\_id AS arc\_id, arc\_table.arc\_layer AS arc\_layer, seg\_table.seg\_id AS seg\_id, seg\_table.seg\_layer AS segment\_id

FROM (

(

SELECT o.idobject AS arc\_id, e.layer AS arc\_layer, points.arc\_x, points.arc\_y

FROM public.object AS o

JOIN entity AS e

ON o.idobject = e.identity

JOIN (

SELECT p.x AS arc\_x, p.y AS arc\_y, op.idobject AS arc\_id\_point

FROM point p

JOIN objpoints op

ON p.idpoint = op.idpoint

) AS points

ON points.arc\_id\_point = o.idobject

JOIN objtype AS ot

ON o.objtype = ot.idobjtype

WHERE ot.name = 'Arc'

) AS arc\_table

JOIN (

SELECT o.idobject AS seg\_id, e.layer AS seg\_layer, points.seg\_x, points.seg\_y

FROM public.object AS o

JOIN entity AS e

ON o.idobject = e.identity

JOIN (

SELECT \_p.x AS seg\_x, \_p.y AS seg\_y, op.idobject AS seg\_id\_point

FROM point AS \_p

JOIN objpoints AS op

ON \_p.idpoint = op.idpoint

) AS points

ON points.seg\_id\_point = o.idobject

JOIN objtype AS ot

ON o.objtype = ot.idobjtype

WHERE ot.name = 'Segment'

) AS seg\_table

ON arc\_x = seg\_x AND arc\_y = seg\_y AND arc\_layer = seg\_layer

);

## 6) Определить существуют ли объекты, такие что для нескольких из них в рамках одного слоя существуют одинаковые ограничения размера

-- Определить существуют ли объекты, такие что

-- для нескольких из них в рамках одного слоя

-- существуют одинаковые ограничения размера

-- выбираем тройки объект-слой-ограничение такие что

-- ограничение имеет тип 1 (т.е. размера)

CREATE OR REPLACE VIEW olc AS

SELECT o.idobject, e.layer, \_c.idconstraint FROM public.object AS o

JOIN entity AS e ON e.identity=o.idobject

JOIN constrinfo AS \_info ON \_info.identity=o.idobject

JOIN public.constraint AS \_c ON \_c.idconstraint=\_info.idconstraint

WHERE \_c.constrtype=1;

-- попарно объединяем тройки если

-- \* не одна и так же запись

-- \* на одном слою

-- \* одно и то же ограничение размера

SELECT EXISTS (

SELECT \* FROM olc AS olc1

JOIN olc AS olc2 ON

olc1.layer=olc2.layer

AND olc1.idconstraint=olc2.idconstraint

AND olc1.idobject != olc2.idobject);

## 7) Определить, есть ли на рисунке отрезок, перпендикулярный семи другим отрезкам?

CREATE OR REPLACE VIEW segments\_points AS

(

SELECT \_o.idobject AS id, \_t.name AS name, \_p.x AS x, \_p.y AS y, \_p.idpoint AS point\_id

FROM public.object AS \_o

JOIN objtype AS \_t ON \_o.objtype = \_t.idobjtype

JOIN objpoints AS \_op ON \_o.idobject = \_op.idobject

JOIN point AS \_p ON \_op.idpoint = \_p.idpoint

WHERE \_t.name = 'Segment'

);

CREATE OR REPLACE VIEW segments\_vectors AS

(

SELECT DISTINCT ON (id) sp1.id as id, sp1.name as name, (sp1.x - sp2.x) as x, (sp1.y - sp2.y) as y

FROM segments\_points as sp1

JOIN segments\_points as sp2

ON sp1.id = sp2.id and sp1.point\_id != sp2.point\_id

);

CREATE OR REPLACE FUNCTION is\_perpendicular(segment\_1 INT, segment\_2 INT)

RETURNS BOOLEAN

LANGUAGE plpgsql

AS

$$

DECLARE x1 FLOAT;

DECLARE y1 FLOAT;

DECLARE x2 FLOAT;

DECLARE y2 FLOAT;

BEGIN

SELECT x, y INTO x1, y1

FROM segments\_vectors AS sv

WHERE sv.id = segment\_1;

SELECT x, y INTO x2, y2

FROM segments\_vectors AS sv

WHERE sv.id = segment\_2;

RETURN (x1 \* x2) + (y1 \* y2) = 0;

END;

$$;

SELECT \_result.id1 as id

FROM

(

SELECT sv1.id AS id1, sv2.id AS id2

FROM segments\_vectors AS sv1

JOIN segments\_vectors AS sv2

ON is\_perpendicular(sv1.id, sv2.id)

) as \_result

GROUP BY id

HAVING (COUNT(\_result.id1) / 2) = 7;

## 8) Определить все объекты, связанные вертикальным ограничением

SELECT public.object.idobject AS id, objtype.name AS "object type"

FROM public.object

JOIN objtype

ON objtype.idobjtype = public.object.objtype

JOIN entity

ON entity.identity = public.object.idobject

JOIN constrinfo

ON constrinfo.identity = entity.identity

JOIN public.constraint

ON public.constraint.idconstraint = constrinfo.idconstraint

JOIN constrtype

ON constrtype.idconstrtype = public.constraint.constrtype

WHERE constrtype.name = 'Vertical';

### 2 вариант

SELECT COUNT(DISTINCT idobject) FROM public.object

JOIN entity AS e ON public.object.idobject = e.identity

JOIN constrinfo AS \_c ON e.identity = \_c.identity

JOIN public.constraint AS c2 ON c2.idconstraint = \_c.idconstraint

WHERE constrtype = 2;

## 9) Определить в базе данных наличие дублирующихся ограничений

CREATE OR REPLACE VIEW entity\_constraint AS

(

SELECT entity.identity, objtype.name AS obj\_name, constrtype.name AS constrtype, constrtype.is\_parametric

FROM entity

JOIN public.object AS o ON o.idobject = entity.identity

JOIN objtype ON objtype.idobjtype = o.objtype

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint AS \_c ON \_c.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = \_c.constrtype

);

￼

SELECT ec.identity, COUNT(ec.constrtype)

FROM entity\_constraint AS ec

GROUP BY ec.identity

HAVING COUNT(ec.constrtype) > 1;

## 10) На каком из слоев находится максимальное количество сущностей, связанных с сущностями из других слоев

DROP VIEW IF EXISTS layer\_entity;

DROP VIEW IF EXISTS constr\_entity\_num;

DROP VIEW IF EXISTS entity\_constraint\_layer;

CREATE OR REPLACE VIEW entity\_constraint\_layer AS

(

SELECT entity.identity AS id\_e, \_c.idconstraint AS id\_c, entity.layer, objtype.name AS obj\_name, constrtype.name AS constr\_name

FROM entity

JOIN layer ON entity.layer = layer.idlayer

FULL JOIN public.object AS o ON o.idobject = entity.identity

FULL JOIN objtype ON o.objtype = objtype.idobjtype

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint AS \_c ON \_c.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = \_c.constrtype

ORDER BY \_c.idconstraint

);

CREATE OR REPLACE VIEW constr\_entity\_num AS

(

SELECT \*

FROM entity\_constraint\_layer AS e

WHERE e.id\_c IN (

SELECT ecl.id\_c

FROM entity\_constraint\_layer AS ecl

GROUP BY ecl.id\_c

HAVING COUNT(id\_e) > 1

)

);

CREATE OR REPLACE VIEW layer\_entity AS

(

SELECT DISTINCT ON (id) cen1.id\_e AS id, cen1.layer AS layer, cen1.id\_c AS constr\_id, cen1.constr\_name AS costr\_name, cen1.obj\_name AS obj\_name

FROM constr\_entity\_num AS cen1

JOIN constr\_entity\_num AS cen2

ON cen1.id\_c = cen2.id\_c AND cen1.layer != cen2.layer

);

SELECT \*

FROM

(

SELECT res.layer, COUNT (res.id)

FROM layer\_entity AS res

GROUP BY res.layer

) AS abc

ORDER BY abc.count DESC

LIMIT 1;

## 11) Определить количество объектов, связанных ограничениями с вертикальными отрезками

DROP VIEW IF EXISTS entity\_constraint;

CREATE OR REPLACE VIEW entity\_constraint AS

(

SELECT entity.identity AS id\_e, \_c.idconstraint AS id\_c, objtype.name AS obj\_name, constrtype.name AS constr\_name

FROM entity

FULL JOIN public.object AS o ON o.idobject = entity.identity

FULL JOIN objtype ON o.objtype = objtype.idobjtype

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint AS \_c ON \_c.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = \_c.constrtype

ORDER BY \_c.idconstraint

);

SELECT COUNT(\*)

FROM entity\_constraint AS ec1

JOIN entity\_constraint AS ec2

ON ec1.id\_c = ec2.id\_c

WHERE ec1.obj\_name = 'Segment' AND ec1.constr\_name = 'Vertical';

## 12) Определить есть ли на каком-то слое объекты с одинаковым ограничением размера

DROP VIEW IF EXISTS entity\_constraint\_layer;

CREATE OR REPLACE VIEW entity\_constraint\_layer AS

(

SELECT entity.identity AS id\_e, \_c.idconstraint AS id\_c, entity.layer, objtype.name AS obj\_name, constrtype.name AS constr\_name

FROM entity

JOIN layer ON entity.layer = layer.idlayer

FULL JOIN public.object AS o ON o.idobject = entity.identity

FULL JOIN objtype ON o.objtype = objtype.idobjtype

JOIN constrinfo ON constrinfo.identity = entity.identity

JOIN public.constraint AS \_c ON \_c.idconstraint = constrinfo.idconstraint

JOIN constrtype ON constrtype.idconstrtype = \_c.constrtype

ORDER BY \_c.idconstraint

);

SELECT DISTINCT ON (ecl1.id\_e, ecl2.id\_e) ecl1.id\_e AS entity\_id\_1, ecl2.id\_e AS entity\_id\_2, ecl1.layer AS layer, ecl1.constr\_name AS constr\_name

FROM entity\_constraint\_layer AS ecl1

JOIN entity\_constraint\_layer AS ecl2

ON ecl1.constr\_name = ecl2.constr\_name AND ecl1.layer = ecl2.layer AND ecl1.id\_e != ecl2.id\_e;