Criação e Manipulação de Geometrias

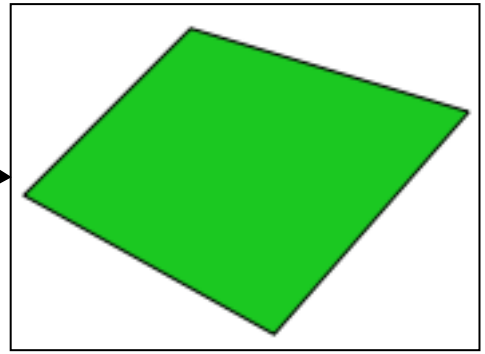
Paulo Trigo Silva

Criar e Manipular Geometrias – modo usual com WKT

```
CREATE TABLE FREGUESIA( nome VARCHAR( 30 ) PRIMARY KEY ) WITH OIDS;
SELECT AddGeometryColumn( '', 'freguesia', 'g_freguesia', -1, 'POLYGON', 2 );
```

para visualização no QuantumGIS
quando a chave primária não é INT

```
INSERT INTO FREGUESIA( nome, g_freguesia )
VALUES( 'Lumiar',
        ST_GeomFromText( 'POLYGON(
        ( 0.916 1.147,
          0.906 1.150,
          0.900 1.144,
          0.909 1.139,
          0.916 1.147 )
        )', -1) );
```



`ST_GeomFromText( WKT )`

Apenas aceita *WKT* (*Well-Known-Text*)

Obriga à construção de grandes expressões de texto.

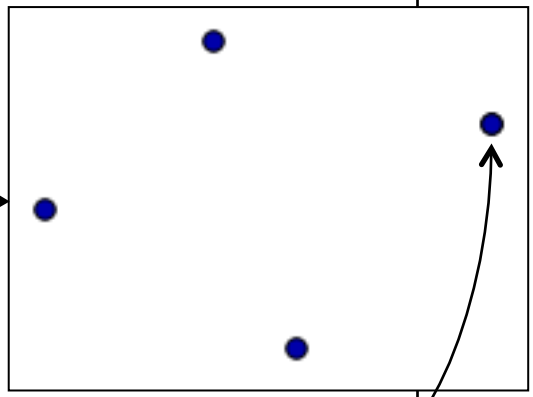
Não é adequada à utilização de directivas preparadas (*prepared statments*).

Paulo Trigo Silva

Manipular Geometrias – com ST_Make* (não WKT)

```
CREATE TABLE LOCAL( id INTEGER PRIMARY KEY );
SELECT AddGeometryColumn( '', 'local', 'g_ponto', -1, 'POINT', 2 );
```

```
INSERT INTO LOCAL VALUES (1, ST_MakePoint( 0.916, 1.147 ) );
INSERT INTO LOCAL VALUES (2, ST_MakePoint( 0.906, 1.150 ) );
INSERT INTO LOCAL VALUES (3, ST_MakePoint( 0.900, 1.144 ) );
INSERT INTO LOCAL VALUES (4, ST_MakePoint( 0.909, 1.139 ) );
INSERT INTO LOCAL VALUES (5, ST_MakePoint( 0.916, 1.147 ) );
```



aqui existem 2 pontos sobrepostos
(o que tem id=1 e o que tem id=5)

`ST_MakePoint( x:double, y:double, z:double )`

Aceita valores numéricos

Origina expressões de leitura (e escrita) mais simples.

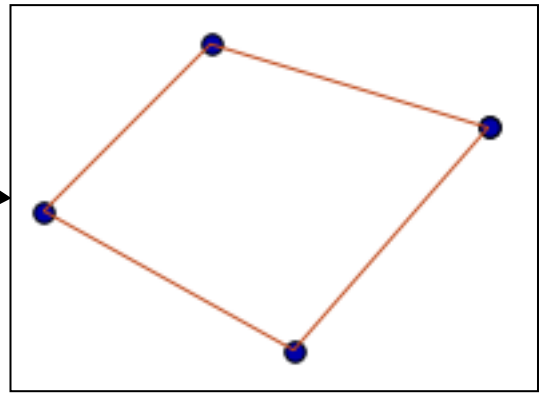
É adequada à utilização de directivas preparadas (*prepared statments*).

Paulo Trigo Silva

Criar Novas Geometrias – ST_Make* (linha)

```
CREATE TABLE CONTORNO( id INTEGER PRIMARY KEY );
SELECT AddGeometryColumn( '', 'contorno', 'g_linha', -1, 'LINESTRING', 2 );
```

```
INSERT INTO CONTORNO( id, g_linha )
SELECT 1, ST_MakeLine( g_ponto )
FROM LOCAL;
```



estão aqui representadas as duas camadas (*layers*):

a camada com os pontos (em LOCAL)

a camada com a linha (em CONTORNO)

```
ST_MakeLine( gs:geometry-set )
ST_MakeLine( p1:point, p2:point )
```

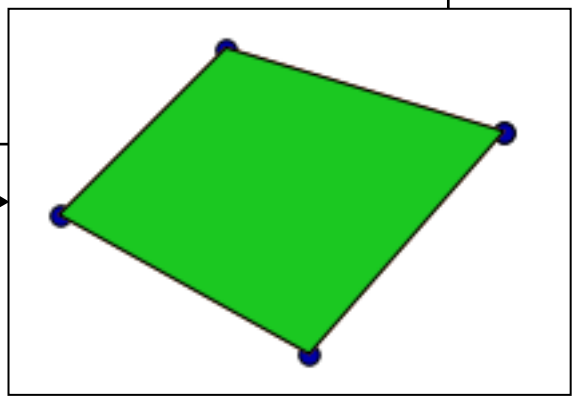
Permite criar uma linha a partir de (um conjunto de) pontos

Paulo Trigo Silva

Criar Novas Geometrias – `ST_Make*` (polígono)

Actualizar o polígono em FREGUESIA com base na linha em CONTORNO

```
UPDATE FREGUESIA
SET g_freguesia =
    ( SELECT ST_MakePolygon( g_linha )
      FROM CONTORNO
      WHERE id = 1 )
WHERE nome = 'Lumiar';
```



estão aqui representadas as três camadas (*layers*):

a camada com os pontos (em LOCAL)

a camada com a linha (em CONTORNO)

a camada com o polígono (em FREGUESIA)

`ST_MakePolygon( g:linestring )`

Permite criar um polígono a partir de um conjunto de uma linha

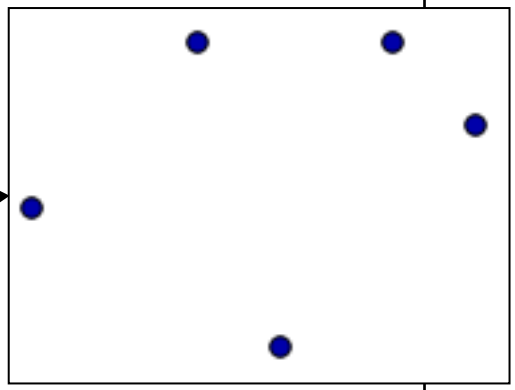
A linha tem que ser definida de modo a ter os extremos coincidentes

Paulo Trigo Silva

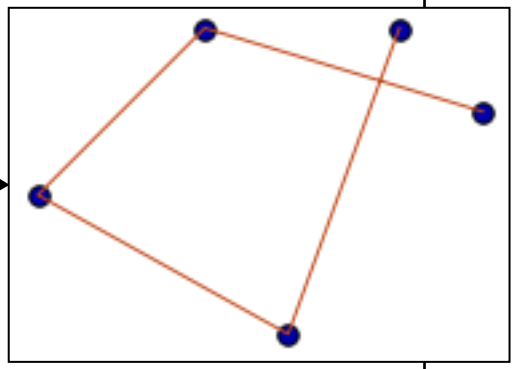
Criação de Novas Geometrias – atenção aos polígonos!

cenário com extremos **não** coincidentes

```
INSERT INTO LOCAL VALUES (1, ST_MakePoint( 0.916, 1.147 ) );
INSERT INTO LOCAL VALUES (2, ST_MakePoint( 0.906, 1.150 ) );
INSERT INTO LOCAL VALUES (3, ST_MakePoint( 0.900, 1.144 ) );
INSERT INTO LOCAL VALUES (4, ST_MakePoint( 0.909, 1.139 ) );
INSERT INTO LOCAL VALUES (5, ST_MakePoint( 0.913, 1.150) );
```



```
INSERT INTO CONTORNO( id, g_linha )
SELECT 1, ST_MakeLine( g_ponto )
FROM LOCAL;
```



```
UPDATE FREGUESIA
SET g_freguesia =
     ( SELECT ST_MakePolygon( g_linha )
       FROM CONTORNO
       WHERE id = 1 )
WHERE nome = 'Lumiar';
```

ERROR:

shell must be closed

Paulo Trigo Silva

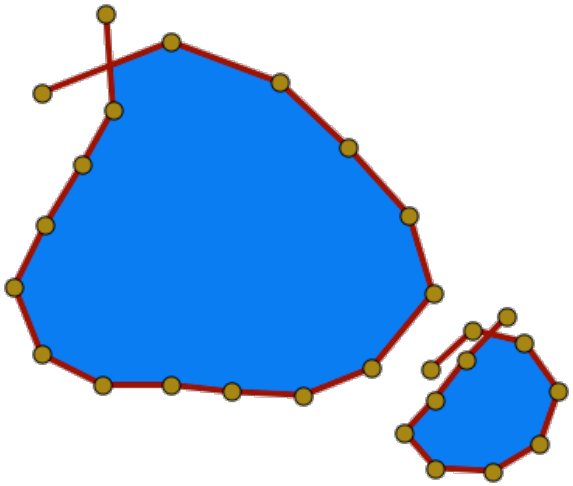
Um Caso Prático

Numa viagem aproveitamos para capturar, com rigor, as coordenadas de alguns locais (terrenos) que nos interessam (e.g., lagoa das 7 cidades!).

Para aumentar o rigor caminhamos, com GPS, pela orla desses terrenos.

+

=



Paulo Trigo Silva

Descrição Completa do Cenário

Admita que tem, numa base de dados, o registo de diferentes tipos de terreno (e.g., lagoa, pântano, cultivo, deserto).

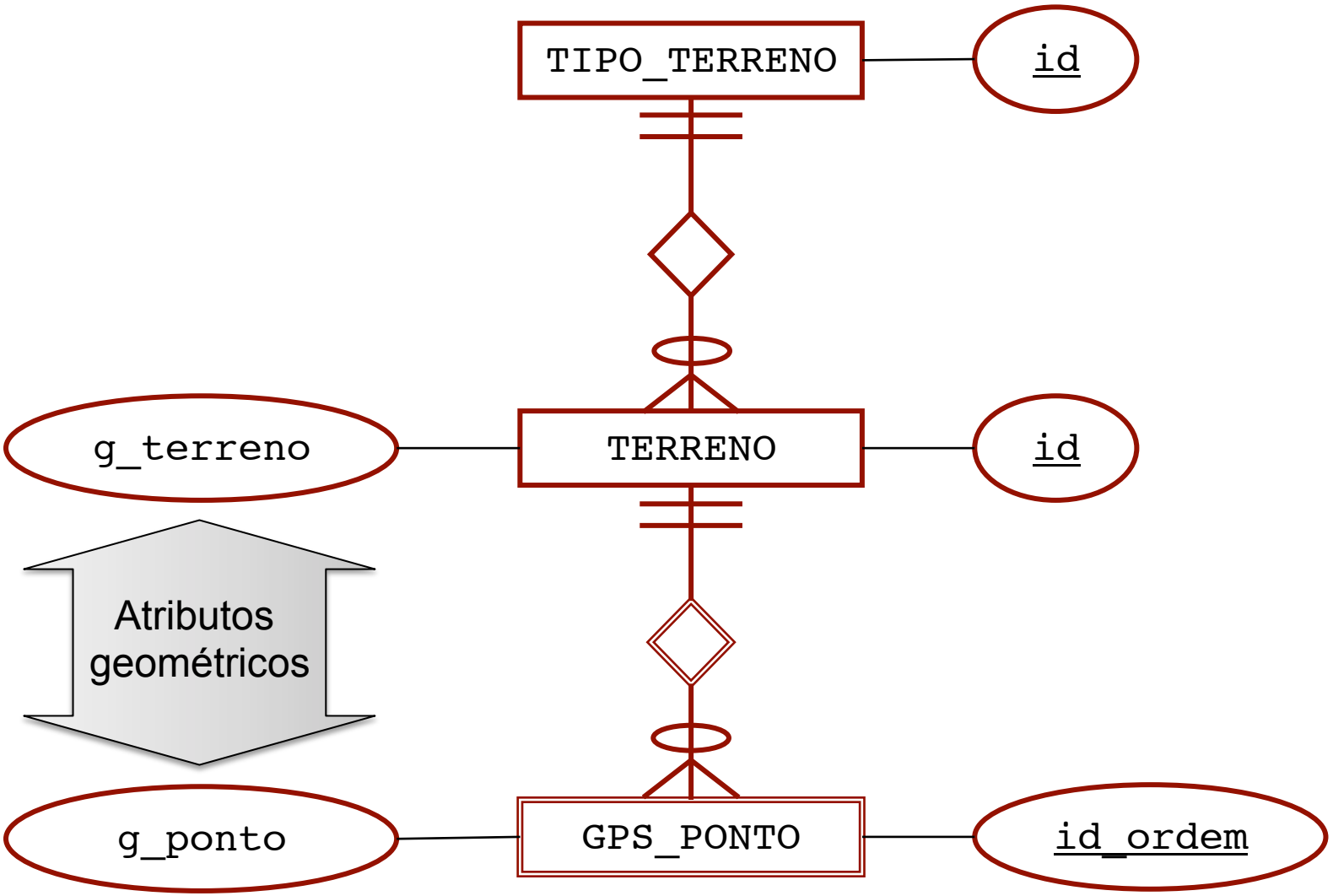
Podem existir vários terrenos de cada tipo sabendo-se que cada que cada terreno pertence a um (e um só) tipo. Cada terreno identifica-se por um identificador único (de valor inteiro) e caracteriza-se pela sua geometria (polígono).

Ao fazer uma viagem pela orla de um terreno vai capturando e registando diversas coordenadas de modo a manter a relação de ordem (pela qual as coordenadas são capturadas). Para simplificar considera-se que a relação de ordem é mantida apenas no contexto de cada terreno.

Construir um modelo conceptual que descreva este cenário.

Paulo Trigo Silva

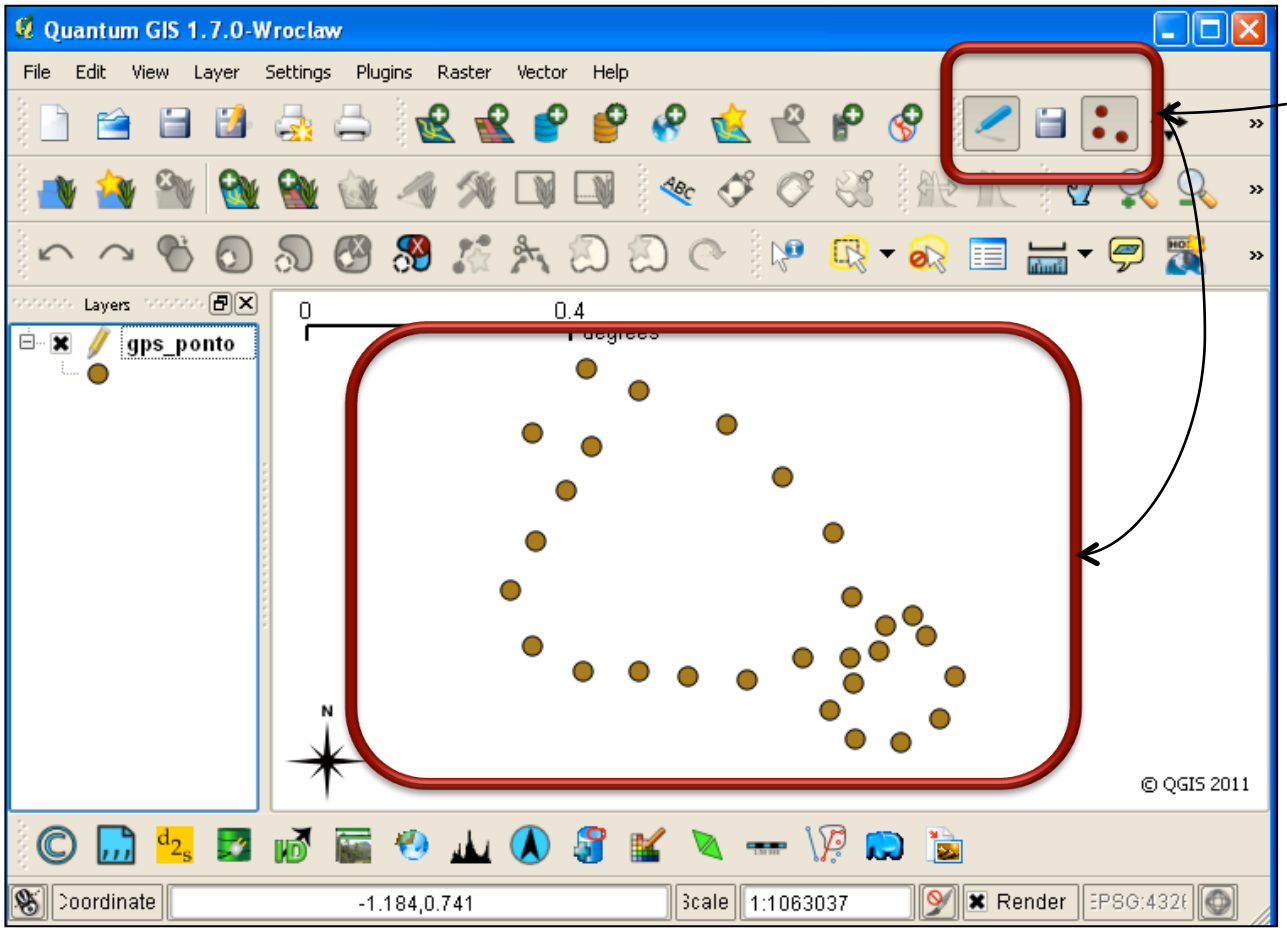
Modelo Conceptual – Entidade Associação



Paulo Trigo Silva

Simular o “caminhar pela orla do terreno”

Usar o QuantunGIS para editar as coordenadas que “estão na orla do terreno” e depois gravar essas coordenadas na base de dados.



Paulo Trigo Silva

... modelo Relacional-Estendido

```
CREATE TABLE tipo_terreno( id_tipo_terreno VARCHAR PRIMARY KEY );
```

```
CREATE TABLE terreno( id_terreno INTEGER PRIMARY KEY, id_tipo_terreno VARCHAR,
            CONSTRAINT fk1 FOREIGN KEY( id_tipo_terreno )
                        REFERENCES tipo_terreno( id_tipo_terreno ) );
SELECT AddGeometryColumn
('', 'terreno', 'g_terreno', -1, 'POLYGON', 2 );
```

```
CREATE TABLE gps_ponto( id_ordem INTEGER, id_terreno INTEGER,
            CONSTRAINT pk_gps_ponto PRIMARY KEY( id_ordem, id_terreno ),
            CONSTRAINT fk1 FOREIGN KEY( id_terreno )
                        REFERENCES terreno( id_terreno ) ) WITH OIDS;
SELECT AddGeometryColumn
('', 'gps_ponto', 'g_ponto', -1, 'POINT', 2 );
```

... povoar o Modelo

```
INSERT INTO tipo_terreno( id_tipo_terreno ) VALUES( 'Lago' );
INSERT INTO tipo_terreno( id_tipo_terreno ) VALUES( 'Pantano' );
```

```
INSERT INTO terreno( id_terreno, id_tipo_terreno ) VALUES( 1, 'Lago' );
INSERT INTO terreno( id_terreno, id_tipo_terreno ) VALUES( 2, 'Lago' );
INSERT INTO terreno( id_terreno, id_tipo_terreno ) VALUES( 3, 'Pantano' );
```

```
INSERT INTO gps_ponto VALUES (1, 1, ST_MakePoint(
                                    -1.21383647798742, 0.534591194968554) );
INSERT INTO gps_ponto VALUES (2, 1, ST_MakePoint(
                                    -1.05450733752621, 0.59748427672956) );
...
INSERT INTO gps_ponto VALUES (1, 2, ST_MakePoint(
                                    -0.733881256233333, 0.194206509128656) );
INSERT INTO gps_ponto VALUES (2, 2, ST_MakePoint(
                                    -0.68217947935112, 0.242078524760334) );
...
```

Obtido via QuantunGIS.

Construir uma nova geometria (linha) a partir dos pontos

Usar `ST_MakeLine` como função agregadora (que aceita conjunto de pontos).

Neste caso notar que a **relação de ordem é muito importante**.

Os pontos são agregados de acordo com a ordem pela qual foram registados.

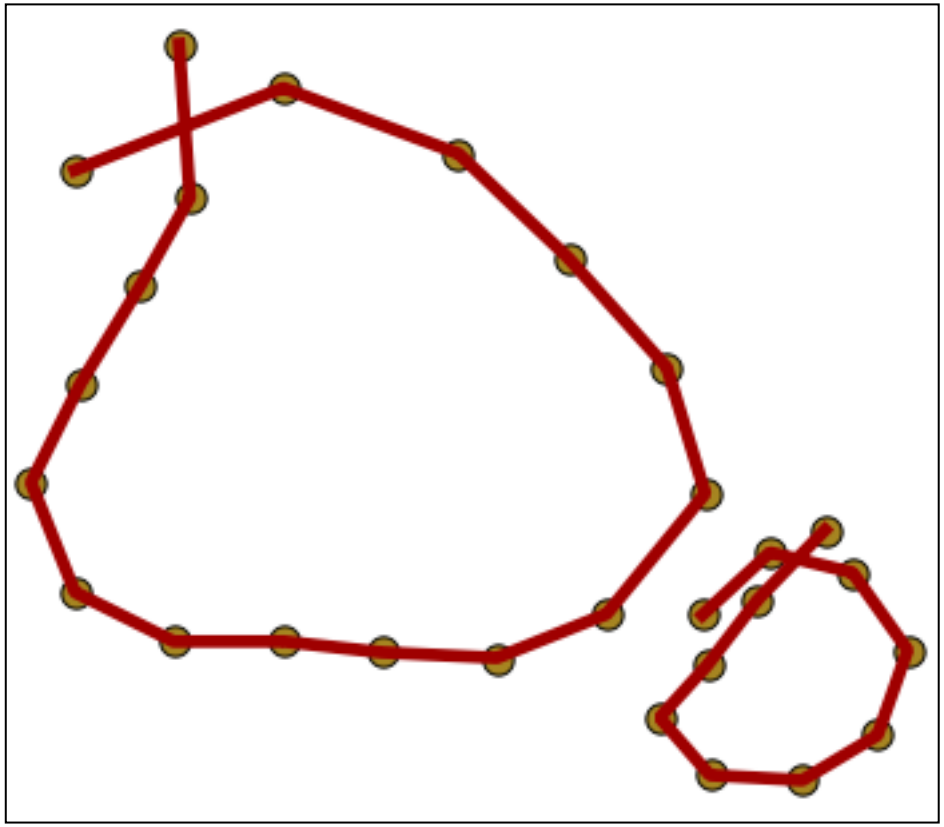
A relação de ordem é imposta, de modo explícito, na sub-interrogação.

Os pontos ordenados são agregados pelo tipo de terreno a que pertencem.

```
CREATE VIEW V_LINHA_CONTORNO( id_terreno, g_linha ) AS


( SELECT ST_MakeLine( g_ponto ), id_terreno
  FROM ( SELECT g_ponto, id_ordem, id_terreno
         FROM gps_ponto
         ORDER BY id_terreno, id_ordem ) AS pontos_ordenados
  GROUP BY id_terreno );
```

Paulo Trigo Silva

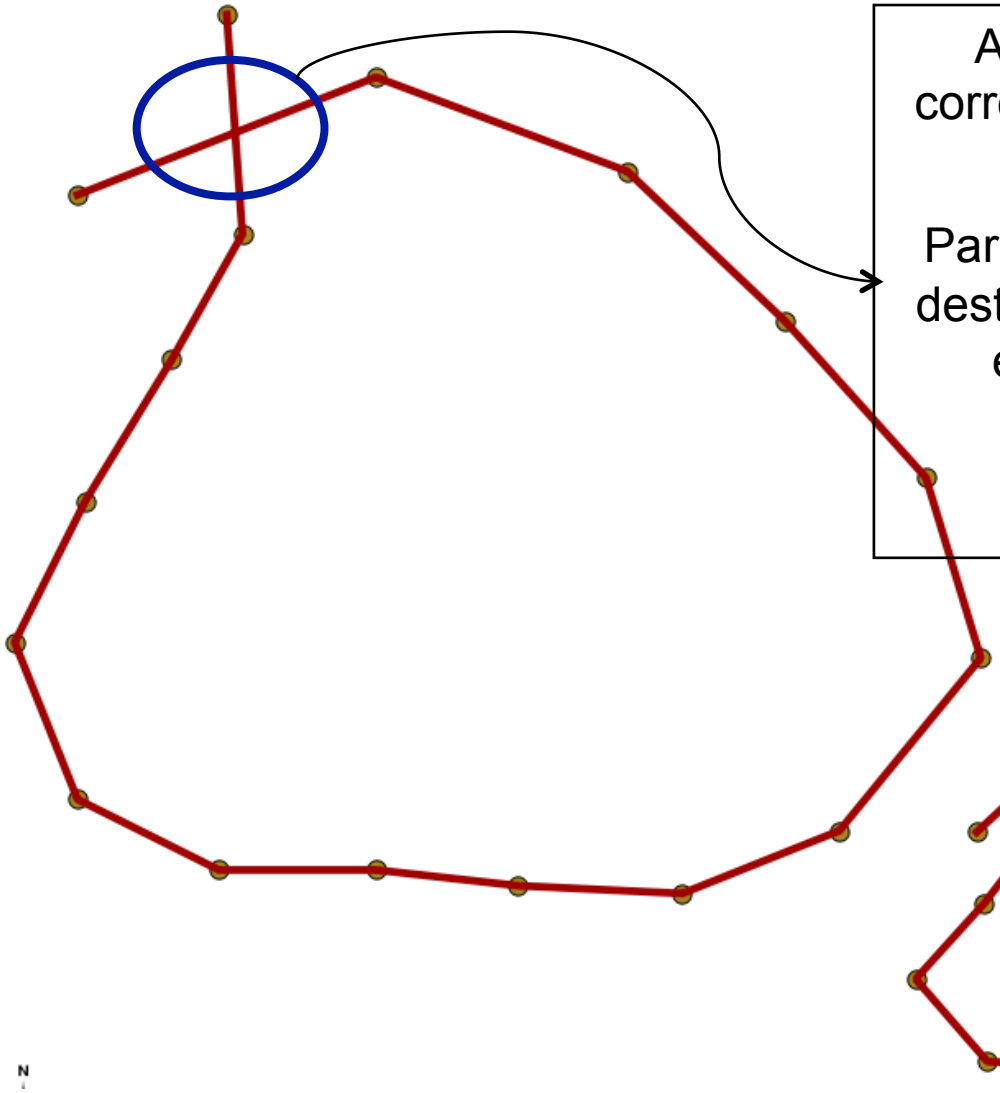
... a nova geometria obtida



`ST_MakeLine( { point } )`

Paulo Trigo Silva

Dificuldade com a nova geometria ...



A linha cruza em local que não corresponde a um ponto usado para descrever a linha.

Para construir um polígono (a partir desta linha) é preciso que os pontos extremos sejam coincidentes.

É necessário “reparar” este cruzamento de linhas!

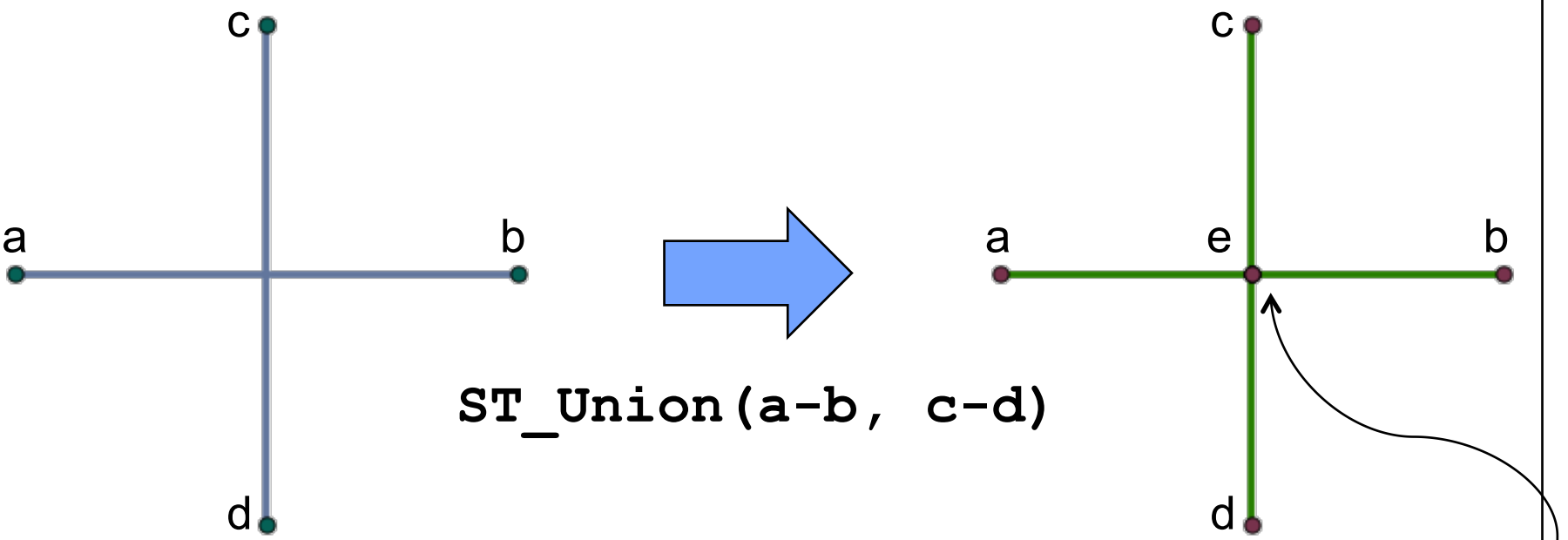
Paulo Trigo Silva

... características do Operador de União (`ST_Union`)

União garante que os interiores têm intersecção vazia.

Apenas as fronteiras, das geometrias resultantes da união, podem ter intersecção não vazia.

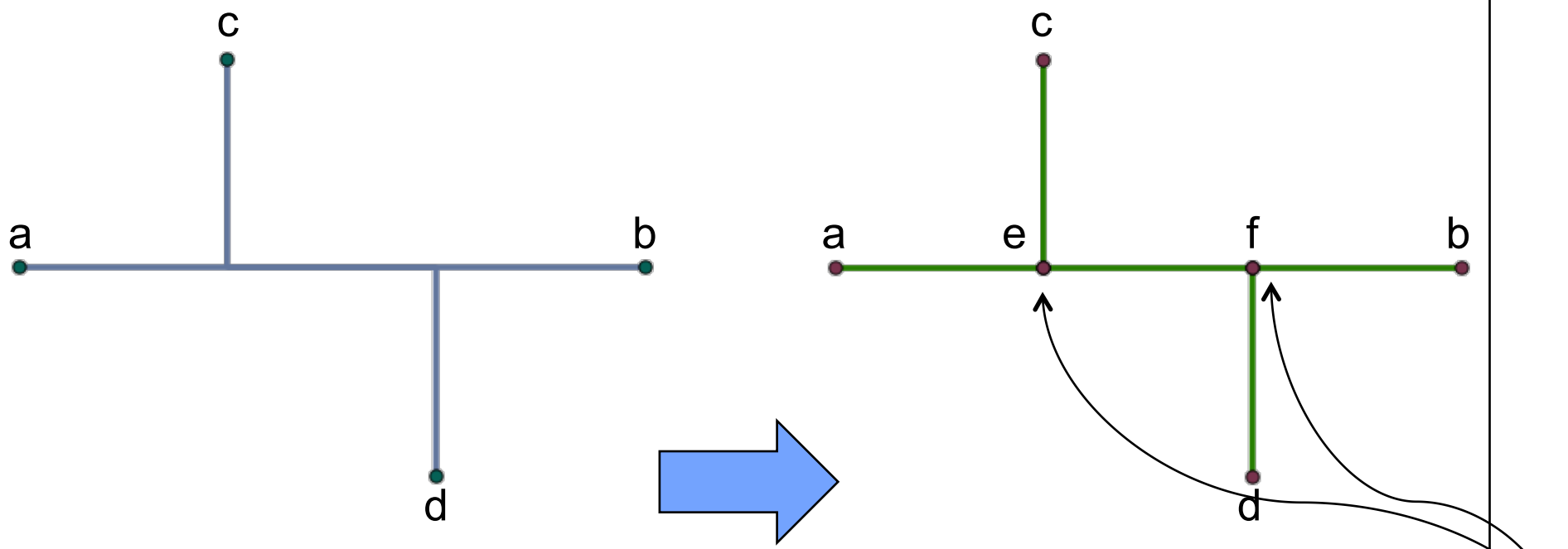
Ou seja, ao fazer a união de geometrias com interiores que se intersectam obtém-se uma geometria onde apenas as fronteiras se intersectam.



Notar que surge este novo ponto (e) pois agora existem 4 segmentos de recta a-e, e-b, c-e, e-d. O novo ponto (e) é fronteira de cada segmento de recta.

Paulo Trigo Silva

... outro exemplo (`ST_Union`)



`ST_Union(a-b, c-d)`

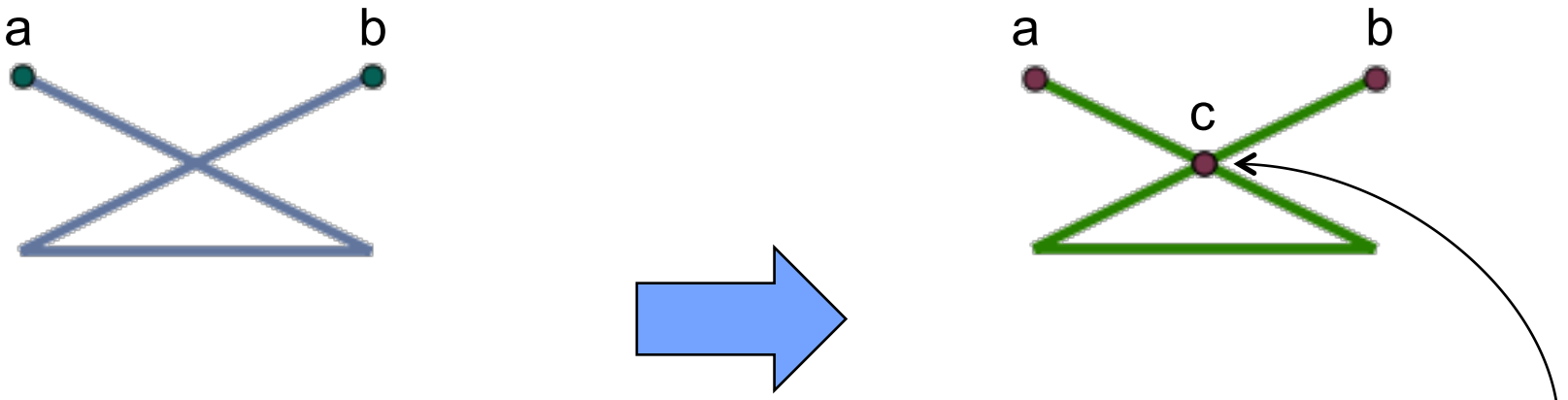
Notar que surgem os novos pontos (e, f) pois agora existem 5 segmentos de recta a-e, e-f, f-b, c-e, f-d. Novos pontos (e, f) são fronteira de cada segmento de recta.

Paulo Trigo Silva

... ST_UnaryUnion (ou com 'LINESTRING EMPTY')

União unária (ou de uma única linha com a linha vazia).

Surge um novo ponto (c) tal que o segmento c-c se pode transformar num polígono, i.e., tem os extremos coincidentes.



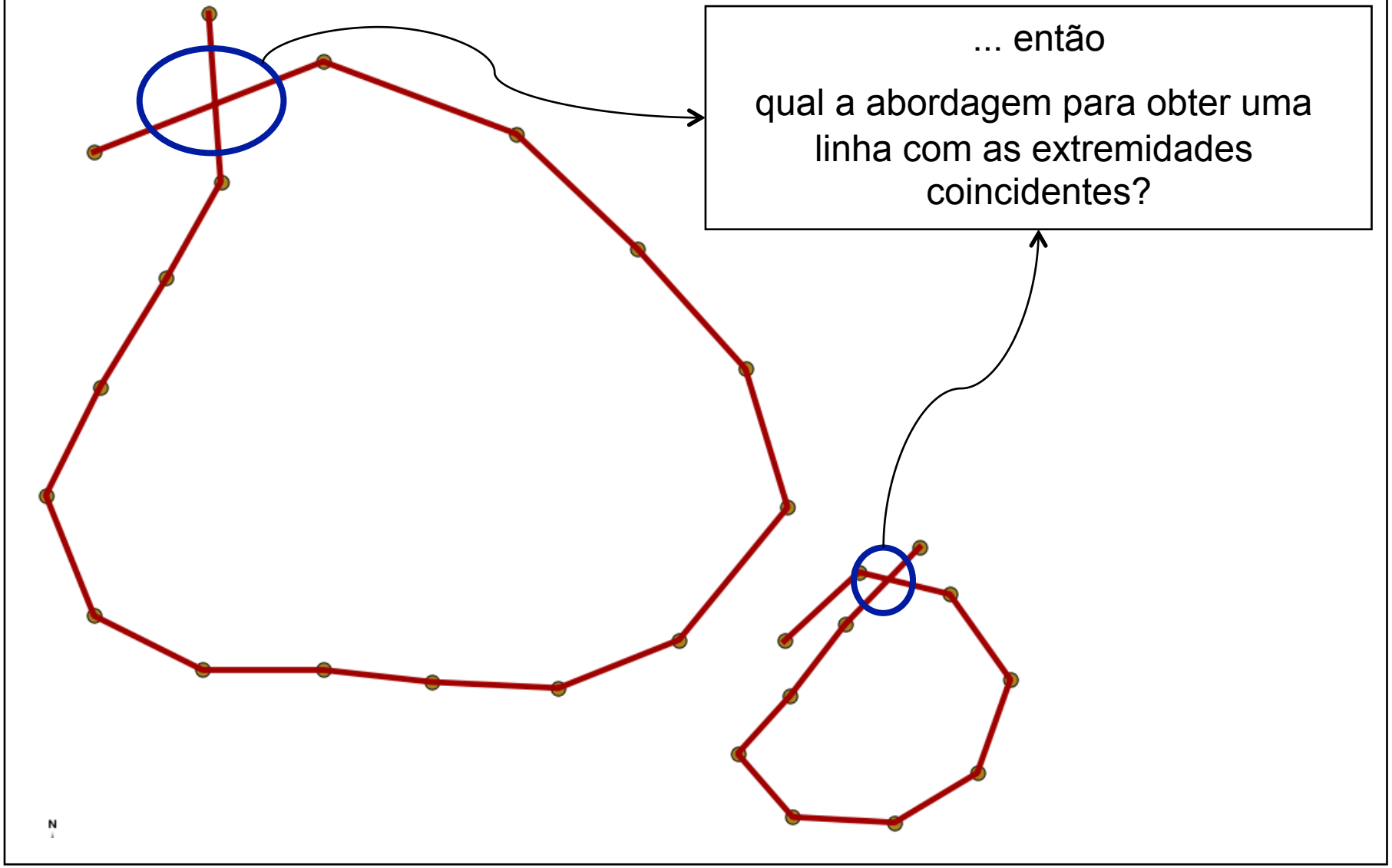
ST_UnaryUnion(a-b)

```
ST_Union(a-b, 'LINESTRING EMPTY') // forma descontinuada nesta versão
```

Notar que surge o novo ponto (c) pois agora existem 3 segmentos de recta a-c, b-c, c-c. O novo ponto (c) é fronteira dos segmentos a-c e b-c.

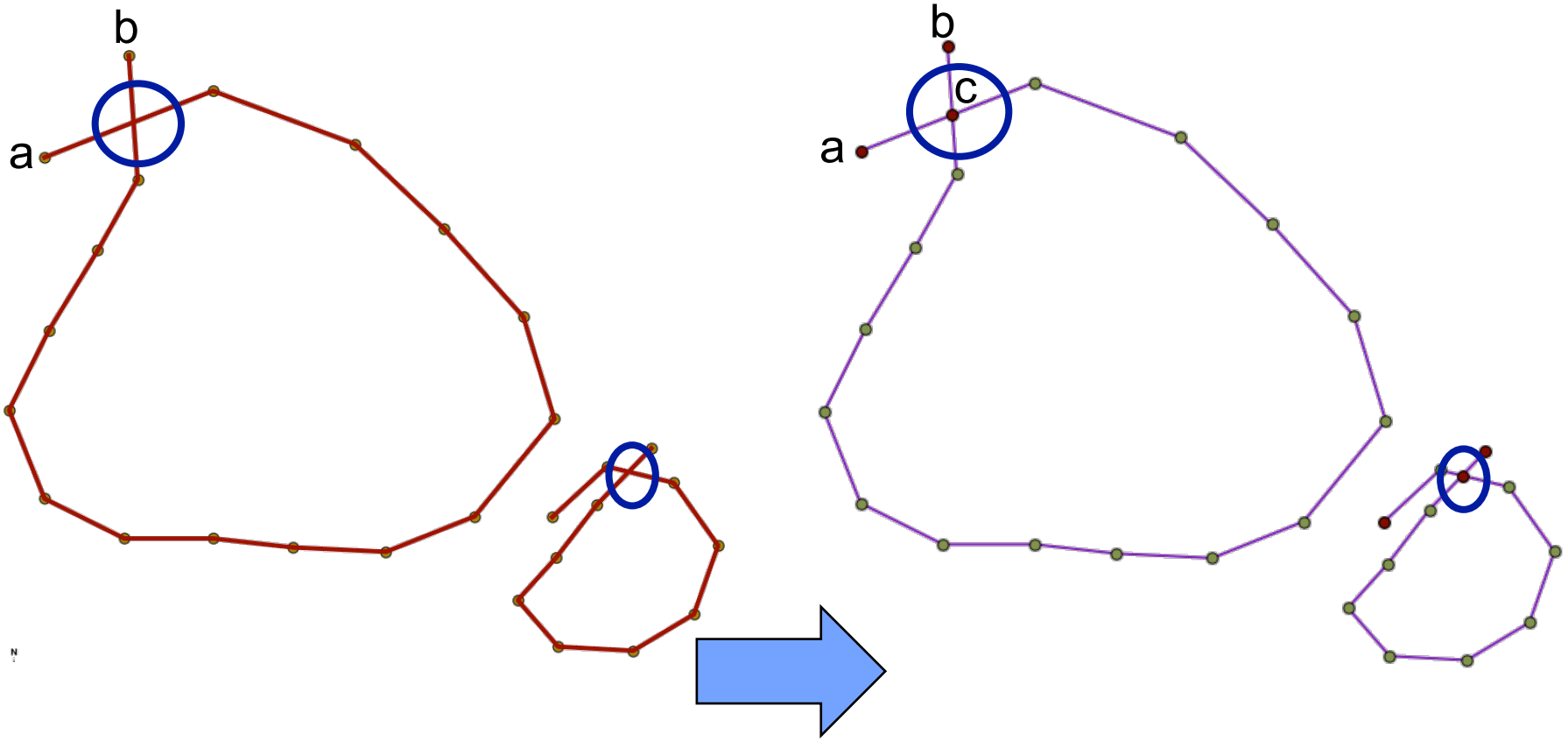
O segmento c-c tem fronteira vazia.

De regresso à dificuldade inicial ...



Paulo Trigo Silva

... obter linha com extremos coincidentes (via `ST_Union`)




```
CREATE VIEW V_LINHA_EXTREMO_COINCIDENTE( id_terreno, g_linha ) AS
SELECT id_terreno,
       ST_UnaryUnion( g_linha ) AS g_linha
FROM V_LINHA_CONTORNO;
```

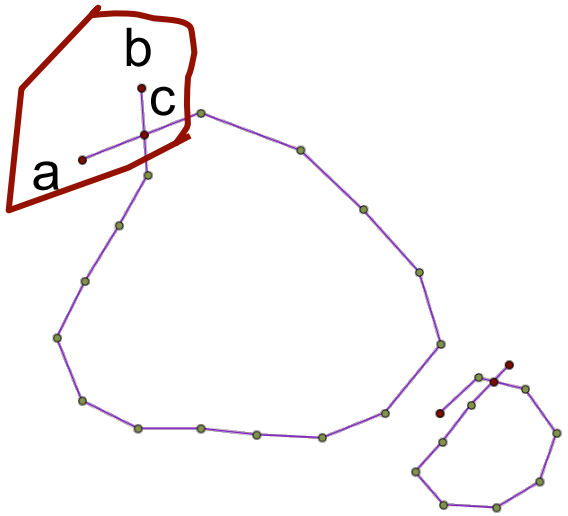
Paulo Trigo Silva

... pormenor adicional

Para obter explicitamente os pontos fronteira de cada segmento é necessário expandir cada 'multilinestring' e obter a sua fronteira.

(posteriormente veremos detalhe sobre a expansão de "multi-geometrias".

Atenção ao caso da fronteira vazia; neste cenário o segmento c-c tem fronteira vazia.




```
SELECT
( CASE WHEN NOT ST_IsEmpty( ST_Boundary( g_linha ) )
       THEN                 ST_Boundary( g_linha ) END )
FROM
( SELECT ST_GeometryN( g_linha,
                       generate_series( 1,
                       ST_NumGeometries( g_linha ) ) ) AS g_linha
  FROM V_LINHA_EXTREMO_COINCIDENTE
 ) AS PontosFronteira;
```

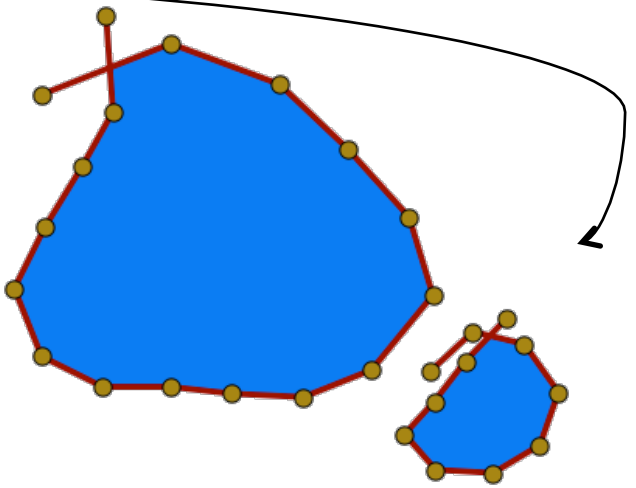
Paulo Trigo Silva

... agora construir o polígono e actualizar a tabela

```
CREATE VIEW V_POLIGONO( id_terreno, g_poligono ) AS
SELECT id_terreno, ST_BuildArea( g_linha )
FROM V_LINHA_EXTREMO_COINCIDENTE;


UPDATE TERRENO
SET g_terreno = ( SELECT g_poligono
                  FROM V_POLIGONO AS VP
                  WHERE VP.id_terreno = TERRENO.id_terreno );
```

+

=



Formas de construir polígonos (sobre outras geometrias)

ST_BuildArea

Aceita uma 'MULTILINESTRING' e constrói o polígono com maior área possível assumindo que existem duas extremidades coincidentes. Os eventuais anéis interiores são respeitados.

Adequada ao cenário atrás exposto.

ST_MakePolygon

Aceita uma 'LINESTRING' com extremidades coincidentes e constrói respectivo o polígono. Tem assinatura para incluir anéis interiores.

Pouco adequado ao cenário atrás exposto pois seria preciso expandir a 'MULTILINESTRING' (resultante de ST_Union) e extrair a 'LINESTRING' de extremos coincidentes (c-c, no exemplo)

ST_Polygonize

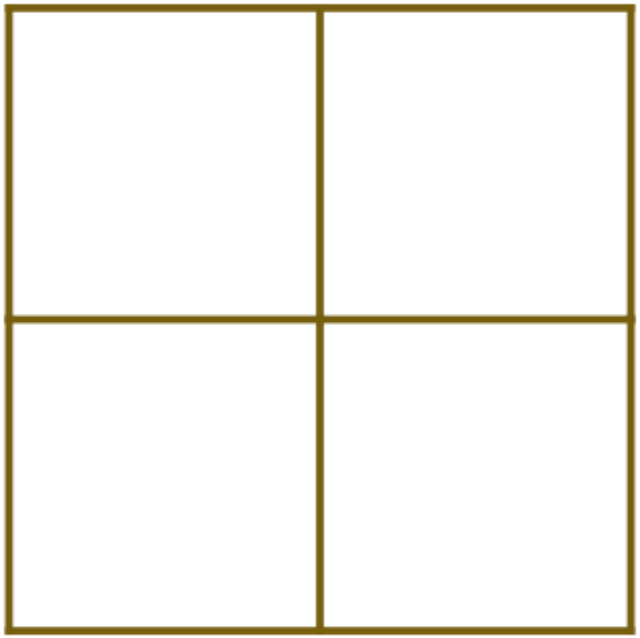
Aceita um conjunto de 'LINESTRING' e constrói uma 'GEOMETRYCOLLECTION' com o maior número de polígonos possível.

Pouco adequado ao cenário pois queremos um polígono por terreno.

Paulo Trigo Silva

Exemplo – ST_BuildArea 'versus' ST_Polygonize

```
CREATE TABLE gis_aresta( id SERIAL PRIMARY KEY );
SELECT AddGeometryColumn
('', 'gis_aresta', 'g_linha', -1,
                   'LINESTRING', 2 );;


INSERT INTO gis_aresta( g_linha )
VALUES(
ST_GeomFromText('LINESTRING(0 0, 1 0)', -1) );

INSERT INTO gis_aresta( g_linha )
VALUES(
ST_GeomFromText('LINESTRING(0 1, 1 1)', -1) );

INSERT INTO gis_aresta( g_linha )
VALUES(
ST_GeomFromText('LINESTRING(0 2, 1 2)', -1) );
...
```



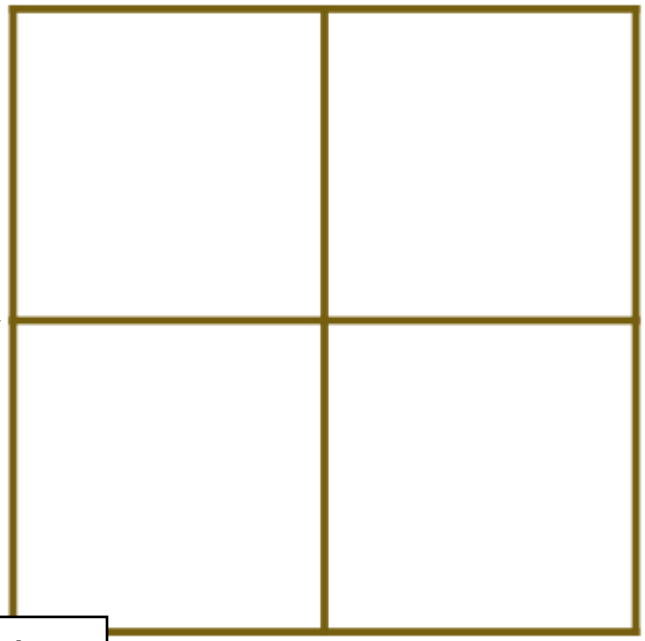
constroem-se 12 'LINESTRING' de modo a ficar com esta figura

Paulo Trigo Silva

... agregam-se aquelas 12 linhas num única multi-linha

```
CREATE TABLE t_collect( id SERIAL PRIMARY KEY );
SELECT AddGeometryColumn
('', 't_collect', 'g_multi_linha', -1, 'MULTILINESTRING', 2 );

INSERT INTO t_collect( g_multi_linha )
( SELECT ST_Collect( g_linha )
  FROM gis_aresta );
```

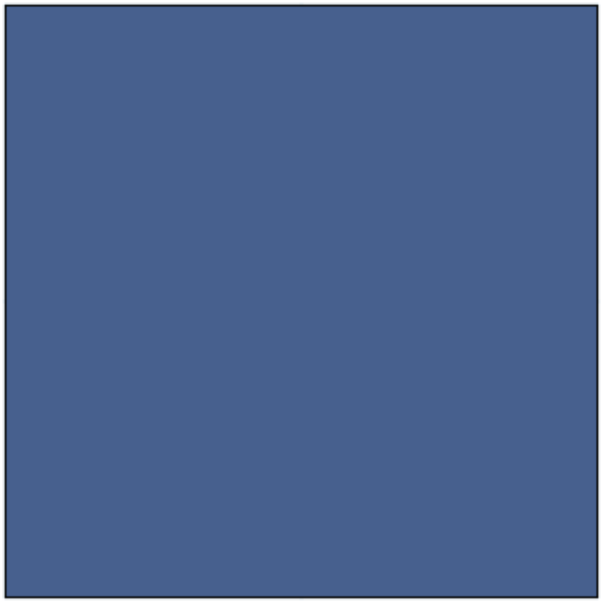
```
SELECT id, ST_AsText( g_multi_linha ) AS ml
FROM t_collect;

id | ml
 1 | MULTILINESTRING((0 0,1 0),
                     (0 1,1 1),
                     (0 2,1 2),
                     (1 0,2 0),
                     (1 1,2 1),
                     (1 2,2 2),
                     (0 0,0 1),
                     ...)
```



tudo colectado numa única 'MULTILINESTRING'

... com o ST_BuilArea usa-se aquela agregação

```
CREATE VIEW V_BUILD_AREA AS
SELECT id, ST_BuildArea( g_multi_linha )
FROM t_collect;
```



ST_BuilArea(ST_Collect(g_linha))

Paulo Trigo Silva

... com o ST_Polygonize usa-se o conjunto de arestas

```
CREATE TABLE t_polygonize( id SERIAL PRIMARY KEY );
SELECT AddGeometryColumn
('', 't_polygonize', 'g_multi_geo', -1, 'GEOMETRYCOLLECTION', 2 );


INSERT INTO t_polygonize( g_multi_pol )
( SELECT ST_Polygonize( g_linha )
  FROM gis_aresta );
```

```
SELECT ST_AsText( g_multi_pol )
FROM t_polygonize;

GEOMETRYCOLLECTION(
  POLYGON((1 0,0 0,0 1,1 1,1 0)),
  POLYGON((1 1,0 1,0 2,1 2,1 1)),
  POLYGON((2 0,1 0,1 1,2 1,2 0)),
  POLYGON((2 1,1 1,1 2,2 2,2 1)))
```

Nota:

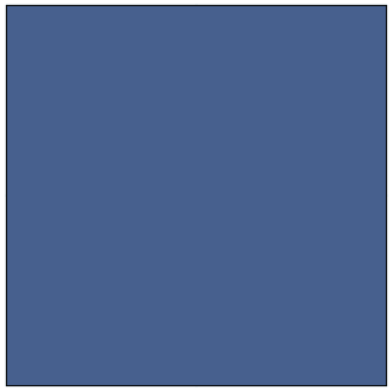
a versão actual do Quantum GIS não suporta apresentação de 'GEOMETRYCOLLECTION'

ST_Polygonize(g_linha)

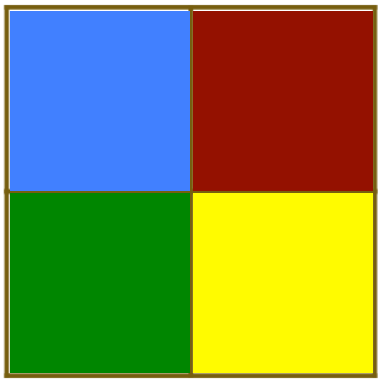
Paulo Trigo Silva

... em resumo

`ST_BuilArea( ST_Collect( g_linha ) )`



`ST_Polygonize( g_linha )`

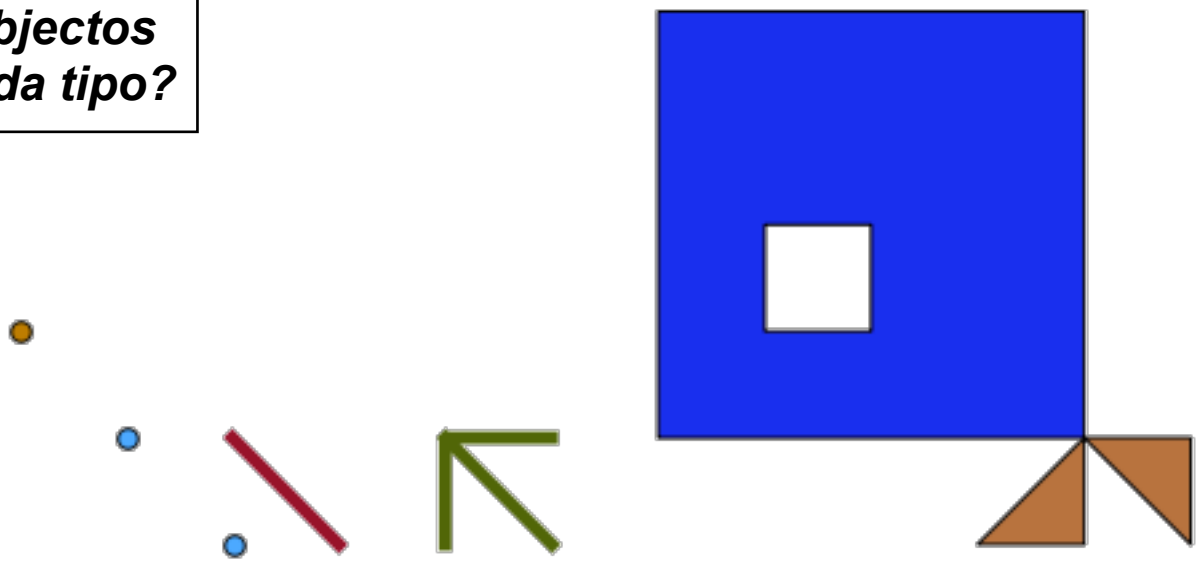


Paulo Trigo Silva

Criar colecção de geometrias – GEOMETRYCOLLECTION

```
CREATE TABLE GEO_COLL( id SERIAL PRIMARY KEY );
SELECT AddGeometryColumn( '', 'geo_coll', 'g_geo_coll', -1, 'GEOMETRYCOLLECTION', 2 );

INSERT INTO GEO_COLL( g_geo_coll ) VALUES (
ST_GeomFromText( 'GEOMETRYCOLLECTION(
                    POINT( -2 2 ),
                    MULTIPOINT( (0 0), (-1 1) ),
                    LINESTRING( 0 1, 1 0 ),
                    MULTILINESTRING( ( 2 0, 2 1, 3 1 ), (2 1, 3 0 ) ),
                    POLYGON( (4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2) ),
                    MULTIPOLYGON( ((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1))))'));
```

Quais destes objectos pertencem a cada tipo?



... e agora, expandir a GEOMETRYCOLLECTION

id	geo
1	**GEOMETRYCOLLECTION**(**POINT**(-2 2), **MULTIPOINT**((0 0), (-1 1)), **LINESTRING**(0 1, 1 0), **MULTILINESTRING**((2 0, 2 1, 3 1), (2 1, 3 0)), **POLYGON**((4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2)), **MULTIPOLYGON**(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1))))

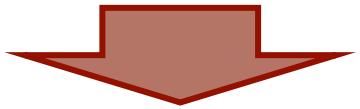
id	geo
1	**POINT**(-2 2)
2	**MULTIPOINT**((0 0), (-1 1))
3	**LINESTRING**(0 1, 1 0)
4	**MULTILINESTRING**((2 0, 2 1, 3 1), (2 1, 3 0))
5	**POLYGON**((4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2))
6	**MULTIPOLYGON**(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1)))

... com ST_GeometryN(*collection, N*)

```
SELECT ST_GeometryN( g_geo_coll, 1 ) AS geo
FROM GEO_COLL;
```

geo
POINT(-2 2)

```
SELECT ST_GeometryN( g_geo_coll, ST_NumGeometries( g_geo_coll ) ) AS geo
FROM GEO_COLL;
```



geo
MULTIPOLYGON(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1)))

Notar que:

```
SELECT ST_NumGeometries( g_geo_coll ) ) FROM GEO_COLL;
>> 6
```

Paulo Trigo Silva

... gerar uma série numérica (útil para `ST_GeometryN`)

```
SELECT S
FROM generate_series( 2, 4 ) AS S;

 S
---
 2
 3
 4
```

```
SELECT generate_series( 5, 1, -2 ) AS S;

 S
---
 5
 3
 1
```

```
SELECT generate_series( 5, 1, -2 )
       + T.S AS X
FROM generate_series( 2, 4 )
     AS T( S );

 X
---
 7
 5
 3
 8
 6
 4
 9
 7
 5
```

```
SELECT current_date + T.A as data
FROM generate_series( 0, 14, 7 ) AS T( A );

 data
------------
 2011-11-07
 2011-11-14
 2011-11-21
```

Expandir toda a GEOMETRYCOLLECTION

```
SELECT id, ST_GeometryN( g_geo_coll,
                         generate_series( 1, ST_NumGeometries( g_geo_coll ) )
           ) AS geo
FROM GEO_COLL;
```

generate_series

devolve um conjunto de tuplos pelo que pode ser usado para gerar um novo tuplo para cada sub-geometria!

id	geo
1	POINT(-2 2)
2	MULTIPOINT((0 0), (-1 1))
3	LINESTRING(0 1, 1 0)
4	MULTILINESTRING((2 0, 2 1, 3 1), (2 1, 3 0))
5	POLYGON((4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2))
6	MULTIPOLYGON(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1)))

Paulo Trigo Silva

Expandir sub-geometrias dentro de sub-geometrias

id	geo
1	**GEOMETRYCOLLECTION(** **POINT**(-2 2), **MULTIPOINT**((0 0), (-1 1)), **LINESTRING**(0 1, 1 0), **MULTILINESTRING**((2 0, 2 1, 3 1), (2 1, 3 0)), **POLYGON**((4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2)), **MULTIPOLYGON**(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1))))

Como usar `generate_series` para:

expandir as sub-geometrias dentro de outra sub-geometria?

Por exemplo, como obter os '`POLYGON`' que estão em '`MULTIPOLYGON`' numa '`GEOMETRYCOLLECTION`'?

id	geo
1	**POLYGON((7 0,8 0,8 1,7 0))**
2	**POLYGON((8 1,9 0,9 1,8 1))**

... obter o tipo de cada sub-geometria – `GeometryType`

```
SELECT GeometryType(
          ST_GeometryN( g_geo_coll,
                        generate_series( 1, ST_NumGeometries( g_geo_coll ) )
          )
       ) AS geo
FROM GEO_COLL;
```

```
  geo
-----------------
 POINT
 MULTIPOINT
 LINESTRING
 MULTILINESTRING
 POLYGON
 MULTIPOLYGON
```

... expandir sub-geometrais dentro de sub-geometrias

```
SELECT ST_AsText(
       ST_GeometryN( geo,
                     generate_series( 1, ST_NumGeometries( geo ) )
                   )
                 ) AS geo_simple
FROM (
     SELECT geo
     FROM (
          SELECT ST_GeometryN( g_geo_coll,
                               generate_series( 1, ST_NumGeometries( g_geo_coll ) )
                 ) AS geo
          FROM GEO_COLL
          ) AS t_geo

     WHERE GeometryType( geo ) = 'MULTIPOLYGON'

     ) AS t_geo_simple;
```

```
 geo_sinple
----------------------------
 POLYGON((7 0,8 0,8 1,7 0))
 POLYGON((8 1,9 0,9 1,8 1))
```

Expandir e obter o '*path*' de cada geometria – ST_Dump

```
SELECT ST_Dump( g_geo_coll )
       AS path_geo
FROM GEO_COLL;
```

```
 path_geo
-------------------------
 ({1},     010100...000)
 ("{2,1}", 010100...000)
 ("{2,2}", 010100...000)
 ({3},     010200...000)
 ("{4,1}", 010200...000)
 ("{4,2}", 010200...000)
 ({5},     010300...000)
 ("{6,1}", 010300...000)
 ("{6,2}", 010300...000)
```

ST_Dump

Devolve um conjunto de tuplos.

Cada tuplo tem uma instância de `geometry_dump`.

Cada `geometry_dump` é composta por:

- `um array` com caminho até geometria,
- a própria geometria.

```
geometry_dump
[
path: array(int)
geom: geometry
]
```

... os constituintes de ST_Dump

```
SELECT              (ST_Dump( g_geo_coll )).path AS path,
         ST_AsText( (ST_Dump( g_geo_coll )).geom ) AS geo
FROM GEO_COLL;
```

```
 path  |                          geo
-------+-----------------------------------------------------
 {1}   | POINT(-2 2)
 {2,1} | POINT(0 0)
 {2,2} | POINT(-1 1)
 {3}   | LINESTRING(0 1,1 0)
 {4,1} | LINESTRING(2 0,2 1,3 1)
 {4,2} | LINESTRING(2 1,3 0)
 {5}   | POLYGON((4 1,8 1,8 5,4 5,4 1),(5 2,6 2,6 3,5 3,5 2))
 {6,1} | POLYGON((7 0,8 0,8 1,7 0))
 {6,2} | POLYGON((8 1,9 0,9 1,8 1))
```

Como aceder, por exemplo, à geometria em [4, 2]?

Ou seja, como aceder à sub-geometria na posição 4 de 'GEOMETRYCOLLECTION' e aí aceder à sub-geometria na posição 2?

Paulo Trigo Silva

... aceder a sub-geometria dentro de sub-geometria

geo
GEOMETRYCOLLECTION(**POINT**(-2 2), **MULTIPOINT**((0 0), (-1 1)), **LINESTRING**(0 1, 1 0), **MULTILINESTRING**((2 0, 2 1, 3 1), (2 1, 3 0)), **POLYGON**((4 1, 8 1, 8 5, 4 5, 4 1), (5 2, 6 2, 6 3, 5 3, 5 2)), **MULTIPOLYGON**(((7 0, 8 0, 8 1, 7 0)), ((8 1, 9 0, 9 1, 8 1))))

```
SELECT p, ST_AsText( geo ) AS geo
FROM (
      SELECT ( ST_Dump( g_geo_coll ) ).path AS p,
             ( ST_Dump( g_geo_coll ) ).geom AS geo
      FROM GEO_COLL
     ) AS T
WHERE p[ 1 ] = 4 AND p[ 2 ] = 1;
```

```
 p   |           geo
-----+-------------------------
 {4,1} | LINESTRING(2 0,2 1,3 1)
```

Detalhe dos cenários atrás expostos (sobre `ST_Union`)

As próximas folhas apresentam o detalhe sobre a forma como foram construídos os cenários (atrás expostos) de exploração de `ST_Union`.

`ST_Union`

(cenários atrás expostos)

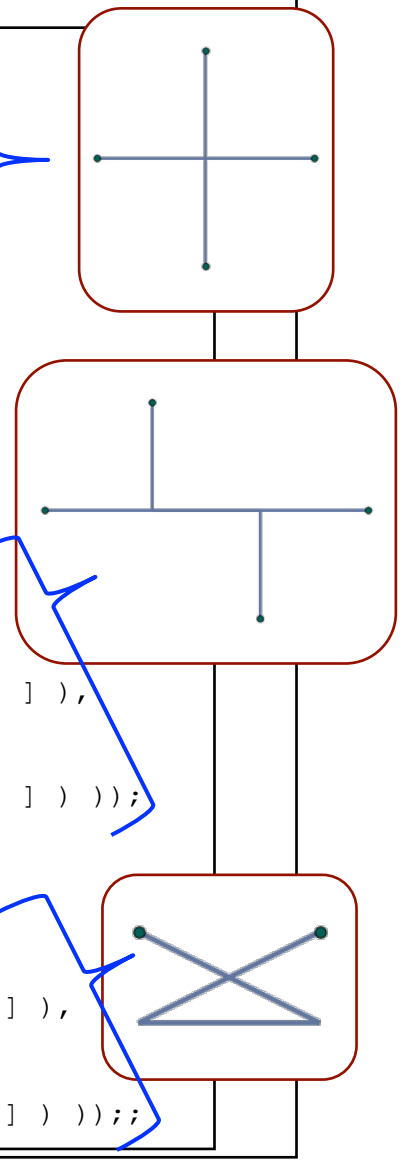
Paulo Trigo Silva

Modelo Relacional-Estendido

```
-- NAS VARIAS TABELAS O ATRIBUTO g_multi_ponto
-- E' SEMPRE USADPO PARA REGISTAR A FRONTEIRA

CREATE TABLE LINHA( id SERIAL PRIMARY KEY );

SELECT AddGeometryColumn( '', 'linha', 'g_linha', -1,
'LINESTRING', 2 );
SELECT AddGeometryColumn( '', 'linha', 'g_multi_ponto', -1,
'MULTIPOINT', 2 );


CREATE TABLE UNIAO_LINHA( id SERIAL PRIMARY KEY );

SELECT AddGeometryColumn( '', 'uniao_linha', 'g_multi_linha', -1,
'MULTILINESTRING', 2 );
SELECT AddGeometryColumn( '', 'uniao_linha', 'g_multi_ponto', -1,
'MULTIPOINT', 2 );


CREATE TABLE UNIAO_LINHA_SEGMENTOS( id SERIAL PRIMARY KEY );

SELECT AddGeometryColumn( '', 'uniao_linha_segmentos', 'g_linha', -1,
'LINESTRING', 2 );
SELECT AddGeometryColumn( '', 'uniao_linha_segmentos', 'g_multi_ponto', -1,
'MULTIPOINT', 2 );
```

Povoar Dados (da tabela `LINHA`)

```
INSERT INTO LINHA( g_linha, g_multi_ponto ) VALUES (
            ST_Makeline( ST_MakePoint( 0.0, 0.0 ), ST_MakePoint( 2.0, 0.0 ) ),
ST_Boundary( ST_Makeline( ST_MakePoint( 0.0, 0.0 ), ST_MakePoint( 2.0, 0.0 ) ) ));
INSERT INTO LINHA( g_linha, g_multi_ponto ) VALUES (
            ST_Makeline( ST_MakePoint( 1.0, -1.0 ), ST_MakePoint( 1.0, 1.0 ) ),
ST_Boundary( ST_Makeline( ST_MakePoint( 1.0, -1.0 ), ST_MakePoint( 1.0, 1.0 ) ) ));
INSERT INTO LINHA( g_linha, g_multi_ponto ) VALUES (
            ST_Makeline( ST_MakePoint( 3.0, 0.0 ), ST_MakePoint( 6.0, 0.0 ) ),
ST_Boundary( ST_Makeline( ST_MakePoint( 3.0, 0.0 ), ST_MakePoint( 6.0, 0.0 ) ) ));

INSERT INTO LINHA( g_linha, g_multi_ponto ) VALUES (
            ST_Makeline( ARRAY[ ST_MakePoint( 4.0, 1.0 ), ST_MakePoint( 4.0, 0.0 ),
                                ST_MakePoint( 5.0, 0.0 ), ST_MakePoint( 5.0, -1.0 ) ] ),
ST_Boundary( ST_Makeline( ARRAY[ ST_MakePoint( 4.0, 1.0 ), ST_MakePoint( 4.0, 0.0 ),
                                ST_MakePoint( 5.0, 0.0 ), ST_MakePoint( 5.0, -1.0 ) ] ) ));

INSERT INTO LINHA( g_linha, g_multi_ponto ) VALUES (
            ST_Makeline( ARRAY[ ST_MakePoint( 9.0, 1.0 ), ST_MakePoint( 7.0, 0.0 ),
                                ST_MakePoint( 9.0, 0.0 ), ST_MakePoint( 7.0, 1.0 ) ] ),
ST_Boundary( ST_Makeline( ARRAY[ ST_MakePoint( 9.0, 1.0 ), ST_MakePoint( 7.0, 0.0 ),
                                ST_MakePoint( 9.0, 0.0 ), ST_MakePoint( 7.0, 1.0 ) ] ) ));;
```



Paulo Trigo Silva

Criar as Uniões entre os Segmentos de Recta

```
INSERT INTO UNIAO_LINHA( g_multi_linha, g_multi_ponto )
SELECT              ST_Union( g_linha ) AS g_linha,
       ST_Boundary( ST_Union( g_linha ) ) AS g_multi_ponto
FROM LINHA
WHERE id = 1 or id = 2;



INSERT INTO UNIAO_LINHA( g_multi_linha, g_multi_ponto )
SELECT              ST_Union( g_linha ) AS g_linha,
       ST_Boundary( ST_Union( g_linha ) ) AS g_multi_ponto
FROM LINHA
WHERE id = 3 or id = 4;


INSERT INTO UNIAO_LINHA( g_multi_linha, g_multi_ponto )
SELECT       ST_UnaryUnion( g_linha ) AS g_linha,
ST_Boundary( ST_UnaryUnion( g_linha ) ) AS g_multi_ponto
FROM LINHA
WHERE id = 5;
```

Paulo Trigo Silva

Expandir os Diversos Segmentos

Expandir os diversos segmentos
(de UNIAO_LINHA para UNIAO_LINHA_SEGMENTOS)

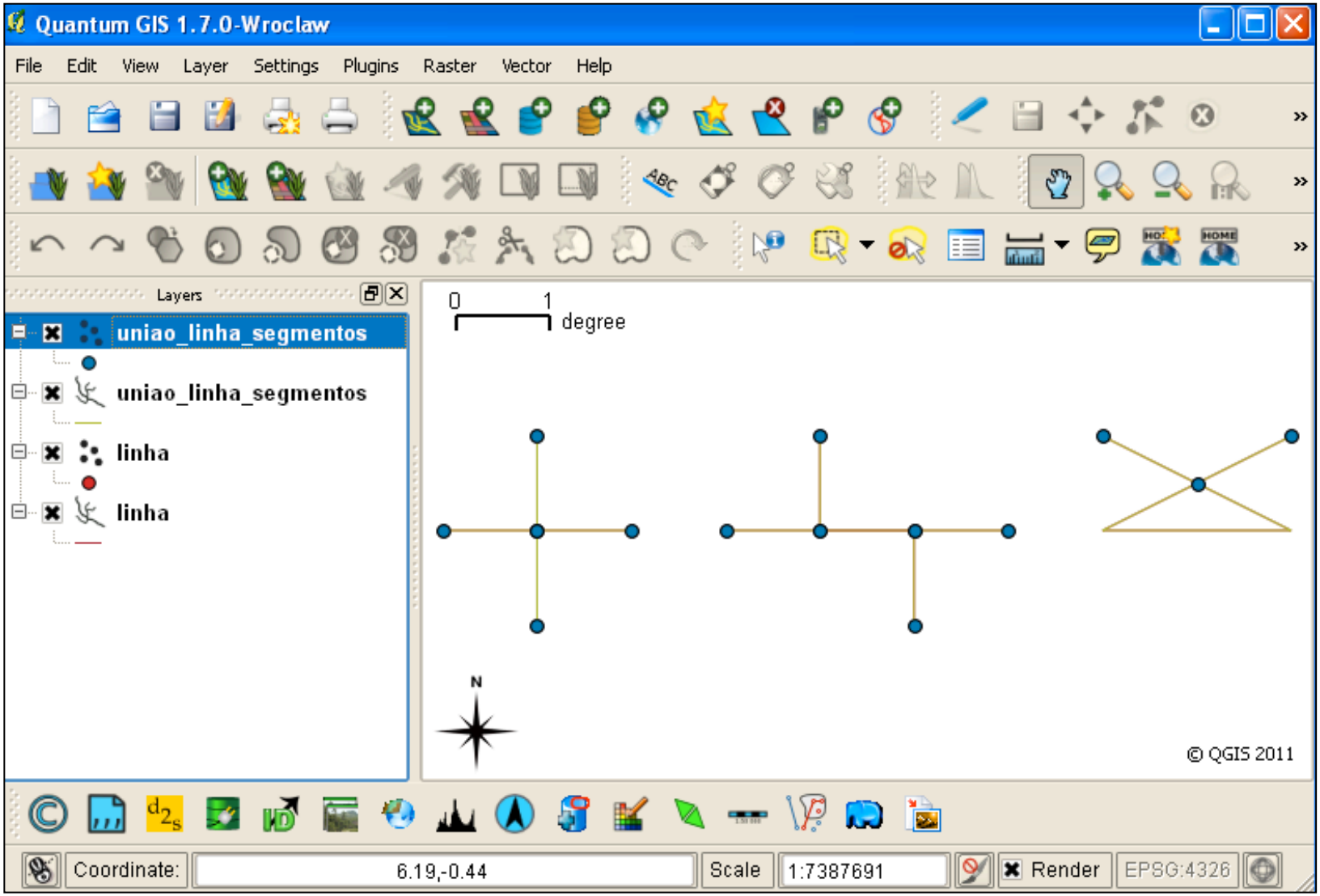
Obter a fronteira de cada segmento.

```
INSERT INTO UNIAO_LINHA_SEGMENTOS( g_linha )
SELECT ST_GeometryN( g_multi_linha, generate_series( 1,
                                        ST_NumGeometries( g_multi_linha ) ) )
                                        AS g_linha
FROM UNIAO_LINHA;



UPDATE UNIAO_LINHA_SEGMENTOS
SET g_multi_ponto = ( CASE WHEN not ST_IsEmpty( ST_Boundary( g_linha ) )
                           THEN                 ST_Boundary( g_linha ) END );
```

Atenção ao caso da fronteira vazia no cenário em que a linha cruza com ela mesmo.

Paulo Trigo Silva

... o resultado (no Quantum GIS)



Paulo Trigo Silva