PostGIS

SGBD espacial
Extensões espaciais para SQL

Clodoveu Davis

Tópicos

- Tipos geométricos no PostGIS
- Tabelas geográficas e alternativas de projeto físico
- Funções geoespaciais em SQL
- Relacionamentos entre geometrias

TIPOS GEOMÉTRICOS

Tipos geométricos

- O PostGIS suporta dois tipos de dados especiais:
  - Geometry
    - Sistema cartesiano “plano”
    - Unidades de medida coerentes com as coordenadas
    - Todas as funções
  - Geography
    - Sistema “esférico”, necessariamente no sistema de projeção WGS84 (EPSG:4326)
    - Medidas em metros
    - Poucas funções, tendência a expansão

Geometry

- Armazena objetos no formato WKT (Well Known Text), especificado pela OGC
  - Existem extensões para a terceira coordenada, chamadas genericamente de XYZ, XYM e XYZM
- A criação de tabelas não permite a inserção imediata de colunas de geometria
  - É necessário criar a tabela convencional primeiro, depois adicionar a coluna geométrica

Geometry

- Exemplo

```
CREATE TABLE munic (
  ID NUMBER,
  NOME VARCHAR(20));
SELECT AddGeometryColumn(‘’,
  ‘munic’, ‘geom’, -1,
  ‘LINESTRING’, 2);
```

  - Observar o formato anômalo desse select, sem cláusula FROM

Geometry

- AddGeometryColumn(
  <schema_name>, --opcional
  <table_name>, --nome tabela
  <column_name>, --coluna geo
  <srid>, --sist. ref. espacial
  <type>, --tipo de geometria
  <dimension> -- 2 ou 3
  );

Geometry

- Metadados
  - Existe uma tabela de metadados denominada geometry_columns para armazenar informação sobre as geometrias em uso

```
CREATE TABLE geometry_columns (
    f_table_catalog VARCHAR(256) NOT NULL,
    f_table_schema VARCHAR(256) NOT NULL,
    f_table_nam VARCHAR(256) NOT NULL,
    f_geometry_column VARCHAR(256) NOT NULL,
    coord_dimension INTEGER NOT NULL,
    srid INTEGER NOT NULL,
    type VARCHAR(30) NOT NULL
)
```

  - A coluna *type* informa o tipo de objeto para toda a coluna
    - Se for interessante restringir a coluna a apenas um tipo, usar o nome
    - Caso contrário, usar 'GEOMETRY'

Geometry

- DropGeometryColumn
  - Deleta a coluna geométrica e metadados correspondentes
- UpdateGeometrySRID
  - Substitui o valor do SRID em todas as linhas e atualiza os metadados correspondentes
- Populate_Geometry_Columns
  - Comando que verifica as tabelas existentes e preenche a tabela geometry_columns
  - A tabela pode ser preenchida manualmente também

Geometry

- Geometrias podem ser criadas a partir de constantes usando a função GeomFromText
- Isso permite a formulação de comandos INSERT

```
INSERT INTO munic (id, nome, geom)
VALUES (1001, 'Belo Horizonte',
GeomFromText('POLYGON(1 2, 2 3, 4 5, 6 7,
  1 2)', 29100);
```

Geometry

- O string passado ao GeomFromText é exatamente a representação WKT
- A partir do WKT, foi especificada a representação WKB (Well Known Binary), mais compacta, que aparece nas colunas geométricas das tabelas PostGIS
- A representação WKT é muito simples, separando vértices com vírgulas e anéis em regiões com parênteses

Geometry

- Exemplos WKT

```
POINT(123 456)
LINESTRING(0 0, 0 1, 1 1, 1 0)
POLYGON((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 1))
MULTIPOINT(0 0, 4 0, 4 4)
MULTILINESTRING((0 0, 1 1, 1 2, 2 2),(3 3, 4 4, 4 6, 6 6))
MULTIPOLYGON((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 1))
GEOMETRYCOLLECTION(POINT(123 456), LINESTRING(0 0, 0 1, 1 1, 1
  0), POINT(4 4))
```

- A representação não inclui o SRID, que vai ser informado no momento da inserção da geometria no banco

Geometry

- Transformações entre WKT e WKB
  - bytea WKB = asBinary(geometry);
  - text WKT = asText(geometry);
  - geometry = GeomFromWKB(bytea WKB, SRID)
  - geometry = GeomFromText(text WKT, SRID)

Geometry

- O PostGIS implementa extensões aos WKT/WKB, que são limitados na definição da OGC a geometrias 2D sem SRID
  - EWKB e EWKT
  - Formas “canônicas”: representações em hexadecimal
    - É o string que retorna após uma consulta à coluna geométrica se não for usado nenhum filtro ou função

Geometry

- EWKT
  - `POINT(0 0 0)  -- ponto 3D`
  - `SRID=29100;POINT(123 300)  -- ponto 2D com SRID`
  - `POINTM(0 0 10)  -- XYM (2,5D)`
  - `POINT(0 0 0 0)  -- XYZM (3,5D?)`
  - Demais tipos incluindo vértices 3D

Geometry

- EWKT e EWKB em comandos INSERT
  - `INSERT INTO pt_cotado (geom, ID) VALUES (GeomFromEWKT(‘SRID=29100;POINTM(-120.2 37.4 82.7)´), 1001);`

Geometry

- Sistemas de referência espaciais
  - Parâmetros armazenados na tabela SPATIAL_REF_SYS

```
CREATE TABLE spatial_ref_sys (
srid INTEGER NOT NULL PRIMARY KEY,
auth_name VARCHAR(256),
auth_srid INTEGER,
srtext VARCHAR(2048),
proj4text VARCHAR(2048)
)
```

  - A coluna *srtext* é especificada em uma forma de WKT para SRS
  - A coluna *proj4text* é usada em uma biblioteca de projeção chamada Proj4

Geometry

- Índices geográficos
  - O PostGreSQL suporta índices em B-Tree, R-Tree e GiST
    - B-Tree: índices convencionais
    - R-Tree: serviria para retângulos envolventes de objetos geográficos, mas a implementação do PostGreSQL não é considerada robusta o suficiente
    - GiST: Generalized Search Tree é uma forma de indexação genérica, que permite uma implementação melhorada da R-Tree
  - `CREATE INDEX nome ON tab USING GIST (geom);`
  - Após a criação de um índice espacial, é interessante forçar o PostGreSQL a coletar estatísticas para levar em conta esse índice na otimização das consultas geográficas
    - `VACUUM ANALYZE table column;`

Geometry

- Existe suporte para geometrias curvas e para 3D, porém o conjunto de funções que trabalham com essas variações é mais limitado, e o suporte a essas geometrias pelo software cliente ainda não é comum

Geography

- O PostGIS inclui um tipo chamado *Geography*, além do já discutido *Geometry*
  - *Geography* é baseado no esferóide, enquanto *Geometry* é baseado em um plano
    - Cálculos sobre o esferóide x cálculos sobre projeções
  - Poucas funções estão disponíveis para *Geography* por enquanto, e o único SRID admitido é o 4326 (WGS84 lon/lat)
  - A unidade de medida em *Geography* é o metro

Geography

- Geography: Não é necessário usar o AddGeometryColumn ou algo parecido

```
CREATE TABLE global_points (
  id SERIAL PRIMARY KEY,
  name VARCHAR(64),
  location GEOGRAPHY(POINT,4326)
);
```

Geography

- Indexação espacial: idem com Geography

```
CREATE INDEX global_points_gix ON global_points
  USING GIST ( location );
```

- Algumas funções de Geography:

```
ST_GeographyFromText('SRID=4326;POINT(-40,-20)');
ST_Area
ST_AsGML
ST_AsKML
ST_AsText
ST_Distance
ST_Covers
ST_CoveredBy
ST_Dwithin
ST_Intersects
ST_Length
```

TABELAS GEOGRÁFICAS

Tabelas Geográficas

- Opções
  - CREATE TABLE seguido de AddGeometryColumn
  - CREATE TABLE contendo GEOGRAPHY
  - Herança em tabelas (table inheritance)

  - Obs: PostGIS 2.0 terá CREATE TABLE contendo GEOMETRY também

Herança em tabelas

- Recurso do PostGreSQL

```
CREATE TABLE estradas(gid serial PRIMARY KEY,
nome_estrada char(100), tipo char(2));
SELECT AddGeometryColumn(‘’,’estradas’, ‘geom’, 4326,
‘LINESTRING’, 2);

CREATE TABLE estradas_pav(CONSTRAINT pkp PRIMARY
KEY(gid)) INHERITS (estradas);
ALTER TABLE estradas_pav
ADD CONSTRAINT pav CHECK (tipo in (‘AS’, ‘CO’));

CREATE TABLE estradas_npv(CONSTRAINT pkn PRIMARY
KEY(gid)) INHERITS (estradas);
ALTER TABLE estradas_npv
ADD CONSTRAINT pav CHECK (tipo in (‘TE’));
```

Visões, regras e gatilhos

- Mecanismos que complementam e expandem as possibilidades de projeto fisico, com mais recursos para implementação de restrições de integridade e flexibilidade estrutural
- Visões (VIEWs)
  - CREATE VIEW big_city as SELECT * FROM city WHERE pop > 1000000
  - big_city se comporta como uma tabela:
    - SELECT * FROM big_city WHERE state = 'MG'
  - Nem sempre:
    - INSERT INTO big_city VALUES (.....) → falha

Visões

- Visões (VIEWs)
  - CREATE VIEW big_city as SELECT * FROM city WHERE pop > 1000000
  - big_city se comporta como uma tabela:
    - SELECT * FROM big_city WHERE state = ‘MG’
  - Nem sempre:
    - INSERT INTO big_city VALUES (…..) → falha
- Visões podem ser complementadas com regras (rewrite rules)

VIEW com regras

```
CREATE OR REPLACE VIEW big_city
AS SELECT * FROM city WHERE pop > 1000000
CREATE OR REPLACE RULE rule_big_city_insert AS
   ON INSERT TO big_city
   DO INSTEAD
      INSERT INTO city(gid, nome, pop, geom)
      VALUES(NEW.gid, NEW.nome, NEW.pop, NEW.geom);
CREATE OR REPLACE RULE rule_big_city_delete AS
   ON DELETE TO big_city
   DO INSTEAD
      DELETE FROM city
      WHERE gid = OLD.gid AND pop > 1000000;
CREATE OR REPLACE RULE rule_big_city_update AS
   ON UPDATE TO big_city
   DO INSTEAD
      UPDATE city
      SET gid = NEW.gid, nome = NEW.nome,
          pop = NEW.pop, geom = NEW.geom
      WHERE gid = OLD.gid AND pop > 1000000
```

Triggers

- Ativados antes ou depois de INSERTs, DELETEs ou UPDATEs
- Exemplo (há muitas variações, que podem receber programação complexa):

```
CREATE TRIGGER trigger1
BEFORE INSERT ON city FOR EACH ROW
EXECUTE PROCEDURE check_insert();
```

Restrições de Integridade

- Com CHECKs, rewrite rules e triggers, pode-se implementar todo tipo de restrição de integridade no PostGIS
- As restrições derivadas diretamente da modelagem não são suportadas como definidas conceitualmente, é necessário adaptar os mecanismos a cada caso
- Tabelas geográficas criadas para PostGIS podem vir acompanhadas de dois tipos simples de restrição de integridade

Restrições de Integridade

- Tabelas geográficas criadas para PostGIS podem vir acompanhadas de alguns tipos simples de restrição de integridade
  - Verificação do número de dimensões

```
CONSTRAINT enforce_dims_geom CHECK
(st_ndims(geom) = 2)
```

  - Verificação do tipo de geometria

```
CONSTRAINT enforce_geotype_geom CHECK
(geometrytype(geom) = 'POINT'::text OR
geom IS NULL)
```

  - Verificação do sistema de coordenadas e projeção

```
CONSTRAINT enforce_srid_geom CHECK
(st_srid(geom) = 4326)
```

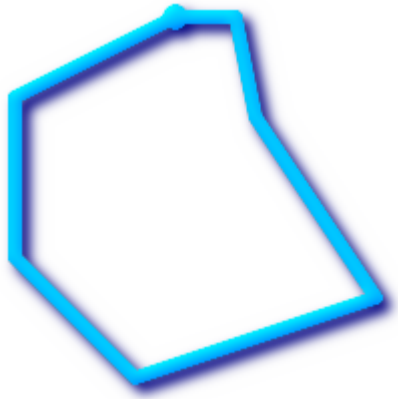
Restrições de Integridade

- Verificação de geometrias: o PostGIS, de acordo com o especificado pelo OGC, exige que as geometrias sejam *simples* e *válidas*
  - Simples: geometria que não tem pontos anômalos, como autointerseções, autotangências ou coincidências indesejadas
    - Aplicável a pontos e linhas
  - Válida: geometria (polígono) em que os anéis exteriores contém inteiramente os anéis interiores (a interseção, se houver, só pode ser em um ponto)

PostGIS

(a)

(b)



(c)

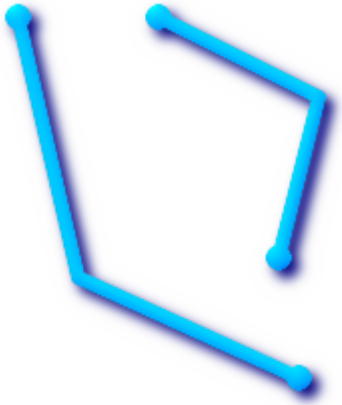
(d)

LINESTRINGS

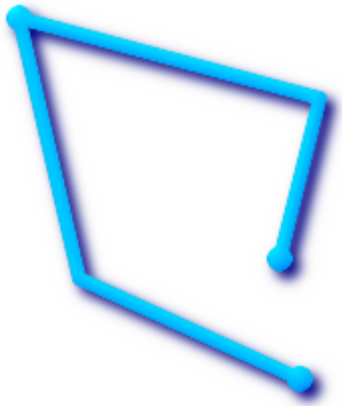
(a) e (c) são simples, (b) e (d) não são

(c) é chamado de anel linear (linear ring); linear rings têm que ter mais de 2 vértices (o último coincide com o primeiro, deve existir pelo menos mais um)

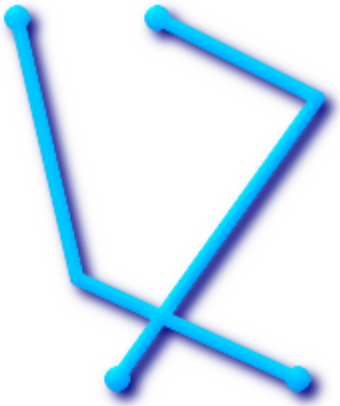
PostGIS



(a)



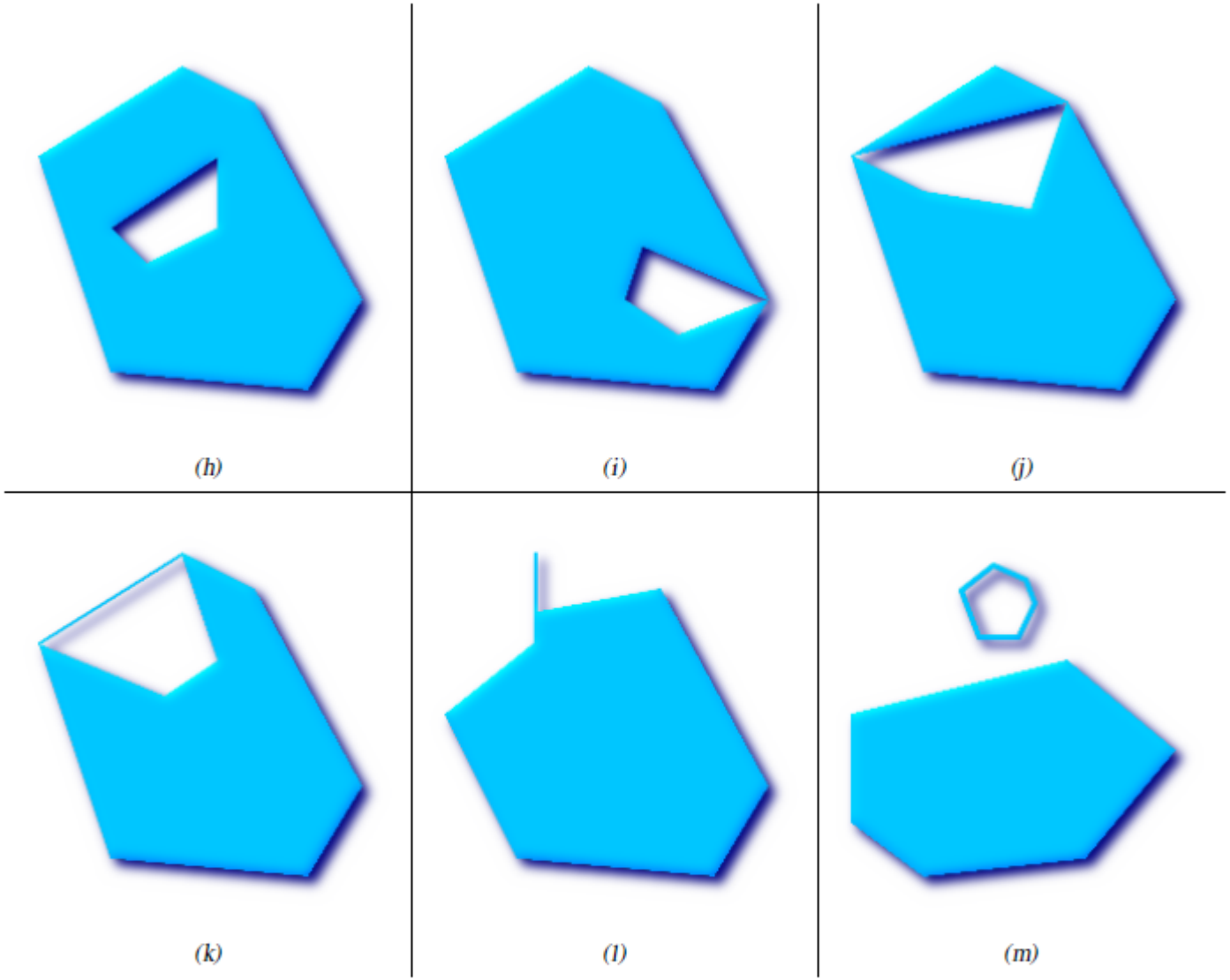
(b)



(c)

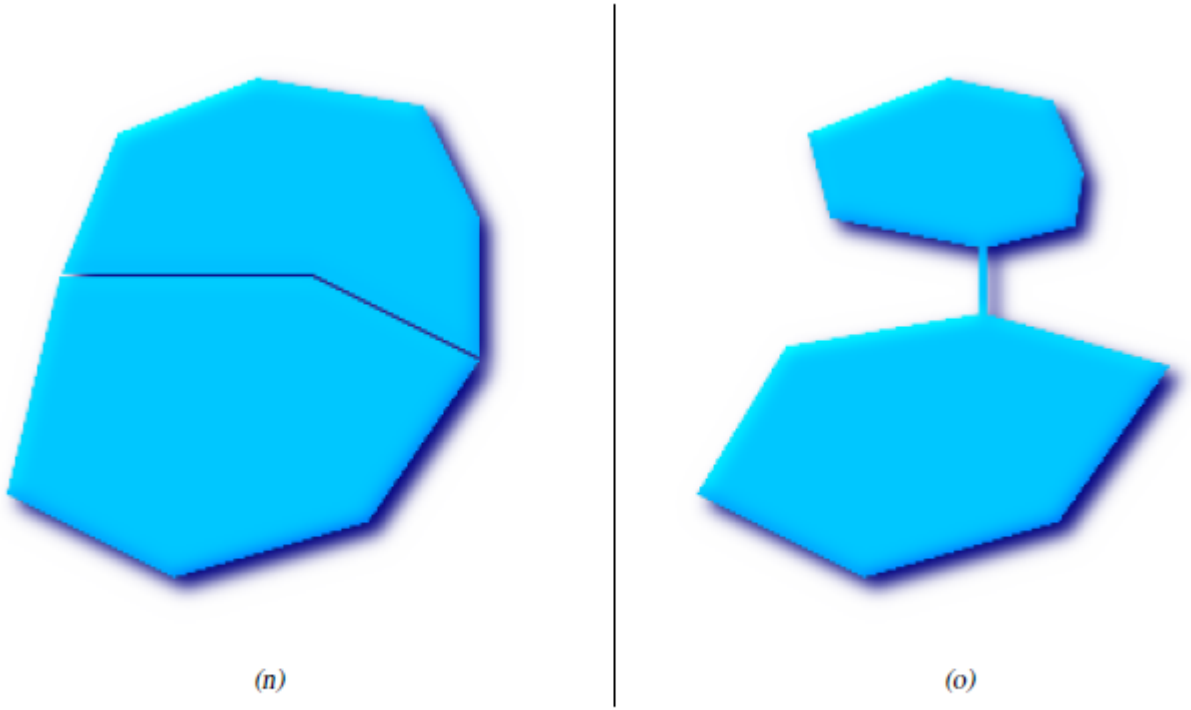
MULTILINESTRINGS

(a) e (b) são simples, (c) não é



(h) and **(i)** are valid `POLYGON`s, **(j-m)** cannot be represented as single `POLYGON`s, but **(j)** and **(m)** could be represented as a valid `MULTIPOLYGON`.

PostGIS



(**n**) and (**o**) are not valid `MULTIPOLYGON`s.

MULTIPOLYGONS só são válidos se os interiores não tiverem interseção. As fronteiras podem ter interseção, mas apenas em pontos

Restrições de Integridade

- Verificação da validade de uma geometria
  - SELECT IsValid(geometria)
  - SELECT IsSimple(geometria)
- O PostGIS não aplica esses testes na entrada de cada geometria por default, pois isso pode demandar muito tempo de CPU
- Essa função não valida geometrias 3D (não OGC)
- Implementação:
  - `ALTER TABLE tab ADD CONSTRAINT nome CHECK (IsValid(geom));`
  - Incorporar no código de um trigger

FUNÇÕES GEOESPACIAIS EM SQL

Grupos de funções

- Construção
- Geração de saída
- Manipulação da geometria
- Medição
- Decomposição
- Composição
- Simplificação
- Testes topológicos

Construção

- ST_GeomFromText
- ST_GeomFromEWKT
- ST_GeomFromWKB
- St_GeomFromEWKB

Geração de saída

- ST_AsText
- ST_AsEWKT
- ST_AsBinary
- ST_AsEWKB
- ST_AsKML
- ST_AsGML
- ST_AsGeoJSON
- ST_AsSVG
- ST_Geohash

Manipulação da geometria

- Sistema de referência espacial
  - ST_SetSRID
  - ST_SRID
  - ST_Transform
- Tipo de geometria e dimensões espaciais
  - ST_GeometryType
  - ST_CoordDim
  - ST_Dimension
- Teste de validade
  - ST_IsValid
  - ST_IsValidReason
  - PostGIS 2.0: ST_IsValidDetail e ST_MakeValid
- Composição
  - ST_NPoints

Medição

- ST_Area(geom)
- ST_Length(geom)
- ST_Length3D(geom)
- ST_Length_Spheroid(geom, spheroid)
- ST_Distance(geom, geom)
- ST_Dwithin(geom, geom, distance)

Distâncias

- ST_Distance_Sphere(point, point)
  - Distância geodésica aproximada, considerando o planeta como se fosse esférico
- ST_Distance_Spheroid(point, point, spheroid)
  - Distância linear entre dois pontos lat/lon para um esferóide em particular, dado em um string:
    - 'SPHEROID["<NAME>",<SEMI-MAJOR AXIS>,<INVERSE FLATTENING>]'
- Uma alternativa é usar reprojeção (conversão para outro sistema de coordenadas, e cálculo de distância euclidiana)
  - ST_Transform(geom, novoSRID)

Decomposição

- ST_Box2D
- ST_Envelope
- ST_X
- ST_Y
- ST_Boundary
- ST_ExteriorRing
- ST_Centroid
- ST_PointOnSurface
- ST_PointN
- ST_GeometryN
- ST_Dump
- ST_DumpPoints
- ST_DumpRings

Composição

- ST_Point
- ST_MakePoint
- ST_MakePolygon
- ST_BuildArea
- ST_Polygonize
- ST_Buffer

Composição

- ST_MakePolygon(linestring [,linestring[]])
  - Cria um polígono a partir de várias linhas
- ST_BuildArea(geom)
  - Cria um polígono a partir de um objeto de linha
- ST_Polygonize(geomSet)
  - Cria uma GeometryCollection
- ST_Collect(geomSet) / ST_Collect(geom, geom)
  - Retorna uma GeometryCollection ou um objeto Multi

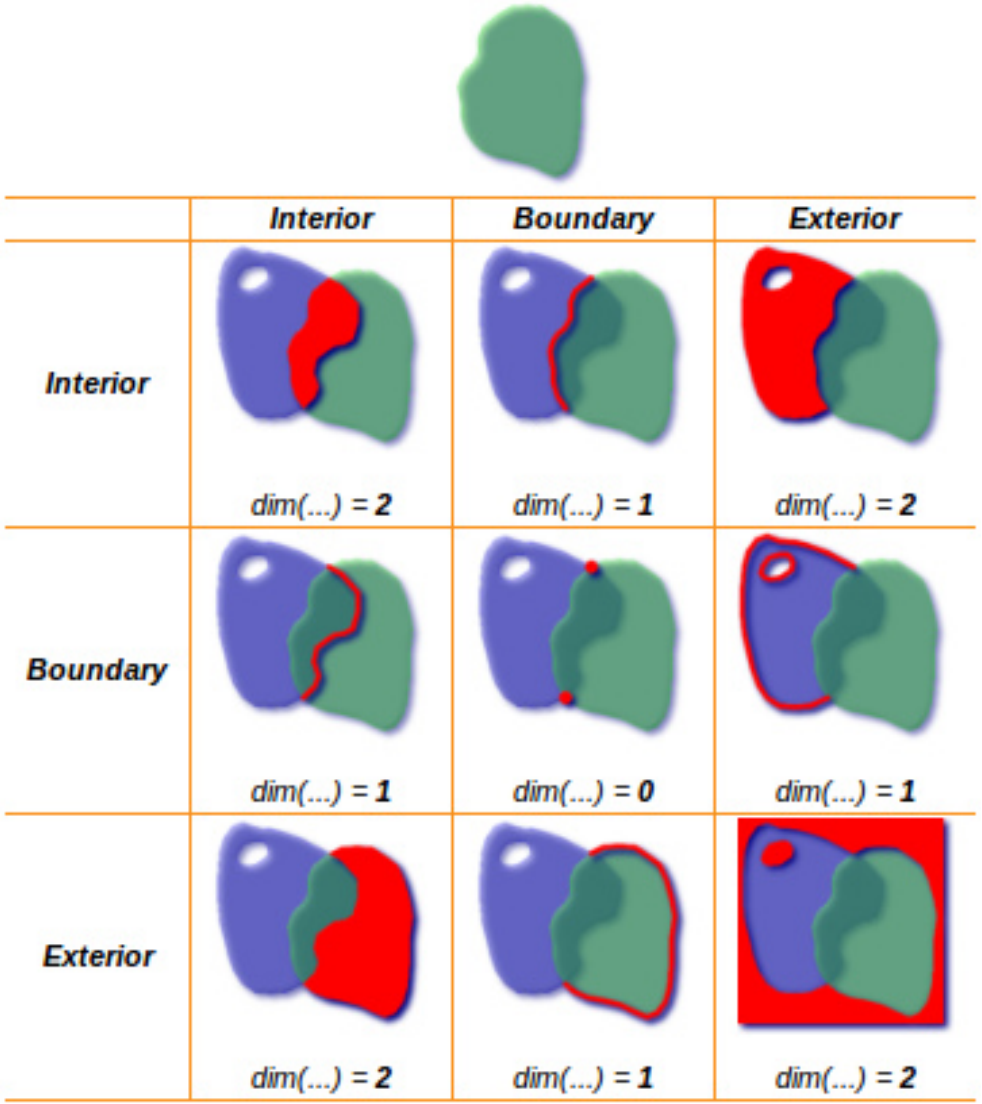
Simplificação

- ST_Simplify
- ST_SimplifyPreserveTopology
- ST_SnapToGrid

Funções Topológicas

- Nomes ligeiramente diferentes da teoria de Egenhofer, mas aderentes aos padrões OpenGIS e SQL-MM
- Usam a matriz de 9 interseções (ordem: interior, fronteira, exterior), numa versao estendida, chamada de **DE-9IM** (*Dimensionally extended 9-intersection matrix*)
- As posições da matriz são preenchidas com
  - 0 -> interseção em ponto (0D)
  - 1 -> interseção em linha (1D)
  - 2 -> interseção em polígono (2D)
  - T -> interseção em ponto, linha ou polígono
  - F -> sem interseção
  - * -> indiferente (em testes)

- Exemplo
- Matriz = 212101212 (leitura esq->dir, top->bottom)



PostGIS

- Exemplo
- Matriz = 1*1***1*

Funções Topológicas

- ST_Equals(geom, geom)
- ST_Disjoint(geom, geom)
- ST_Intersects(geom, geom)
  - _ST_Intersects(geom, geom): idem, mas evita o uso do índice
  - == NOT disjoint(geom, geom) (ANYINTERSECT do Oracle)
- ST_Touches(geom, geom)
- ST_Crosses(geom, geom)
- ST_Within(geom, geom)
- ST_Overlaps(geom, geom)
- ST_Contains(geom, geom)
- ST_Covers(geom, geom)
- ST_CoveredBy(geom, geom)
- ST_Relate(geom, geom, intPatternMatrix)
- ST_Relate(geom, geom)
  - Retorna a DE-9IM (dimensionally extended 9-intersection matrix)

Funções topológicas

- ST_Equals(geom1, geom2)
  - Retorna TRUE se as geometrias forem “espacialmente equivalentes”
- ST_Disjoint(geom1, geom2)
  - Retorna TRUE se as geometrias forem disjuntas
- ST_Intersects(geom1, geom2)
  - Retorna TRUE se as geometrias tiverem qualquer ponto de interseção
  - Usar _ST_Intersects para evitar o uso do índice espacial
  - Intersects == NOT disjoint

Funções topológicas

- ST_Touches(geom1, geom2)
  - Retorna TRUE se as geometrias se tocarem (4IM): interseção entre fronteiras é não vazia, interseção dos interiores é vazia
- ST_Crosses(geom1, geom2)
  - Retorna TRUE se as geometrias se cruzam, ou seja, se elas tiverem alguns pontos do interior em comum, mas não todos
  - Para evitar o uso do índice, usar _ST_Crosses
  - DE-9IM: T*T****** para ponto/linha, ponto/polígono e linha/polígono
  - DE-9IM: T*****T** para linha/ponto, polígono/ponto e polígono/linha
  - DE-9IM: 0******** para linha/linha

$$a.Crosses(b) \Leftrightarrow (dim(I(a) \cap I(b)) < max(dim(I(a)), dim(I(b)))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

Funções topológicas

- ST_Within(geom1, geom2)
  - Retorna TRUE se a geom1 está totalmente dentro da geom2
  - Para evitar o uso do índice, usar _ST_Within
  - DE-9IM: T*F**F***
- ST_Overlaps(geom1, geom2)
  - Retorna TRUE se as geometrias compartilharem algum espaço, forem da mesma dimensão, e uma não contiver inteiramente a outra
  - Não é o mesmo que ST_Intersects

Funções topológicas

- ST_Contains(geom1, geom2)
  - Retorna TRUE se a geom1 contiver espacialmente a geom2, ou seja, nenhum ponto de geom2 pode estar no exterior de geom1, e pelo menos um ponto de geom2 está no interior de geom1
  - Equivale a ST_Within(geom2, geom1)
- ST_ContainsProperly(geom1, geom2)
  - Retorna TRUE se geom2 interceptar o interior de geom1, mas não sua fronteira ou o exterior
  - DE-9IM: T**FF*FF*
  - Este é o relacionamento CONTAINS do 4IM

Funções topológicas

- ST_Covers(geom1, geom2)
  - Retorna TRUE se nenhum ponto de geom2 estiver no exterior de geom1
  - É uma função definida em 4IM, mas curiosamente não é parte do padrão OGC; apesar disso, existe no Oracle e no PostGIS
- ST_CoveredBy(geom1, geom2)
  - Inverso de ST_Covers

Funções topológicas

- ST_Relate(geom1, geom2)
  - Retorna a DE-9IM que ocorre entre as geometrias
- ST_Relate(geom1, geom2, matriz)
  - Retorna TRUE se o relacionamento existente entre as duas geometrias atender ao especificado na matriz

Funções topológicas

- ST_OrderingEquals(geom1, geom2)
  - Retorna TRUE se as geometrias forem iguais e seus vértices estiverem dispostos na mesma ordem/direção
  - ST_Reverse(geom) retorna uma geometria com a sequência de vértices invertida
- ST_DWithin(geom1, geom2, distância)
  - Retorna TRUE se as geometrias estiverem à distância especificada uma da outra
- ST_DFullyWithin(geom1, geom2, distância)
  - Retorna TRUE se as geometrias estiverem inteiramente dentro da distância especificada uma da outra

Funções de agregação geométrica

- ST_Intersection(geom, geom)
- ST_Difference(geom, geom)
- ST_Union(geom, geom)
  - Variação: ST_Union(GeomSet)
  - Variação: ST_MemUnion(GeomSet)
- ST_SymDifference(geom, geom)