Extensões Espaciais para SQL

Clodoveu Davis

Variações Principais

- Oracle Spatial
- PostGreSQL/PostGIS

- Decorrentes do padrão OGC *Simple Features Specification for SQL*

Oracle Spatial

Oracle

- Estrutura dos objetos do tipo `SDO_GEOMETRY`:

```
CREATE TYPE SDO_GEOMETRY AS OBJECT {
  SDO_GTYPE           NUMBER,
  SDO_SRID            NUMBER,
  SDO_POINT           SDO_POINT_TYPE,
  SDO_ELEM_INFO       MDSYS.SDO_ELEM_INFO_ARRAY,
  SDO_ORDINATES       MDSYS.SDO_ORDINATE_ARRAY };
```

Oracle

- SDO_GTYPE: tipo de geometria do objeto
  - Previsão para 3D, mas falta um tipo para superfícies

Valor de *SDO_GTYPE*	Tipo de geometria
d000	Desconhecido
d001	Ponto
d002	Poligonal
d003	Polígono (são permitidos buracos)
d004	Coleção heterogênea de elementos
d005	Múltiplos pontos
d006	Múltiplas poligonais
d007	Múltiplos polígonos (ilhas)

Oracle

- SDO_SRID: identificação do sistema de coordenadas usado *na codificação do objeto*
  - Podem existir na mesma tabela objetos cujas coordenadas estão expressos em sistemas diferentes
  - É tarefa do SIG (interface) interpretar e projetar essas coordenadas sobre a tela no momento da visualização
  - O valor do campo é numérico, e serve de chave para outra tabela em que os parâmetros das diversas projeções cartográficas estão codificados
    - Ex: SAD-69, hemisfério Sul, fuso 23 (45º WGr): código 83701

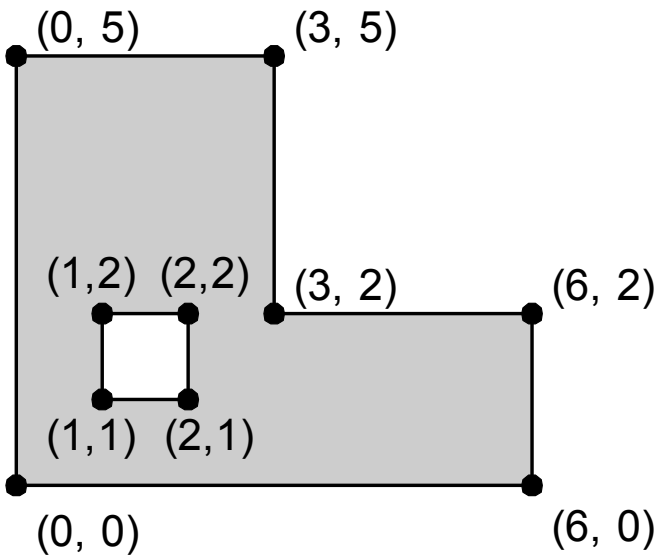
Objetos Geográficos em BD objeto-relacional

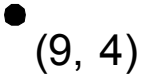
- SDO_POINT: coordenadas de um vértice tridimensional (variável do tipo SDO_POINT_TYPE)
  - Se o objeto for do tipo ponto, toda a geometria está definida
  - Se o objeto for de outro tipo, este campo é ignorado
  - Recurso válido para acelerar a seleção e apresentação de objetos do tipo ponto, muito comuns nas aplicações

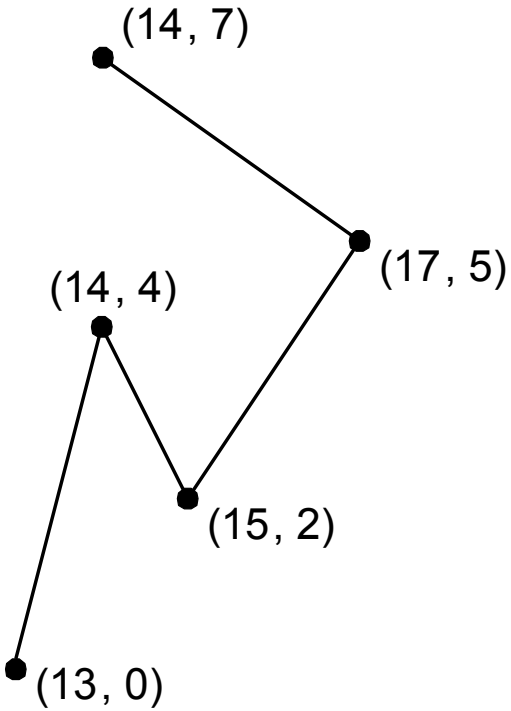
Objetos Geográficos em BD objeto-relacional

- SDO_ELEM_INFO e SDO_ORDINATES: dois vetores coordenados que codificam a geometria de objetos lineares e poligonais
  - SDO_ORDINATES: uma seqüência de conjuntos de coordenadas 3D
  - SDO_ELEM_INFO: separação das coordenadas em grupos, dando a cada grupo uma definição de comportamento

Exemplo







Objeto	Campo	Valor	Comentário
1	SDO_GTYPE	2003	Polígono simples, 2D
1	SDO_SRID	83201	UTM, Fuso 23, hemisfério sul, SAD-69
1	SDO_POINT	<nulo>	Desnecessário
1	SDO_ELEM_INFO	(1, 1003, 1, 8, 2003, 1)	Começando do par de coordenadas número 1, trata-se de um polígono externo (1003), com arestas retas (1); do par 8 em diante, é um polígono interno ou buraco (2003), com arestas retas (1).
1	SDO_ORDINATES	(0,0,6,0,6,2, 3,2,3,5,0,5, 0,0, 1,1,1,2,2, 2,2,1,1,1)	Seqüência x, y em sentido anti-horário no polígono externo e em sentido horário para o polígono interno (indica buraco); o último ponto repete o primeiro nos dois casos.
2	SDO_GTYPE	2001	Ponto 2D
2	SDO_SRID	82301	UTM, Fuso 23, hemisfério sul, SAD-69
2	SDO_POINT	(9, 4, 0)	Coordenadas do ponto
2	SDO_ELEM_INFO	<nulo>	Desnecessário
2	SDO_ORDINATES	<nulo>	Desnecessário
3	SDO_GTYPE	2002	Poligonal 2D
3	SDO_SRID	82301	UTM, Fuso 23, hemisfério sul, SAD-69
3	SDO_POINT	<nulo>	Desnecessário
3	SDO_ELEM_INFO	(1, 2, 1)	Começando do par de coordenadas número 1, trata-se de uma poligonal aberta (2), com arestas retas (1).
3	SDO_ORDINATES	(13,0,14,4, 15,2,17,5, 14,7)	Seqüência x, y.

Criação de tabelas espaciais

```
CREATE TABLE mg_munic
( codmuni number,
  nommuni varchar2(30),
   geom MDSYS.SDO_GEOMETRY);

CREATE INDEX mg_munic_sdx ON
mg_munic(geom) INDEX TYPE IS
MDSYS.SPATIAL_INDEX;
```

OBS.: A criação de índice só é permitida quando os metadados correspondentes à tabela foram criados

Oracle

- As extensões espaciais estão concentradas no esquema MDSYS
- A maioria das funções depende da existência de índices espaciais para funcionar
- Existem duas modalidades de licenciamento
  - Oracle Enterprise (ou SE): inclui o Oracle Spatial
  - Oracle XE: inclui o Oracle Locator
    - Não inclui várias funções geométricas nem os pacotes adicionais LRS, SDO_SAM, geocoding, georaster, modelo de dados topológico, modelo de redes

Oracle

- Sintaxe genérica dos operadores espaciais

```
<sp_op> (
  table_geom IN SDO_GEOMETRY,
  Query_geom IN SDO_GEOMETRY
  [, parameter_string IN VARCHAR2
  [, tag IN NUMBER]] ) = 'TRUE'
```

Exemplo

```
SELECT COUNT(*)
FROM munic m, munic n
WHERE m.nomemunic = "Betim"
AND SDO_WITHIN_DISTANCE(m.geom,
  n.geom, 'DISTANCE=50 UNIT=KM')
  = 'TRUE';
```

Operadores

- Proximidade simples: SDO_WITHIN_DISTANCE
- Vizinhos mais próximos: SDO_NN
- Relacionamentos espaciais/topológicos: SDO_RELATE e variações
- Funções geométricas: SDO_GEOM (buffer, distância, união, interseção, diferença, XOR, área, comprimento/perímetro, MBR, convex hull, centróide, ponto no polígono, ...)

Oracle

- SDO_WITHIN_DISTANCE(geom1, geom2, paramString)
  - Determina se a geometria 1 (*table geometry*) está próxima à geometria 2 (*query geometry*)
  - O paramString contém, entre aspas simples, a distância e a unidade de medida empregadas
  - A unidade default é o metro
- Ex:

```
SELECT count(*)
FROM lojas l, clientes c
WHERE l.id = 1 AND SDO_WITHIN_DISTANCE(c.geom, l.geom,
   'DISTANCE=1 UNIT=km') = 'TRUE';
```

- Ex2:

```
SELECT c.id, SDO_WITHIN_DISTANCE(c.geom, l.geom, 'DISTANCE=1
  UNIT=m') dist
FROM lojas l, clientes c
WHERE l.id = 1 AND SDO_WITHIN_DISTANCE(c.geom, l.geom,
   'DISTANCE=1 UNIT=km') = 'TRUE'
ORDER BY c.id
```

Oracle

- SDO_NN(geom1, geom2, paramString, tag): retorna os n vizinhos mais próximos
- Tag especifica (opcionalmente) um operador auxiliar de distância
- Ex (retorna todos os clientes, em ordem de proximidade):

```
SELECT c.id, c.nome
FROM lojas l, clientes c
WHERE l.id = 1 AND SDO_NN(c.geom, l.geom) = TRUE;
```

- Ex2 (retorna os 5 clientes mais próximos):

```
SELECT c.id, c.nome
FROM lojas l, clientes c
WHERE l.id = 1
AND SDO_NN(c.geom, l.geom, 'SDO_NUM_RES=5') = TRUE;
```

- Alternativa

```
SELECT c.id, c.nome
FROM lojas l, clientes c
WHERE l.id = 1
AND SDO_NN(c.geom, l.geom) = TRUE
AND ROWNUM <= 5;
```

Oracle

- SDO_RELATE(geom1, geom2, paramString): retorna o relacionamento espacial existente entre as duas geometrias
  - Valem as regras da matriz de 4 interseções
- Ex:

```
SELECT l.id, l.nome
FROM loja l, municipio m
WHERE SDO_RELATE(l.geom, m.geom,
  'MASK=INSIDE') = TRUE;
```

Oracle

- MASK na função SDO_RELATE:
  - INSIDE
  - COVEREDBY
  - CONTAINS
  - COVERS
  - TOUCH
  - EQUAL
  - OVERLAPBDYINTERSECT (= overlap entre polígonos)
  - ON (linha sobre a fronteira de polígono)
  - OVERLAPBDYDISJOINT (~= overlap entre linhas, segmentos se tocam mas extremos não)
  - ANYINTERACT: qualquer um dos acima
  - DISJOINT: quando ANYINTERACT for falso

Oracle

- Cada máscara tem um operador correspondente, que equivale ao SDO_RELATE mas sem que seja necessário especificar a máscara
  - SDO_INSIDE
  - SDO_CONTAINS
  - SDO_COVEREDBY
  - SDO_ON
  - SDO_COVERS
  - SDO_TOUCH
  - SDO_OVERLAPBDYINTERSECT
  - SDO_OVERLAPBDYDISJOINT
  - SDO_EQUAL
  - SDO_ANYINTERACT

Oracle

- Múltiplas máscaras: usar ‘+’ no paramString
- Ex.:

```
SELECT b.nome
FROM bairro b, regional r
WHERE r.nome='CENTRO-SUL' AND
  SDO_RELATE(b.geom, r.geom, 'MASK=INSIDE
  +OVERLAPBDYINTERSECT+ COVEREDBY') =
  TRUE;
```

Oracle

- SDO_FILTER: obtém geometrias cujo retângulo envolvente mínimo interceptam o REM de uma geometria
  - Sempre retorna um superconjunto da seleção com ANYINTERACT

```
SELECT b.nome
  FROM bairro b, regional r
  WHERE r.nome = 'Centro-Sul'
  AND SDO_FILTER(b.geom, r.geom) =
  'TRUE'
```

Oracle: funções geométricas

- Obs: várias estão ausentes do Oracle Locator
- SDO_GEOM.SDO_BUFFER(geom, dist, tolerancia, paramString)
  - Gera um buffer à distância especificada de uma geometria
- Ex.:

```
CREATE TABLE area_risco AS
SELECT id, SDO_GEOM.SDO_BUFFER(f.geom, 500, 1) FROM
  favela f
```

Oracle

- SDO_GEOM.SDO_DISTANCE(geom1, geom2, tolerancia, paramString)
  - Gera um buffer à distância especificada de uma geometria
- Ex.: clientes a menos de 500 m da loja 2

```
SELECT c.id, c.nome
FROM lojas l, clientes c
WHERE l.id = 2
AND SDO_GEOM.SDO_DISTANCE(c.geom, l.geom,
  0.5, 'unit=km') < 0.5;
```

Oracle

- SDO_GEOM.RELATE(geomA, mask, geomB, tol)
  - Retorna um string contendo o nome do relacionamento entre as geometrias A e B, ou TRUE/FALSE
  - Máscara:
    - DETERMINE: determina o relacionamento entre as geometrias
    - INSIDE, COVEREDBY, COVERS, CONTAINS, ...: retorna TRUE se o relacionamento especificado for observado

```
SELECT b.nome
   FROM bairro b, regional r
   WHERE r.nome = 'Barreiro'
   AND SDO_GEOM.RELATE(b.geom, 'INSIDE', r.geom) =
   'INSIDE'
```

Obs: SDO_GEOM.RELATE não usa o índice espacial, ao contrário de SDO_RELATE; uso típico: fazer FILTER ou relacionamento com ANYINTERACT, e depois usar esta função para refinamento

Oracle

- Operadores geométricos
  - SDO_UNION(geomA, geomB, tol)
  - SDO_INTERSECTION(geomA, geomB, tol)
  - SDO_DIFFERENCE(geomA, geomB, tol)
  - SDO_XOR(geomA, geomB, tol)

Oracle

- Funções de análise geométrica
  - SDO_AREA(geom, tol [, paramUnidade])
  - SDO_LENGTH(geom, tol [, paramUnidade])

```
SELECT SDO_AREA(b.geom, 0.005, 'unit=sq_m'),
  nome FROM bairro b;
```

Oracle

- SDO_MBR(geom)
  - Retorna o retângulo envolvente de uma geometria
- SDO_MIN_MBR_ORDINATE(geom, dim)
  - Retorna o limite inferior esquerdo do MBR da geometria na dimensão indicada
- SDO_MAX_MBR_ORDINATE(geom, dim)
  - Retorna o limite superior direito do MBR da geometria na dimensão indicada

Oracle

- Funções geométricas diversas
  - SDO_CONVEXHULL(geom, tol)
  - SDO_CENTROID(geom, tol)
  - SDO_POINTONSURFACE(geom, tol)

Oracle

- Agregações geométricas
  - SDO_AGGR_MBR(geom)
  - SDO_AGGR_UNION(geom)
  - SDO_AGGR_CONVEXHULL(geom)
  - SDO_AGGR_CENTROID(geom)

PostGIS

PostGIS Setup

- Após a instalação, ativar o PostGIS como extensão:

```
-- Enable PostGIS (includes raster)
CREATE EXTENSION postgis;
```

Esquemas

- Por default, novas tabelas serão criadas no esquema *public*
- No entanto, o esquema *public* conterá as tabelas do usuário, tabelas do PostGIS e funções
- Isso complica o backup, pois a restauração levará junto funções de uma determinada versão do PostGIS

Esquemas

- Criar tabelas do usuário em um novo esquema
  - É mais fácil fazer backup de dados que estão em um esquema separado
  - É mais simples restaurar dados que estão em um esquema separado, foco na aplicação
  - É mais fácil gerenciar versões de sistemas com mudança de estrutura
  - Usuários podem ficar restritos a acessar o conteúdo de um esquema, permitindo separar o que pode e o que não pode ser modificado

Esquemas

```
-- exemplo de nome
CREATE SCHEMA geodata
-- Esquema novo no caminho de busca (sessão)
SET search_path = geodata, public
-- E/OU alteração do default do usuário
ALTER USER postgres SET search_path =
geodata, public

-- mudar uma tabela para o novo esquema
ALTER TABLE tab SET SCHEMA geodata
```

Users

```
CREATE USER user WITH ROLE postgis_writer;

-- User schema
CREATE SCHEMA user AUTHORIZATION user;

-- verificar
show search_path
```

PostGIS

- 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

PostGIS

- 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'

Tabelas PostGIS

- Armazenam objetos no formato WKT (Well Known Text), especificado pela OGC
  - Existem extensões para a terceira coordenada, chamadas genericamente de XYZ, XYM e XYZM
- Até a versão 2.0, 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

PostGIS

- Versões antigas: comandos mantidos para compatibilidade

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

  - Observar o formato anômalo desse select, sem cláusula FROM

PostGIS

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

- Era necessário depois acrescentar uma linha na tabela geometry_columns, via INSERT ou usando a função *populate_geometry_columns*. A partir da versão 2.0, isso é automático, e a tabela geometry_columns não é mais editável.
  ```

PostGIS

- Criação de tabelas a partir da versão 2.0
  - Suporte para *typmod* em objetos geometry
  - Os modificadores typmod são o tipo de geometria OGC e o SRID, usados diretamente na tabela geometry_columns
  - Exemplo:

```
CREATE TABLE x (
nome VARCHAR(100),
geom GEOMETRY(Point, 4326)
);
```

PostGIS

- Exemplo de ALTER TABLE: reprojeção dos dados geométricos, alterando de 3D para 2D; a atualização de geometry_columns é automática

```
ALTER TABLE geotbl
ALTER COLUMN geom
SET DATA TYPE geometry(Point,26910)
USING ST_Transform(ST_Force_2D(geom),4326)
```

PostGIS

- 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);
```

PostGIS

- 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

PostGIS

- 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

PostGIS

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

PostGIS

- 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

PostGIS

- 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

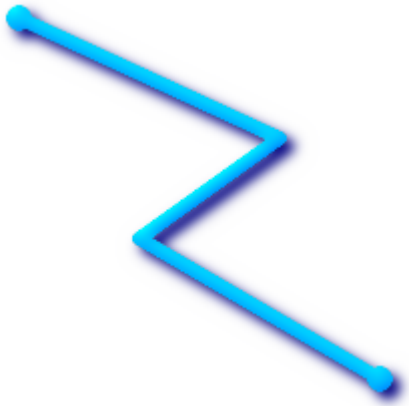
PostGIS

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

PostGIS

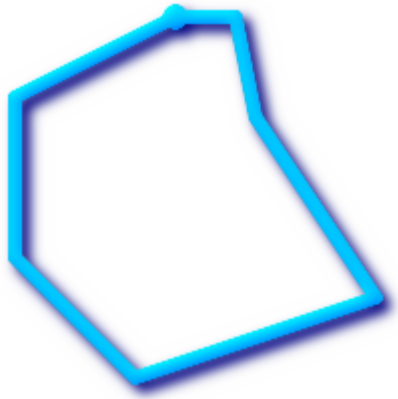
- Verificação de geometrias: o PostGIS, de acordo com o especificado pelo OGC, exige que as geometrias sejam *simples* e *válidas*
  - Simples: geometria que não tem pontos anômalos, como autointerseções, autotangências ou coincidências indesejadas
    - Aplicável a pontos e linhas
  - Válida: geometria (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)

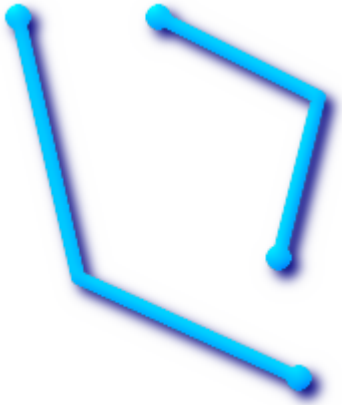
(d)

LINESTRINGS

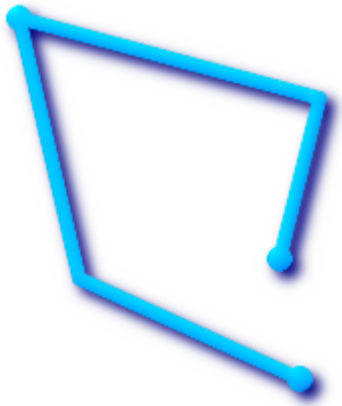
(a) e (c) são simples, (b) e (d) não são

(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)

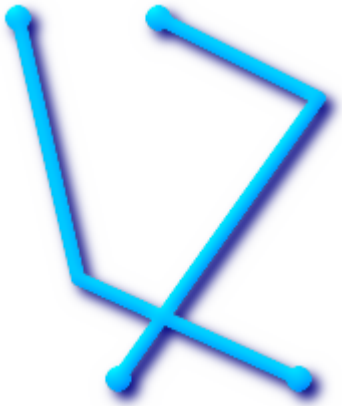
PostGIS



(a)



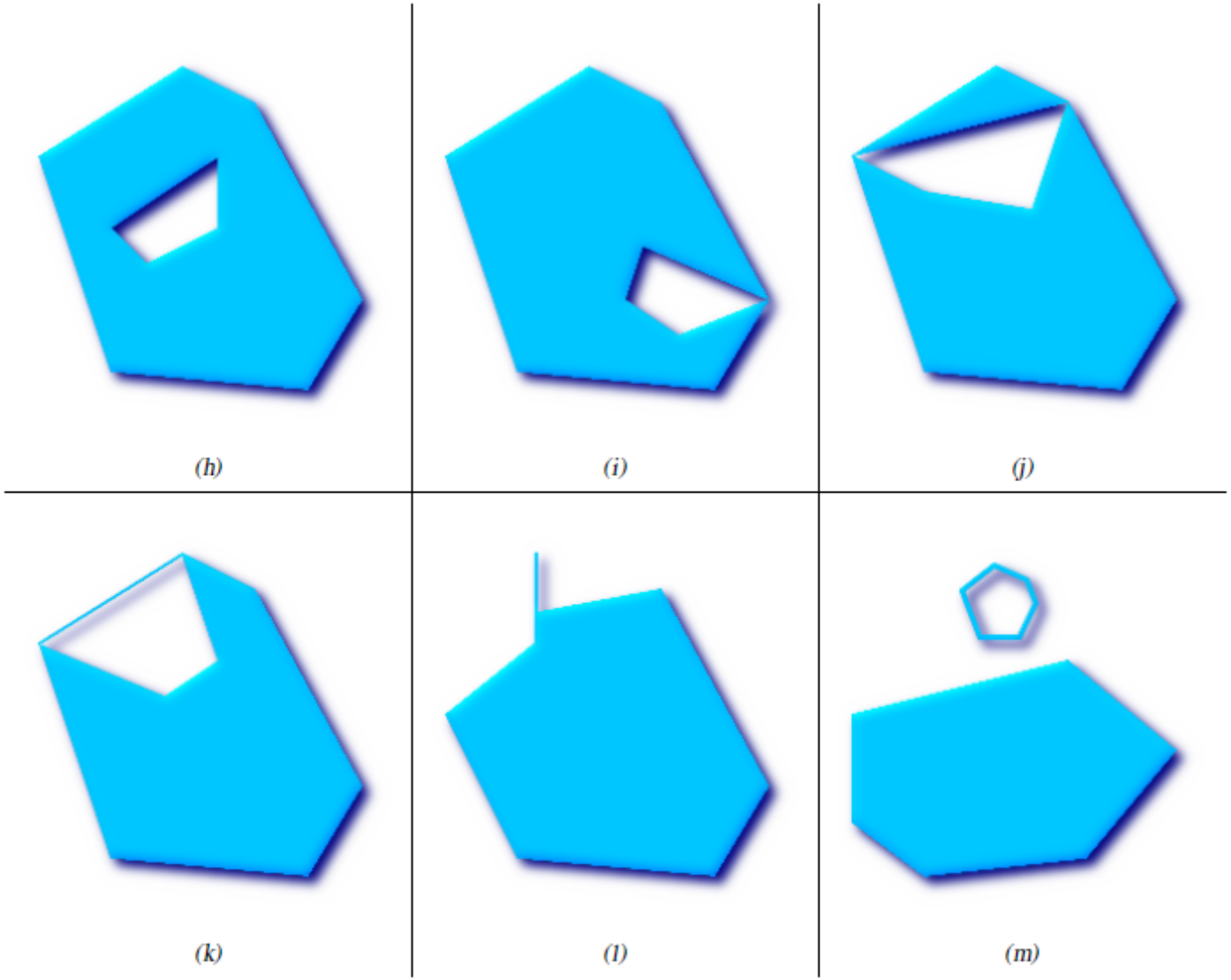
(b)



(c)

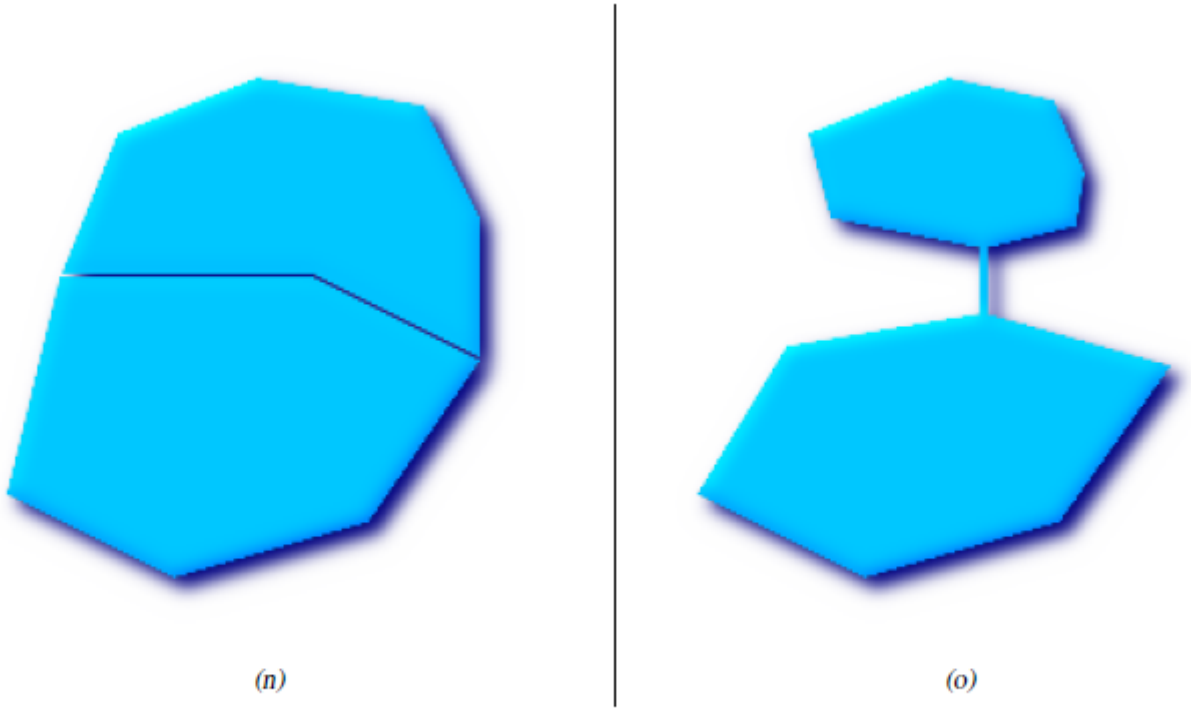
MULTILINESTRINGS

(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

PostGIS

- Verificação da validade de uma geometria
  - SELECT IsValid(geometria)
  - SELECT IsSimple(geometria)
- O PostGIS não aplica esse teste na entrada de cada geometria, pois isso pode demandar muito tempo de CPU
- Essa função não valida geometrias 3D (não OGC)
- Restrição de integridade para implementar isso:
  - `ALTER TABLE tab ADD CONSTRAINT nome CHECK (IsValid(geom));`

PostGIS

- 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;`

PostGIS

- 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

PostGIS

- Geography: como no caso de geometry com typmod, não é necessário usar o AddGeometryColumn

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

PostGIS

- 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
```

PostGIS

- Além de Geometry e Geography, o PostGIS também oferece suporte a funções do padrão SQL/MM3 (Multimedia)

PostGIS

- Funções geométricas
  - ST_Distance(geom, geom)
  - ST_DWithin(geom, geom, distance)
  - ST_Centroid(geom)
  - ST_Area(geom)
  - ST_Length(geom)
  - ST_PointOnSurface(geom)
  - ST_Boundary(geom)
  - ST_Buffer(geom, distance, numCircleSegments)
  - St_ConvexHull(geom)

PostGIS

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

PostGIS

- ST_AsText(geom)
- ST_AsBinary(geom)
- ST_SRID(geom)
- ST_Dimension(geom)
- ST_Envelope(geom)
- ST_IsEmpty(geom)
- ST_IsSimple(geom)
  - True se não houverem autointerseções
- ST_IsClosed(geom)
- ST_IsRing(geom)
  - True se a curva for simples e fechada
- ST_NumGeometries(geom)
  - Número de elementos em uma coleção

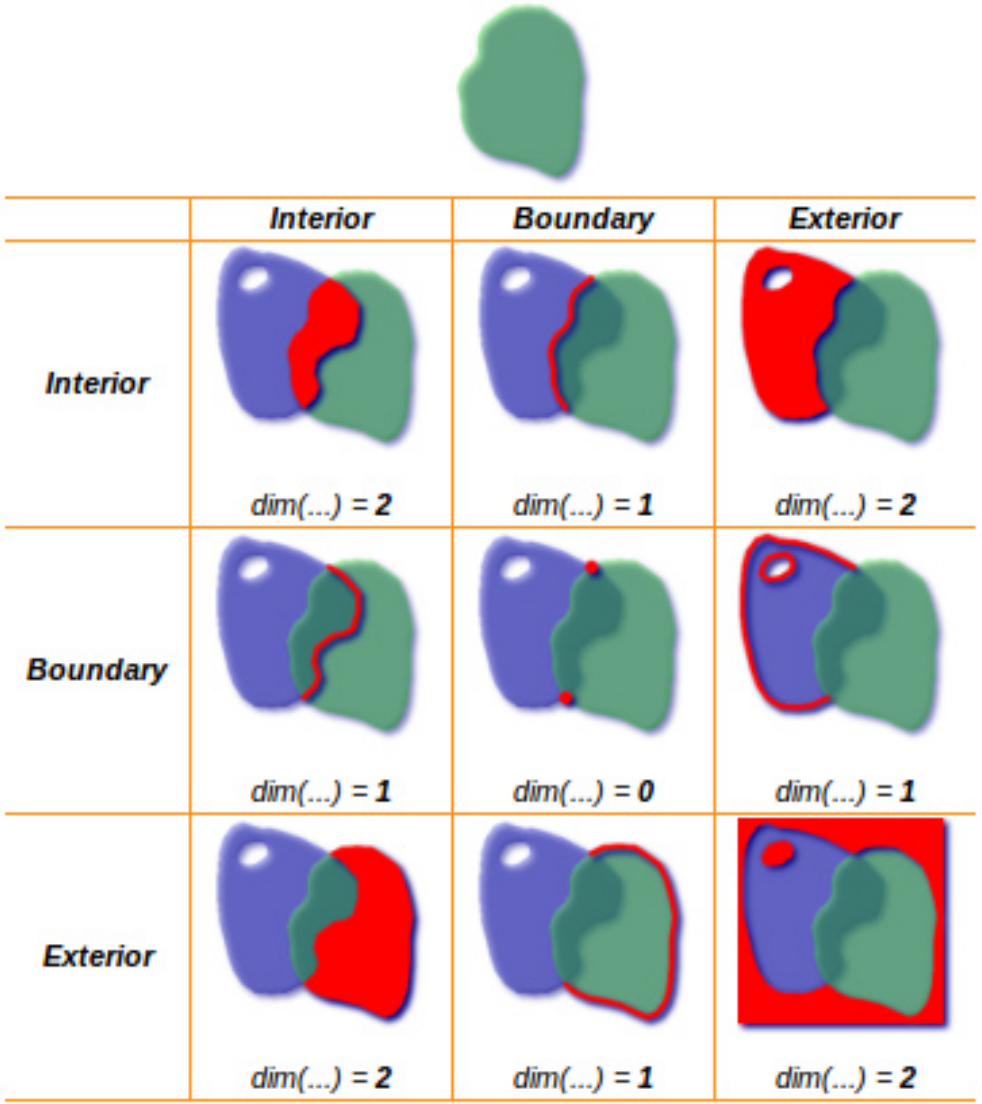
PostGIS

- ST_GeometryN(geom, int)
  - Retorna a n-ésima geometria em uma coleção
- ST_NumPoints(geom)
  - Retorna o número de vértices do primeiro linestring na geometria
- ST_PointN(geom, int)
  - Retorna o n-ésimo vértice da geometria
- ST_ExteriorRing(geom)
- ST_NumInteriorRings(geom)
- ST_InteriorRing(geom, int)
- ST_EndPoint(geom)
- ST_StartPoint(geom)
- GeometryType(geom) / ST_GeometryType(geom)
  - Retorna string com o nome do tipo da geometria
- ST_X(geom)
- ST_Y(geom)
- ST_Z(geom)

PostGIS

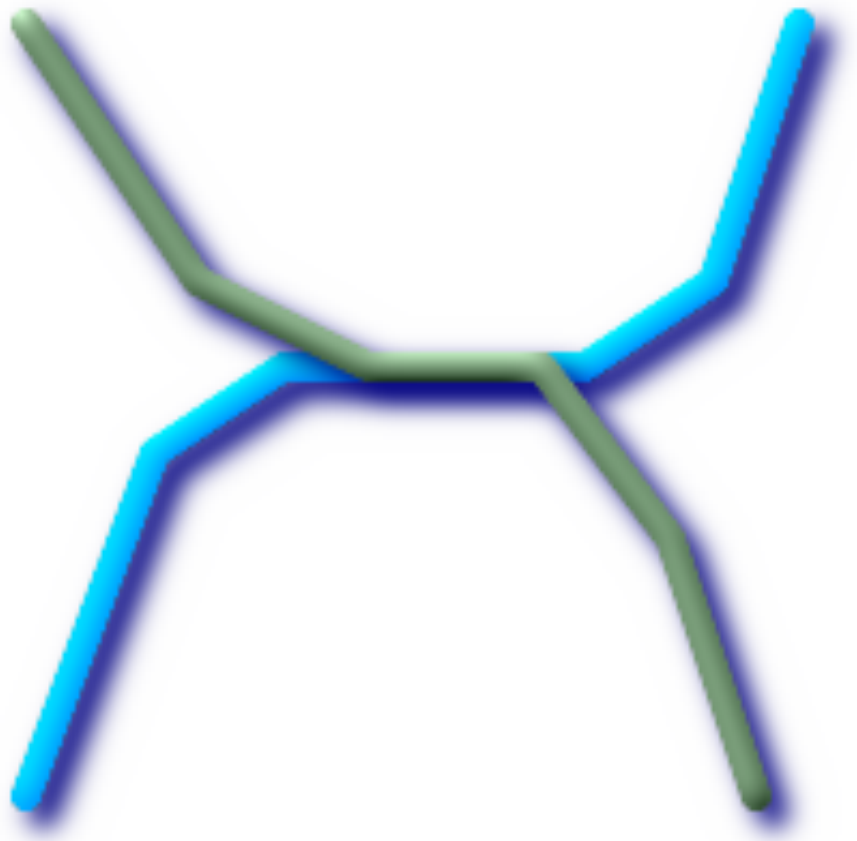
- Funções topológicas
  - Nomes ligeiramente diferentes da teoria de Egenhofer, mas aderentes aos padrões OpenGIS e SQL-MM
  - Usam a matriz de 9 interseções (ordem: interior, fronteira, exterior), numa versao estendida, chamada de **DE-9IM** (*Dimensionally extended 9-intersection matrix*)
  - As posições 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

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



PostGIS

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



PostGIS

- 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

PostGIS

- Operadores
  - A = B -> true se os MBRs coincidirem
  - A &< B -> true se o MBR de A intercepta ou está à esquerda do de B
  - A &> B -> true se ocorrer o inverso
  - A << B -> true se o MBR de A estiver à esquerda do de B
  - A >> B -> true se ocorrer o inverso
  - Outros operadores para MBR: abaixo (&<|, <<|), acima (|&>, |>>), iguais (~=), contém (~), contido (@), overlaps (&&)

PostGIS

- 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)

PostGIS

- ST_AsGML([version,] geom [, precision])
  - Version = 2 ou 3
  - Precision: default é 15 digitos significativos
- ST_AsKML([version,] geom [,precision])

PostGIS

- 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

PostGIS

- ST_Summary(geom)
- ST_ndims(geom)
- ST_npoints(geom)
- ST_nrings(geom)
- ST_isvalid(geom)
- ST_box2d(geom)
- ST_box3d(geom)