

BASES DE DATOS GEOGRÁFICAS

Implementaciones

“Lo más grande es el espacio, porque lo encierra todo”
Tales de Mileto

Docentes : Ing. Fernando Sato
 A.S. Sebastian Trossero



RESUMEN

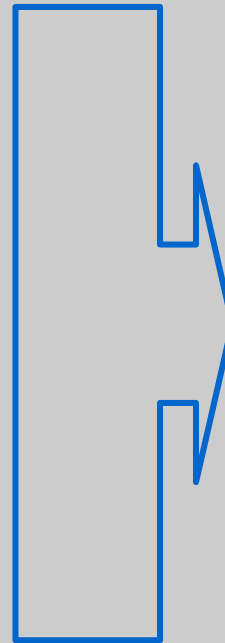
- Características PostGIS: Imp. de Referencia
- Jerarquia GEOMETRY
- PostGIS formatos nativos
- Constructores de TIPOS
- Metodos de la SuperClase GEOMETRY
- Metodos de Analisis Espacial
- Instalación PostGIS
- Anexo Transformación WKT to x,y
- Fuentes

Bases de Datos Geográficas

Implementación de Referencia

Implementación

La Implementación elegida es **PostGIS**.



Bases de Datos Geográficas

PostGIS - Introducción

PostGIS

- ✓ Es una extensión que convierte **PostgreSQL** en una base de datos espacial.
- ✓ Esta certificada por Open Geospatial Consortium (**OGC**).
- ✓ Cuenta con **Licencia Publica GNU**.
- ✓ Es un **producto sólido**, de **amplia inserción en el mercado GIS**, soportado por numerosas plataformas de GIS.

Bases de Datos Geográficas

PostGIS – Conectividad

- ✓ Proporciona Conectividad **ODBC**, standard de Conectividad de Base de Datos Abierta.
- ✓ Incluye extensiones a los controladores ODBC de PostgreSQL permitiendo el acceso transparente a objetos GIS de PostGIS.
- ✓ La conectividad **ODBC** es parte del estándar **OGC**.
- ✓ También proporciona Java Database Connectivity **JDBC**, (no forma parte del estándar **OGC**).

Bases de Datos Geográficas

PostGIS – Características Importantes

- ✓ **GiST** (Generalized Search Tree) proporciona una indexación espacial de alta velocidad.
- ✓ **PROJ.4** es una biblioteca de código abierto que proporciona reproyección de coordenadas para convertir entre distintos sistemas de coordenadas y un gran número de funciones específicas para el dominio de GIS.
- ✓ **GEOS** (Geometry Engine, Open Source) es una biblioteca utilizada por PostGIS para realizar todas las operaciones en las características simples de OpenGIS para la especificación de SQL. La biblioteca GEOS se utiliza para proporcionar pruebas de geometría (`ST_Touches ()`, `ST_Contains ()`, `ST_Intersects ()`) y operaciones (`ST_Buffer ()`, `ST_Union ()`, `ST_Intersection ()`, `ST_Difference ()`) dentro de PostGIS.

Bases de Datos Geográficas

PostGIS – Características Importantes

- ✓ PostGIS implementa el SFS de OGC para SQL.
- ✓ Esto significa, que soporta todos los tipos de datos OGC y las funciones asociadas:

Point

LineString

Polygon

Multipoint

Multiline

MultiPolygon

GeometryCollection

- ✓ PostGIS utiliza el formato de texto bien conocido **WKT** de OGC.

Bases de Datos Geográficas

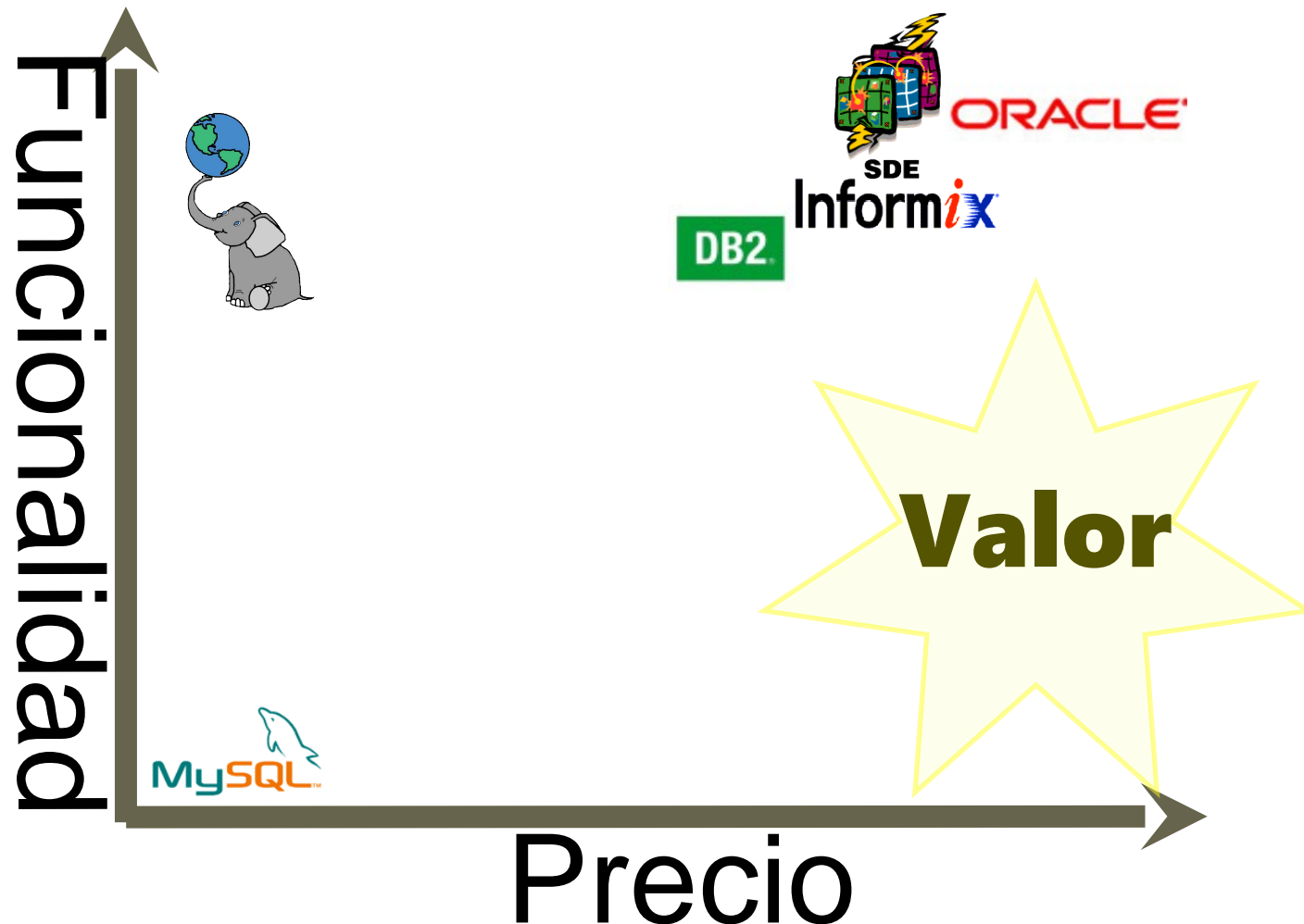
PostGIS – Características Importantes

- ✓ Las alternativas para disponer de una tabla habilitada espacialmente son:
 - Convertir una tabla temática: El Stored Procedure `AddGeometryColumn` (**versiones anterior a la 1**), permite agregar una columna espacial convirtiendo a tabla geométrica
 - Crear una tabla geométrica directamente con `Create Table`.

Aspecto Importante: Se integra total y transparentemente a la capa de postgresQL.

Bases de Datos Geográficas

Comparativa Precio / Funcionalidad



Bases de Datos Geográficas

Costo Licencias

“Enterprise”	1 Dual-Core	2 Quad-Core
Oracle	U\$S 40 k	U\$S 160 K
IBM DB2	U\$S 36,4 K	U\$S 145 K
SQL Server	U\$S 25 K	U\$S 50 K
IBM Informix	U\$S 50 K	U\$S 200 K
PostGIS	\$0	\$0

Bases de Datos Geográficas

PostGIS – Tipos, Índices y Funciones

Tipos de datos Espaciales Geography

Este tipo de dato proporciona soporte nativo para objetos espaciales representados por coordenadas geográficas (a veces llamadas coordenadas geodésicas, o "lat/lon").

Las coordenadas geográficas son coordenadas esféricas expresadas en unidades angulares (grados). La base del tipo geográfico de PostGIS es una esfera. El camino mas corto entre dos puntos en la esfera es el arco de circunferencia mas corto que une los dos puntos. esto significa que los cálculos geográficos (áreas, distancias, longitudes, intersecciones, etc) deben calcularse en la esfera, utilizando matemáticas mas complejas. Para medidas mas precisas, los cálculos deben tomar la forma esferoidal actual del mundo en cuenta, y las matemáticas se vuelven aun mas complejas.

Bases de Datos Geográficas

PostGIS – Tipos, Índices y Funciones

Tipos de datos Espaciales Geometry

Tipo Básico que da soporte nativo para objetos espaciales representados según SFS de OGC,

- Geometry es un tipo de datos Postgis fundamental, usado para representar una feature (característica) en un sistema de coordenadas euclidiano.

→

La implementación de PostGIS hace que cuando se agrega un atributo de este tipo a una tabla, inserta una `geometry_column` internamente.

→

→

→

Bases de Datos Geográficas

PostGIS – Tipos, Índices y Funciones

¿Cuándo utilizar Geometry o Geography?

- El tipo debe estar condicionado por la extensión del área de trabajo de la aplicación futura.
 - ¿Los datos se extienden por el globo o una zona continental grande?
 - ¿O es una provincia, región o municipio?
- Si tus datos están en un área pequeña usar **GEOMETRY**.
- Si tus datos son globales o cubren una región continental usar **GEOGRAPHY**.

Bases de Datos Geográficas

PostGIS – Tipos, Índices y Funciones

Resumiendo, PostGIS aporta a PostgreSQL:

Tipos de datos Espaciales

- Geometry
- Geography

Índices Espaciales

- R-tree
- quad-tree
- kd-tree

Funciones Espaciales

- Mas de 1000 funciones, extendiendo el estándar.

Bases de Datos Geográficas

PostGIS – Niveles de Representación

Layers (Capas)



Geometrias (Geometry)



Colecciones (Collection)



Primitivas Graficas
(Graphical Primitives)



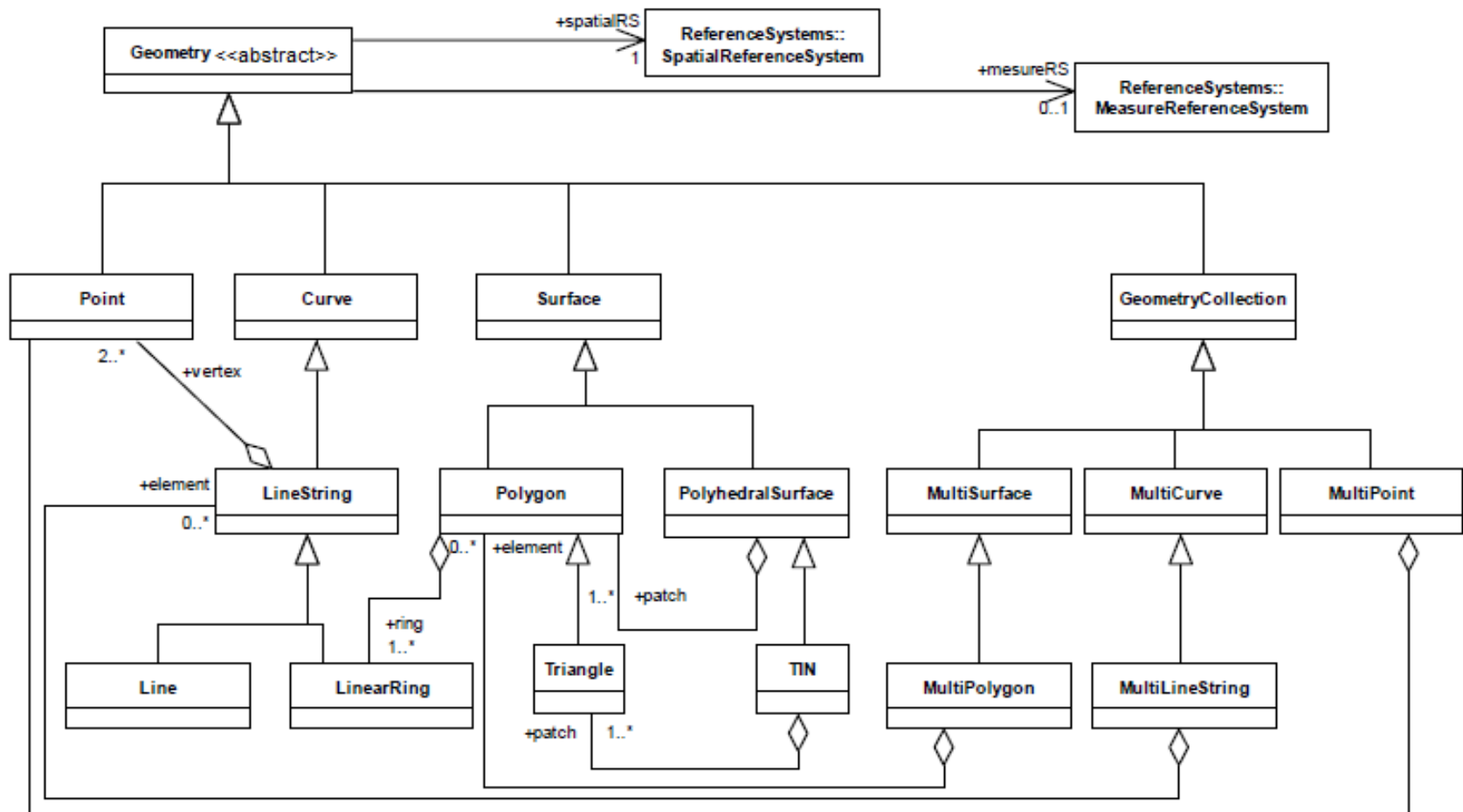
Coordenadas (Coordinates)

- ✓ Rutas, Ríos, Parcelas, Terrenos uso / cobertura
- ✓ Una o más de Rutas, ríos, parcelas de tierra polígonos de en una capa simple.
- ✓ Uno o mas segmentos de ríos o rutas, líneas de parcelas y uso del suelo
En una geometría principal.
- ✓ Puntos individuales, lineas y polígonos que son usados para representar una geometría.
- ✓ (x,y) o (coordenadas planas) o (latitud, longitud)

Bases de Datos Geográficas

PostGIS – Tipos de datos

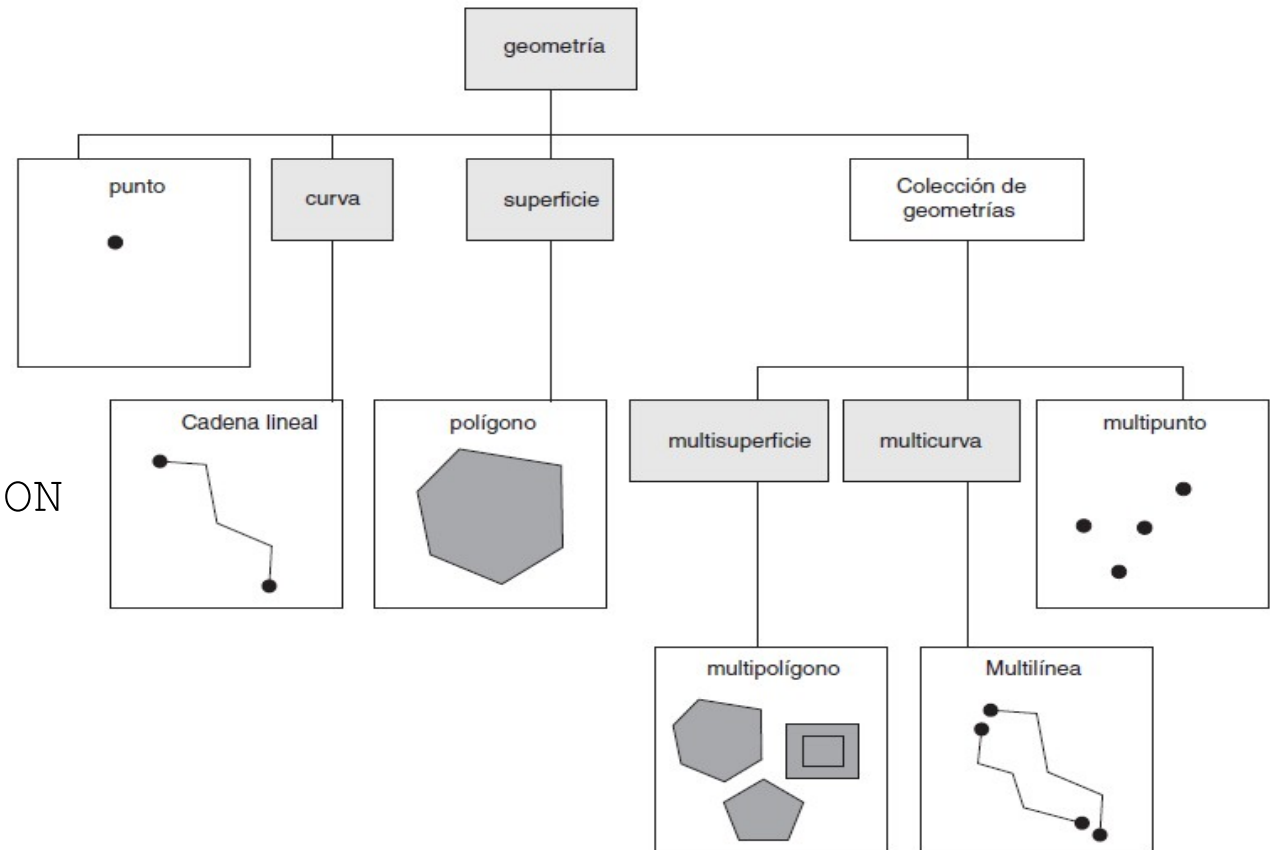
Jerarquía: Tipos agregados (tienen prefijo ST_)



Bases de Datos Geográficas

PostGIS – Tipos - Representación

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION



Bases de Datos Geográficas

PostGis – Formatos Nativos – WKT y WKB

Utiliza los formatos OGC:

- **WKT** (Well Known Text): Gramática específica para definir objetos espaciales expresados en forma vectorial textual. Algunos:

Tipo de geometria	WKT
POINT	“POINT(0 0)”
LINESTRING	“LINESTRING(0 0, 1 1, 2 2, 3 4)”
POLYGON	“POLYGON(0 0, 0 1, 1 1, 0 0)”
MULTIPOINT	“MULTIPOINT(0 0, 1 1, 2 2)”
MULTILINESTRING	“MULTILINESTRING ((10 10, 2 2, 10 40), (40 40, 30 30, 40 20, 30 10))”
MULTIPOLYGON	“MULTIPOLYGON (((3 2, 0 0, 5 4, 3 2)))”
GEOMETRY COLLECTION	“GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))”

- **WKB** (Well Known Binary): Gramática específica para definir objetos espaciales expresados en forma vectorial binaria. (Mas Rápido).

Bases de Datos Geográficas

PostGis – Formatos Nativos – WKT y WKB

Mas Ejemplos.

- `POINT(0 0)`
- `LINESTRING(0 0,1 1,1 2)`
- `POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))`
- `MULTIPOINT(0 0,1 2)`
- `MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))`
- `MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))`
- `GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))`

Bases de Datos Geográficas

PostGis – Formatos Nativos – EWKT y EWKB

Los formatos WKT solo soportan geometrías 2D, tampoco soportan SRID.

Los formatos extendidos de PostGIS son un superconjunto de los OGC, actualmente (todo WKB/WKT valido es un EWKB/EWKT valido), pero esto puede variar en el futuro, por ej si el OGC saca un nuevo formato que crea conflictos con PostGIS.

PostGIS EWKB/EWKT agrega soporte 2dm, 3d y 3dm con SRID.

Vamos con los asociados a Puntos (0 dimensión):

- `POINT(0 0 0) -- XYZ`
- `POINTM(0 0 0) -- XYM`₍₁₎
- `POINT(0 0 0 0) -- XYZM`₍₁₎
- `MULTIPOINTM(0 0 0,1 2 1)`

Todos con posibilidad de asociar un SRID.

Nota ₁: M de measure (medida), modela una medida asociada, ej en una ruta el label de "km" o "mojon", o si hay estación de servicio o no.

Bases de Datos Geográficas

PostGIS – Tipos - Referencia

OGC también requiere que el almacenamiento interno de objetos espaciales incluya el sistema de referencia espacial (SRID). El SRID es necesario al crear objetos espaciales postGIS.

La Entrada/Salida de estos formatos están disponibles utilizando las interfaces siguientes:

```
bytea WKB = ST_AsBinary(geometry);  
text WKT = ST_AsText(geometry);  
geometry = ST_GeomFromWKB(bytea WKB, SRID);  
geometry = ST_GeometryFromText(text WKT, SRID);
```

Por ejemplo un comando valido para crear un punto en una relación resultado sería:

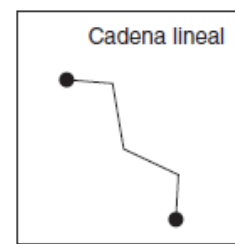
```
SELECT ST_GeomFromText('POINT(-126.4 45.32)', 312) unPto;
```

Bases de Datos Geográficas

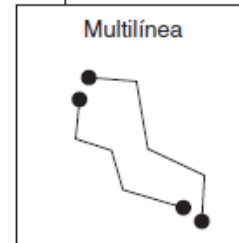
PostGIS – Tipos - EWKT y EWKB

Seguimos con los asociados a Lineas (1 dimensión):

- `LINESTRINGM(2 3 4, 3 4 5)` ₍₁₎



- `MULTILINESTRING((0 0 0,1 1 0,1 2 1), (2 3 1,3 2 1,5 4 1))`



Todos con posibilidad de asociar un SRID.

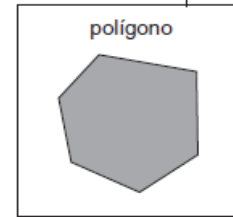
Nota ₁: M de measure (medida), modela una medida asociada, ej en una ruta el label de "km" o "mojon", o si hay estación de servicio o no.

Bases de Datos Geográficas

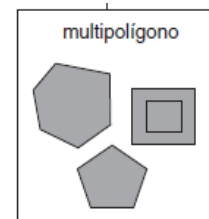
PostGIS – Tipos - EWKT y EWKB

Seguimos con los asociados a Superficies (2 dimensión):

- `POLYGON((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0))`



- `MULTIPOLYGON(((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)),((-1 -1 0,-1 -2 0,-2 -2 0,-2 -1 0,-1 0)))`



- `TRIANGLE ((0 0, 0 9, 9 0, 0 0)) --> Extensión de PostGis`

Bases de Datos Geográficas

PostGIS – Prefijo ST

Todas las funciones de PostGIS comienzan con "**ST_**".

ST es el acrónimo de **S**patial **T**ype (tipo espacial).

De esta manera, el operador de conjunto **UNION** de **OGC**, en postgis tendrá la misma sintaxis, pero, su nombre comenzará con st (**ST_UNION**).

Ejemplo:

```
SELECT
```

```
ST_DIMENSION(ST_GeomFromText(' POINT (1 4)', 312)) P1;
```

Nota: Esta convención no se aplica a WKT ni EWKT.

Bases de Datos Geográficas

Constructor **GeomFromText** Point

Tipos de Puntos: *Postgis, como dijimos admite varios tipos de puntos, veamos su definición EWKT de cada uno de ellos:*

ST_GeomFromText:

```
SELECT
  ST_ASTEXT(st_geomfromtext('POINT(10 15)'))      AS xy,
  ST_ASTEXT(st_geomfromtext('POINT(10 15 9)'))    AS pointz,
  ST_ASTEXT(st_geomfromtext('POINTM(10 15 3)'))   AS pointm,
  ST_ASTEXT(st_geomfromtext('POINT(10 15 9 21)')) AS pointzm;
```

Ambos son Validos

Resultado:

Salida de datos				
Comentar Mensajes Historial				
	point text	pointz text	pointm text	pointzm text
1	POINT(10 15)	POINT Z (10 15 9)	POINT M (10 15 9)	POINT ZM (10 15 9 21)

Recordar: M de measure (medida).

Bases de Datos Geográficas

Constructor **GeomFromText** **LineString**

Tipos de LineString: *Postgis, como dijimos admite varios tipos, veamos su definición EWKT de cada uno de ellos:*

ST_GeomFromText:

```
SELECT
  ST_ASTEXT(st_geomfromtext('LINESTRING(0 0, 1 3, 2 2)'),
  ST_ASTEXT(st_geomfromtext('LINESTRINGZ(0 0 0, 1 3 0, 2 2 1)'),
  ST_ASTEXT(st_geomfromtext('LINESTRINGM(0 0 21, 1 3 23, 2 2 31)'),
  ST_ASTEXT(st_geomfromtext('LINESTRING(0 0 0 21, 1 3 2 23, 2 2 1
                                     31)')) AS LINESTRINGZM
```

Ambos son Validos

Salida de datos Comentar Mensajes Historial				
	linestring text	linestringz text	linestringm text	linestringzm text
1	LINESTRING(0 0,1 3,2 2)	LINESTRING Z (0 0 0,1 3 0,2 2 1)	LINESTRING M (0 0 21,1 3 23,2 2 31)	LINESTRING ZM (0 0 0 21,1 3 2 23,2 2 1 31)

Bases de Datos Geográficas

Constructor **GeomFromText Polygon**

Tipos de Polygon: *Postgis, como dijimos admite varios tipos, veamos su definición EWKT de cada uno de ellos:*

ST_GeomFromText:

SELECT

```
ST_ASTEXT(st_geomfromtext('POLYGON((0 0, 0 1, 1 0, 0 0))'),
ST_ASTEXT(st_geomfromtext('POLYGONZ((0 0 0, 0 1 1, 1 0 1, 0 0 1))') ,
ST_ASTEXT(st_geomfromtext('POLYGONM((0 0 21, 1 3 23, 2 2 31, 0 0
20))')),
ST_ASTEXT(st_geomfromtext('POLYGONZM((0 0 0 21, 1 3 2 0, 2 2 2 -1, 0 0
0 0))')) AS POLYGONZM
```

Resultado:

Ambos son Validos

Salida de datos				
Comentar Mensajes Historial				
	polygon text	polygonz text	polygonm text	polygonzm text
1	POLYGON((0 0,0 1,1 0,0 0))	POLYGON Z ((0 0 0,0 1 1,1 0 1,0 0 1))	POLYGON M ((0 0 21,1 3 23,2 2 31,0 0 20))	POLYGON ZM ((0 0 0 21,1 3 2 0,2 2 2 -1,0 0 0 0))

Bases de Datos Geográficas

Constructores Propios de PostGis

St_MakePoint: Construye un objeto **St_Point**, recibiendo como parámetros *x* e *y* como tipo *Double Precision*. **No recibe Srid.**

Idem St_MakePointM Construye un objeto punto, recibiendo como parámetros *x*, *y*, *m* como tipo *Double Precision*. **No recibe Srid.**

Ejemplos:

Podemos asignar luego SRID con St_SetSRid

SELECT

```
ST_ASTEXT(st_makepoint(1, 2))           AS Pointxy,
ST_ASTEXT(st_makepoint(1, 2, 1))        AS Pointxyz,
ST_ASTEXT(st_makepoint(1, 2, 1, 0.5))    AS Pointxyzm,
ST_ASTEXT(st_makepointm(1, 2, 0.5))     AS Pointxym
```

Resultado:

Salida de datos Comentar Mensajes Historial				
	pointxy text	pointxyz text	pointxyzm text	pointxym text
1	POINT (1 2)	POINT Z (1 2 1)	POINT ZM (1 2 1 0.5)	POINT M (1 2 0.5)

Bases de Datos Geográficas

Constructores Propios de PostGis

St_MakeLine: Construye un objeto **St_LineString**, a partir de puntos multipuntos o líneas.

St_MakePolygon: Construye un objeto **St_Polygon**, a partir de *linestring*.

Algunos Ejemplos:

```
SELECT
```

```
    ST_AsText(ST_MakeLine(ST_MakePoint(1,2), ST_MakePoint(3,4)))  
    AS st_linestring,
```

```
    ST_AsText(ST_MakePolygon(ST_GeomFromText('LINESTRINGM(75.15 29.53  
1,77 29 1,77.6 29.5 2, 75.15 29.53 2)')) AS st_polygon_M
```

Resultado:

Salida de datos		
Comentar		
Mensajes		
Historial		
	st_linestring text	st_polygon_m text
1	LINESTRING(1 2,3 4)	POLYGON M ((75.15 29.53 1,77 29 1,77.6 29.5 2,75.15 29.53 2))

Bases de Datos Geográficas

Constructores Multiples y Colecciones

St_Collect(g1,g2): Construye una colección de(*MultiPoint*, *MultiLineString*, *MultiPolygon* o una *GeometryCollection*) a partir de *g1* y *g2* .

Algunos Ejemplos:

```
SELECT ST_ASTEXT(ST_Collect(g1,g2)) as coll1,
       ST_ASTEXT(ST_Collect(g1,g4)) as coll2 from
(SELECT
  ST_GEOFROMTEXT('POINT(4 4)') g1,
  ST_GEOFROMTEXT('POINT(1 2)') g2,
  ST_GEOFROMTEXT('LINESTRING(2 6, 6 2)') g3,
  ST_GEOFROMTEXT('POLYGON((1 2, 3 2, 3 6, 1 6, 1 2))') g4
) AS TEMP
```

Resultado:

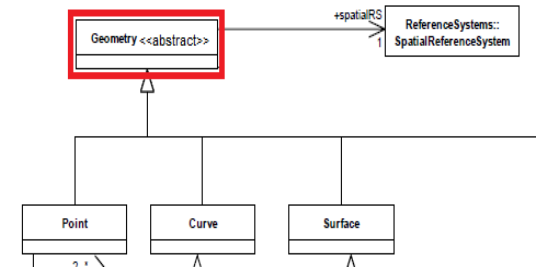
coll1 text	coll2 text
MULTIPOINT(4 4,1 2)	GEOMETRYCOLLECTION(POINT(4 4),POLYGON((1 2,3 2,3 6,1 6,1 2)))

Bases de Datos Geográficas

OGC – Operaciones de Geometry

Operaciones Básicas sobre Geometry:

La especificación OGC define las siguientes operaciones básicas (G representa un tipo geometry, una geometría):



- ✓ **ST_Dimension(g)**: devuelve la dimensión de la geometría g.
- ✓ **ST_NDims(g)**: devuelve la dimensión de las coordenadas de g
- ✓ **ST_GeometryType(g)**: devuelve el nombre del tipo geométrico de g (por ej, ST_LINESTRING, ST_POLYGON, ST_MULTICURVE, ...).
- ✓ **ST_SRID(g)**: devuelve el ID del sistema de referencia espacial de g.
- ✓ **ST_SETSRID(g,srid)**: setea un nuevo ID del sistema de referencia espacial de g.

Bases de Datos Geográficas

OGC – Operaciones de Geometry

Operaciones Básicas sobre Geometry (continuación):

- ✓ **ST_IsEmpty(g)**: comprueba, si g está vacío.
- ✓ **ST_IsSimple(g)**: comprueba, si g es simple (definido en el modelo de geometría OGC).
- ✓ **ST_Envelope(g)**: genera un recuadro mínimo, rectángulo mínimo con lados paralelos a los ejes que rodea el objeto ("minimum bounding rectangle", MBR).
- ✓ **ST_Boundary(g)**: devuelve el cierre del limite combinatorio de g, según la definición de OGC.

Bases de Datos Geográficas

PostGis - Definiciones Básicas - Dimensiones

Definiciones generales Básicas, sobre Geometry:

La especificación OGC define la siguiente operación básica.

- ✓ **St_Dimension(g)**: devuelve la dimensión de la geometría g.



```
1) select st_dimension(st_makepoint(3,2,1));
```

```
2) select  
st_dimension(st_MakeLine(st_makepoint(0,0),st_makepoint(1,1)))
```

Que no debemos confundir con St_NDims que

- ✓ **St_NDims**: devuelve las dimensiones de las coordenadas de g.

```
3) select st_ndims(st_makepoint(3,2,1));
```

```
4) select st_ndims(st_makeline .... sigue como 2)
```

Resultado: 1) = 0, 2) = 1, 3) = 3 4) = 2

Bases de Datos Geográficas

PostGis - Def Básicas – Tipo

Definiciones generales Básicas, sobre Geometry:

La especificación OGC define la siguiente operación básica.

- ✓ **St_GeometryType(g)**: devuelve el tipo de la geometría g.



```
select st_geometrytype(st_makepoint(3,2,1)),  
       st_geometrytype(st_linefromtext('LINESTRING(0 0 , 1 1)')),  
       st_geometrytype(  
         st_geomfromtext('POLYGON((0 0, 11 0, 1 1, 0 0))'))
```

Resultado:

	st_geometrytype text	st_geometrytype text	st_geometrytype text
1	ST Point	ST LineString	ST Polygon

Bases de Datos Geográficas

PostGis - Def Básicas – SRid

Definiciones generales Básicas, sobre Geometry:

La especificación OGC define las siguiente operación básica.



- ✓ **St_SRid(g):** devuelve el identificador de referencia espacial como se define en la tabla spatial_ref_sys de la geometría g.

```
select st_srid(st_linefromtext('LINESTRING(0 0 ,1 1)',22175)),  
       st_srid(st_geomfromtext('POLYGON((0 0,11 0,1 1,0 0))',22175))
```

- ✓ **Resultado:**

	st_srid integer	st_srid integer
1	22175	22175

Bases de Datos Geográficas

PostGis - Def Básicas – SRid

Definiciones generales Básicas, sobre Geometry:

La especificación OGC define la siguiente operación básica.



- ✓ **St_SetSRid(g, entero srid):** Setea el identificador de referencia espacial srid a la geometría g.

Ver script siguiente

Bases de Datos Geográficas

PostGis - Def Básicas – SRid

```
DO $$
DECLARE
    geometrial geometry;
BEGIN
    geometrial = st_geomfromtext('POLYGON((0 0,    11 0,    10 1, 1 1, 0 0))');

    RAISE NOTICE
        'Función ST_ varias con ST_SETSRID
        -----
        Asignacion: %

        en formato WKT: %
            st_srid: %

        Nuevo Srid: %
        -----
        ',
        geometrial,
        st_astext(geometrial),
        st_srid(geometrial),
        st_srid(st_setsrid(geometrial,22175));
END$$;
```



...

• • •

Función ST_varias con ST_SETSRID

en formato WKT: POLYGON((0 0,11 0,10 1,1 1,0 0))

Nuevo Srid: 22175

Bases de Datos Geográficas

PostGis - Def Básicas – isEmpty

Definiciones generales Básicas, sobre Geometry:

La especificación OGC define la siguiente operación.



- ✓ **St_IsEmpty(g):** devuelve True si la Geometría es una colección vacía, polígono vacío, punto vacío etc.

```
select      st_isempty(st_geometryfromtext('LINESTRING EMPTY')),
            st_isempty(st_geometryfromtext('POINT EMPTY')),
            st_isempty(st_geometryfromtext('POINT(1 0)'))
```

✓ Resultado:

True para las dos primeras columnas, False para la ultima.

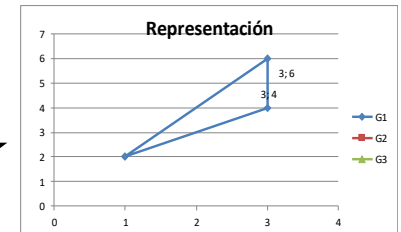
Bases de Datos Geográficas

PostGis - Def Básicas – IsSimple

Definiciones generales Básicas, sobre Geometry:
La especificación OGC define la siguiente operación.



- `St_IsSimple(g)`: devuelve True si la Geometría no tiene autointersecciones.



Ejemplos

```
SELECT ST_IsSimple(ST_GeomFromText('POLYGON((1 2, 3 4, 3 6, 1 2))'));
st_issimple
```

```
-----
t
(1 row)
```

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)'));
st_issimple
```

```
-----
f
(1 row)
```

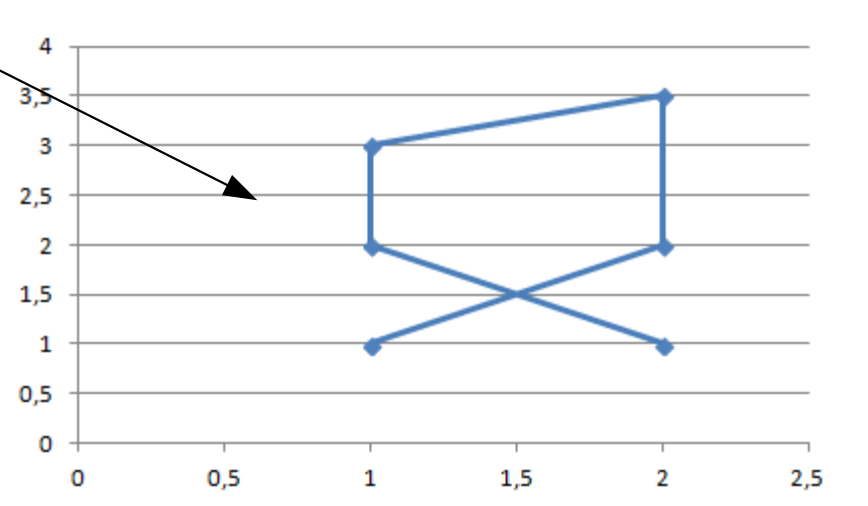

Bases de Datos Geográficas

PostGis - Def Básicas – IsSimple

Ejemplos

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)'));  
st_issimple
```

f
(1 row)



Bases de Datos Geográficas

PostGis - Def Básicas – Boundary

Caso 1: *POINT (10 130)*

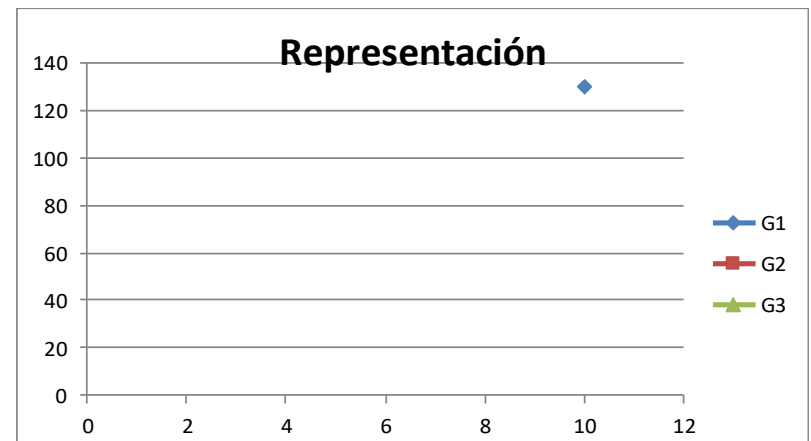
- ✓ **St_Boundary(g):** devuelve un contorno común del objeto.



```
SELECT ST_Boundary(geom)
FROM (SELECT 'POINT(10 130)::geometry' As geom) As f
```

- ✓ **Resultado:**

```
-- ST_AsText output
GEOMETRYCOLLECTION EMPTY
```



Bases de Datos Geográficas

PostGis - Def Básicas – Boundary

Caso 2: `POLYGON ((10 130,50 190,110 190,140 150, 150 80, 100 10, 20 40, 10 130), (70 40, 100 50, 120 80, 80 110,50 90, 70 40))`

✓ **St_Boundary(g):** devuelve un contorno común del objeto.

```
SELECT ST_Boundary(geom)
FROM (SELECT 'POLYGON ((10 130,50 190,110 190,140 150,
                        150 80, 100 10, 20 40, 10 130 ),
                        ( 70 40, 100 50, 120 80, 80 110,50 90, 70 40 ))'::geometry
      As geom) As f
```



✓ **Resultado:**



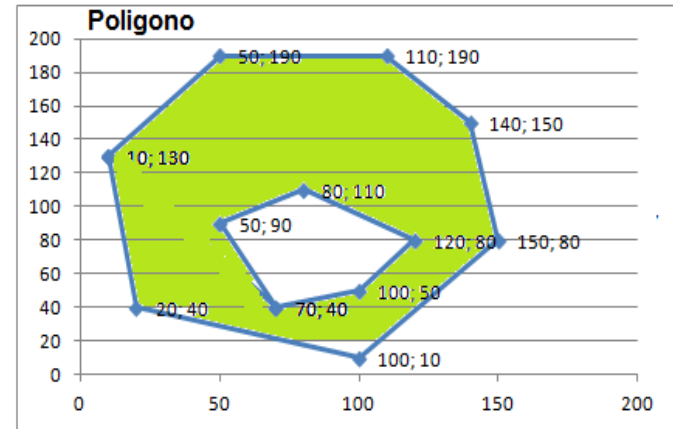
Bases de Datos Geográficas

PostGis - Def Básicas – Boundary

Caso 2:

```
POLYGON((10 130,50 190,110 190,140 150,150 80, 100 10,20 40,10 130 ),  
        (70 40, 100 50, 120 80, 80 110,50 90, 70 40 ))
```

✓ Polígono:

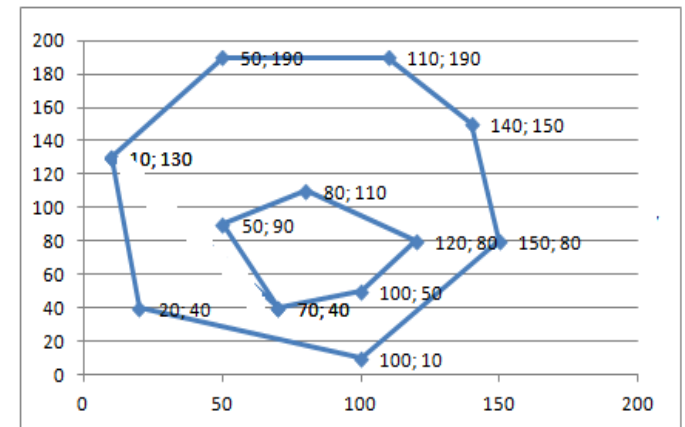


✓ ST_Boundary:

-- ST_AsText output

```
MULTILINESTRING((10 130,50 190,  
110 190,140 150,...
```

--2 Multilinestring con misma
-geom que los componentes del polígono



Bases de Datos Geográficas

PostGis - Def Básicas – Boundary

Caso 3: `LINESTRING(100 150,50 60,70 80, 160 170)`

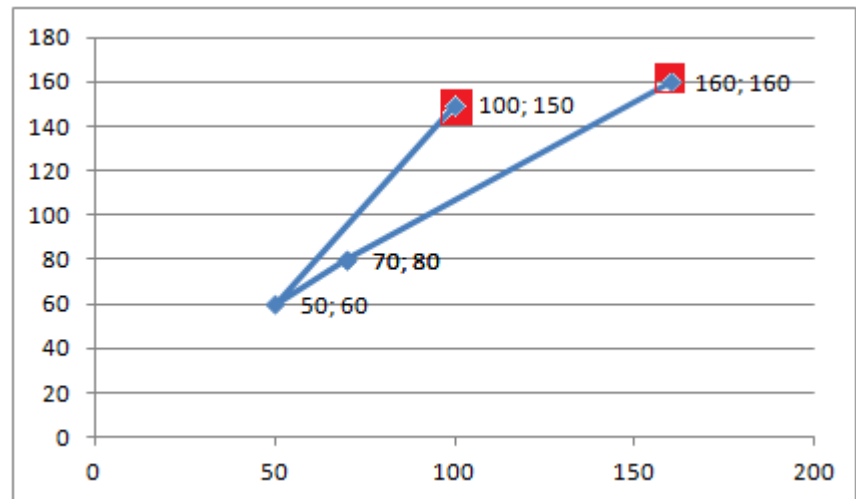


- ✓ **St_Boundary(g):** devuelve el cierre del limite combinatorio de g, según la definición de OGC.

```
SELECT ST_Boundary(geom)
FROM
(SELECT 'LINESTRING(100 150,50 60,70 80, 160 170)::geometry
As geom) As f
```

✓ Resultado:

```
-- ST_AsText output
MULTIPOINT(100 150,160 170)
```



Bases de Datos Geográficas

PostGis - Def Básicas – Envelope

Caso 1:

La especificación OGC define la siguiente operación.



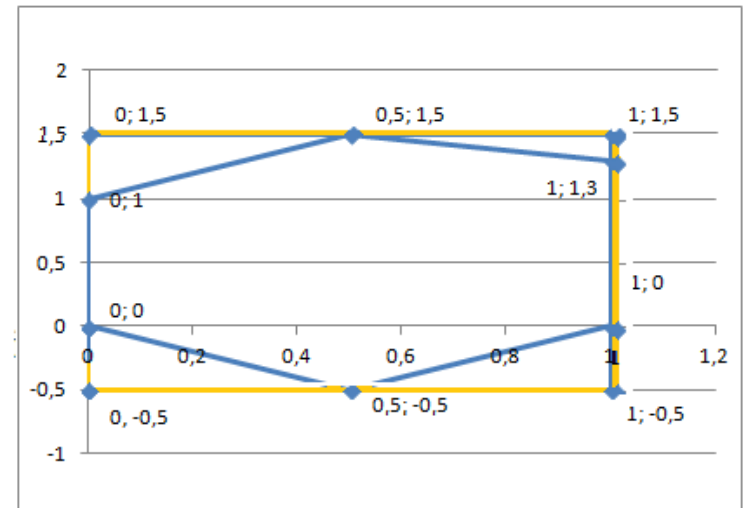
- ✓ **St_Envelope(g)**: devuelve el MBR (Minimum Bourding Rectangle al rededor de g.

```
SELECT st_astext(      st_envelope( st_geomfromtext
    ('Polygon((0 0,0 1, 0.5 1.5,1 1.3, 1 0,0.5 -0.5, 0 0)) '))
    )
    );
```

○

✓ Resultado:

```
"POLYGON((0 -0.5 , 0 1.5,
          1 1.5 , 1 -0.5,
          0 -0.5))"
```



Bases de Datos Geográficas

PostGis - Def Básicas – Envelope

Caso 2:



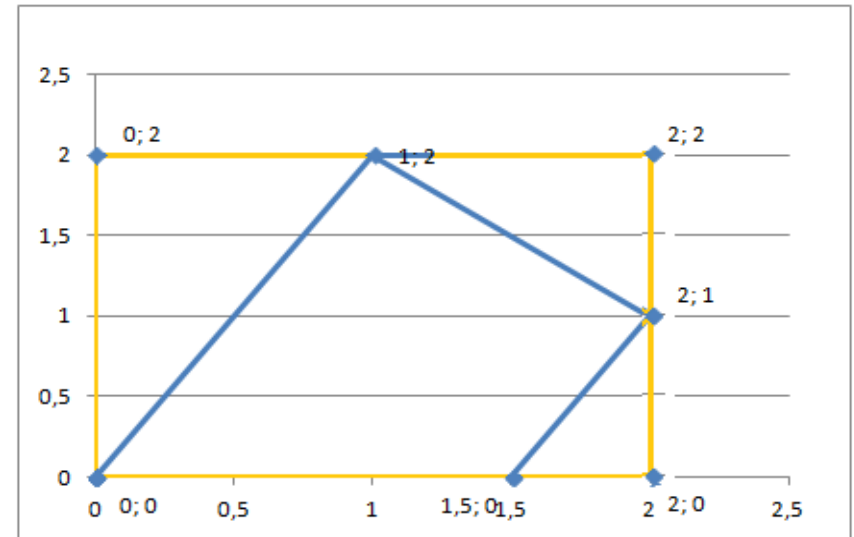
✓ Ej con ST_LineString

```
SELECT st_astext(      st_envelope( st_geomfromtext
    ('LineString(0 0,1 2, 2 1, 1.5 0) ')
                        )
      );
```

○

✓ Resultado:

```
"POLYGON( (0 0 , 0 2,
           2 2 , 2 0,
           0 0) )"
```



Bases de Datos Geográficas

PostGis - Def Básicas – Envelope

Caso 3:



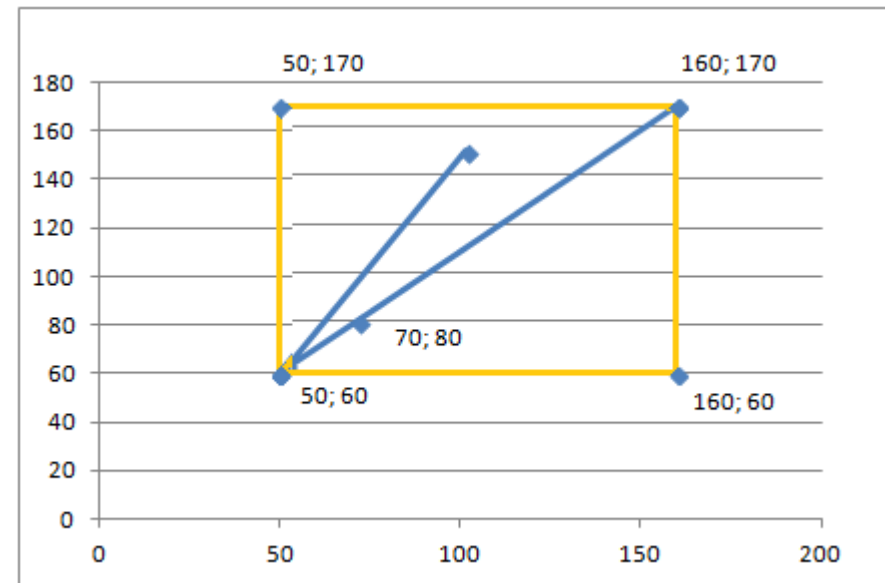
✓ Ej con ST_LineString

```
SELECT st_astext(      st_envelope(  st_geomfromtext
    ('LineString(0 0,1 2, 2 1, 1.5 0) ')
    )
    );
```

○

✓ Resultado:

```
"POLYGON((50 60 , 50 170,
          160 170 , 160 60,
          50 60))"
```



Bases de Datos Geográficas

Analisis Espacial con SQL de Postgis

Postgis dispone de un gran numero de operadores y funciones para análisis espacial, veamos algunos de los mas importantes:

Devuelven Float

- ✓ ***ST_Length*** *retorna el largo de una geometría.*
- ✓ ***ST_Perimeter*** *retorna el perímetro de una figura.*
- ✓ ***ST_Area*** *retorna el área de una figura.*

Bases de Datos Geográficas

Analisis Espacial - Longitud

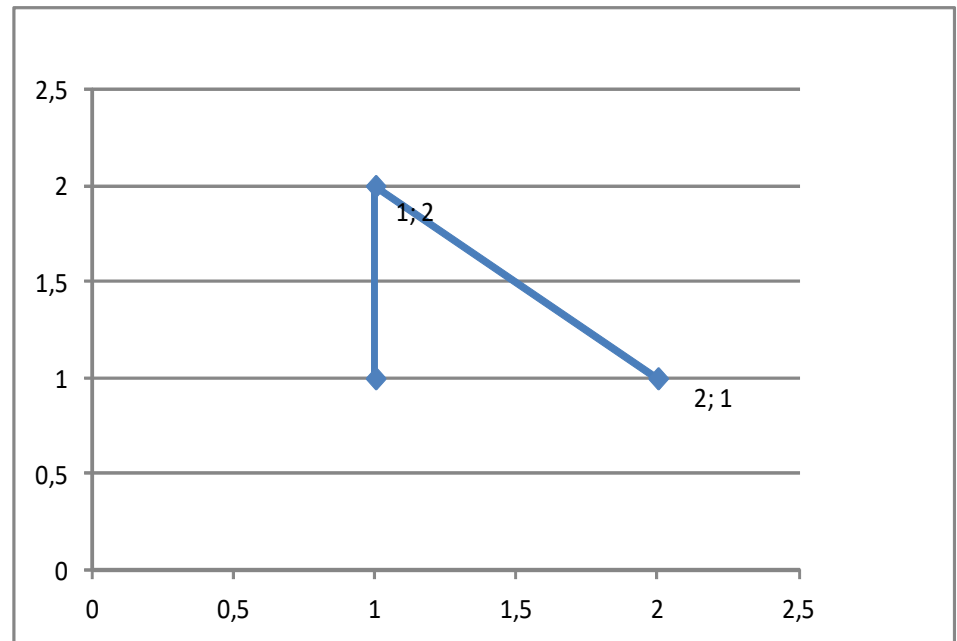
St_Length: Devuelve la longitud de la geometría de un linestring o multistring.

✓ **Ej con ST_LineString**

```
SELECT st_length(st_geomfromtext('LineString(1 1,1 2,2 1)'));
```

✓ **Resultado:**

	st_length double precision
1	2.41421356237309



Bases de Datos Geográficas

Analisis Espacial - Longitud

Nota sobre St_Length en POL: El largo de un polígono es siempre cero.

✓ Ej con ST_LineString

```
Select st_length(g1) largolin, st_length(g2) largopol from
(select g1, st_makepolygon(g1) as g2 from
(select ST_GEOMFROMTEXT('LINESTRING(10 30,20 30,20
35,10 35,10 30)') as g1
) tmp
) tmp2
```

✓ Resultado:

	largolin double precision	largopol double precision
1	30	0

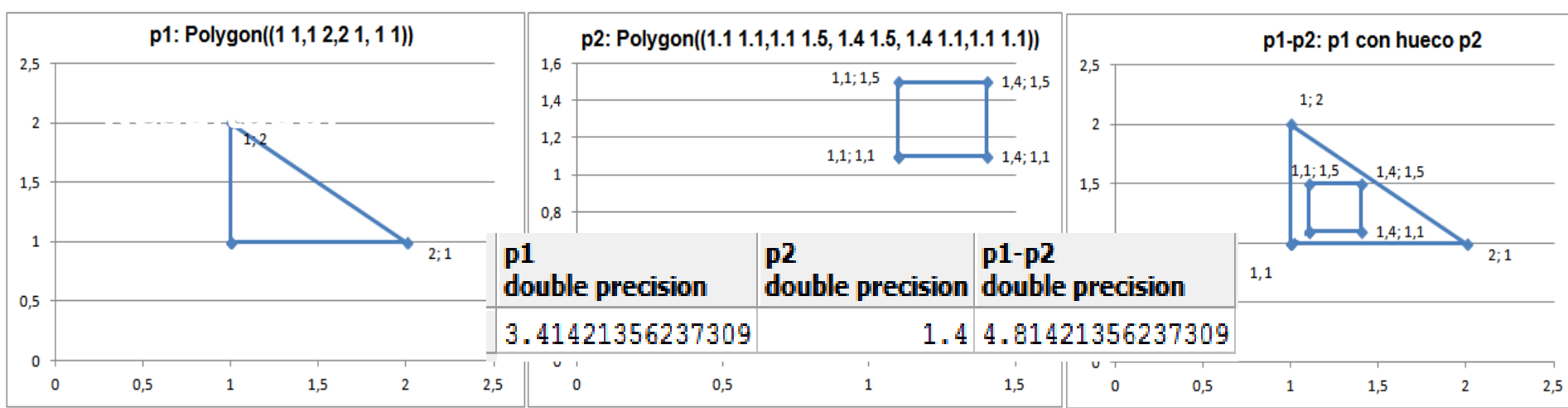
Bases de Datos Geográficas

Analisis Espacial - Perimeter

St_Perimeter: Devuelve la longitud de la geometría de un Polígono
 Prestar atención que computa también el perímetro de huecos.

✓ **Ej con ST_Polygon** *SELECT*

```
st_perimeter(st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))'))
                                as p1,
st_perimeter( st_geomfromtext('Polygon((1.1 1.1,1.1 1.5, 1.4
                                1.5, 1.4 1.1,1.1 1.1))')) as p2,
st_perimeter( st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1),
                                (1.1 1.1,1.1 1.5, 1.4 1.5, 1.4 1.1,1.1 1.1))')) as "p1-p2";
```



Bases de Datos Geográficas

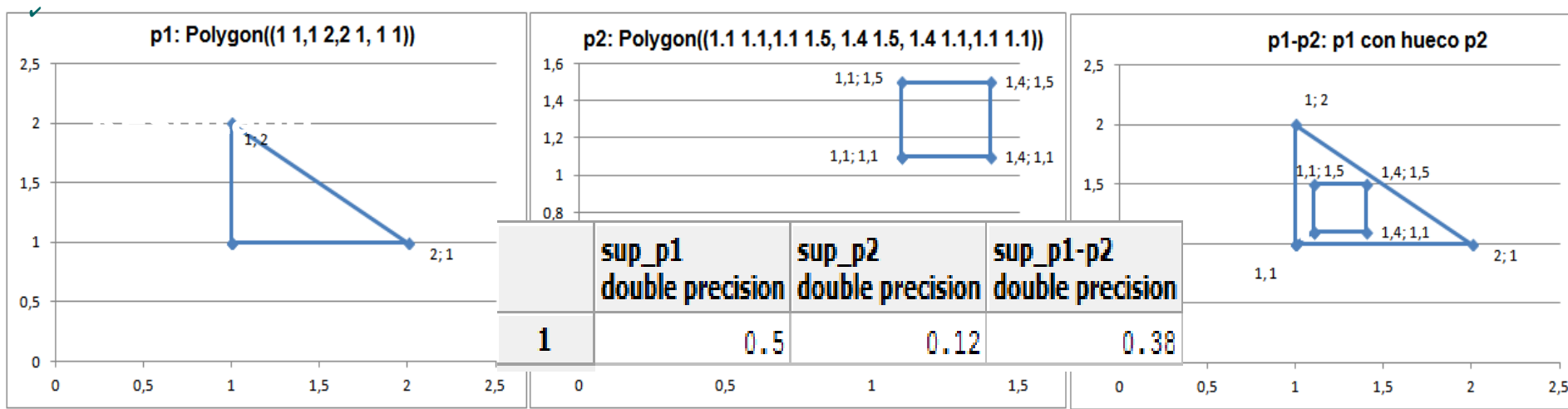
Analisis Espacial - Area

St_Area: Devuelve superficie de la geometría de un Polígono
 Prestar atención que computa también la superficie con huecos.

✓ **Ej con ST_Polygon** *SELECT*

```

st_area(st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))'))
                                                    as sup_p1,
st_area(  st_geomfromtext('Polygon((1.1 1.1,1.1 1.5, 1.4
                                1.5, 1.4 1.1,1.1 1.1))')) as sup_p2,
st_area(  st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1),
                                (1.1 1.1,1.1 1.5, 1.4 1.5, 1.4 1.1,1.1 1.1))')) as "s_p1-p2";
    
```



Bases de Datos Geográficas

Caso de Estudio

En una base de datos Postgis, se tiene una tabla de departamentos con las columnas nombre, provincia y geometría de cada departamento (Poligono).

```
departamentos(gid, provincia_id, nombre, supgeom)  
provincias(id, nombre)
```

Ejemplo:

¿Determine el ranking de los departamentos de Santa Fe y Entre Ríos ordenados de superficie de mayor a menor ? Projete provincia, departamento y superficie.

¿Sume la superficie total y la cantidad de departamentos registrados de santa fe y córdoba?.

Bases de Datos Geográficas

Analisis Espacial – Fxs de Conjunto

Veamos las operaciones Básicas de conjunto, todas generan geometrías:

Devuelven Geometrías

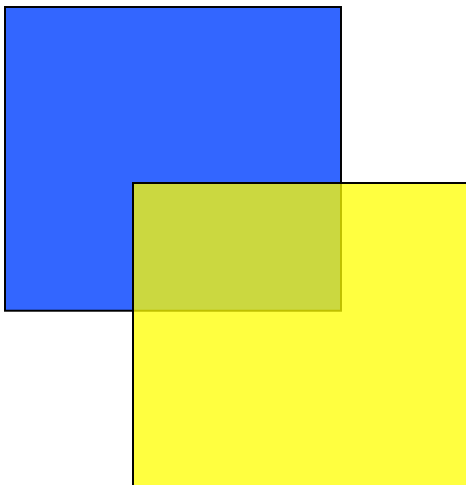
- ✓ ***ST_Union*** *retorna la geometría unión*
- ✓ ***ST_Intersection*** *retorna la geometría intersección.*
 - ✓ *ST_Intersects y ST_Disjoint (Devuelven Boolean).*
- ✓ ***ST_Difference*** *retorna la geometría Diferencia.*
- ✓ ***ST_SymDifference*** *retorna la Diferencia Simétrica*

Bases de Datos Geográficas

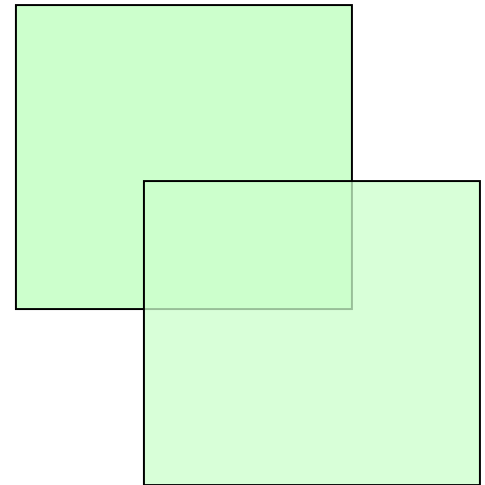
Analisis Espacial - Union

La función siguiente devuelve la **union** de las dos geometrias:

- ✓ ***ST_union***(A,B) retorna la **geometría unión** de A y B.



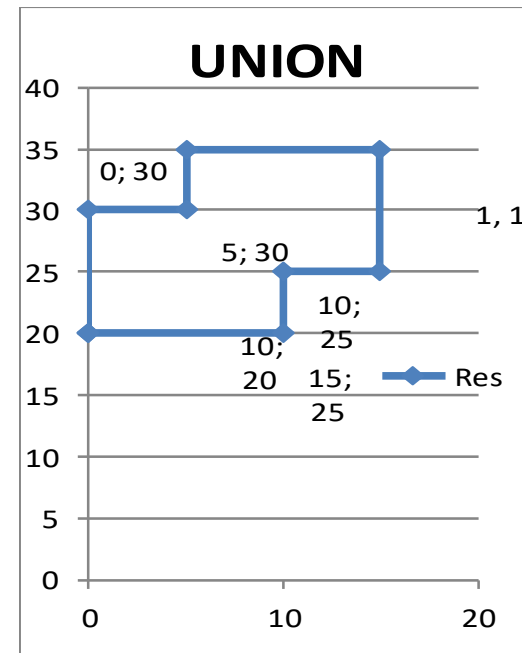
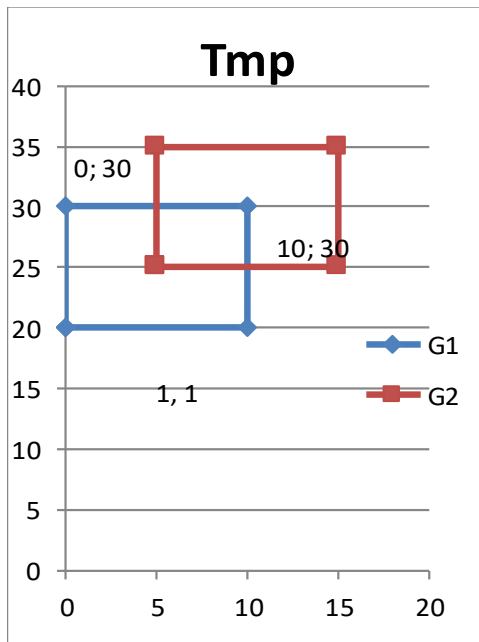
ST_union() →



Bases de Datos Geográficas

Analisis Espacial - Union

```
SELECT st_astext(ST_UNION(g1, g2)) as Res from
  (SELECT st_geomfromtext('POLYGON((0 20,0 30,10 30,10 20,0 20))')
    as g1,
    st_geomfromtext('POLYGON((5 25,5 35,15 35,15 25,5 25))')
    as g2) tmp
```

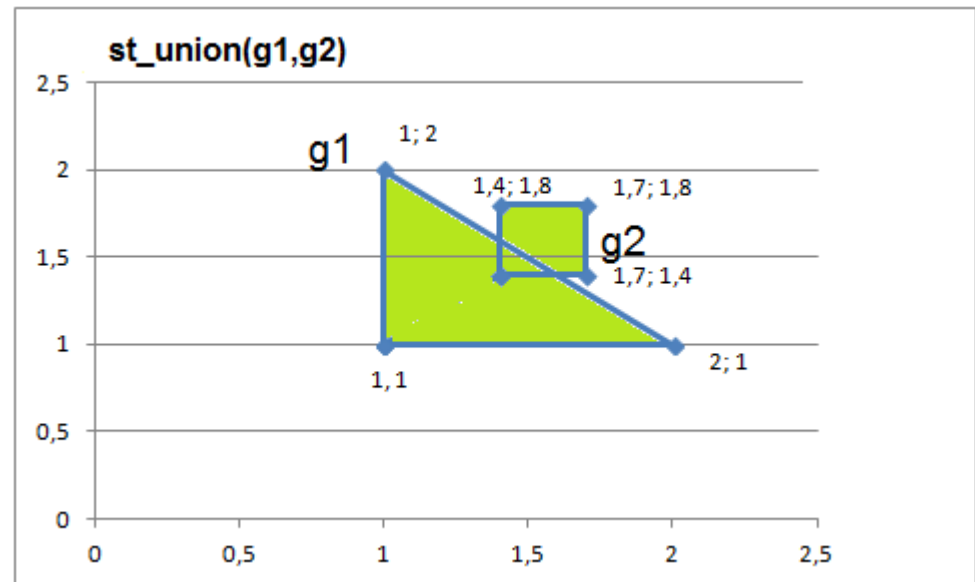


Res: POLYGON((0 20,0 30,5 30,5 35,15 35,15 25,10 25,10 20,0 20))

Bases de Datos Geográficas

Analisis Espacial - Union

```
select st_union(g1, g2) from (SELECT  
st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))')  
as g1,  
st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,  
1.7 1.4,1.4 1.4))') as g2)  
AS TEMP
```

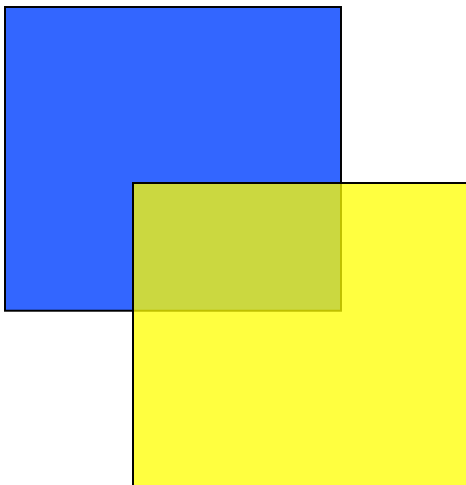


Bases de Datos Geográficas

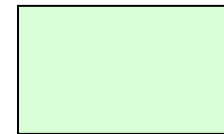
Analisis Espacial - Intersección

La función siguiente devuelve la Intersección de las dos geometrías:

- ✓ ***ST_Intersection***(a,b) retorna la geometría intersección.



ST_Intersection() →

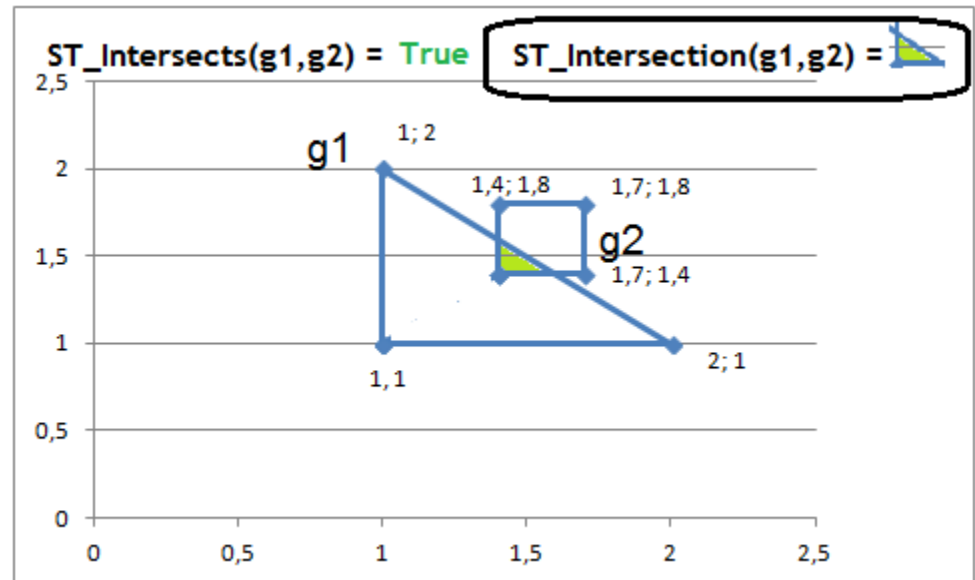


Bases de Datos Geográficas

Analisis Espacial - Intersecciones

```
select st_Intersects(g1, g2), st_Intersection(g1,g2) from  
(SELECT st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))')  
as g1,  
st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,  
1.7 1.4,1.4 1.4))') as g2)
```

AS TEMP

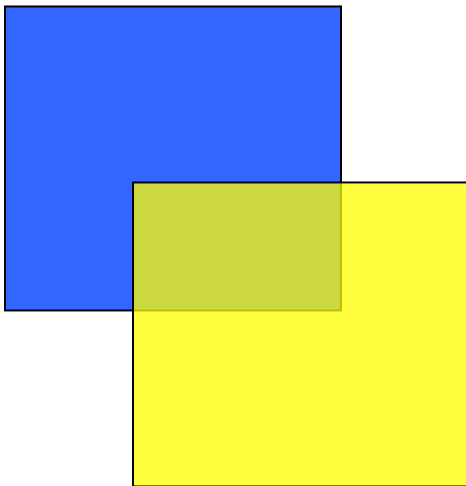


Bases de Datos Geográficas

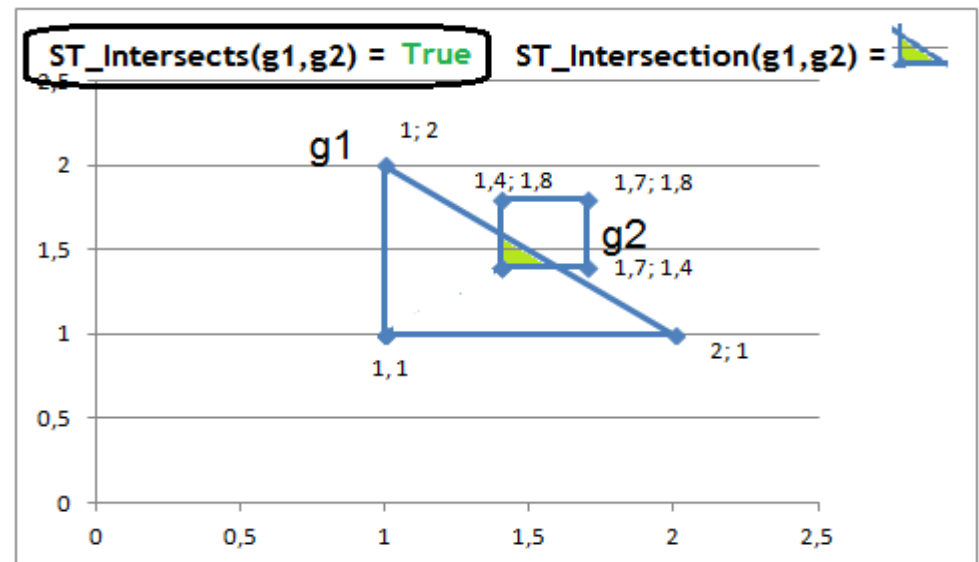
Analisis Espacial - Intersecciones

Además contamos con una función booleana que retorna TRUE si a y b se interceptan:

- ✓ ***ST_Intersects***(a,b) esto nos permite hacer Joins y otras operaciones geometrías mas eficientes



ST_Intersects() = TRUE



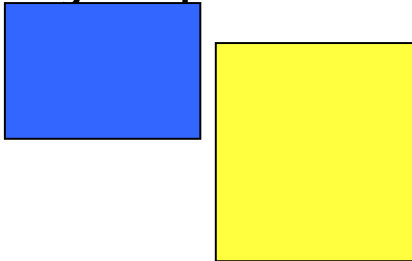
Bases de Datos Geográficas

Analisis Espacial - Disjunto

Una función espacial que podemos considerar opuesta a Intersects es Disjoint:

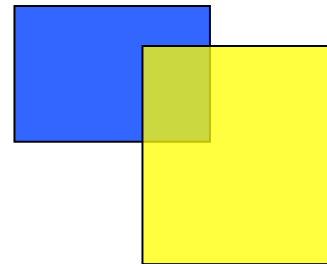
- ✓ ***ST_Disjoint***(a,b) retorna un True si a y b no se intersectan.

Ejemplos:



ST_Disjoint() = TRUE

ST_Intersects() = FALSE



ST_Disjoint() = FALSE

ST_Intersects() = TRUE

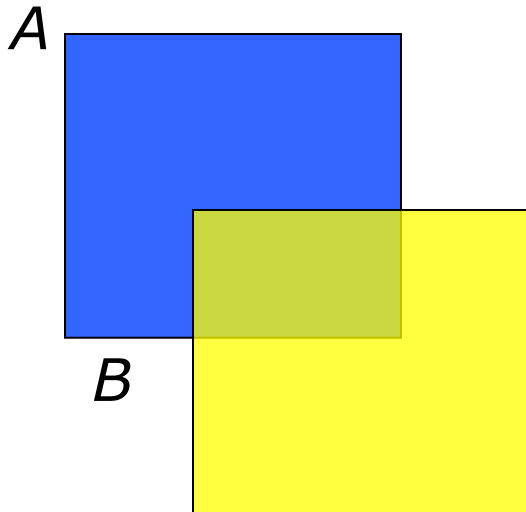
Bases de Datos Geográficas

Analisis Espacial - Diferencia

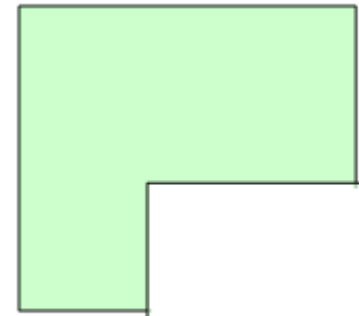
La función siguiente devuelve la diferencia de las dos geometrías:

- ✓ ***ST_difference***(A,B) retorna la geometría diferencia de $A - B$.

Nota: Operación NO Conmutativa.



ST_difference()



Bases de Datos Geográficas

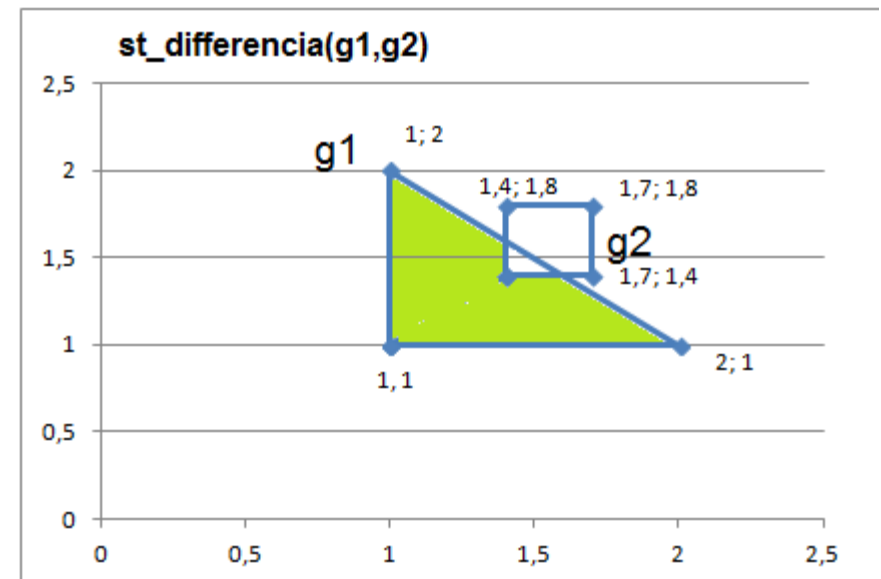
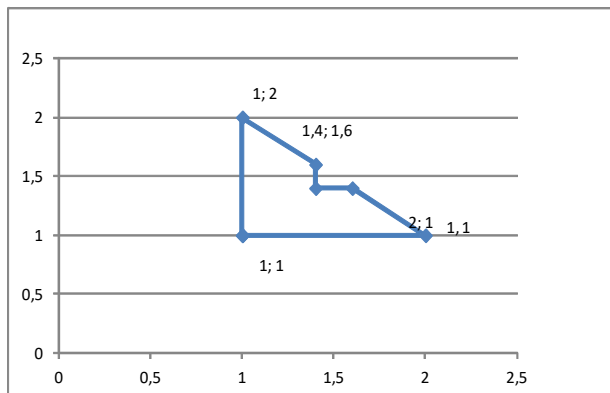
Analisis Espacial - Diferencia

```
select st_difference(g1, g2) from (SELECT  
st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))')  
as g1,  
st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,  
1.7 1.4,1.4 1.4))') as g2)
```

AS TEMP

st_astext
text

POLYGON((1 1,1 2,1.4 1.6,1.4 1.4,1.6 1.4,2 1,1 1))

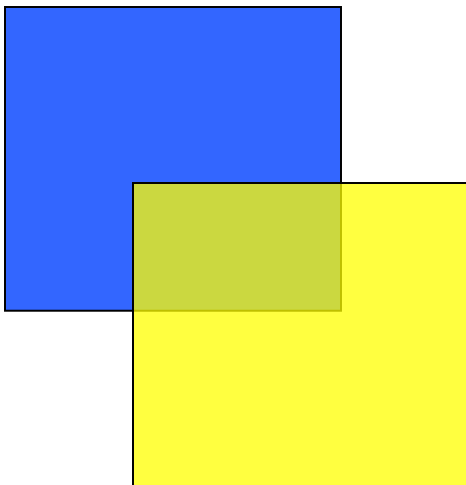


Bases de Datos Geográficas

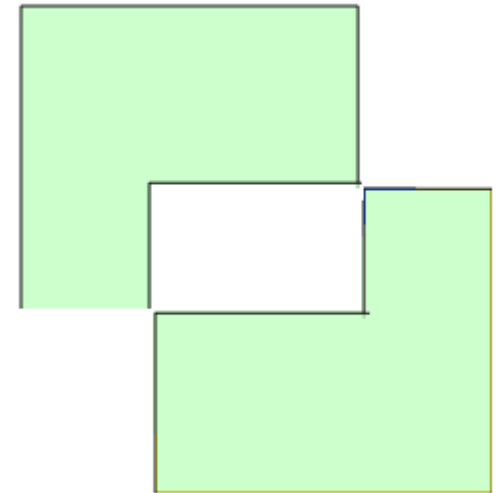
Analisis Espacial – Diferencia Simetrica

Esta Función representa la diferencia simétrica:

- ✓ ***ST_SymDifference***(A,B) se llama dif simétrica por $St_symdiferencia(A,B) = St_symdiferencia(B,A)$.



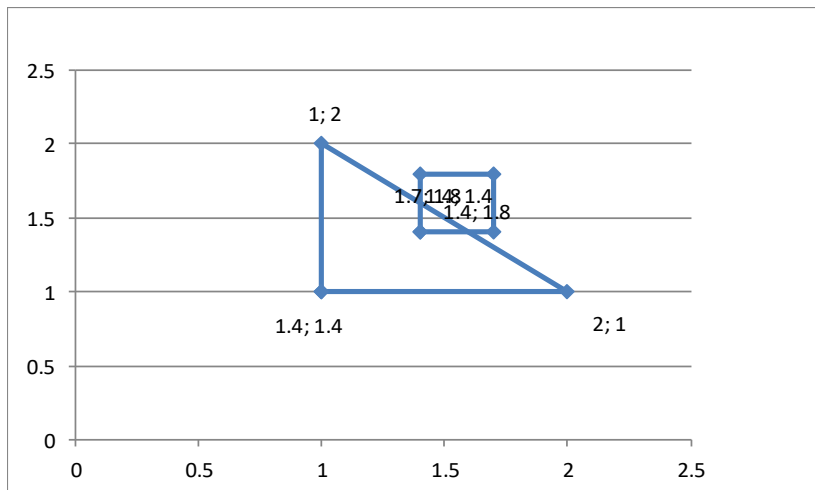
ST_Symdifference()



Bases de Datos Geográficas

Analisis Espacial – Diferencia Simétrica

```
select st_astext(st_SymDifference(g2,g1)) st_SymDifference_g1_g2
from
  (SELECT st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))') g1,
    st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7
      1.8,1.7 1.4,1.4 1.4))') as g2) AS TEMP
```



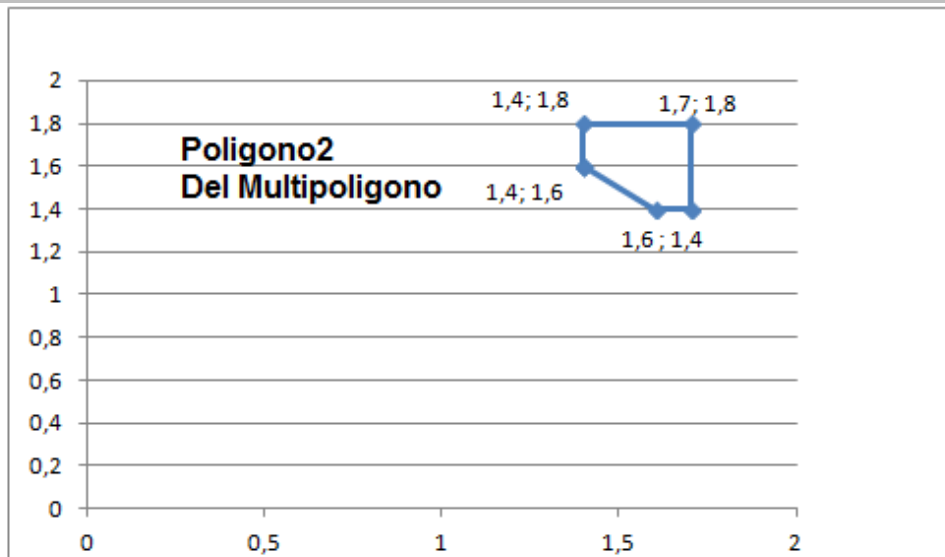
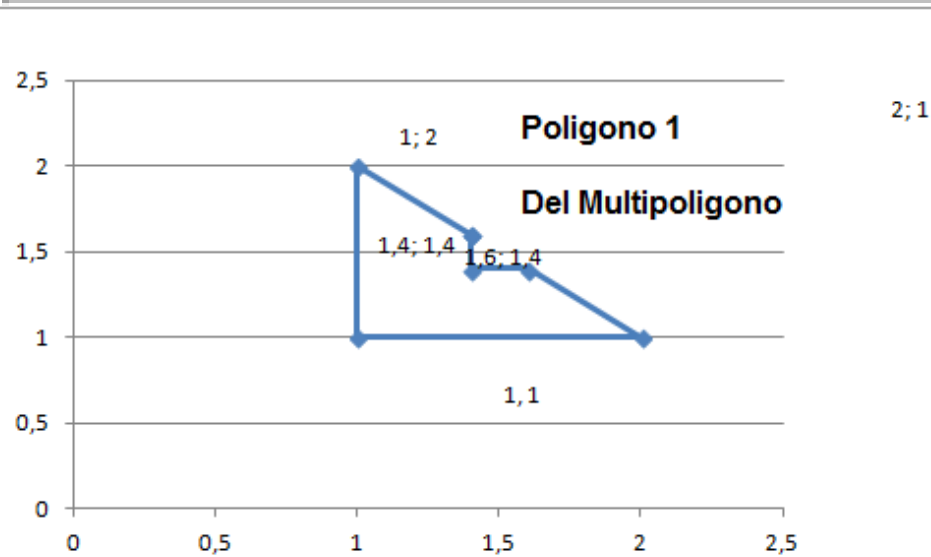
Bases de Datos Geográficas

Analisis Espacial – Diferencia Simétrica

```
select st_astext(st_SymDifference(g2,g1)) st_SymDifference_g1_g2
from
  (SELECT st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))') g1,
    st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7
      1.8,1.7 1.4,1.4 1.4))') as g2) AS TEMP
```

st_symdifference_g1_g2
text

MULTIPOLYGON(((1.4 1.6,1.4 1.4,1.6 1.4,2 1,1 1,1 2,1.4 1.6)),((1.4 1.6,1.4 1.8,1.7 1.8,1.7 1.4,1.6 1.4,1.4 1.6)))



Bases de Datos Geográficas

Analisis Espacial – Otras Funciones

Veamos las operaciones Básicas de conjunto, todas generan geometrías:

Devuelven Tipos Varios

- ✓ ***ST_Equals*** *retorna True si tienen la misma geometría.*
- ✓ ***ST_Distance*** *retorna la mínima distancia entre dos geometrías.*
- ✓ ***ST_Buffer*** *calcula la geometría de un buffer de tamaño alrededor de la geometría g.*
- ✓ ***St_ConvexHull*** *Genera la geometría que es el casco convexo de g.*

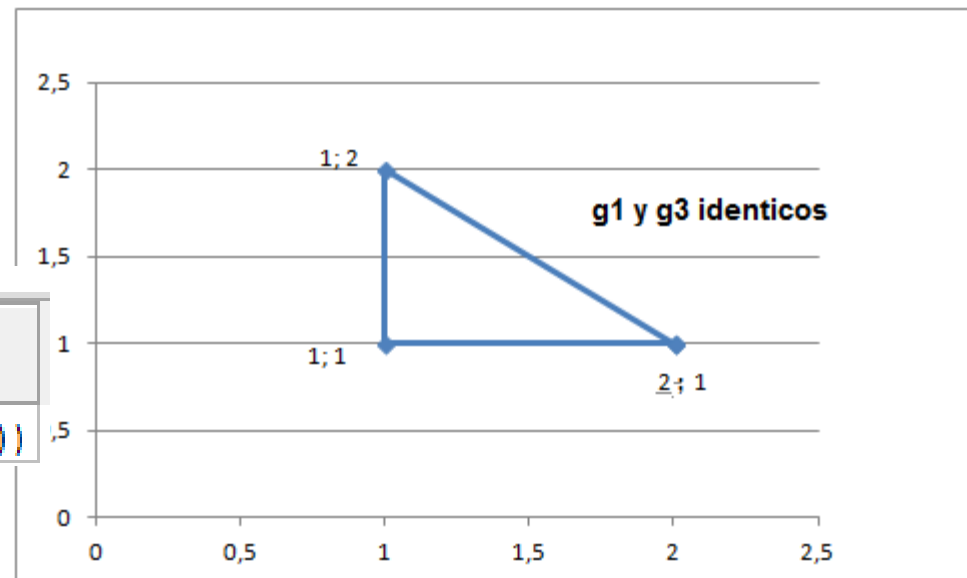
Bases de Datos Geográficas

Analisis Espacial – Igualdad

```
select st_equals(g3,g1) st_equals_g2_g1, st_overlaps(g3,g1)
st_overlaps_g3_g1, st_astext(g1) g1, st_astext(g3) g3 from
(SELECT  st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))')g1,
        st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))')g3 )
AS TEMP;
```

st_equals_g2_g1 boolean	st_overlaps_g3_g1 boolean
t	f

g1 text	g3 text
POLYGON((1 1,1 2,2 1,1 1))	POLYGON((1 1,1 2,2 1,1 1))

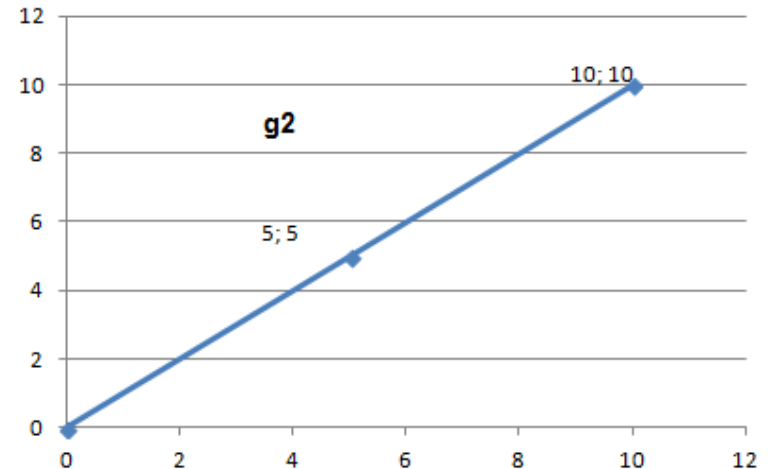
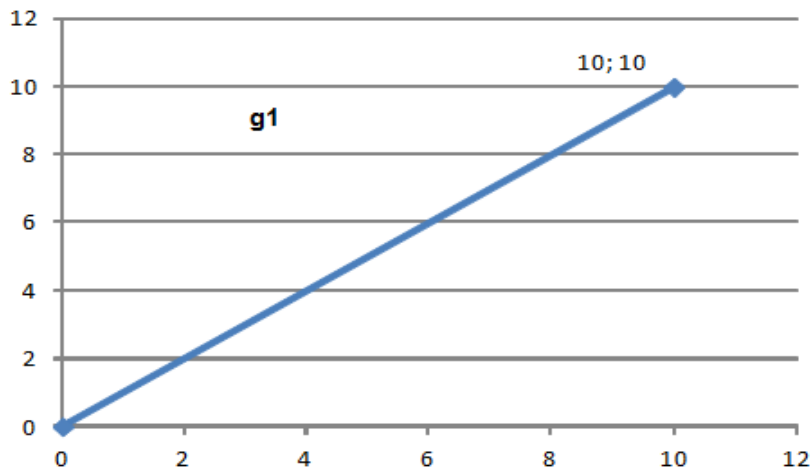


Bases de Datos Geográficas

Analisis Espacial – Igualdad

```
select st_equals(g2,g1) st_equals_g2_g1, st_overlaps(g3,g1),  
       st_astext(g1) g1, st_astext(g2) g2 from  
(SELECT ST_GeomFromText ('LINESTRING (0 0, 10 10)') g1,  
       ST_GeomFromText ('LINESTRING (0 0, 5 5, 10 10)') g2)  
AS TEMP;
```

st_equals_g2_g1 boolean	st_overlaps_g2_g1 boolean	g1 text	g2 text
t	f	LINESTRING(0 0,10 10)	LINESTRING(0 0,5 5,10 10)



Bases de Datos Geográficas

Analisis Espacial – Casco Convexo

El casco convexo de una geometría representa la geometría convexa mínima que encierra todas las geometrías dentro del conjunto:

ST_ConvexHull(**geo**) → retorna una geométrica que representa el casco convexo de geo.



Casco convexo de una MultiLineString y un MultiPoint visto junto con MultiLineString y MultiPoint

```
SELECT ST_AsText (ST_ConvexHull (
    ST_Collect (
        ST_GeomFromText ('MULTILINESTRING ((100 190,10 8), (150 10, 20 30))'),
        ST_GeomFromText ('MULTIPOINT (50 5, 150 30, 50 10, 10 10)')
    ));

--- st_astext---
POLYGON ((50 5,10 8,10 10,100 190,150 30,150 10,50 5))
```

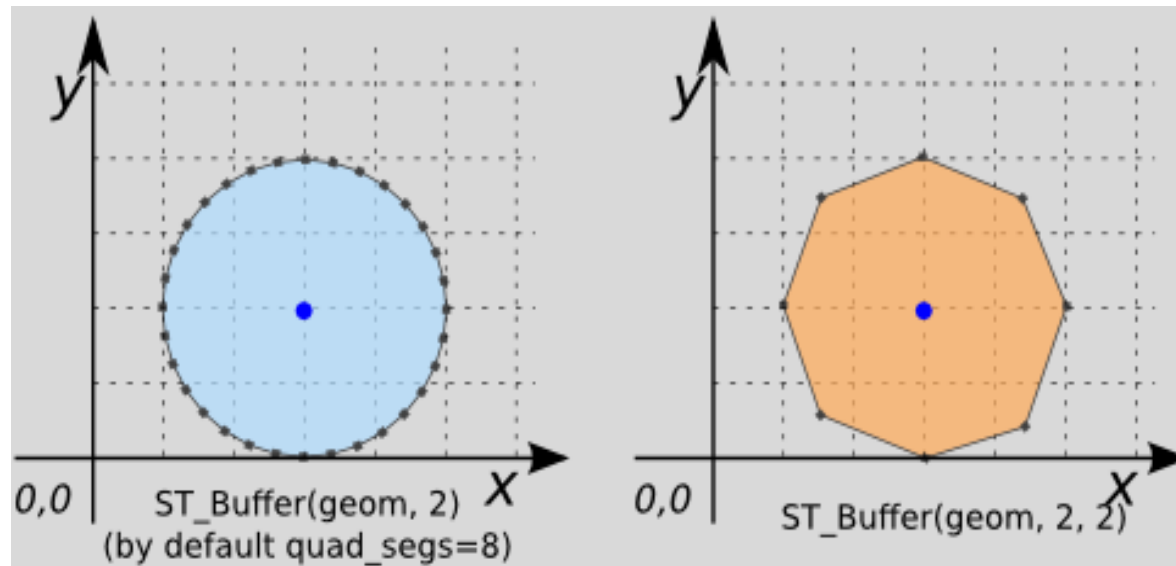
Bases de Datos Geográficas

Analisis Espacial – Buffer

El `St_Buffer` devuelve una geometría que cubre todos los puntos dentro de una distancia dada de la geometría de entrada:

`ST_Buffer(geo, radio)` → retorna una geométrica.

Nota: La función tiene un tercer parametro que permite cambiar el redondeo o el contorno de la cobertura.



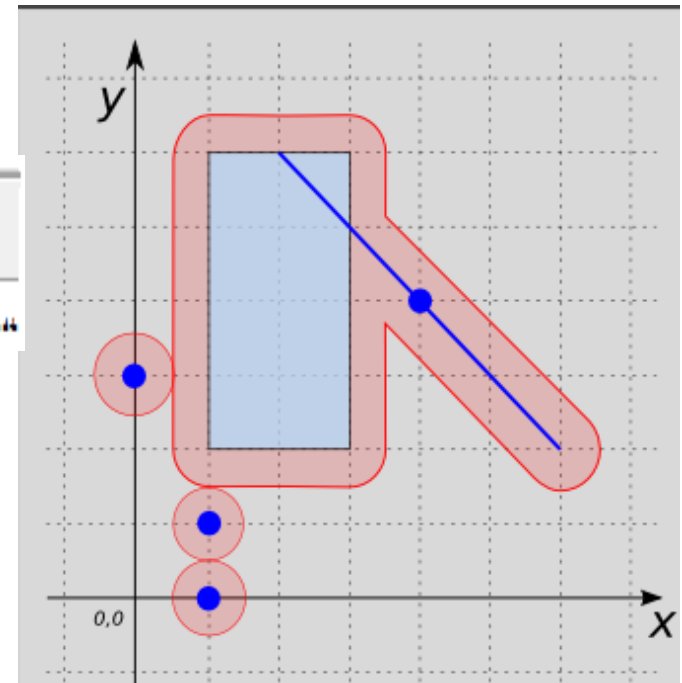
Para ver este parametro se sugiere ver la documentación:
http://www.postgis.net/docs/ST_Buffer.html

Bases de Datos Geográficas

Analisis Espacial – Buffer

```
SELECT st_astext(st_buffer(coll1,0.5)) FROM
(SELECT ST_Collect(ST_Collect(g1,g2),g3) as coll1 from
(SELECT
  ST_GEOFROMTEXT('MULTIPOINT((4 4),(1 1),(1 0),(0 3))') g1,
  ST_GEOFROMTEXT('LINESTRING(2 6, 6 2)') g2,
  ST_GEOFROMTEXT('POLYGON((1 2, 3 2, 3 6, 1 6, 1 2))') g3
) AS TEMP) AS TEMP2;
```

st_astext
text
MULTIPOLYGON(((0.5 2,0.5 3,0.5 6,0.509607359798385 6.09754516100806,0.53'...

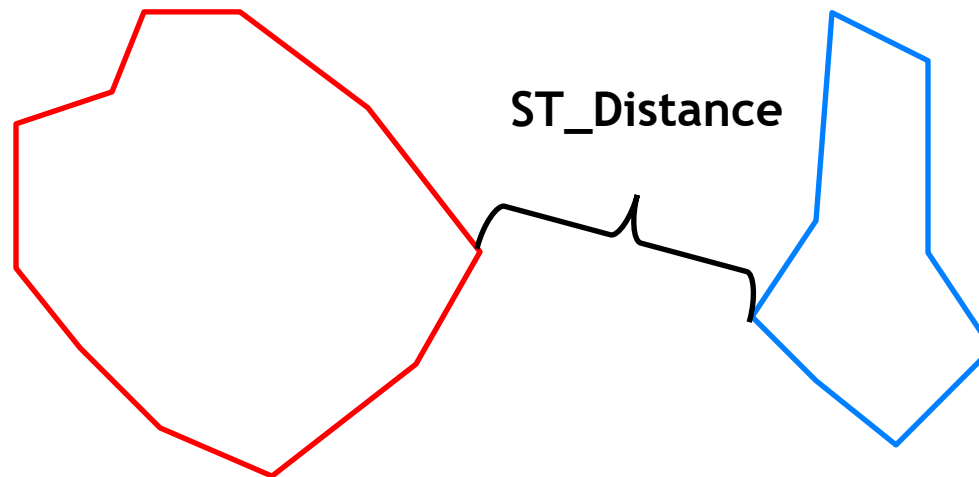


Bases de Datos Geográficas

Analisis Espacial – Distancia

La función distancia devuelve la magnitud de la mínima distancia entre ambas geometrías:

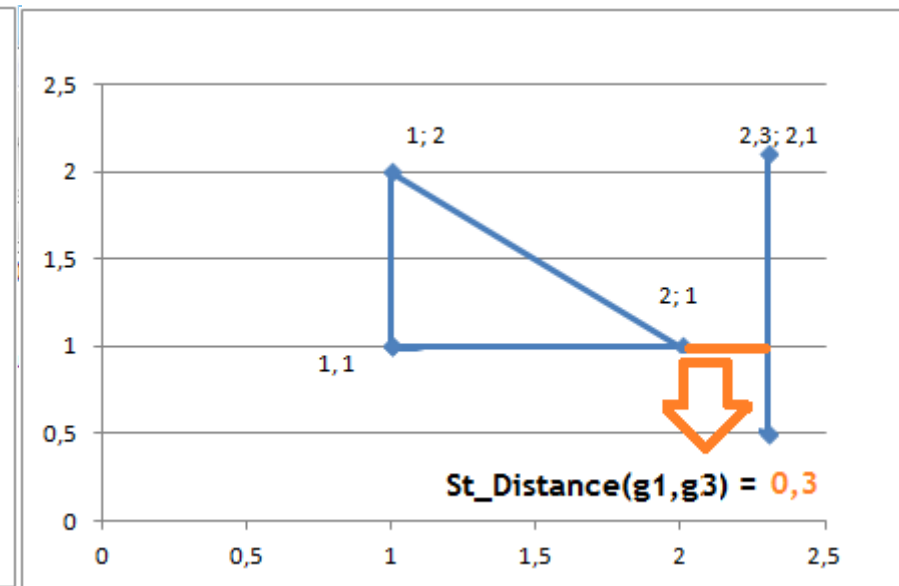
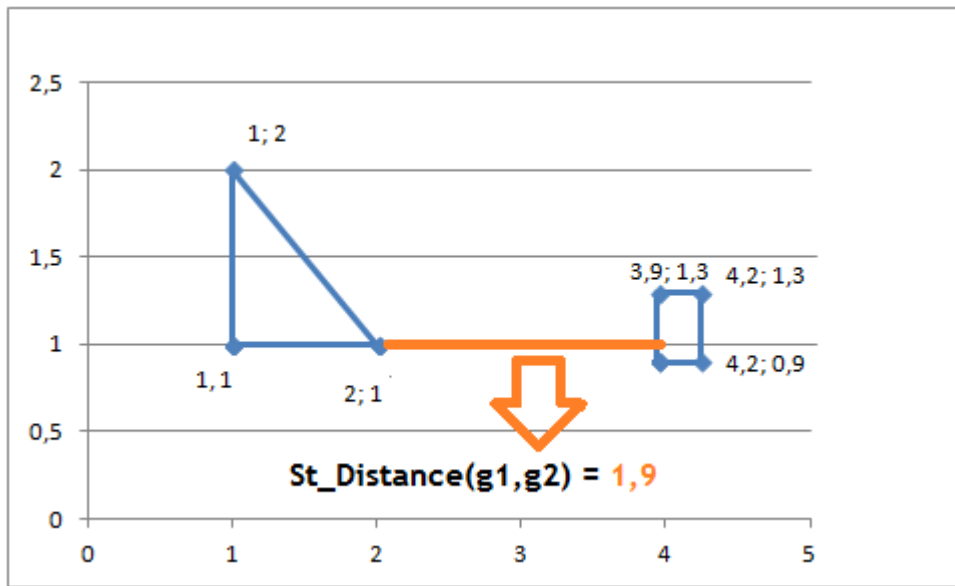
- ✓ ***ST_Distance***(**A**, **B**) → retorna la distancia mínima entre A y B.



Bases de Datos Geográficas

Analisis Espacial - Distancia

```
select st_Distance(g1, g2), st_Distance(g1, g3) from
( SELECT st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))') as g1,
  st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,
                                1.7 1.4,1.4 1.4))') as g2,
  st_geomfromtext('LineString(2.3 0.5, 2.3 2.1)') as g3
)
AS TEMP
```

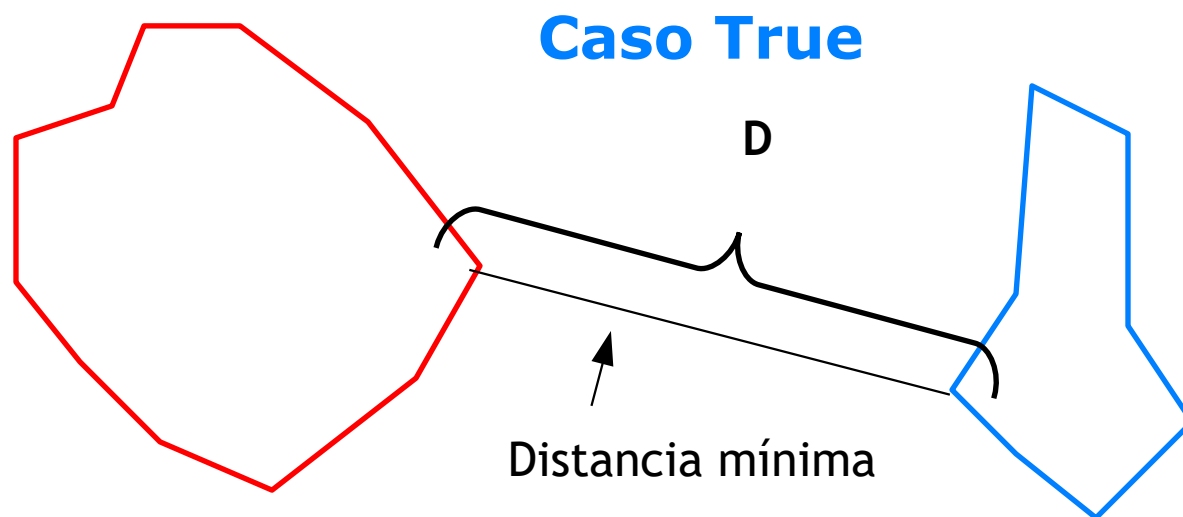


Bases de Datos Geográficas

Analisis Espacial – Dentro de Distancia

La función `Dwithin` es una función lógica, que devuelve verdadero sí las dos geometrías están a menor distancia que la especificada como tercer argumento:

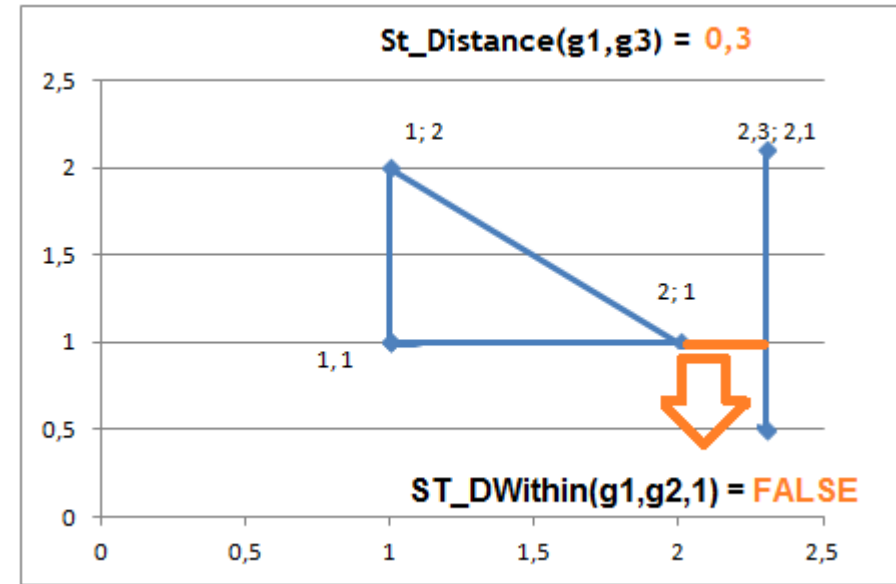
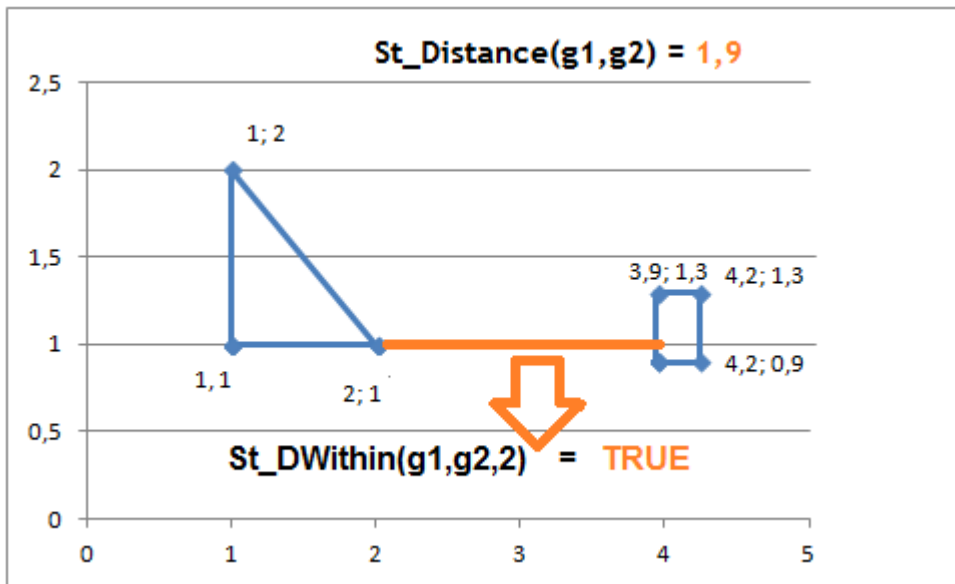
- ✓ ***ST_DWithin***(**A**, **B**, **D**) → **A** a distancia < **D** de **B**, retorna un boolean que asociado a esta premisa.



Bases de Datos Geográficas

Analisis Espacial – Dentro de Distancia

```
select st_DWithin(g1,g2,2),st_DWithin(g1,g3,1) from
( SELECT st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))') as g1,
  st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,
    1.7 1.4,1.4 1.4))') as g2,
  st_geomfromtext('LineString(2.3 0.5, 2.3 2.1)') as g3
)
AS TEMP
```

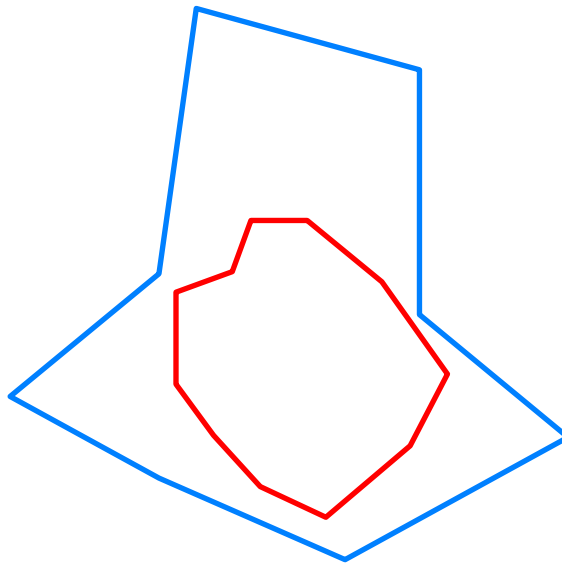


Bases de Datos Geográficas

Analisis Espacial – Contiene / Dentro

Existen 2 funciones íntimamente asociadas inclusiones:

- ✓ ***ST_Contains***(**A**, **B**) → **A** Contiene íntegramente a **B**, retorna un boolean asociado a esta premisa.
- ✓ ***ST_Within***(**B**, **A**) **B** Dentro de **A**, retorna un boolean asociado a esta premisa.



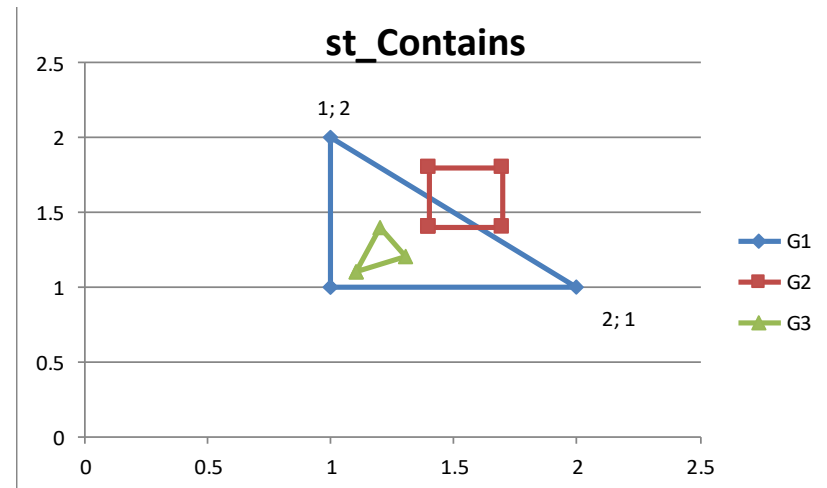
Bases de Datos Geográficas

Analisis Espacial – Contains

```
select st_contains(g1,g2) ,st_contains(g1,g3) from
(SELECT st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))')g1,
st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7
1.8,1.7 1.4,1.4 1.4))') as g2,
st_geomfromtext('Polygon((1.1 1.1,1.3 1.2,1.2
1.4,1.1 1.1))') as g3 ) AS TEMP
```

g1 text	g2 text	g3 text
POLYGON((1 1,1 2,2 1,1 1))	POLYGON((1.4 1.4,1.4 1.8,1.7 1.8,1.7 1.4,1.4 1.4))	POLYGON((1.1 1.1,1.3 1.2,1.2 1.4,1.1 1.1))

st_contains_g1_g2 boolean	st_contains_g1_g3 boolean
f	t

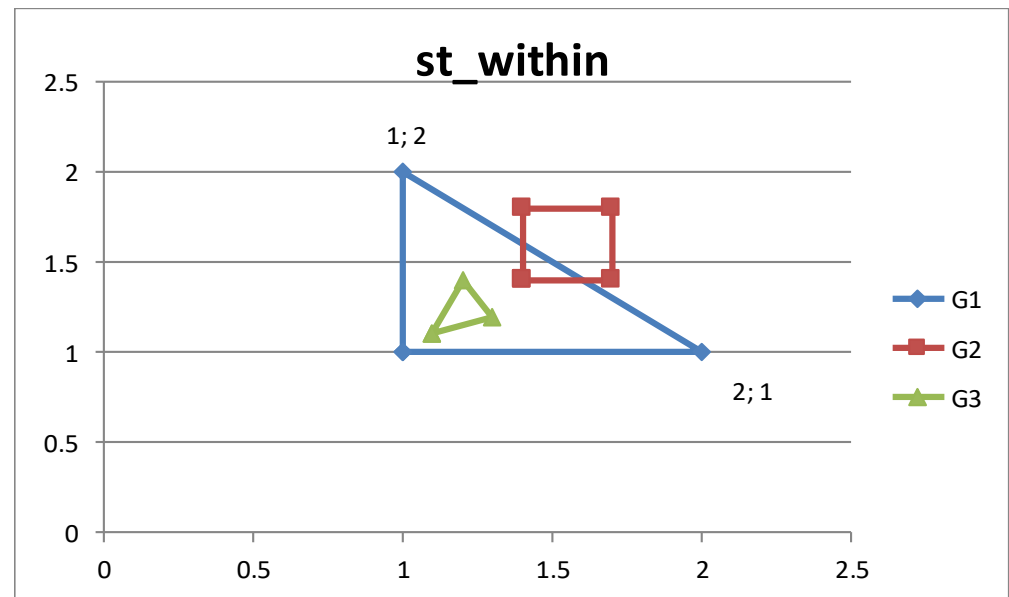


Bases de Datos Geográficas

Analisis Espacial – Within

```
select st_within(g2,g1) ,st_within(g3,g1) from
  (SELECT st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))')g1,
    st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7
      1.8,1.7 1.4,1.4 1.4))') as g2,
    st_geomfromtext('Polygon((1.1 1.1,1.3 1.2,1.2
      1.4,1.1 1.1))') as g3 ) AS TEMP
```

G1		G2		G3	
X	Y	X	Y	X	Y
1	1	1.4	1.4	1.1	1.1
1	2	1.4	1.8	1.3	1.2
2	1	1.7	1.8	1.2	1.4
1	1	1.7	1.4	1.1	1.1
		1.4	1.4		



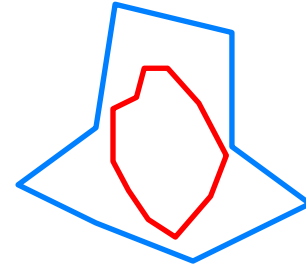
st_within_g2_g1	st_within_g3_g1
boolean	boolean
f	t

Bases de Datos Geográficas

Analisis Espacial – Superposición

La función Overlaps devuelve TRUE si las Geometrías comparten espacio y no son de la misma dimensión, y no están completamente contenidas entre sí.

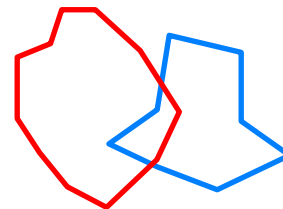
✓ ***ST_Overlaps***(**A**, **B**) → FALSE



✓ ***ST_Overlaps***(**B**, **Bprima**) → FALSE



✓ ***ST_Overlaps***(**B**, **A**) → TRUE



Bases de Datos Geográficas

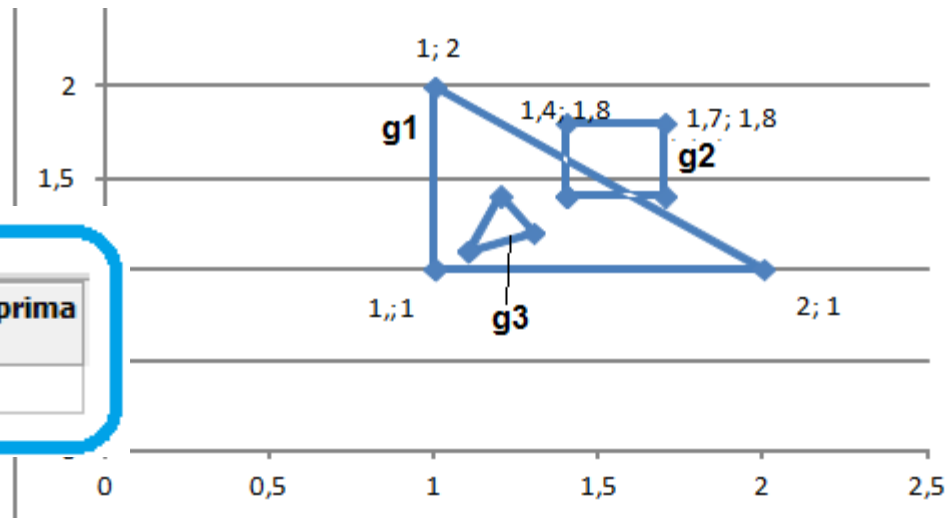
Analisis Espacial – Superposición

```
select st_overlaps(g2,g1) ,st_overlaps(g3,g1) from
(SELECT st_geomfromtext('Polygon((1 1,1 2,2 1,1 1))') g1,
st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7
1.8,1.7 1.4,1.4 1.4))') as g2,
st_geomfromtext('Polygon((1.1 1.1,1.3 1.2,1.2
1.4,1.1 1.1))') as g3 ) AS TEMP
```

g1 text	g2 text	g3 text
POLYGON((1 1,1 2,2 1,1 1))	POLYGON((1.4 1.4,1.4 1.8,1.7 1.8,1.7 1.4,1.4 1.4))	POLYGON((1.1 1.1,1.3 1.2,1.2 1.4,1.1 1.1))

g1prima text
POLYGON((1 1,1 2,2 1,1 1))

overlaps_g1_g2 boolean	overlaps_g1_g3 boolean	overlaps_g1_g1prima boolean
t	f	f



Bases de Datos Geográficas

Caso de Estudio Empresa de Seguros

Después de una reciente inundación en río X, una compañía de seguros quiere corregir la información sobre asegurados que están en la zona de la inundación, y representan un riesgo creciente para la compañía.

La base de datos contiene una tabla de ríos que contiene los ríos y sus zonas de inundación y otra tabla de edificios con los datos de los hogares de los titulares de pólizas.

Las tablas de la Base de datos tienen la siguiente definición:

```
rivers(name, water_amount, river_line, flood_zones)
buildings(customer_name, street, city, zip, ground_plot)
```

La columna `river_line` (`LINEA_RIO`) contiene los linestrings que representa el cauce segmentado de los ríos.

La columna de `zonas_de_inundación` (`flood_zones`) muestra esta área para cada río. El Terreno de cada edificio se almacena en parcela.

Bases de Datos Geográficas

Caso de Estudio Empresa de Seguros

Las tablas se crean con los sql siguientes.

```
CREATE TABLE rivers (  
  name VARCHAR(30) PRIMARY KEY,  
  water_amount DOUBLE PRECISION,  
  river_line ST_LineString,           --cauce segmentado del rio  
  flood_zones ST_MultiPolygon);      --zonas de inundación
```

```
CREATE TABLE buildings (  
  customer_name VARCHAR(50) PRIMARY KEY,  
  street VARCHAR(50),  
  city VARCHAR(20),  
  zip VARCHAR(10),  
  ground_plot ST_Polygon);           --terreno (geometría)
```

Bases de Datos Geográficas

Caso de Estudio Empresa de Seguros

La primera tarea es actualizar la información sobre las zonas de inundación. Las zonas de inundación para cada río se considera que se va a extender por 2 kilómetros en cada dirección. La función **ST_Buffer** se utiliza para este fin y la instrucción SQL siguiente para extender las zonas de inundación por el radio especificado.

```
UPDATE rivers
    SET flood_zones = ST_Buffer(flood_zones,2000)
WHERE name = 'X'
```

En el siguiente paso, la empresa quiere encontrar todos los clientes que están ahora en la zona de inundaciones de cada río. Usamos la función espacial **ST_Intersects** para encontrar todos esos edificios.

```
SELECT customer_name, street, city, zip
    FROM buildings AS b, rivers AS r
WHERE ST_INTERSECTS(b.ground_plot,r.flood_zones) = TRUE
```

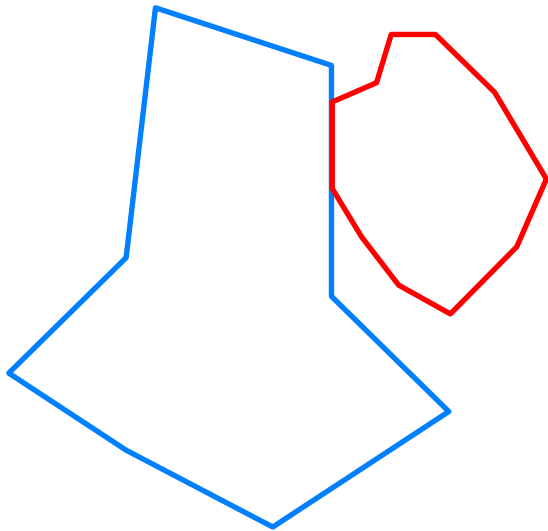
Los clientes pueden ser informados de Cualquier cambio en sus pólizas con las direcciones recuperadas en el sql anterior.

Bases de Datos Geográficas

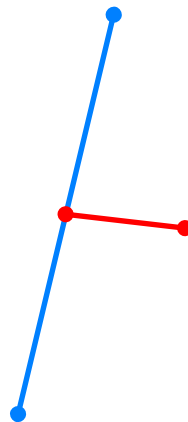
Analisis Espacial – Toca (Linda)

Esta función tiene este cometido:

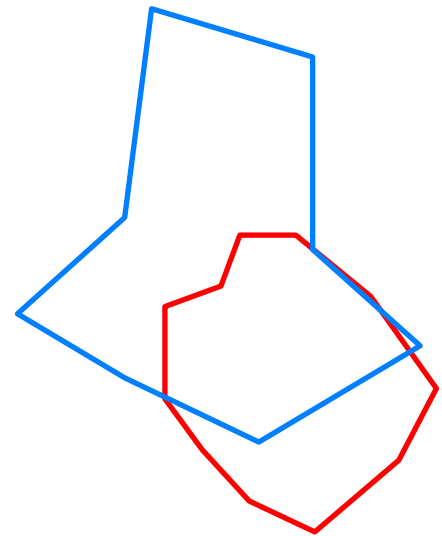
- ✓ ***ST_Touches***(**A**, **B**) → **A** toca a **B**, retorna un boolean asociado a esta premisa.



✓ ***Verdadero***



✓ ***Verdadero***



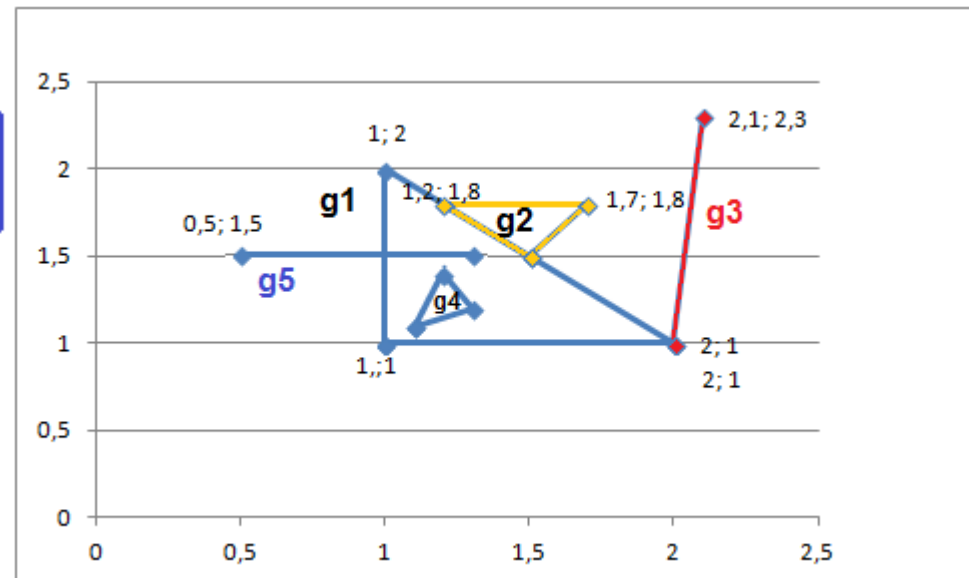
✓ ***Falso***

Bases de Datos Geográficas

Analisis Espacial – Toca (Linda)

```
Select st_Touches(g1,g2), st_Touches(g3,g1),
       st_Touches(g1,g4), st_Touches(g1,g5)           from
(SELECT st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))') as g1,
       st_geomfromtext('Polygon((1.4 1.4,1.4 1.8, 1.7 1.8,
                                1.7 1.4,1.4 1.4))') as g2,
       st_geomfromtext('LineString(2 1, 2.3 2.1)') as g3,
       st_geomfromtext('Polygon((1.1 1.1,1.3 1.2,1.2 1.4,1.1
                                1.1))') as g4 ) AS TEMP
```

touche_g1g2 boolean	touche_g1g3 boolean	touche_g1g4 boolean	touche_g1g5 boolean
t	t	f	f

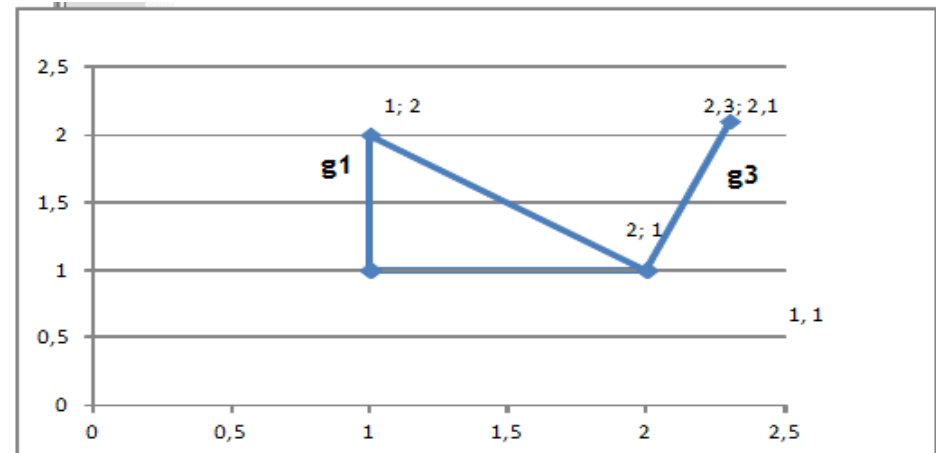
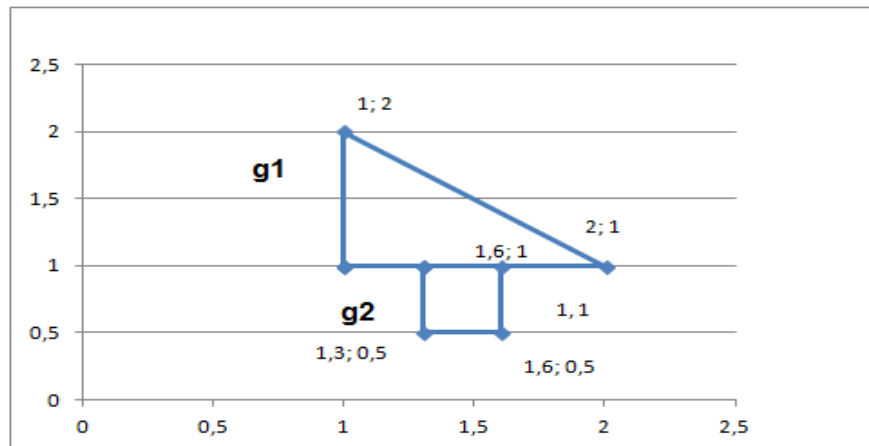


Bases de Datos Geográficas

Analisis Espacial – Toca (Linda)

```
select st_Touches(g1,g2), st_Touches(g1,g3) from ( SELECT
st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))') as g1,
      st_geomfromtext('Polygon((1.3 1,1.6 1,1.6 0.5,
      1.3 0.5,1.3 1))') as g2,
      st_geomfromtext('LineString(2 1, 2.3 2.1)') as g3 )
AS TEMP
```

st_touches boolean	st_touches boolean	g1 text	g2 text	g3 text
t	t	POLYGON((1 1,1 2,2 1,1 1))	POLYGON((1.3 1,1.6 1,1.6 0.5,1.3 0.5,1.3 1))	LINESTRING(2 1,2.3 2.1)



Bases de Datos Geográficas

Caso de Estudio



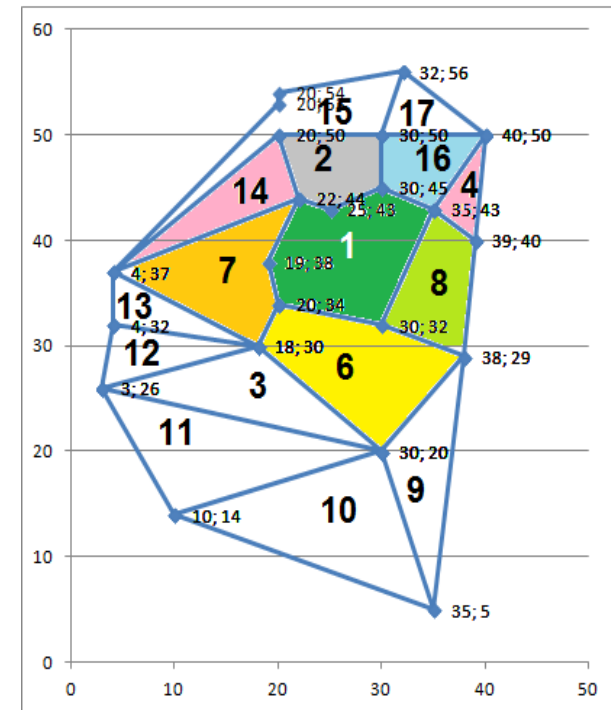
En una base de datos Postgis, se tiene una tabla de departamentos con las sig columnas: nombre, provincia y geometría de cada departamento (Polígono).

```
departamentos(gid, provincia, nombre, supgeom)
```

La dirección de catastro cuenta con una deficiente vectorización con algunos errores Como la que se muestra.

Resuelva:

- 1) ¿Que departamentos de entre ríos lindan con Villaguay, que geometría representa el limite?.



Bases de Datos Geográficas

Caso de Estudio

Definición de la tabla:

`departamentos (gid, provincia, nombre, supgeom)`

Resuelva:

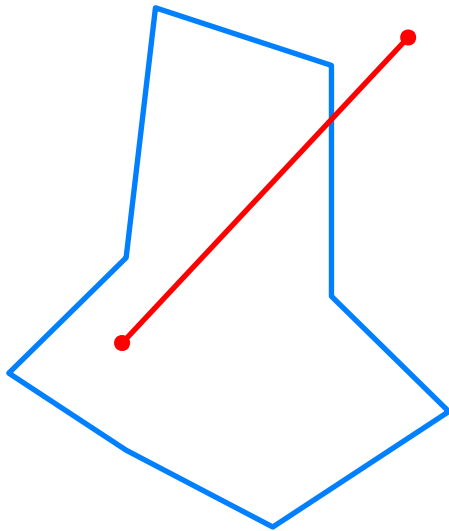
¿Que departamentos de entre ríos
lindan con Villaguay, que geometría
representa el limite?.

Bases de Datos Geográficas

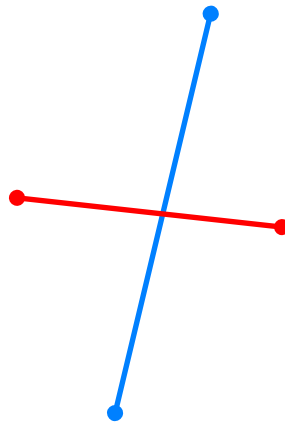
Analisis Espacial – Cruce

Una función interesante :

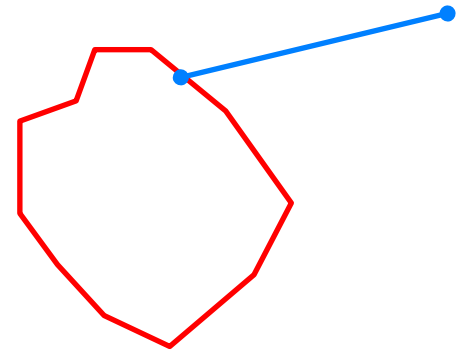
- ✓ ***ST_Crosses***(**A**, **B**) → **A** se Crusa con **B**, retorna un boolean asociado a esta premisa.



✓ ***Verdadero***



✓ ***Verdadero***



✓ ***Falso***

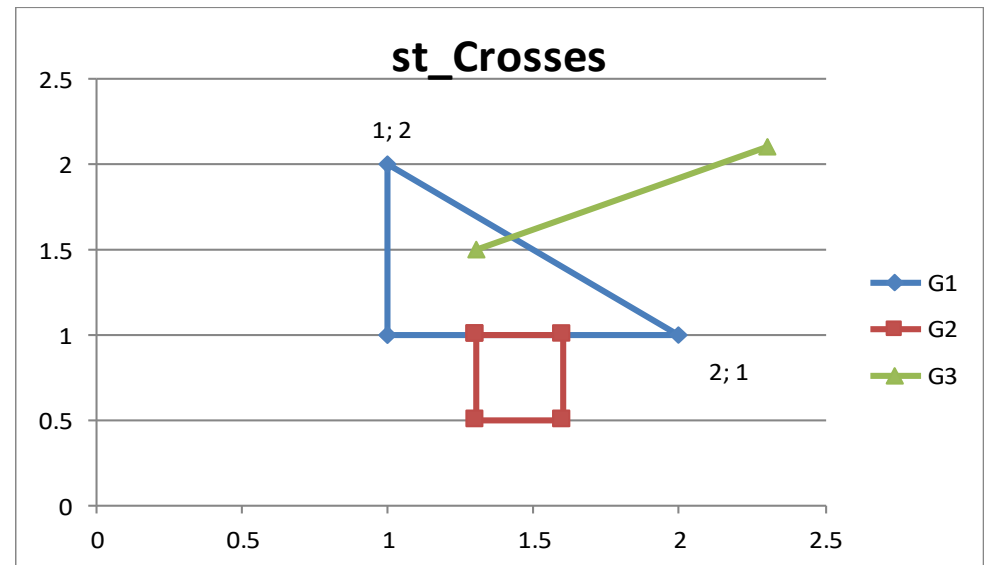
Bases de Datos Geográficas

Analisis Espacial – Cruce

```
select st_Crosses(g1,g3),st_Crosses(g1,g2) from ( SELECT
st_geomfromtext('Polygon((1 1,1 2,2 1, 1 1))') as g1,
      st_geomfromtext('Polygon((1.3 1,1.6 1,1.6 0.5,
                                1.3 0.5,1.3 1))') as g2,
      st_geomfromtext('LineString(1.3 1.5, 2.3 2.1)')as g3 )
AS TEMP
```

crosses_g1_g3 boolean	crosses_g1_g2 boolean
t	f

G1		G2		G3	
X	Y	X	Y	X	Y
1	1	1.3	1	1.3	1.5
1	2	1.6	1	2.3	2.1
2	1	1.6	0.5		
1	1	1.3	0.5		
		1.3	1		



Bases de Datos Geográficas

Caso de Estudio

En una base de datos Postgis, se tiene una tabla de ciudades con las nombre y geometría de cada ciudad (Poligono). Además Río tiene entre otras un Multilinestring que representa el cauce y un poligono que representa la zona de inundación.

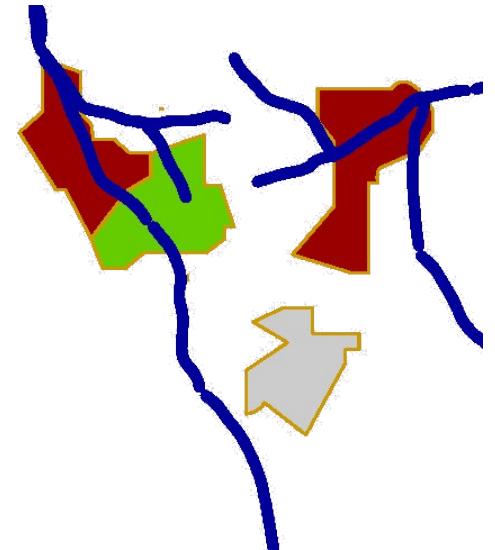
```
ciudades(nombre, cantidadHabitantes, geometria)
```

```
Rios(nombre, lineaCauce, zonaInundacion)
```

Ejemplo:

¿Qué ciudades son atravesadas por Ríos?

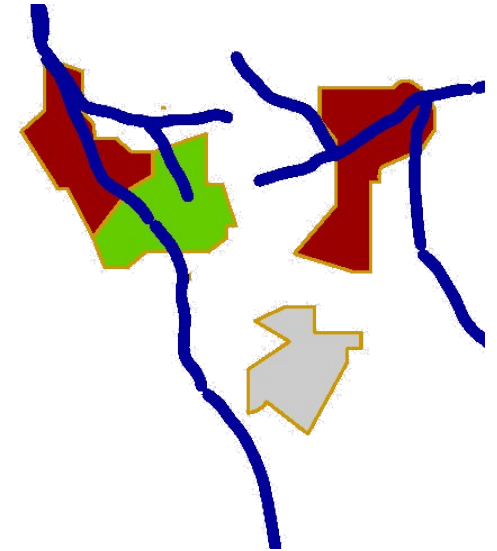
¿Como resolvemos esto?.



Bases de Datos Geográficas

Caso de Estudio

La función ST_Crosses permite saber si las geometrias se cruzan (cauce) y geometria de la superficie de la ciudad.



```
SELECT Ciudades.nombre  
FROM Ciudades JOIN Rios ON  
    ST_CROSSES(Ciudades.geometria, Rios.lineaCauce) = TRUE
```

Bases de Datos Geográficas

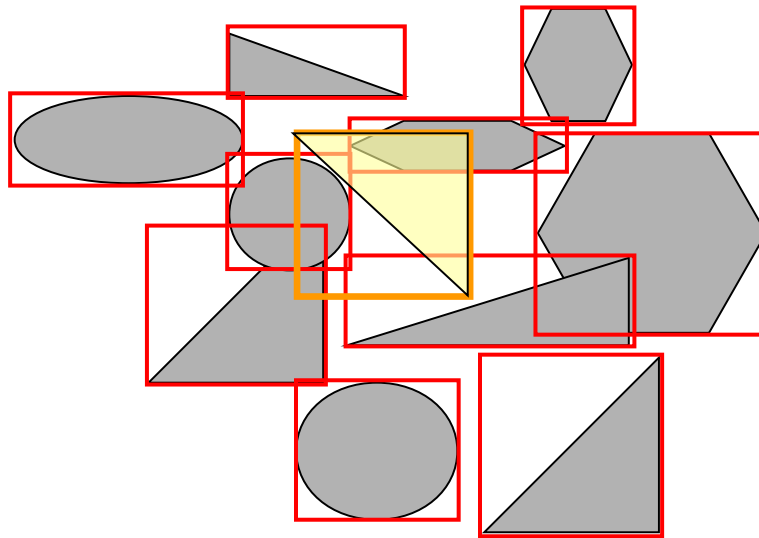
Creando Indices GIST

PostGIS Implementa Indices R-Tree implementado el sistema de indexación GiST

Organiza estos como rectángulos MBR mínimos anidados para agilizar las búsquedas.

```
CREATE INDEX rutas_indice_gist  
  ON calles USING GIST (atributo_geografico);
```

*Agrega el
tipo indice GIST*
USING GIST



Bases de Datos Geográficas

Caso de Estudio Banco

Un Banco tiene sucursales distribuidas geográficamente y administra sus clientes. Cada cliente puede tener una o más cuentas, y cada cuenta es administrada por una sucursal del banco. El banco realiza un análisis de sus clientes, para mejorar la calidad de servicios. Este análisis implica también controlar sus componentes espaciales, las ubicaciones de los edificios del cliente y de las sucursales del banco.

Las tablas de la Base de datos del banco tienen la siguiente definición:

```
CREATE TABLE accounts (                                --cuentas bancarias
    account_id INTEGER PRIMARY KEY,                    --id de cuenta
    routing_no INTEGER NOT NULL,
    customer_id INTEGER NOT NULL,                      --idCliente
    branch_id INTEGER NOT NULL,
    type VARCHAR(10) NOT NULL,                         --tipo de cuenta
    balance DECIMAL(14, 2) NOT NULL,                   --saldo
    FOREIGN KEY(customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY(branch_id) REFERENCES branches(branch_id) );
```


Bases de Datos Geográficas

Caso de Estudio Banco

```
CREATE TABLE customers (  
  customerId INTEGER PRIMARY KEY,  
  name VARCHAR(20),  
  street VARCHAR(25),  
  city VARCHAR(10),  
  state VARCHAR(2),  
  zip VARCHAR(5),  
  type VARCHAR(10),  
  location ST_Point);
```

```
CREATE TABLE branches (  
  branchId INTEGER PRIMARY KEY,  
  name VARCHAR(12),  
  manager VARCHAR(20),  
  street VARCHAR(20),  
  city VARCHAR(10),  
  state VARCHAR(2),  
  zip VARCHAR(5),  
  location ST_Point,  
  zone ST_Polygon);
```

La primera consulta determina todos los clientes con saldo de cuenta superior a \$ 10,000.- en cualquiera de las cuentas y que viven más de 20 millas de distancia de su sucursal.

```
SELECT DISTINCT c.customer_id, c.name  
  FROM customers AS c JOIN accounts AS a ON  
    ( c.customer_id = a.customer_id )  
 WHERE a.balance > 10000 AND a.location.ST_Distance(  
   ( SELECT b.location FROM branches WHERE b.branch_id = a.branch_id ),  
   'MILES') > 20
```

Bases de Datos Geográficas

Caso de Estudio Banco

El banco quiere obtener todas las partes de las zonas de ventas asignadas de las sucursales que se superpongan. No se pretende que haya más de una sucursal asignada a una determinada área, y cualquier duplicado debe ser encontrado y corregido. La consulta recupera los identificadores para cada dos sucursales que tienen una superposición en las zonas y también la zona de superposición, codificada en WKT.

```
SELECT b1.branch_id, b2.branch_id,  
b1.zone.ST_Overlaps(b2.zone).ST_AsText()  
FROM branches AS b1 JOIN branches AS b2 ON  
( b1.branch_id < b2.branch_id )  
WHERE b1.zone.ST_Overlaps(b2.zone).ST_IsEmpty() = 0
```

Bases de Datos Geográficas

Caso de Estudio **Banco**

Para reducir la competencia entre las sucursales para sus propios clientes, el banco desea encontrar a todos los clientes que viven dentro de un radio de 10 millas de una sucursal, que no gestiona sus cuentas. Las cuentas se transferirán a una sucursal más cercana si el cliente está de acuerdo.

```
SELECT c.name, c.phone, b.branch_id
FROM branches AS b, customers AS c
WHERE b.location.ST_Buffer(10, 'MILES').ST_Contains(c.location) = 1
AND NOT EXISTS ( SELECT 1
                  FROM accounts AS a
                  WHERE a.customer_id = c.customer_id
                      AND a.branch_id = b.branch_id )
```

RESUMEN

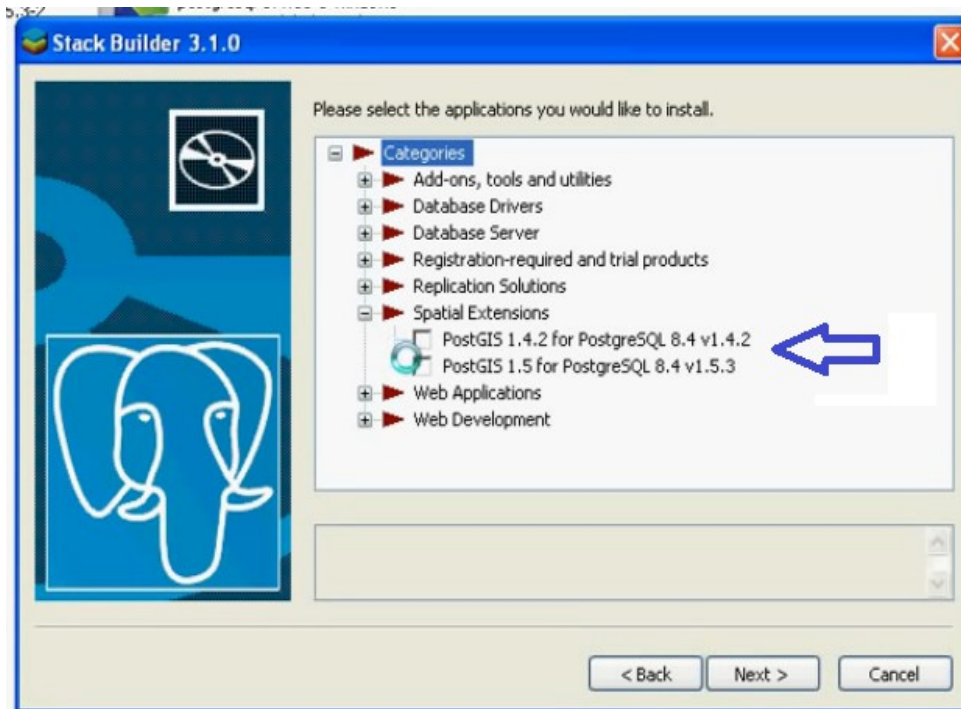
- Características PostGIS: Imp. de Referencia
- Jerarquia GEOMETRY
- PostGIS formatos nativos
- Constructores de TIPOS
- Metodos de la SuperClase GEOMETRY
- Metodos de Analisis Espacial
- **Instalación PostGIS**
- Anexo Transformación WKT to x,y

Bases de Datos Geográficas

Implementación

○ Instalando

Se instala como extensión de PostgreSQL, en el mismo proceso de instalación de este.



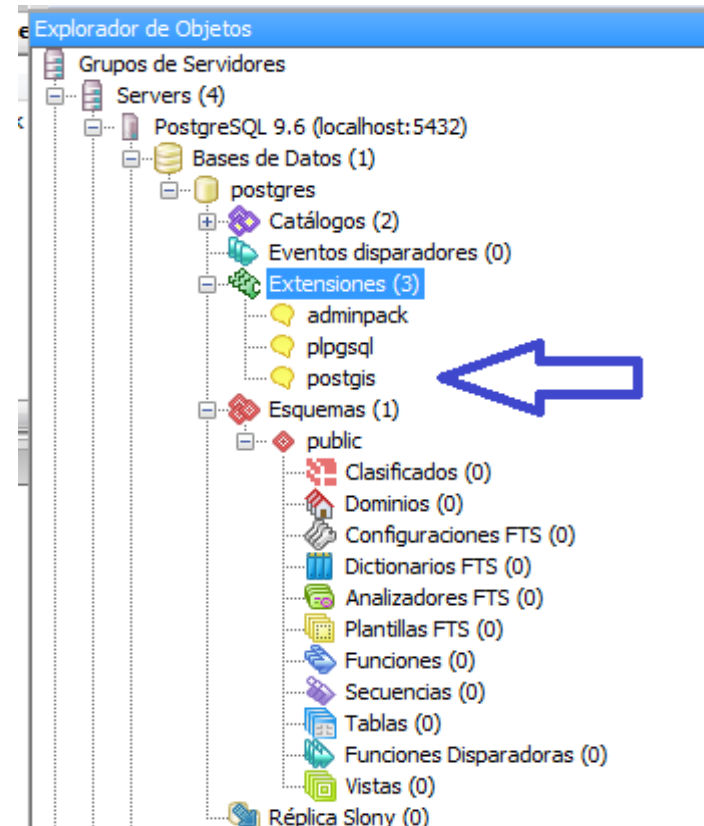
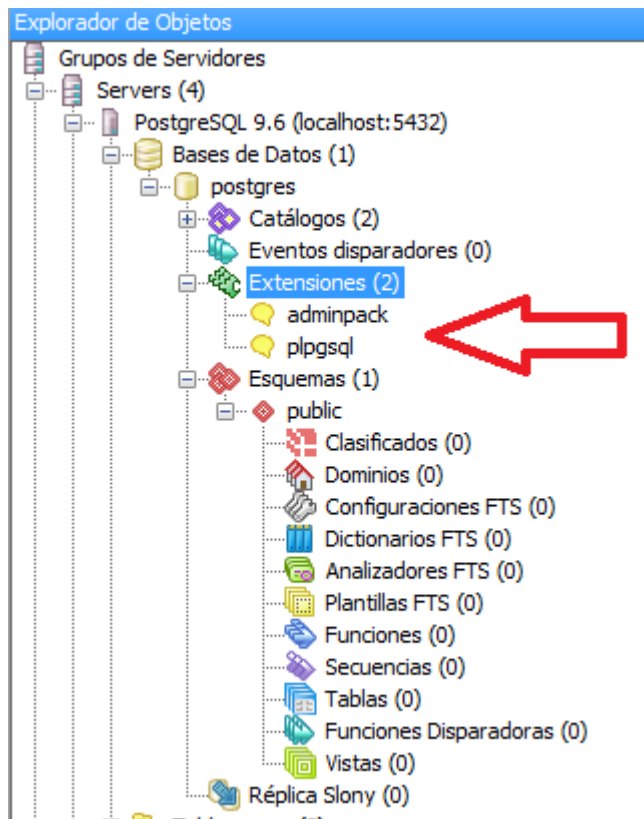
Bases de Datos Geográficas

Implementación stGIS

○ Instalando

Si no esta creada la extensión en la base de datos deseada debemos crearla:

```
CREATE EXTENSION postgis;
```



Bases de Datos Geográficas

Implementación PostGIS

○ Instalando

PostGIS, añade soporte de objetos geográficos, ofreciendo características avanzadas de almacenamiento y análisis de datos gracias a la implementación de un gran número de funciones espaciales que lo convierten en un estándar de facto dentro de los SIG

Es posible trabajar de forma directa con bases de datos PostGIS desde SIG de escritorio como:

QGIS **gvSIG** o incluso **ArcGIS**.

Y en servidores de datos espaciales como:

GeoServer o **MapServer**, y un sinfín de aplicaciones basadas en tecnologías SIG.

Bases de Datos Geográficas

Implementación stGIS

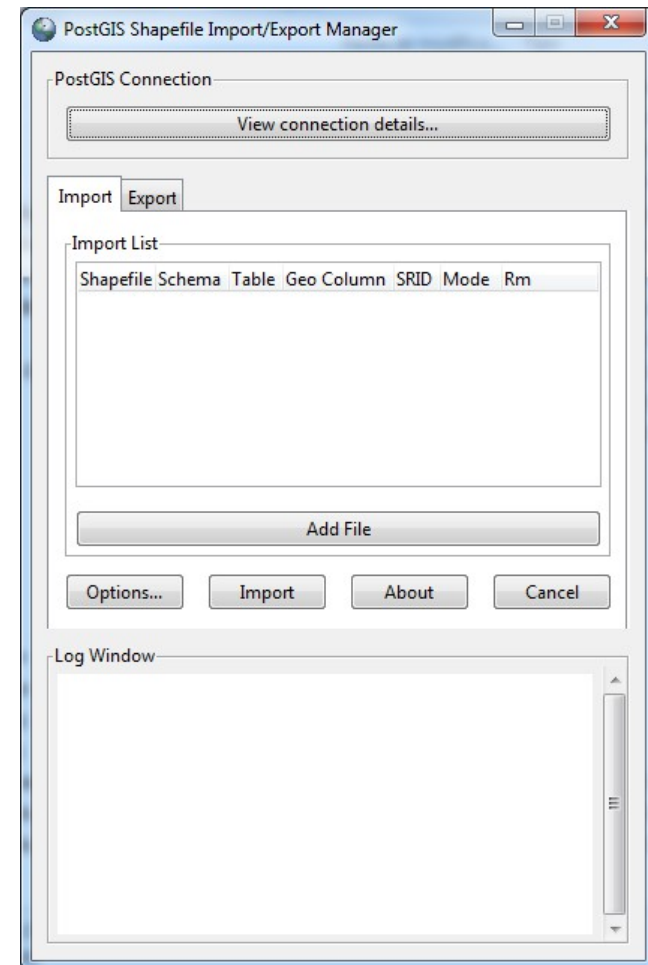
PostGis 2.0 ShapeFile and DBF Loader Export

El proyecto PostGIS tiene una herramienta que permite cargar y exportar información desde y hacia formatos estandarizados.

Es una herramienta gráfica, e intuitiva y se instala desde el instalador de PostGIS.

Carga de Datos: Crea una tabla con el nombre del archivo shapefile o tabla dbf a importar. La tabla no debe existir.

Exportar una tabla: Se exporta por tabla, y cada tabla genera un archivo ShapeFile o tabla dbf.

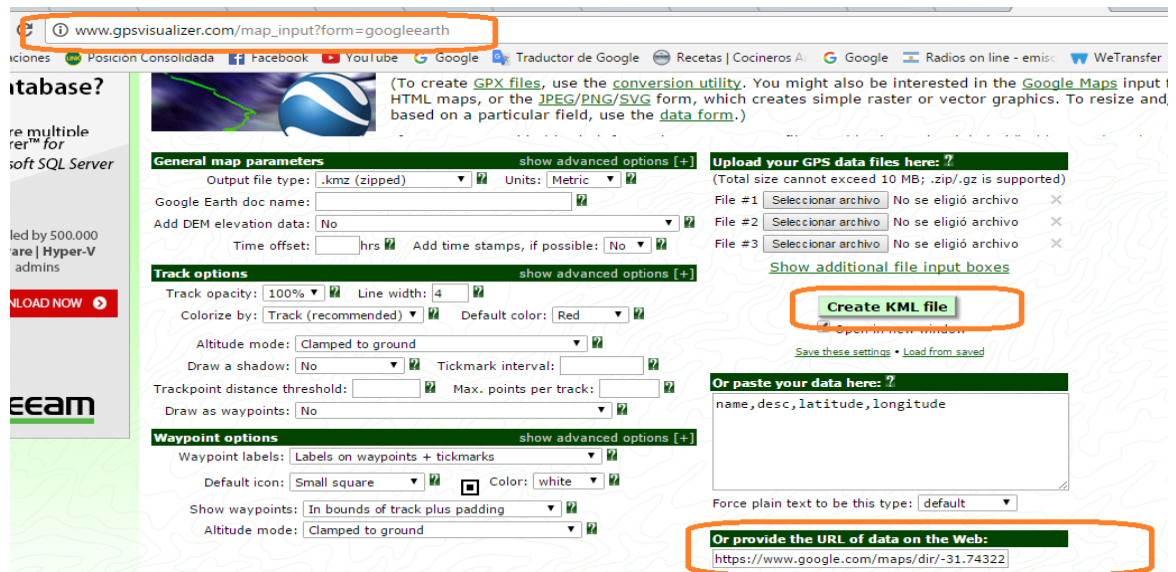


Bases de Datos Geográficas

Implementación PostGIS

○ Exportar trayectorias Google Maps a kml

- Se genera la trayectoria con “Ruta desde aquí” o “Ruta hacia aquí”.
- Luego se genera el kml con la url de maps entrando antes a:
http://www.gpsvisualizer.com/map_input?form=googleearth



The screenshot shows the GPS Visualizer website interface. Several elements are highlighted with orange boxes:

- The URL in the browser address bar: www.gpsvisualizer.com/map_input?form=googleearth
- The "Create KML file" button in the "Upload your GPS data files here" section.
- The "Or provide the URL of data on the Web:" section at the bottom, which contains the URL: <https://www.google.com/maps/dir/-31.74322>

The interface includes various configuration options for map generation, such as "General map parameters", "Track options", and "Waypoint options".

Bases de Datos Geográficas

Implementación stGIS

○ Importar kml o kmz

- Nota para importar debe descomprimir los rar kmz con winrar o algun zipeador y luego ejecutar la utilidad **ogr2ogr.exe**.

```
C:\Archivos de programa\qgis 2.18\bin>ogr2ogr.exe -f "PostgreSQL"  
PG:"host=127.0.0.1 port=5432 dbname=bdatp2017 user=postgres  
password=postgre" d:/backup/rutas_  
provinciales.kml
```

RESUMEN

- Características PostGIS: Imp. de Referencia
- Jerarquia GEOMETRY
- PostGIS formatos nativos
- Constructores de TIPOS
- Metodos de la SuperClase GEOMETRY
- Metodos de Analisis Espacial
- Instalación PostGIS
- Anexo Transformación WKT to x,y

Bases de Datos Geográficas

Función WKT to puntos x,y

A continuación se presenta una función plpgsql wkt2puntos y una planilla excel para representación: poligonosbdag.xls (adjunto).

```
CREATE OR REPLACE FUNCTION wkt2puntos(wkt varchar)
  RETURNS TABLE(x NUMERIC, y NUMERIC) AS
$BODY$
declare
Begin
  wkt = upper(wkt);
  wkt = regexp_replace(wkt, ' {2,}', ' ', 'g');

  wkt = ltrim(rtrim(replace(wkt, 'POLYGONZ', '')));
  wkt = ltrim(rtrim(replace(wkt, 'POLYGONZM', '')));
  wkt = ltrim(rtrim(replace(wkt, 'POLYGONM', '')));
  wkt = ltrim(rtrim(replace(wkt, 'POLYGON', '')));

  wkt = ltrim(rtrim(replace(wkt, 'LINESTRINGZ', '')));
  wkt = ltrim(rtrim(replace(wkt, 'LINESTRINGZM', '')));
  wkt = ltrim(rtrim(replace(wkt, 'LINESTRINGM', '')));
  wkt = ltrim(rtrim(replace(wkt, 'LINESTRING', '')));
```

Bases de Datos Geográficas

Función WKT to puntos x,y

continuación función plpgsql wkt2puntos

```
wkt = ltrim(rtrim(replace(wkt, 'POINTZ', '')));
wkt = ltrim(rtrim(replace(wkt, 'POINTZM', '')));
wkt = ltrim(rtrim(replace(wkt, 'POINTM', '')));
wkt = ltrim(rtrim(replace(wkt, 'POINT', '')));

wkt = split_part(wkt, '(', 2);
wkt = split_part(wkt, ')', 1);

CREATE LOCAL TEMP TABLE IF NOT EXISTS tmp (xy VARCHAR) ON COMMIT
        DELETE ROWS;

INSERT INTO tmp SELECT CAST(regexp_split_to_table(wkt, ',') AS
VARCHAR) AS xy;

RETURN QUERY
        SELECT CAST(split_part(xy, ' ', 1) AS NUMERIC) as x,
               CAST(split_part(xy, ' ', 2) AS NUMERIC) as y
        FROM tmp;

End $BODY$ LANGUAGE plpgsql;
```

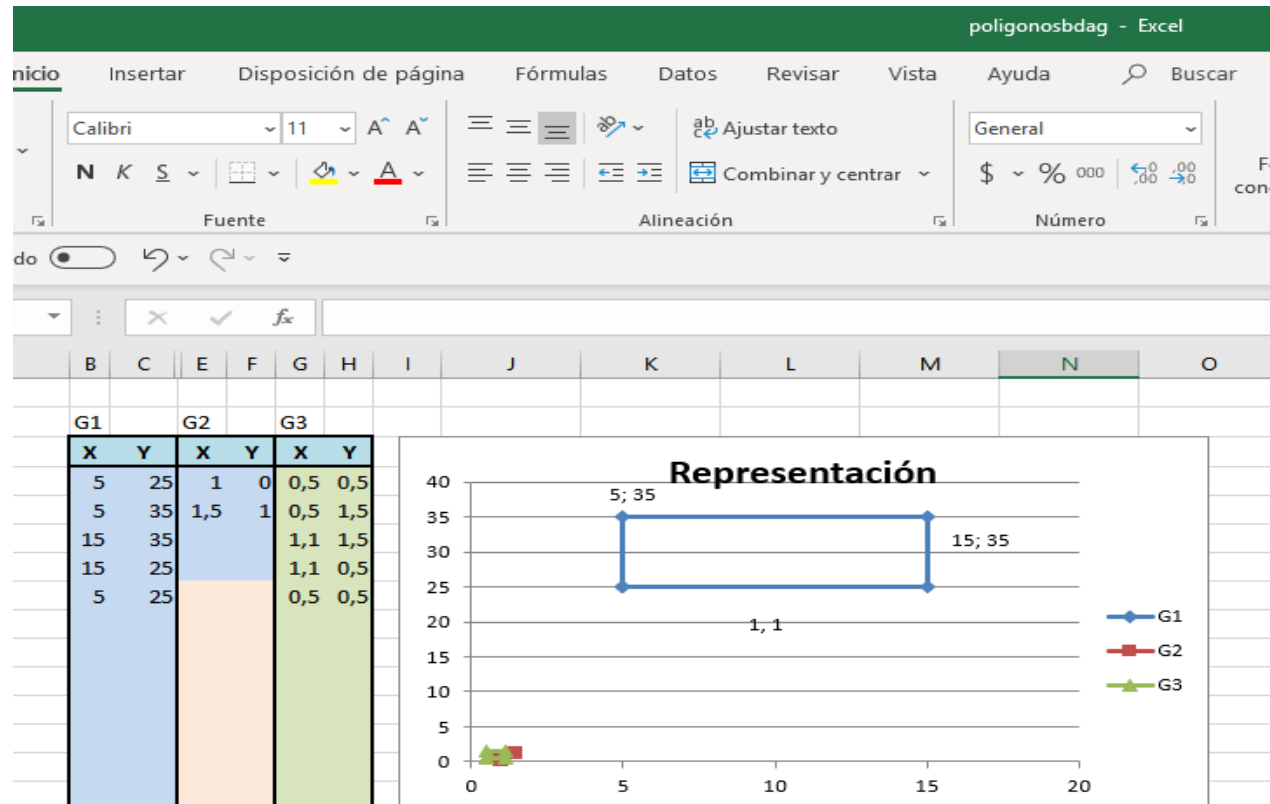
Bases de Datos Geográficas

Función WKT to puntos x,y

Ejemplo de Utilización:

```
select * from wkt2puntos('POLYGON((5 25,5 35,15 35,15 25,5 25))')
```

	x numeric	y numeric
1	5	25
2	5	35
3	15	35
4	15	25
5	5	25



Bases de Datos Geográficas



Fuentes

Manual PostGIS 3.0 :

Postgis-3.0-es.pdf

Otras fuentes digitales asociadas:

<http://www.opengeospatial.org/standards/sfs>