

## **Modulo “Analisis de Datos Cientificos y Geograficos”**

### **DATOS ESPACIALES**



## RECORDEMOS SQL...

```
SELECT c1,c2 ... cN  
FROM tabla1,...tablaN  
WHERE condicionW  
GROUP BY campo  
HAVING condicionH  
ORDER BY N
```

- 5: columnas que se desean obtener
- 1: tablas a usar
- 2: tuplas a recuperar (que cumplan condicion)
- 3: grupos a armar (por igualdad de campo)
- 4: grupos a recuperar (que cumplan condicion)
- 6: resultado ordenado por N

## SQL: EJEMPLO 1

```
SELECT a3, b2  
FROM A, B  
WHERE a1=b1  
ORDER BY b2
```

a3	b2
2	4
3	4
1	7

A			B	
a1	a2	a3	b1	b2
8	9	2	3	7
3	1	1	8	4
8	2	3		

a1	a2	a3	b1	b2
8	9	2	3	7
8	9	2	8	4
3	1	1	3	7
3	1	1	8	4
8	2	3	3	7
8	2	3	8	4

## SQL: EJEMPLO 2

```
SELECT b2, a3      5  
FROM tA, B        1  
WHERE a1=b1       2  
GROUP BY b2       3  
HAVING b2<5      4  
ORDER BY b2       6
```

A			B	
a1	a2	a3	b1	b2
8	9	2	3	7
3	1	1	8	4
8	2	3		

a1	a2	a3	b1	b2
8	9	2	3	7
8	9	2	8	4
3	1	1	3	7
3	1	1	8	4
8	2	3	3	7
8	2	3	8	4

## SQL: EJEMPLO 2

ERROR

```
SELECT b2, a3 5  
FROM tA, B 1  
WHERE a1=b1 2  
GROUP BY b2 3  
HAVING b2<5 4  
ORDER BY b2 6
```

A			B	
a1	a2	a3	b1	b2
8	9	2	3	7
3	1	1	8	4
8	2	3		

a1	a2	a3	b1	b2
8	9	2	3	7
8	9	2	8	4
3	1	1	3	7
3	1	1	8	4
8	2	3	3	7
8	2	3	8	4

## SQL: EJEMPLO 3

```
SELECT b2 , max(a3) 5  
FROM tA, B 1  
WHERE a1=b1 2  
GROUP BY b2 3  
HAVING b2<5 Rta  
ORDER BY b2
```

b2	max(a3)
4	3

A			B	
a1	a2	a3	b1	b2
8	9	2	3	7
3	1	1	8	4
8	2	3		

a1	a2	a3	b1	b2
8	9	2	3	7
8	9	2	8	4
3	1	1	3	7
3	1	1	8	4
8	2	3	3	7
8	2	3	8	4

## SQL: EJEMPLO 4

```
SELECT b2 , sum(a3) 5  
FROM tA, B 1  
WHERE a1=b1 2  
GROUP BY b2 3  
HAVING b2<5 Rta  
ORDER BY b2
```

b2	sum(a3)
4	5

A			B	
a1	a2	a3	b1	b2
8	9	2	3	7
3	1	1	8	4
8	2	3		

a1	a2	a3	b1	b2
8	9	2	3	7
8	9	2	8	4
3	1	1	3	7
3	1	1	8	4
8	2	3	3	7
8	2	3	8	4

# CONSULTAS ESPACIALES

```
SELECT ...  
FROM  tabla1, tabla2  
WHERE ...  
GROUP BY ...  
HAVING ...  
ORDER BY ...
```

## Funciones Espaciales:

AREA  
LENGTH  
BUFFER  
CENTROID  
DIFFERENCE  
INTERSECTION  
UNION  
etc.



# FUNCIONES ESPACIALES

## ST\_GeomFromText

Toma una representación de texto conocido y un Id. de referencia espacial y devuelve un objeto de geometría.

### *Sintaxis*

`st_geomfromtext (wkt text, srid integer)`

- POINT(x y)
- MULTIPOINT(x1 y1, x2 y2, ...)
- LINESTRING(x1 y1, x2 y2)
- MULTILINESTRING((x1 y1 x2 y2),(x3 y3 x4 y4) ...)
- POLYGON((x1 y11, x12 y12, x13 y13 ... x11 y11),(x21 y21, x22 y22, x23 y23, ... x21 y21))
- MULTIPOLYGON( ((...),(...)), ((...)(...)) )



# FUNCIONES ESPACIALES

## ST\_GeomFromText

### *Ejemplo*

```
INSERT INTO geatable ( name, geom)
VALUES ('radiobase01', ST_GeomFromText('POINT(-126.4 45.32)', 4326));
```

```
INSERT INTO geatable ( name, geom)
VALUES ('terreno', ST_GeomFromText('POLYGON((0 0, 4 0, 4 4, 0 4, 0 0))', 4326));
```

```
INSERT INTO geatable ( name, geom)
VALUES ('rio', ST_GeomFromText('MULTILINESTRING((1 3, 7 8)(7 8, 21 15)
(21 15 32 11)', 4326));
```



## FUNCIONES ESPACIALES

### ST\_AsText

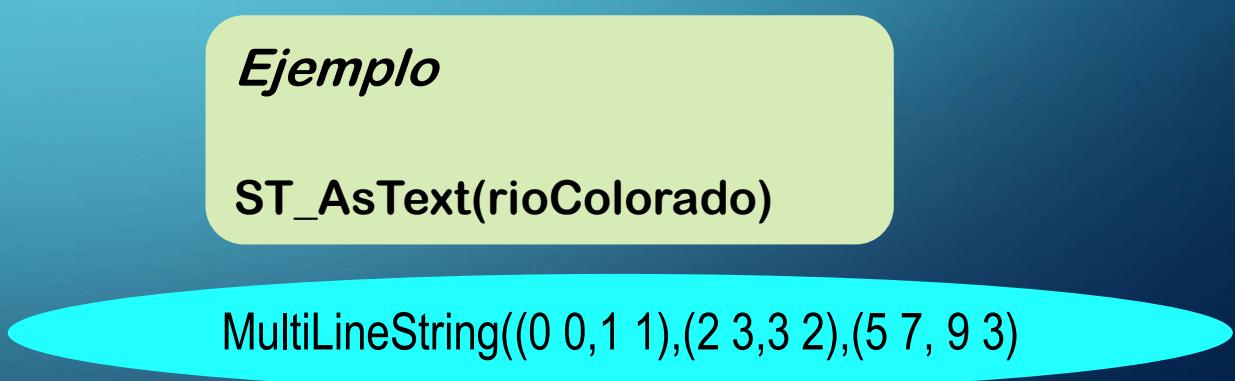
Toma una representación de texto conocido y un Id. de referencia espacial y devuelve un objeto de geometría.

*Sintaxis*

`st_astext(geometry)`

*Ejemplo*

`ST_AsText(rioColorado)`



`MultiLineString((0 0,1 1),(2 3,3 2),(5 7,9 3))`



## FUNCIONES ESPACIALES: De consulta

### ST\_Length

Devuelve la longitud de un objeto tipo lineal.

#### *Sintaxis*

`st_length (linestring)`

`st_length (multilinestring)`

Si no es línea o multilinea,  
retorna valor 0



## FUNCIONES ESPACIALES: De consulta

### ST\_Perimeter

Devuelve la longitud del perímetro de un polígono o multipolígono.

#### *Sintaxis*

`st_Perimeter (polygon)`

`st_Perimeter (multipolygon)`

**Si no es polígono o multipolígono,  
retorna valor 0**



## FUNCIONES ESPACIALES: De consulta

### ST\_Area

Devuelve el área de un polígono o multipolígono.

#### *Sintaxis*

`st_area (polygon)`

`st_area (multipolygon)`

Si no es polígono o multipolígono,  
retorna valor 0



## FUNCIONES ESPACIALES: De consulta

### **ST\_Buffer**

Toma un objeto de geometría G y una distancia N, y devuelve un objeto geometrico formado por el conjunto de puntos a distancia N de la geometría G (zona de influencia en torno a G).

#### *Sintaxis*

`st_buffer (geometry, double_precision)`

## FUNCIONES ESPACIALES: De consulta

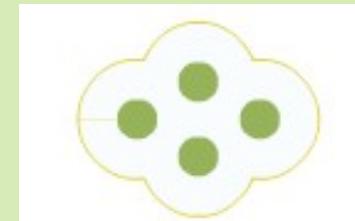
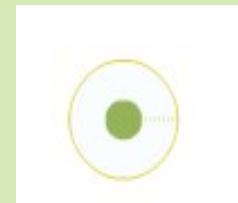
### ST\_Buffer

Toma un objeto de geometria y crea un nuevo objeto geometrico formado por el primer y una buffer de distancia N de la geometria.

### Sintaxis

`st_buffer (geometry, distance)`

### Ejemplo





## FUNCIONES ESPACIALES: De consulta

### ST\_Envelope

Devuelve el MBR de un objeto geometrico.

#### *Sintaxis*

`st_envelope (geometry)`

## FUNCIONES ESPACIALES - Parte 1

**ST\_Envelope**

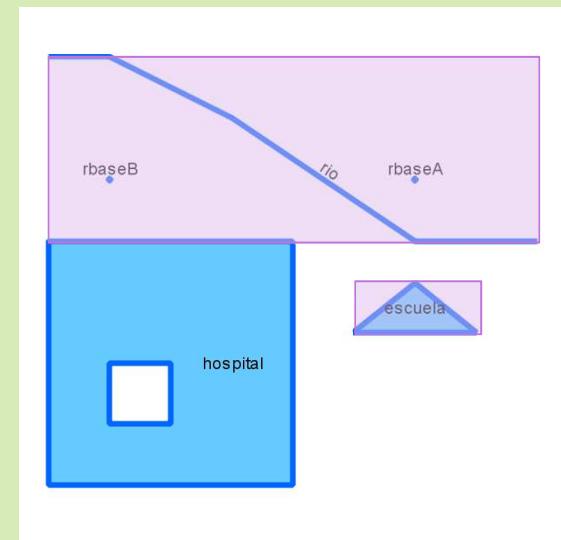
Devuelve el M

*Sintaxis*

`st_envelope (`

*Ejemplo*

```
SELECT ST_Envelope(geom)
FROM gis_example
WHERE descrip = 'escuela' OR descrip='rio';
```





## FUNCIONES ESPACIALES: De consulta

### **ST\_ConvexHull**

Devuelve la envoltura convexa de un objeto **ST\_Geometry**.

#### *Sintaxis*

`st_convexhull (geometry)`

`st_convexhull (ST_Collect(geometry1, geometry2, ...))`

## FUNCIONES GE

**ST\_ConvexHull**

Devuelve la

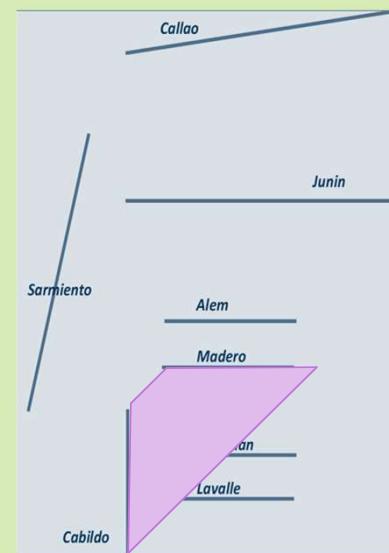
*Sintaxis*

`st_convexhull`

`st_convexhull`

### *Ejemplo*

```
SELECT st_convexhull(st_Collect(a.the_geom, b.the_geom))
FROM bc_calle a, bc_calle b
WHERE a.name='Cabildo' and b.name='Madero'
```





## FUNCIONES ESPACIALES: De consulta

### **ST\_Centroid**

Toma un polígono o multipolígono y devuelve el punto que está en el centro del MBR de la geometría.

#### *Sintaxis*

`st_centroid (geometry)`

## FUNCIONES ESPACIALES - Parte 1

### ST\_Centroid

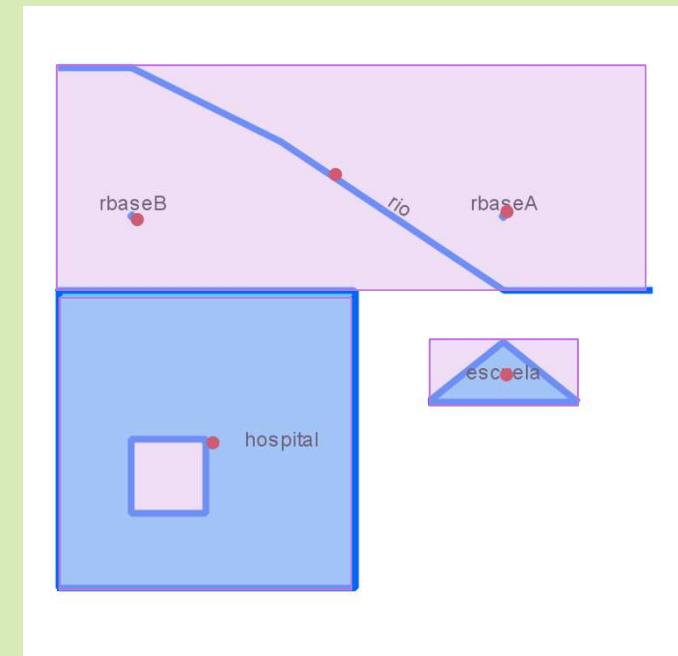
Toma un polígono y  
devuelve un punto en el centro

### Sintaxis

`st_centroid(geom)`

### Ejemplo

```
SELECT ST_Centroid(geom)  
FROM gis_example
```





## FUNCIONES ESPACIALES: De consulta

### **ST\_Distance**

Devuelve la distancia entre dos geometrías. La distancia se mide desde el vértice más cercano de las dos geometrías.

#### *Sintaxis*

**ST\_distance (geometry1, geometry2)**

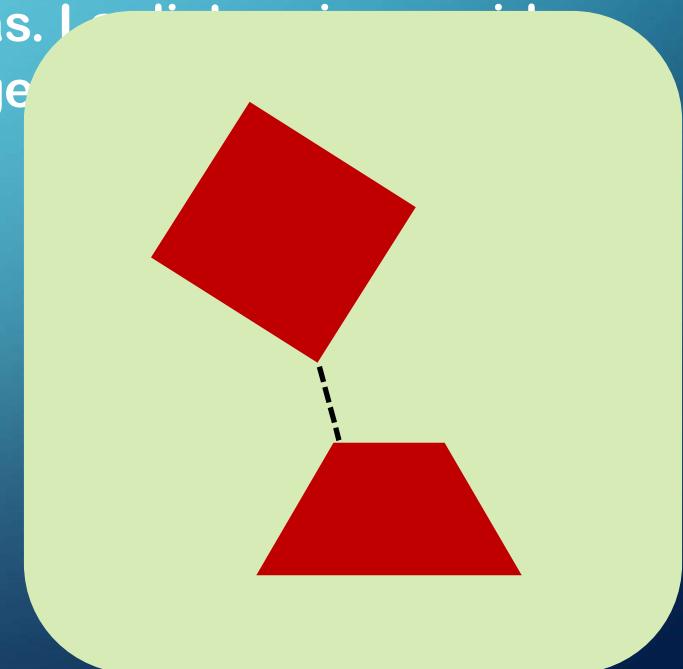
## FUNCIONES ESPACIALES: De consulta

### **ST\_Distance**

Devuelve la distancia entre dos geometrías. La medida se calcula desde el vértice más cercano de las dos ge

### *Sintaxis*

**ST\_distance (geometry1, geometry2)**





## FUNCIONES ESPACIALES: De consulta

**Operador <->**

Devuelve la distancia entre los centros de los MBR de las dos geometrías.

*Sintaxis*

`geometry1 <-> geometry2`

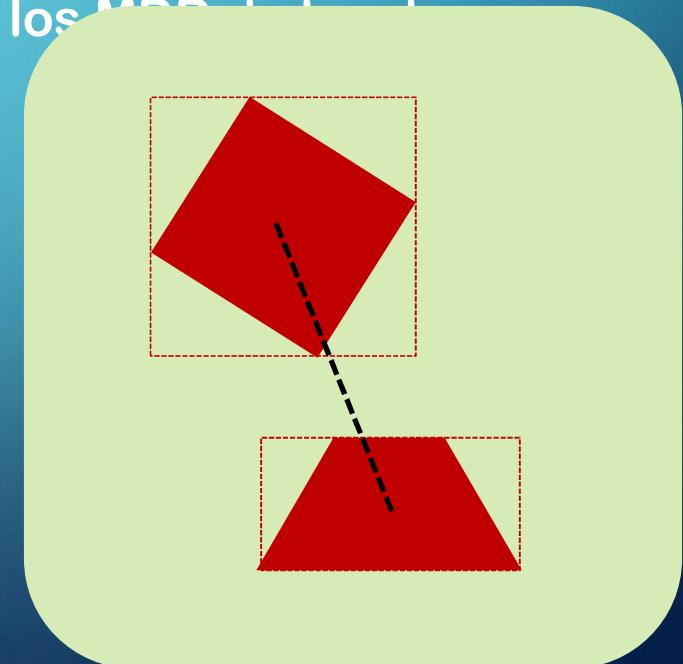
## FUNCIONES ESPACIALES: De consulta

**Operador <->**

Devuelve la distancia entre los centros de los **MEDIDAS** de las geometrías.

*Sintaxis*

`geometry1 <-> geometry2`





## FUNCIONES ESPACIALES: De consulta

**Operador <#>**

Devuelve la distancia entre los MBR de las dos geometrías.

*Sintaxis*

geometry1 <#> geometry2

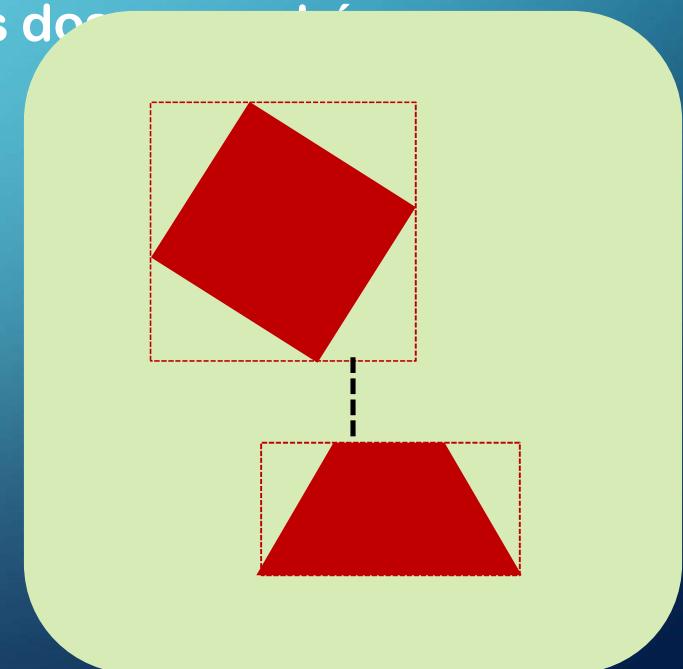
# FUNCIONES ESPACIALES: De consulta

**Operador <#>**

Devuelve la distancia entre los MBR de las dos geometrías

*Sintaxis*

`geometry1 <#> geometry2`



# FUNCIONES ESPACIALES: De consulta

Ejemplo: **ST\_Distance, <->, <#>**

*Ejemplo*

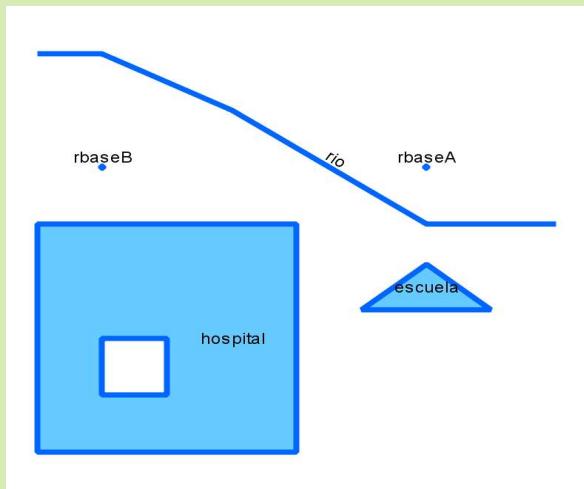
```
SELECT ST_Distance(g1.geom, g2.geom) Rta: 1.094  
FROM gis_example g1, gis_example g2  
WHERE g1.descrip = 'hospital' and g2.descrip='rio';
```

```
SELECT g1.geom <-> g2.geom Rta: 4.03
```

...

```
SELECT g1.geom <#> g2.geom Rta: 0
```

...





## FUNCIONES ESPACIALES: De consulta

kNN

Devuelve los N objetos mas cercanos a la geometria dada.

*Sintaxis*

*Lo hacemos a traves de ORDER BY y LIMIT*



## FUNCIONES ESPACIALES: De consulta

kNN

Devuelve los N o

*Sintaxis*

*Lo hacemos a tr*

*Ejemplo*

**ERROR**

```
SELECT c.name, ST_Distance(c.the_geom, g.geom) as d
FROM gis_example g, bc_calle c
WHERE g.descrip='escuela'
ORDER BY d
LIMIT 3
```



## FUNCIONES ESPACIALES: De consulta

kNN

Devuelve los N objetos mas cercanos a la geometria dada.

*Sintaxis*

*Lo hacemos a traves de ORDER By y LIMIT*



## FUNCIONES ESPACIALES: De consulta

kNN

Devuelve

*Sintaxis*

*Lo hacemos*

*Ejemplo*

```
SELECT c.name,  
ST_Distance(ST_GeomFromText(ST_AsText(c.the_geom),4326),  
ST_GeomFromText(ST_AsText(g.geom),4326)) as d  
FROM gis_example g, bc_callao c  
WHERE g.descrip='escuela'  
ORDER BY d  
LIMIT 3
```

Rta:

"Junin"; 0.700  
"Callao"; 1.767  
"Alem"; 2.700



## FUNCIONES ESPACIALES: De consulta

### **ST\_Disjoint**

Retorna TRUE si la intersección de las dos geometrías produce un conjunto vacío; de lo contrario, devuelve FALSE.

#### *Sintaxis*

`st_disjoint (geometry1, geometry2)`

## FUNCIONES ESPACIALES: De consulta

### ST\_Disjoint

Retorna TRUE si la intersección entre dos geometrías es un conjunto vacío; de lo contrario, FALSE.

### Sintaxis

`st_disjoint (geometry1, geometry2)`

### Ejemplo

Retorna TRUE:





## FUNCIONES ESPACIALES: De consulta

### **ST\_Intersects**

Devuelve TRUE si la intersección de dos geometrías no genera un conjunto vacío; de lo contrario, devuelve FALSE.

#### *Sintaxis*

`st_intersects (geometry1, geometry2)`

## FUNCIONES ESPACIALES: De consulta

### **ST\_Intersects**

Devuelve TRUE si la intersección no es un conjunto vacío; de lo contrario, FALSE.

#### *Sintaxis*

`st_intersects (geometry1, geometry2)`

#### *Ejemplo*

*Retorna TRUE:*





## FUNCIONES ESPACIALES: De consulta

### ST\_Intersection

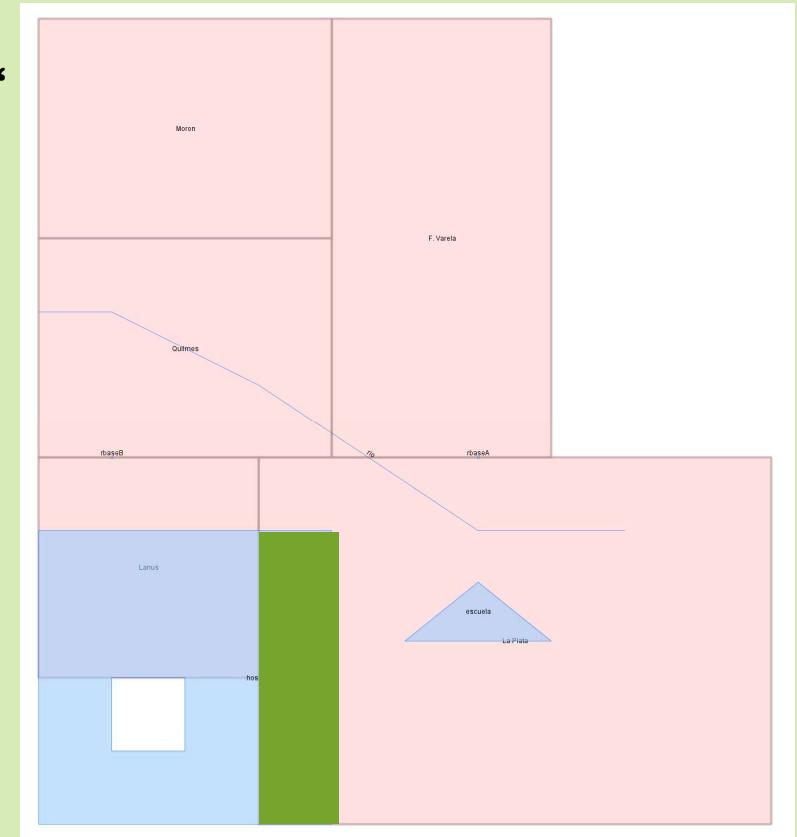
Toma dos objetos de geometría y devuelve el conjunto de intersección como un objeto de geometría bidimensional.

### *Sintaxis*

`st_intersection (geometry1, geometry2)`

## *Ejemplo*

```
SELECT ST_Intersection(g.geom, c.the_geom)
FROM gis_example g, bc_ciudad c
WHERE g.descrip='hospital'
AND c.name='La Plata'
```



**FUNC**

**ST\_Intersection**

Toma la  
intersección

*Sintaxis*

**st\_intersection**

Especializa



## FUNCIONES ESPACIALES: De consulta

### ST\_Union

Devuelve un objeto de geometría que es la combinación de dos objetos de origen.

#### *Sintaxis*

`st_union (geometry1, geometry2)`

## FUNCIONES

### ST\_Union

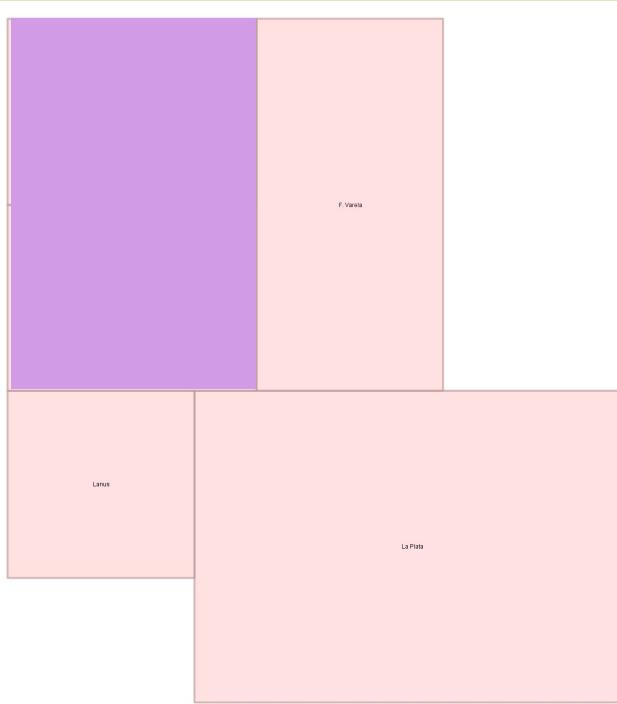
Devuelve un objeto de geometria que es la unión de los objetos de origen.

#### Sintaxis

`st_union (geom1, geom2)`

#### Ejemplo

```
SELECT ST_Union(c1.the_geom, c2.the_geom)
FROM bc_ciudad c1, bc_ciudad c2
WHERE c1.name='Moron' AND c2.name='Quilmes';
```





## FUNCIONES ESPACIALES: De consulta

### **ST\_Difference**

Toma dos objetos de geometría y devuelve un objeto de geometría que es la diferencia de los objetos de origen.

#### *Sintaxis*

**st\_difference (geometry1, geometry2)**

## FUNCIONES

**ST\_D**

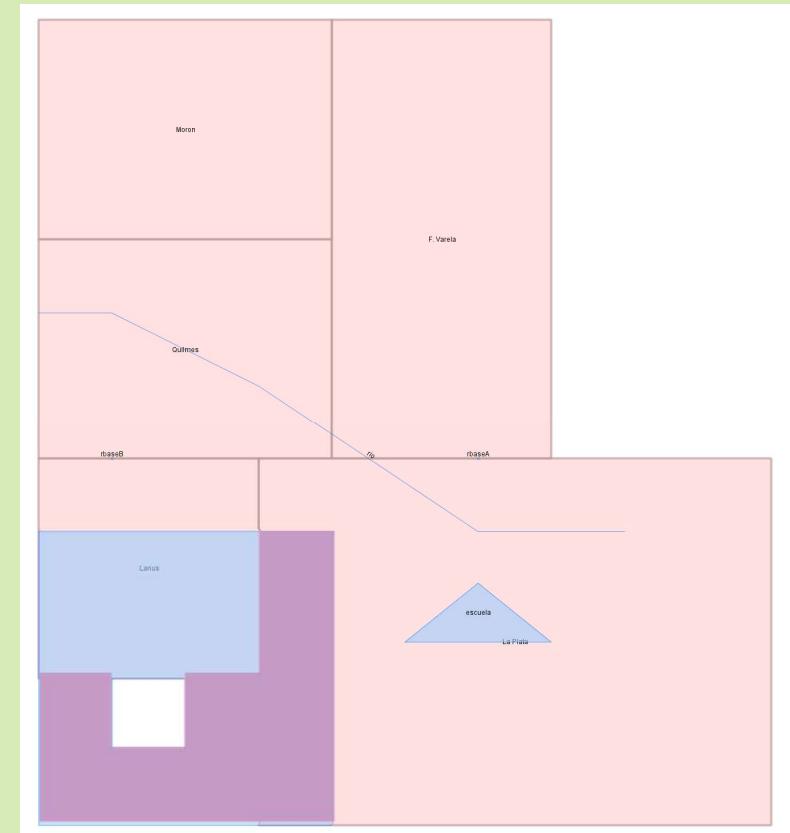
Toma  
que es

*Sintaxis*

**st\_dif**

### Ejemplo

```
SELECT ST_Difference(g.geom, c.the_geom)
FROM gis_example g, bc_ciudad c
WHERE g.descrip='hospital'
AND c.name='Lanus'
```





## FUNCIONES ESPACIALES: De consulta

**ST\_NumInteriorRings**

Retorna el numero de anillos interiores de un polígono.

*Sintaxis*

integer **ST\_NumInteriorRings(polygon);**



## FUNCIONES ESPACIALES: De consulta

**ST\_NumInteriorRings**

Retorna el numero de anillos interiores de un polígono.

*Sintaxis*

integer **ST\_NumInteriorRings(polygon);**

## FUNCIONES ESPACIALES - Parte 2

**ST\_NumInteriorRings**

Retorna el numero de anillos interiores.

*Sintaxis*

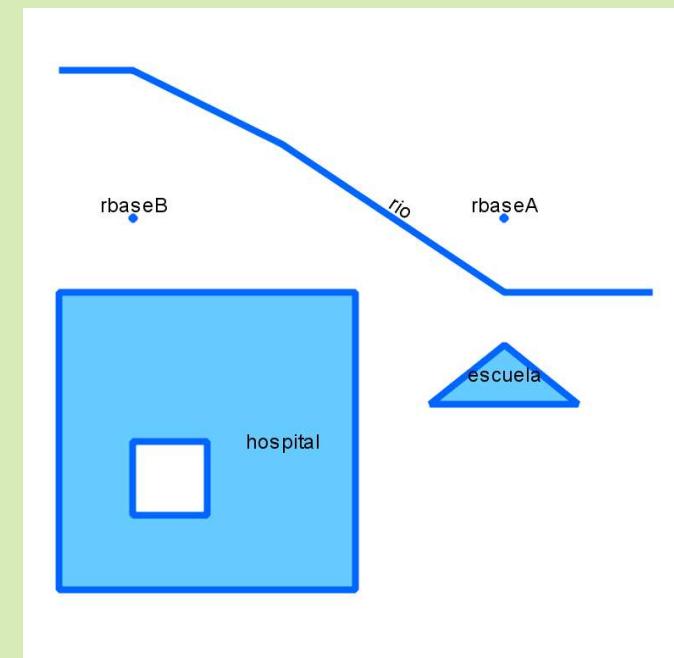
integer **ST\_NumInteriorRings**(  
  geometry geom)

*Ejemplo*

```
SELECT descrip, ST_NumInteriorRings(geom)  
FROM gis_example
```

Rta:

```
"rbaseA";  
"rbaseB";  
"hospital"; 1  
"escuela"; 0  
"rio";
```





## FUNCIONES ESPACIALES: De consulta

### ST\_Contains

Retorna True si ningun punto de la segunda geometria esta en el exterior de la primera y al menos un punto de la segunda esta en el interior de la primera. De lo contrario retorna FALSE.

#### *Sintaxis*

`st_contains (geometry1, geometry2)`

# FUNCIONES ESPACIALES: D

## ST\_Contains

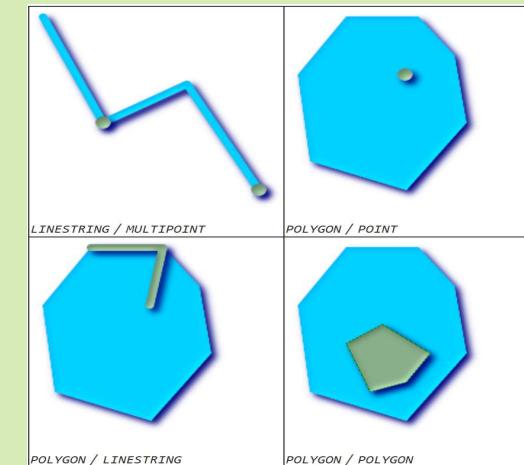
Retorna True si ningun punto de la segunda figura se encuentra en el exterior de la primera y al menos uno de los puntos se encuentra en el interior de la primera. Devuelve FALSE en caso contrario.

### Sintaxis

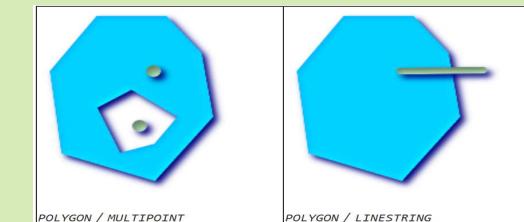
`st_contains (geometry1, geometry2)`

### Ejemplo

Retorna TRUE:



Retorna FALSE:





## FUNCIONES ESPACIALES: De consulta

### ST\_Within

Es exactamente el contrario de ST\_Contains.

#### *Sintaxis*

`st_within(geometry1, geometry2)`

# FUNCIONES ESPACIALES: De consulta

## ST\_Within

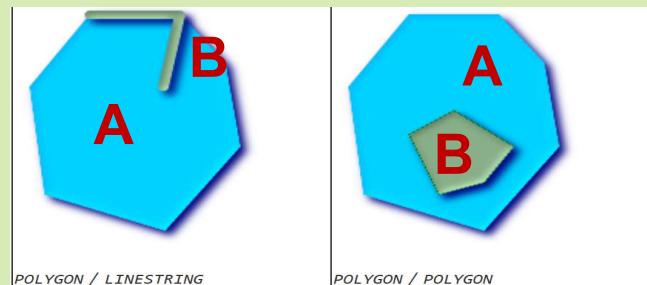
Es exactamente el contrario de ST\_Contains

### Sintaxis

`st_within(geometry1, geometry2)`

### Ejemplo

Si  $ST\_Contains(A,B)$  es TRUE:



$ST\_Within(A,B)$  es False

pero

$ST\_Within(B, A)$  es True



## FUNCIONES ESPACIALES: De consulta

### **ST\_Covers**

Retorna TRUE si ninguno punto de la segunda geometria esta en el exterior de la primera (el primer objeto cubre completamente al Segundo). De lo contrario, devuelve FALSE

#### *Sintaxis*

`st_covers (geometry1, geometry2)`

## FUNCIONES ESPACIALES: De consulta

### **ST\_Covers**

Retorna TRUE si ninguno punto de la segunda figura se encuentra en el exterior de la primera (el primer objeto cubre completamente al Segundo). De lo contrario, retorna FALSE.

#### *Sintaxis*

`st_covers (geometry1, geometry2)`

#### *Ejemplo*

**Retorna TRUE:**





## FUNCIONES ESPACIALES: De consulta

### **ST\_Crosses**

Devuelve TRUE si su intersección genera un objeto de geometría cuya dimensión es un número menor que la dimensión máxima de los objetos de origen. De lo contrario, devuelve FALSE.

#### *Sintaxis*

`st_crosses(geometry1, geometry2)`

## FUNCIONES ESPACIALES: De consulta

### ST\_Crosses

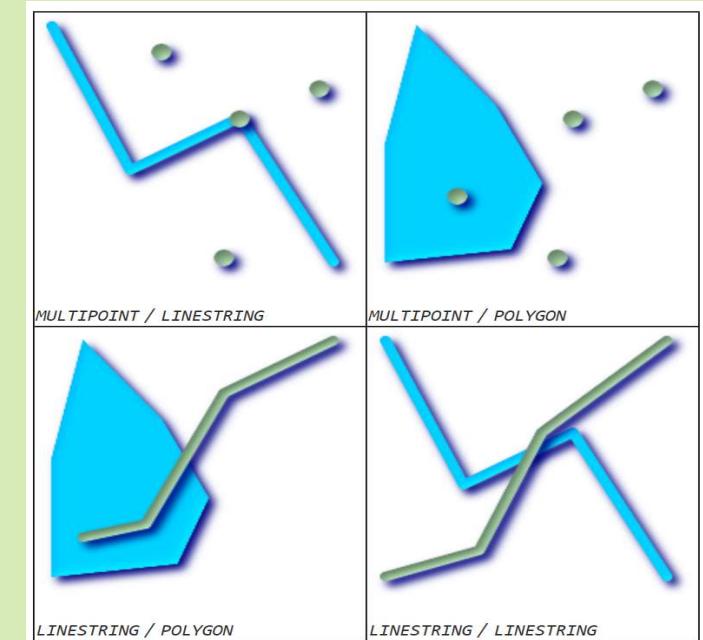
Devuelve TRUE si su intersección genera un resultado que tiene una dimensión menor que los objetos de origen. De lo contrario, devolverá FALSE.

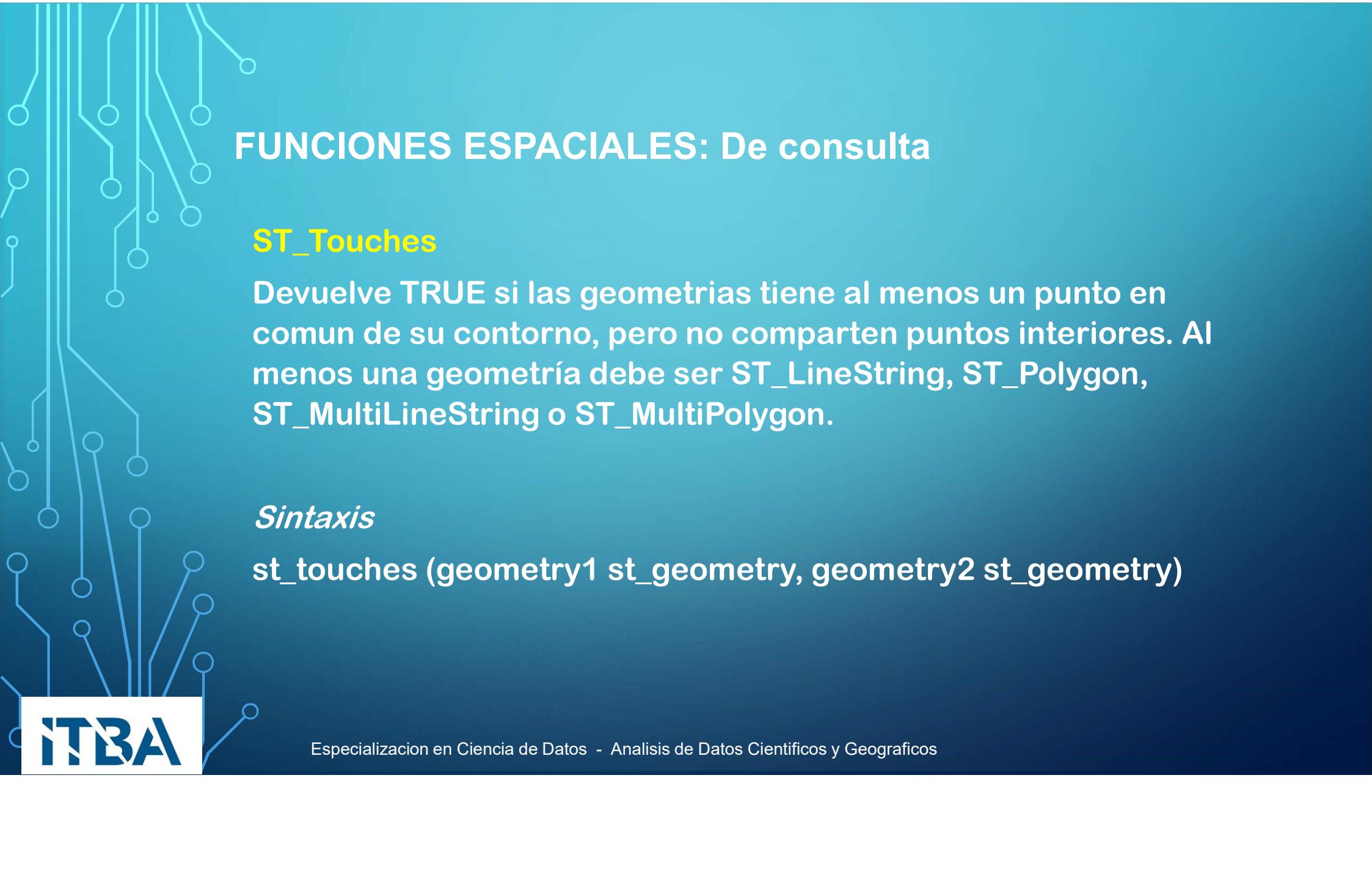
#### *Sintaxis*

`st_crosses(geometry1, geometry2)`

#### *Ejemplo*

*Retorna TRUE:*





## FUNCIONES ESPACIALES: De consulta

### **ST\_Touches**

Devuelve TRUE si las geometrias tiene al menos un punto en comun de su contorno, pero no comparten puntos interiores. Al menos una geometría debe ser ST\_LineString, ST\_Polygon, ST\_MultiLineString o ST\_MultiPolygon.

### *Sintaxis*

`st_touches (geometry1 st_geometry, geometry2 st_geometry)`

## FUNCIONES ESPACIALES: D

### ST\_Touches

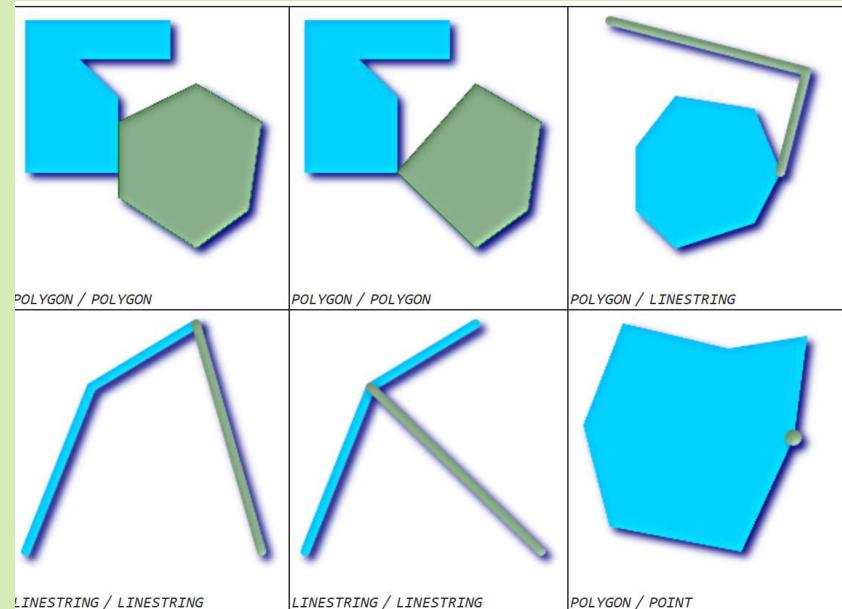
Devuelve TRUE si las geometrías tienen al menos un punto en común de su contorno, pero no se tocan en más de un punto. Al menos una geometría debe ser de tipo ST\_MultiLineString o ST\_MultiPolygon.

### Sintaxis

`st_touches (geometry1 st_geometry2)`

### Ejemplo

Retorna TRUE:





## FUNCIONES ESPACIALES: De consulta

### **ST\_Overlaps**

Retorna TRUE si la intersección de los objetos genera un objeto de geometría de la misma dimensión pero distinto de los dos objetos de origen (no completamente incluido una en el otra).

De lo contrario, devuelve FALSE.

#### *Sintaxis*

`st_overlaps(geometry1, geometry2)`

## FUNCIONES ESPACIALES: De consulta

### ST\_Overlaps

Retorna TRUE si la intersección de geometría de la misma dimensión de los objetos de origen (no complejos).

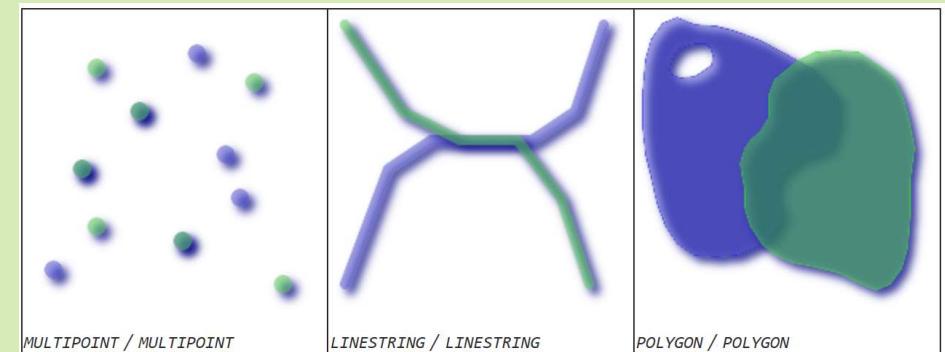
De lo contrario, devuelve FALSE.

### Sintaxis

`st_overlaps(geometry1, geometry2)`

### Ejemplo

Retorna TRUE:





## FUNCIONES ESPACIALES: De consulta

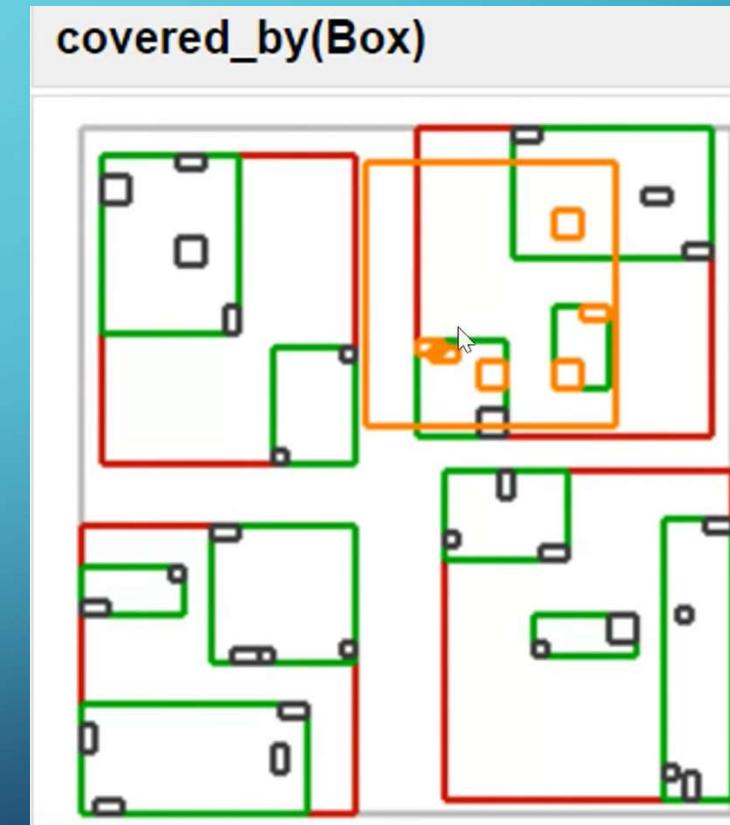
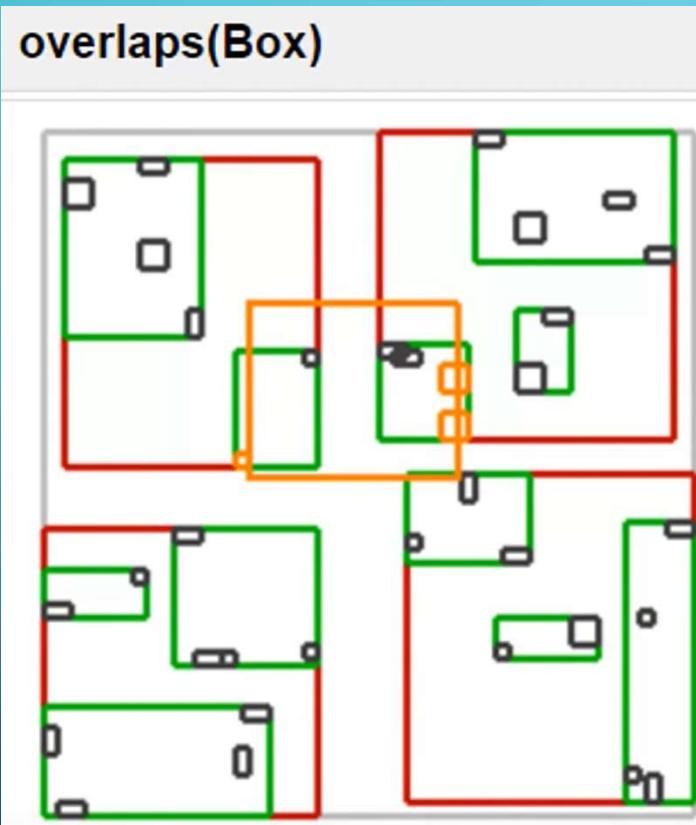
### **ST\_CoveredBy**

Devuelve TRUE si la intersección de los objetos genera un objeto de geometría de la misma dimensión pero distinto de los dos objetos de origen (ninguno punto de la primera está en el exterior de la segunda). De lo contrario, devuelve FALSE.

### *Sintaxis*

**st\_overlaps (geometry1 st\_geometry, geometry2 st\_geometry)**

# OPERACIONES COSTOSAS...



## PRACTIQUEMOS ...

```
CREATE TABLE gis_example (id integer, descrip varchar, geom geometry)
```

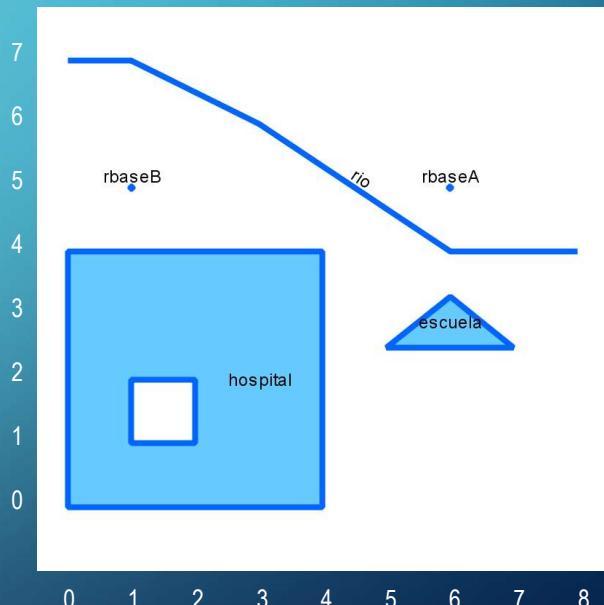
```
INSERT INTO gis_example  
VALUES (1, 'rbaseA', ST_GeomFromText('POINT(6 5)', 4326 ));
```

```
INSERT INTO gis_example  
VALUES (2, 'rbaseB', ST_GeomFromText('POINT(1 5)', 4326 ));
```

```
INSERT INTO gis_example  
VALUES (3, 'hospital',  
ST_GeomFromText('POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1))', 4326 ));
```

```
INSERT INTO gis_example  
VALUES (4, 'escuela',  
ST_GeomFromText('POLYGON((5 2.5,7 2.5,6 3.3,5 2.5))', 4326 ));
```

```
INSERT INTO gis_example  
VALUES (5, 'rio', ST_GeomFromText('LINESTRING(0 7,1 7,3 6,6 4,8 4)', 4326 ));
```

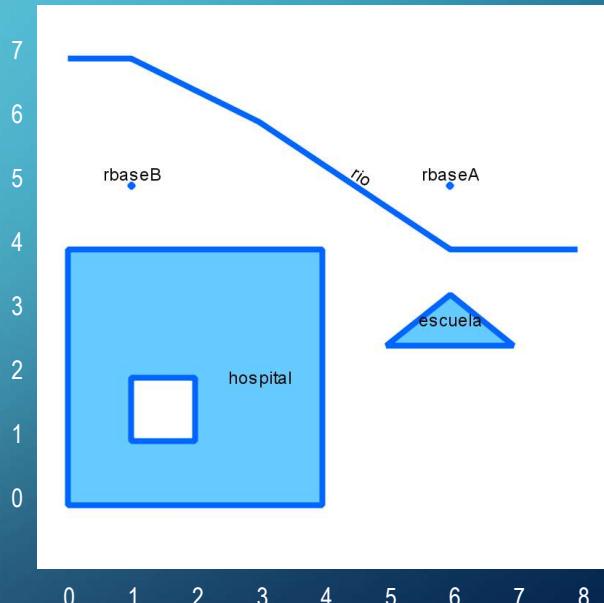


## PRACTIQUEMOS ...

Calcular la distancia entre las radiobases

```
SELECT g1.descrip, g2.descrip, ST_Distance(g1.geom, g2.geom)  
FROM gis_example g1, gis_example g2  
WHERE g1.descrip LIKE 'rbase%' AND g2.descrip LIKE 'rbase%'  
      AND g1.id < g2.id
```

descrip character varying	descrip character varying	st_distance double precision
rbaseA	rbaseB	5

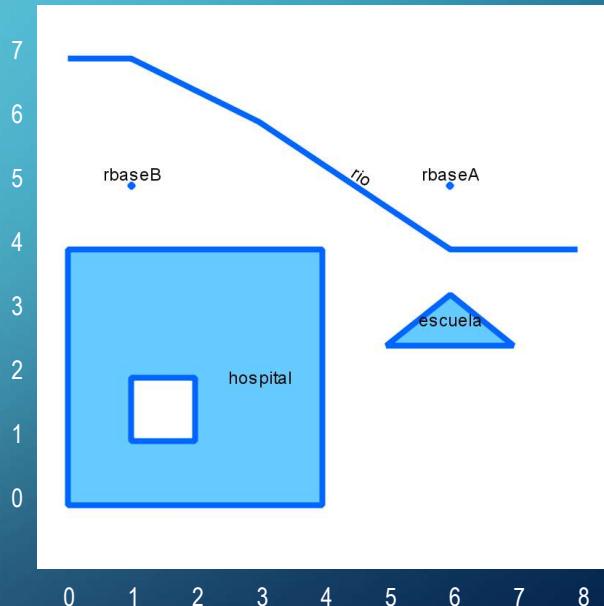


## PRACTIQUEMOS ...

Calcular la distancia entre la radiobaseB  
y el hospital

```
select g1.descrip, g2.descrip, ST_Distance(g1.geom, g2.geom)  
from gis_example g1, gis_example g2  
where g1.descrip = 'rbaseB' and g2.descrip = 'hospital';
```

Rta: 1

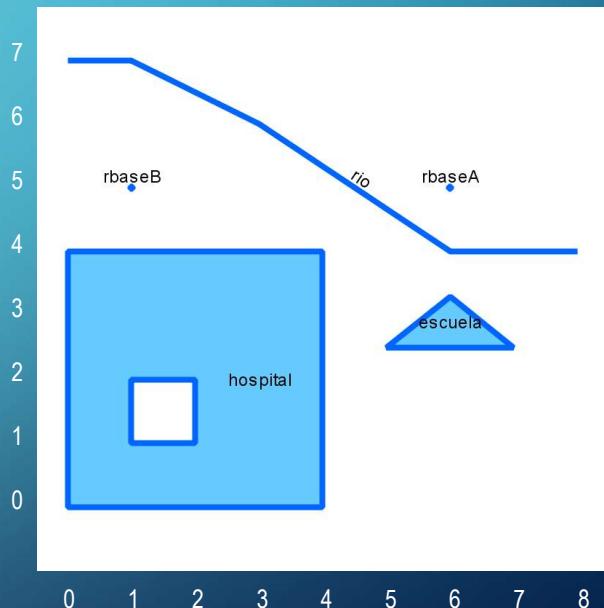


## PRACTIQUEMOS ...

Calcular la distancia entre la radiobaseB  
y el hospital

```
select g1.descrip, g2.descrip, g1.geom<->g2.geom  
from gis_example g1, gis_example g2  
where g1.descrip = 'rbaseB' and g2.descrip = 'hospital';
```

Rta: 3.162277

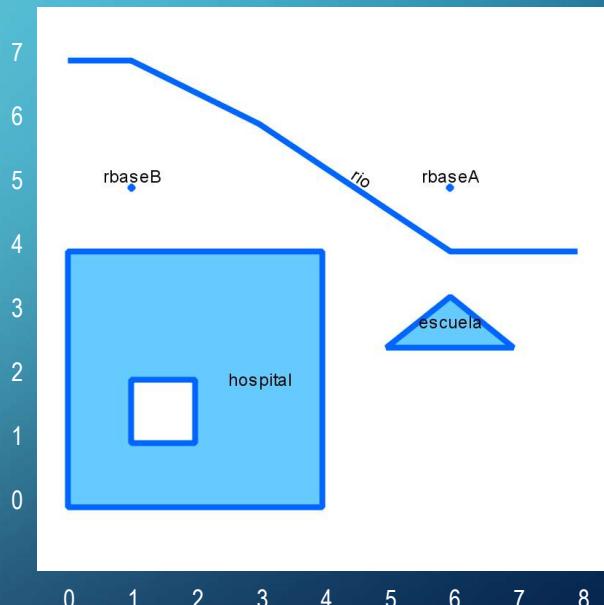


## PRACTIQUEMOS ...

Calcular la distancia entre la radiobaseB  
y el hospital

```
select g1.descrip, g2.descrip, g1.geom<#> g2.geom  
from gis_example g1, gis_example g2  
where g1.descrip = 'rbaseB' and g2.descrip = 'hospital';
```

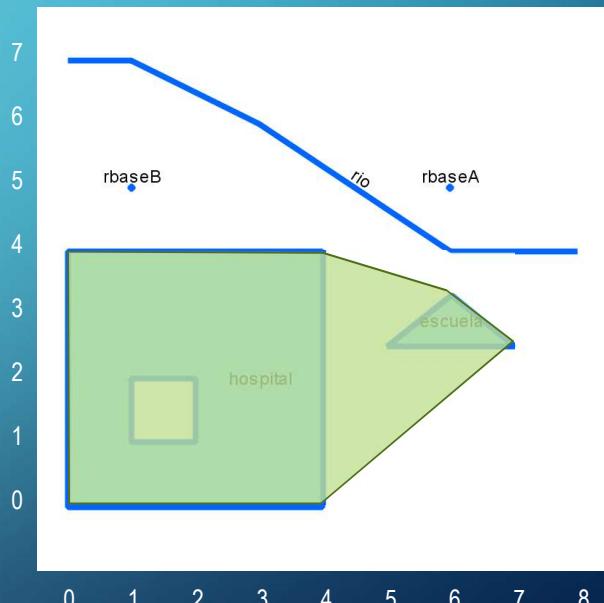
Rta: 1



## PRACTIQUEMOS ...

Como puedo obtener el polígono convexo que abarque al hospital y la escuela?

```
select st_convexhull(st_union(c1.geom, c2.geom))  
from gis_example c1, gis_example c2  
where c1.descrip='hospital' and c2.descrip='escuela';
```

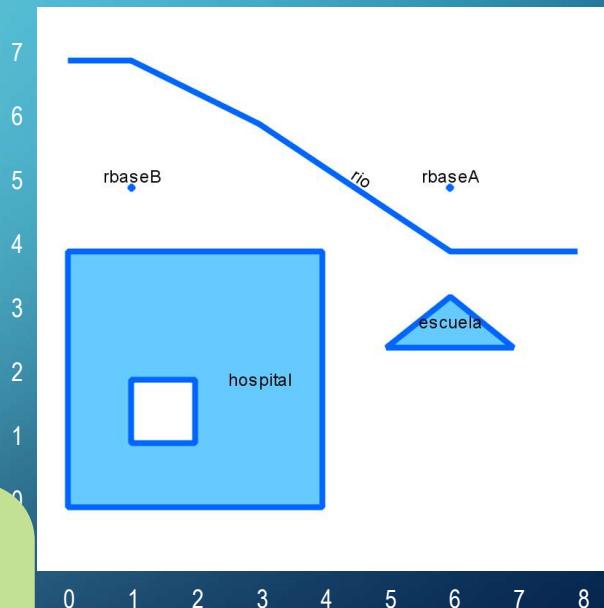


## PRACTIQUEMOS ...

Detectar que elementos serían alcanzados por el río, si este se desbordara 1 m en todo su borde.

```
SELECT g2.descrip, g2.geom  
FROM gis_example g1, gis_example g2  
WHERE (st_intersects(st_buffer(g1.geom, 1.0), g2.geom)) AND  
g1.descrip='rio' and g1.id<>g2.id;
```

Rta:  
rbaseA  
escuela  
hospital



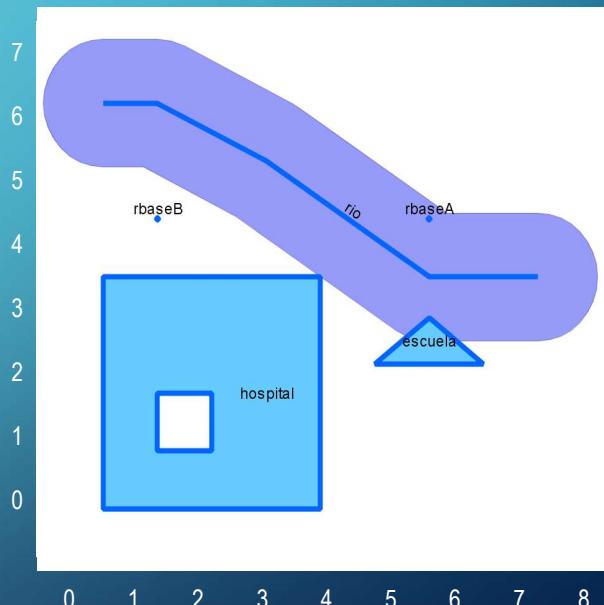
## PRACTIQUEMOS ...

Detectar que elementos serian alcanzados por el rio, si este se desbordara 1 m en todo su borde.

```
SELECT g2.descrip, g2.geom  
FROM gis_example g1, gis_example g2  
WHERE (st_intersects(st_buffer(g1.geom, 1.0), g2.geom)) AND  
g1.descrip='rio' and g1.id<>g2.id;
```

**Lo quiero ver?**

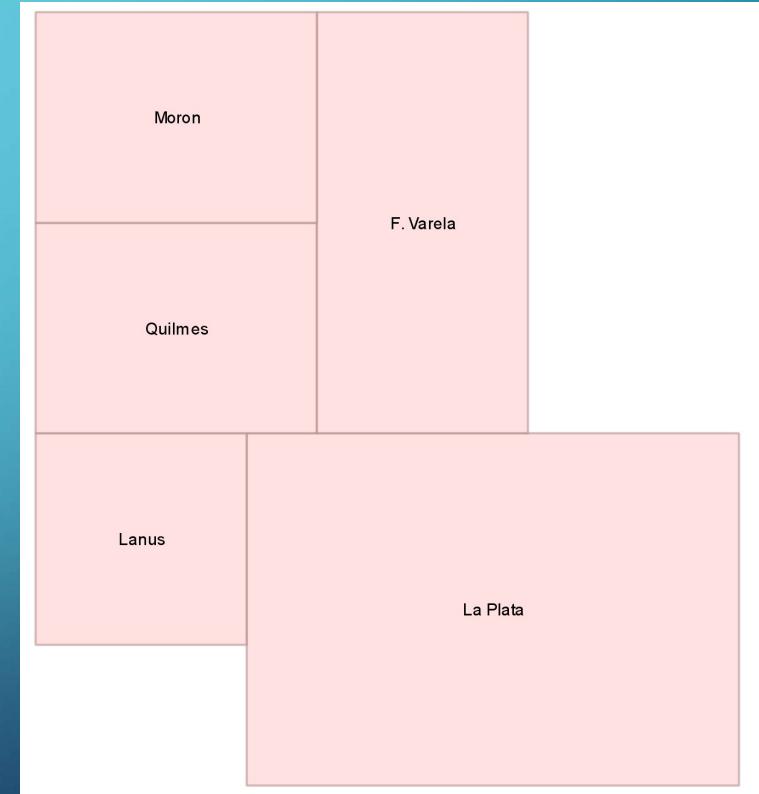
```
SELECT st_buffer(geom, 1.0)  
FROM gis_example  
WHERE descrip='rio';
```



## CONSULTAS ENTRE TODOS...

Ahora contamos con la capa **Ciudades**, dada por:

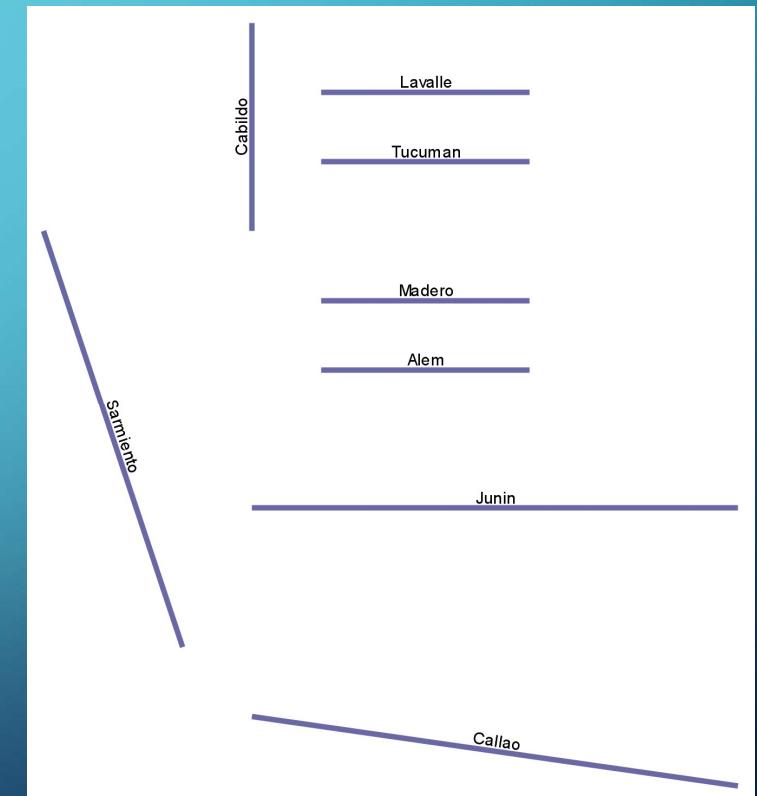
```
CREATE TABLE bc_ciudad  
(gid INT, name STRING, the_geom STRING);
```



# CONSULTAS ENTRE TODOS...

Y con la capa **Calles**, representada por:

```
CREATE TABLE calles  
(gid INT, name STRING, the_geom STRING);
```

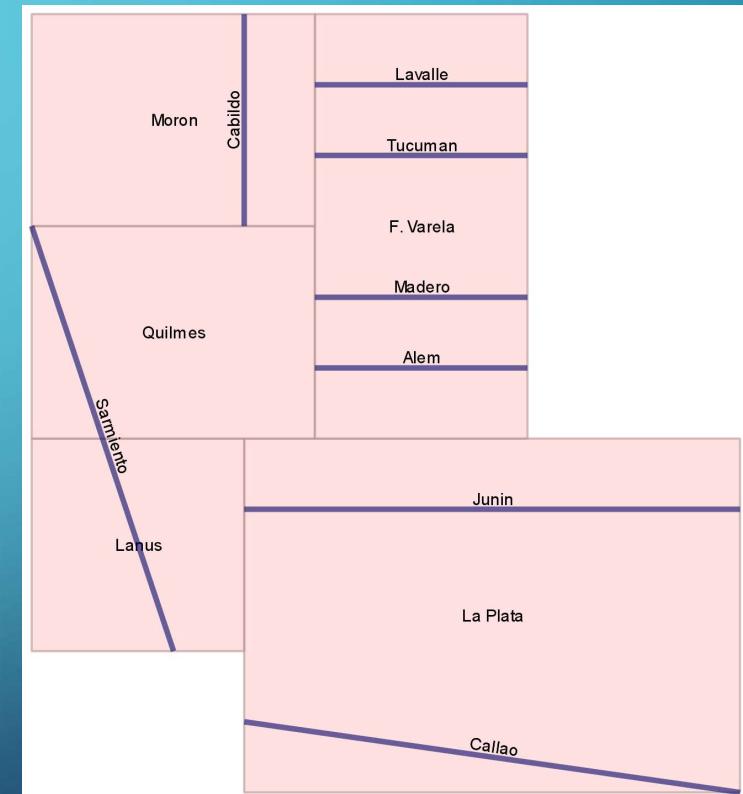


# CONSULTAS ENTRE TODOS...

Cual es la extension de la ciudad de La Plata?

```
SELECT ST_Area(the_geom) AS Hect  
FROM bc_ciudad  
WHERE name = 'La Plata';
```

Hect  
-----  
32657.9103824927

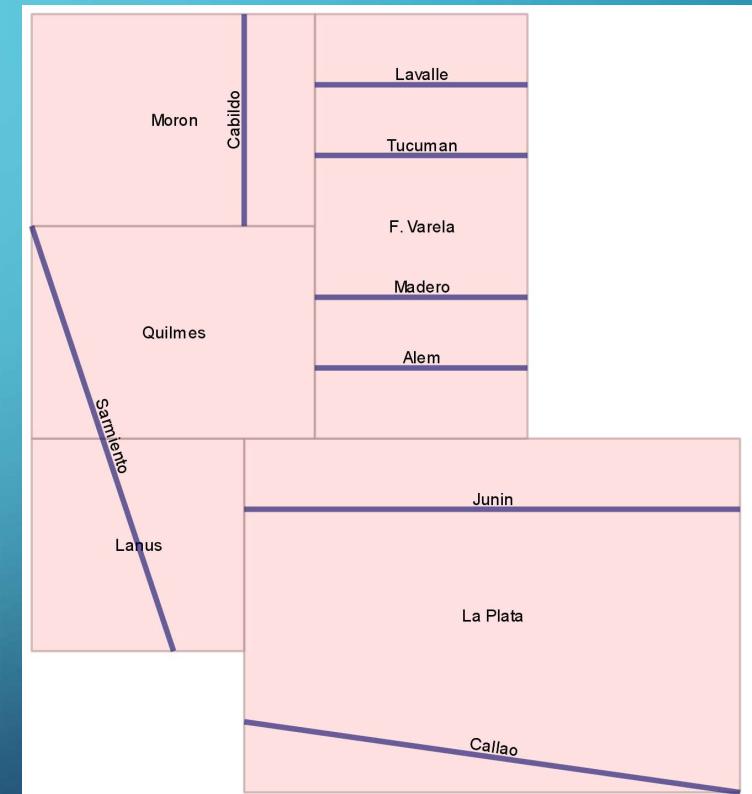


# CONSULTAS ENTRE TODOS...

Cual es la ciudad de mayor extension?

```
SELECT name, ST_Area(the_geom) AS sup  
FROM bc_ciudad  
ORDER BY sup DESC  
LIMIT 1;
```

"La Plata"; 35

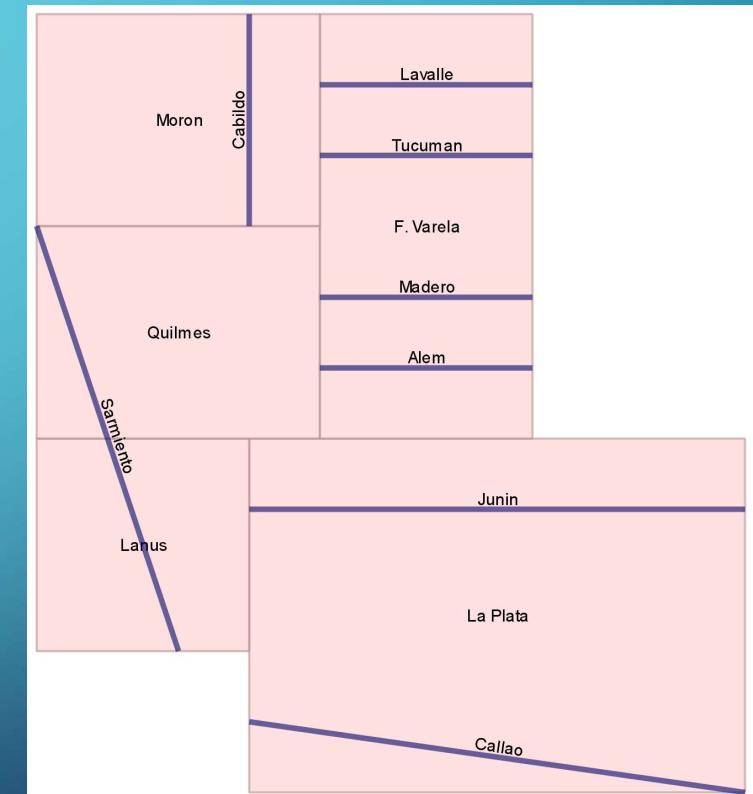


# CONSULTAS ENTRE TODOS...

Cual es la longitud total de todas las calles?

```
SELECT SUM(ST_Length(the_geom)) AS lg  
FROM bc_calle;
```

**lg**  
-----  
**70842.1243039643**

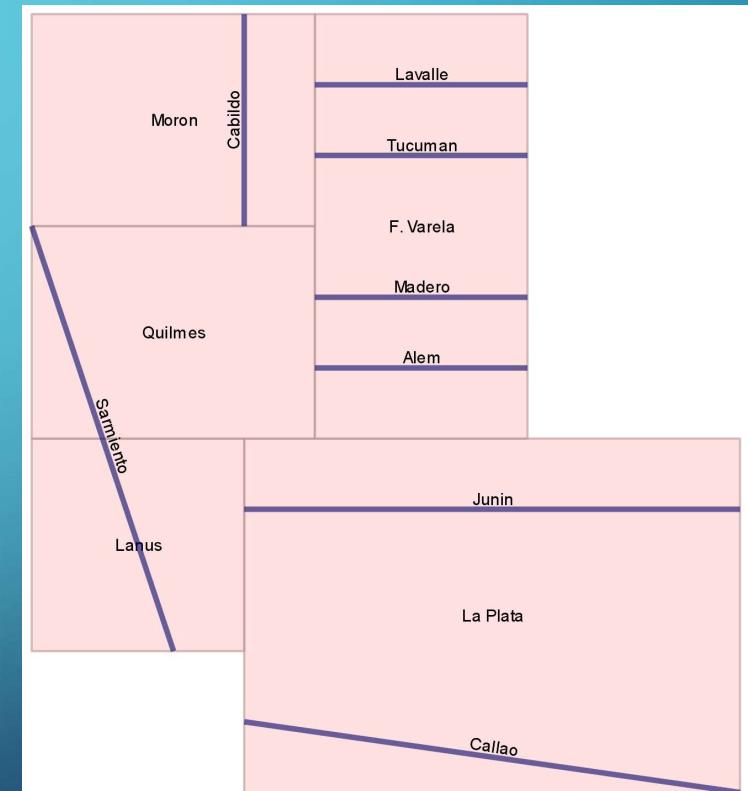


# CONSULTAS ENTRE TODOS...

Cual es la longitud de todas las rutas contenidas en cada ciudad?

```
SELECT ci.name, SUM(  
ST_Length(ST_intersection(ca.the_geom,ci.the_geom)))  
FROM bc_calle ca, bc_ciudad ci  
WHERE ST_Intersects(ci.the_geom, ca.the_geom)  
GROUP BY ci.name;
```

```
"Quilmes"; 3.16227766016838  
"La Plata"; 14.0710678118655  
"Moron"; 3  
"Lanus"; 3.16227766016838  
"F. Varela"; 12
```

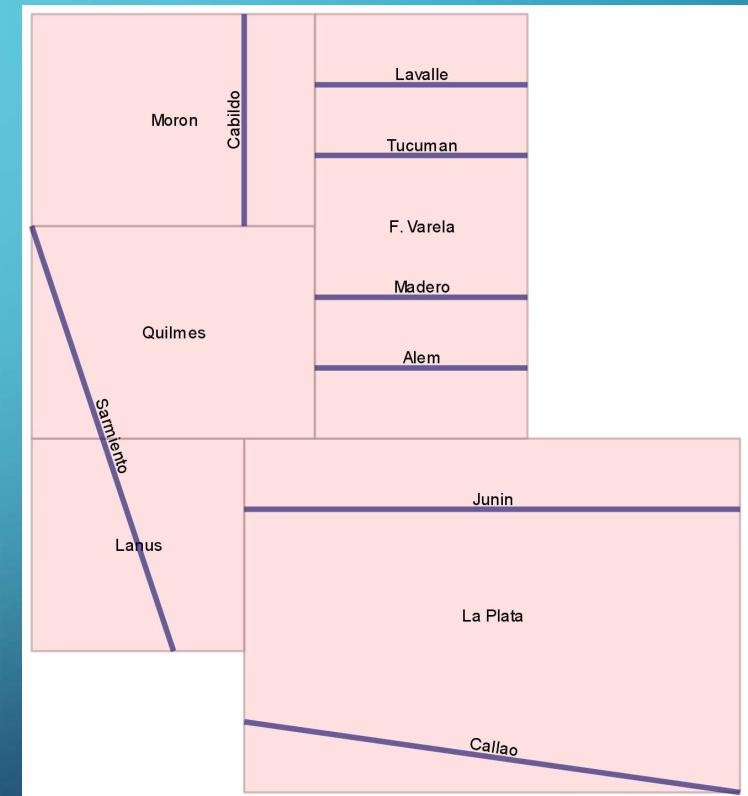


# CONSULTAS ENTRE TODOS...

Buscar las ciudades limitrofes con  
F. Varela

```
SELECT c1.name  
FROM bc_ciudad c1, bc_ciudad c2  
WHERE ST_Touches(c1.the_geom, c2.the_geom)  
AND c2.name = 'F. Varela'
```

"La Plata"  
"Quilmes"  
"Moron"

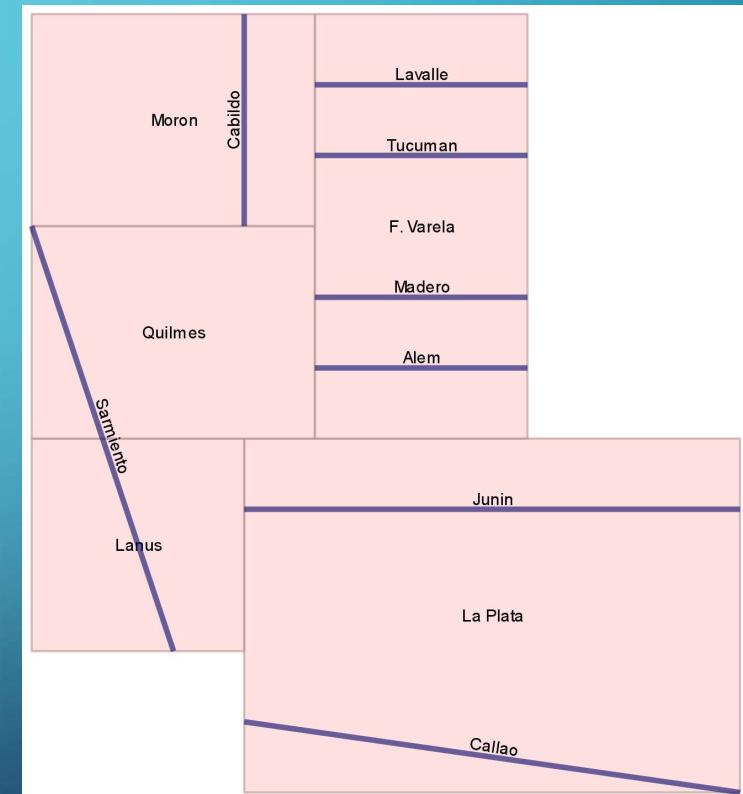


# CONSULTAS ENTRE TODOS...

Listar las ciudades ordenadas (de mayor a menor) por cantidad de ciudades limítrofes

```
SELECT c1.name,
      (SELECT COUNT(*)
       FROM bc_ciudad c2
       WHERE ST_Touches(c1.the_geom, c2.the_geom))
      FROM bc_ciudad c1
      ORDER BY 2 DESC
```

"Quilmes";4  
"La Plata";3  
"F. Varela";3  
"Lanus";2  
"Moron";2



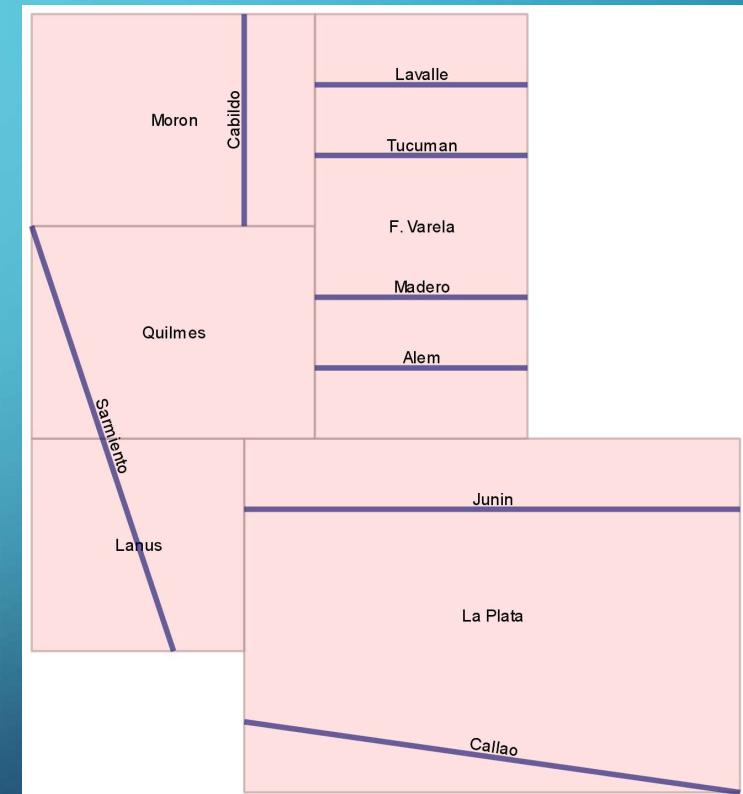
# CONSULTAS ENTRE TODOS...

Listar las ciudades ordenadas  
ascendentemente por cantidad de calles  
internas

```
SELECT c1.name,  
       (SELECT COUNT(*)  
        FROM bc_calle c2  
        WHERE ST_Within(c2.the_geom, c1.the_geom))  
    FROM bc_ciudad c1  
   ORDER BY 2
```



```
"Lanus";0  
"Quilmes";0  
"Moron";1  
"La Plata";2  
"F. Varela";4
```

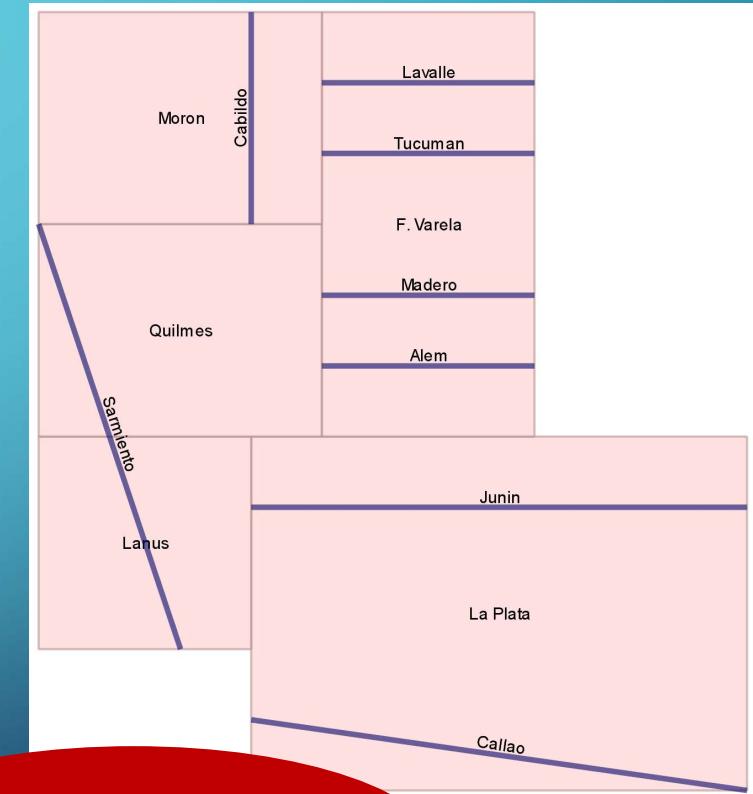


# CONSULTAS ENTRE TODOS...

Listar las ciudades ordenadas  
ascendentemente por cantidad de calles  
internas

```
SELECT c1.name,  
       (SELECT COUNT(*)  
        FROM bc_calle c2  
        WHERE ST_Within(c2.the_geom, c1.the_geom)  
        OR ST_Crosses(c2.the_geom, c1.the_geom))  
    FROM bc_ciudad c1  
   ORDER BY 2
```

"Lanus"; 1  
"Quilmes"; 1  
"Moron"; 1  
"La Plata"; 1  
"F. Varela"; 1



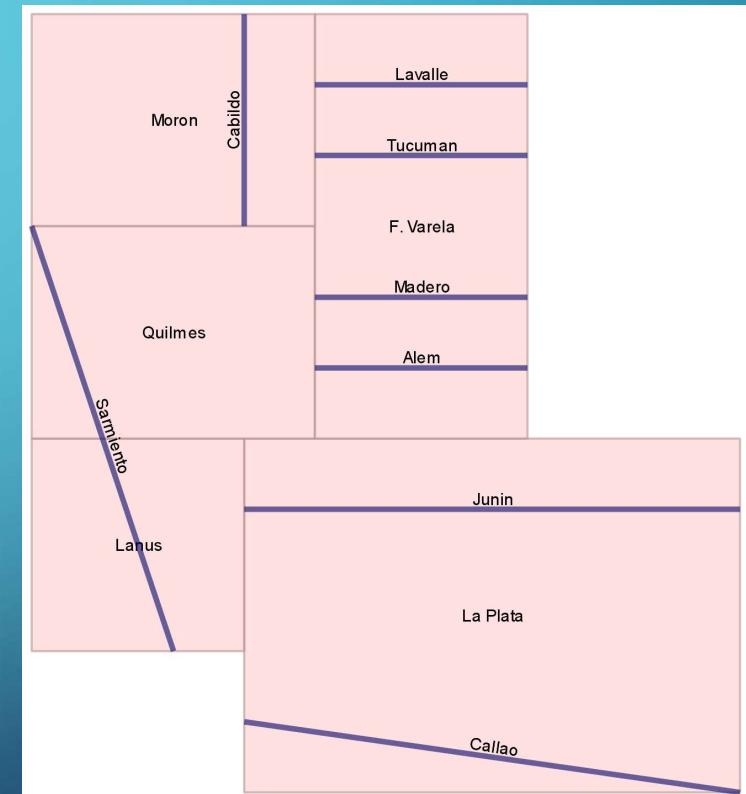
Hubiera servido  
ST\_Intersects?

# CONSULTAS ENTRE TODOS...

Cual es la longitud de la calle  
Sarmiento en Quilmes?

```
SELECT ci.name,  
SUM(ST_Length(ST_intersection(ca.the_geom,ci.the_geom)))  
FROM bc_calle ca, bc_ciudad ci  
WHERE ST_Intersects(ci.the_geom, ca.the_geom) AND  
ci.name='Quilmes'
```

"Quilmes"; 3.16227766016838

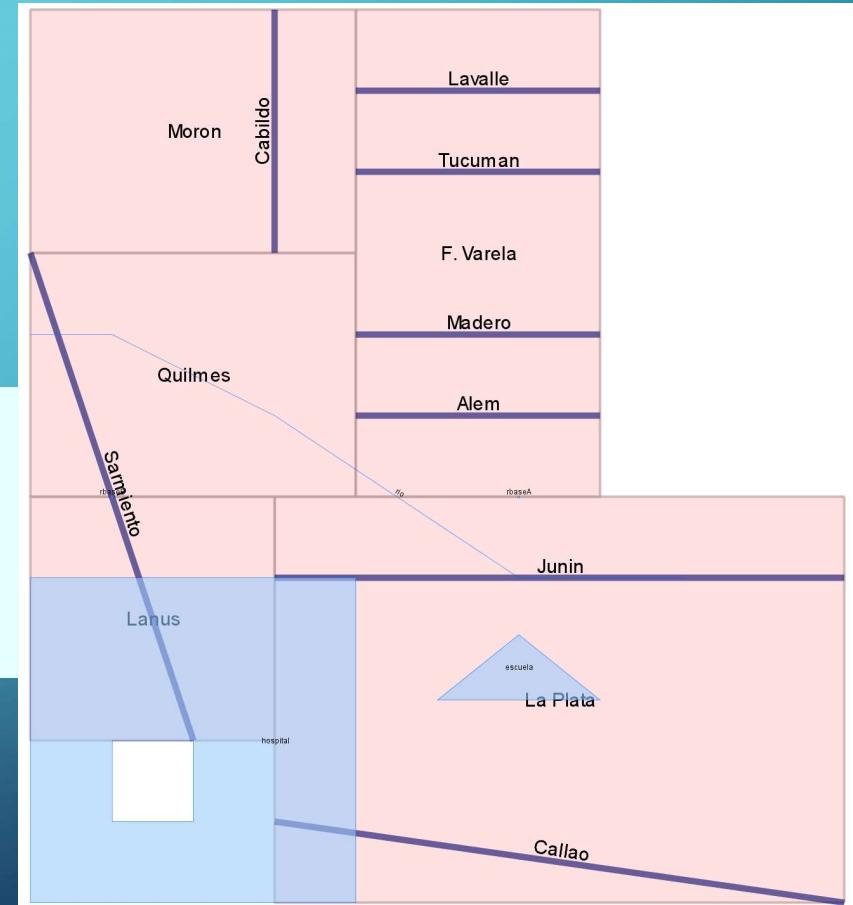


## UN PASITO MAS...

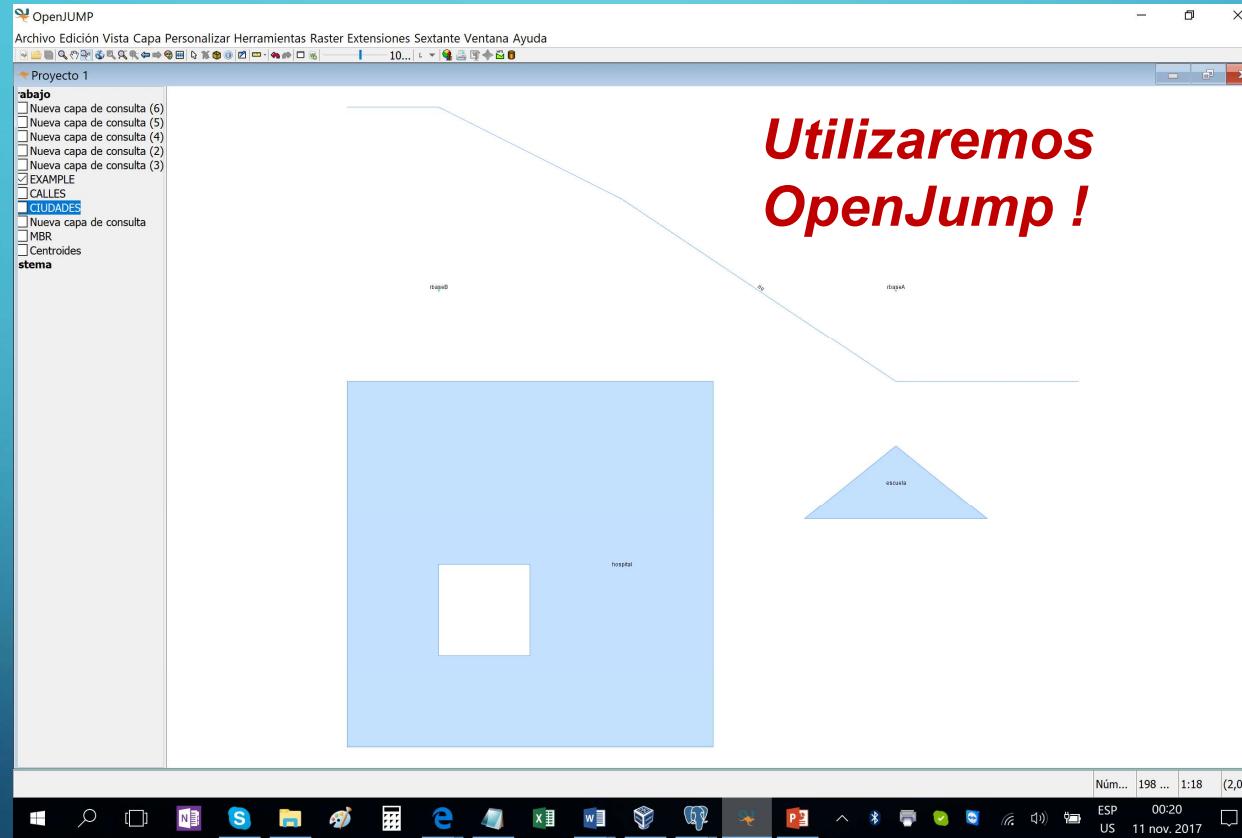
Usando todos los mapas, buscar las 2 calles mas cercanas a la escuela.

```
SELECT c.name, ST_Distance(c.the_geom, g.geom) as d  
FROM gis_example g, bc_calle c  
WHERE g.descrip='escuela'  
ORDER BY d  
LIMIT 2;
```

"Junin"; 0.7  
"Callao"; 1.76776695296637



# PODEMOS VISUALIZAR LOS RESULTADOS?





# MANOS A LA OBRA !!!

<https://postgis.net/docs/reference.html>