

Bases de Datos Geográficas PostGIS

8as Jornadas de Latinoamérica y Caribe gvSIG

Ing. Bruno Rienzi
MSc. Ing. Raquel Sosa



Agenda General

- ▶ Estándar SFS–SQL de OGC
- ▶ Ejemplos de Operadores Espaciales en SQL
- ▶ PostGIS
- ▶ Ejercicios

SFS – SQL

- ▶ Se definen 3 tipos de tablas:
 - FEATURE_TABLE: Es toda tabla que almacena un conjunto de *features* (objetos geográficos). Corresponde al concepto de *layer* (capa geográfica). Cada fila es un objeto geográfico y cada columna es una propiedad de ese objeto. Una de esas columnas debe corresponder a la geometría de ese objeto (o una FK a la misma).
 - Ej. Una capa de polígonos que representan ciudades la representamos como una *feature table* Ciudad(nombre, país, población, geometría, *gid*)

SFS – SQL

- GEOMETRY_TABLE : Es toda tabla que almacena geometrías, en el caso que la *feature table* correspondiente no las almacene directamente. Cada fila posee un identificador geográfico (GID).
- GEOMETRY_COLUMNS: Tabla de metadatos que posee una fila por cada columna geometría de la base, con los siguientes atributos:
 - ID de la *feature table* a la que corresponde la columna geometría
 - El nombre de la columna geometría
 - El SRID de la columna geometría
 - El tipo de geometría (Point, LineString, etc)
 - La dimensión del SRS utilizado (2D, 3D, etc.)
 - El ID de la *geometry_table* que almacena la geometría (podría ser la misma *feature table*)

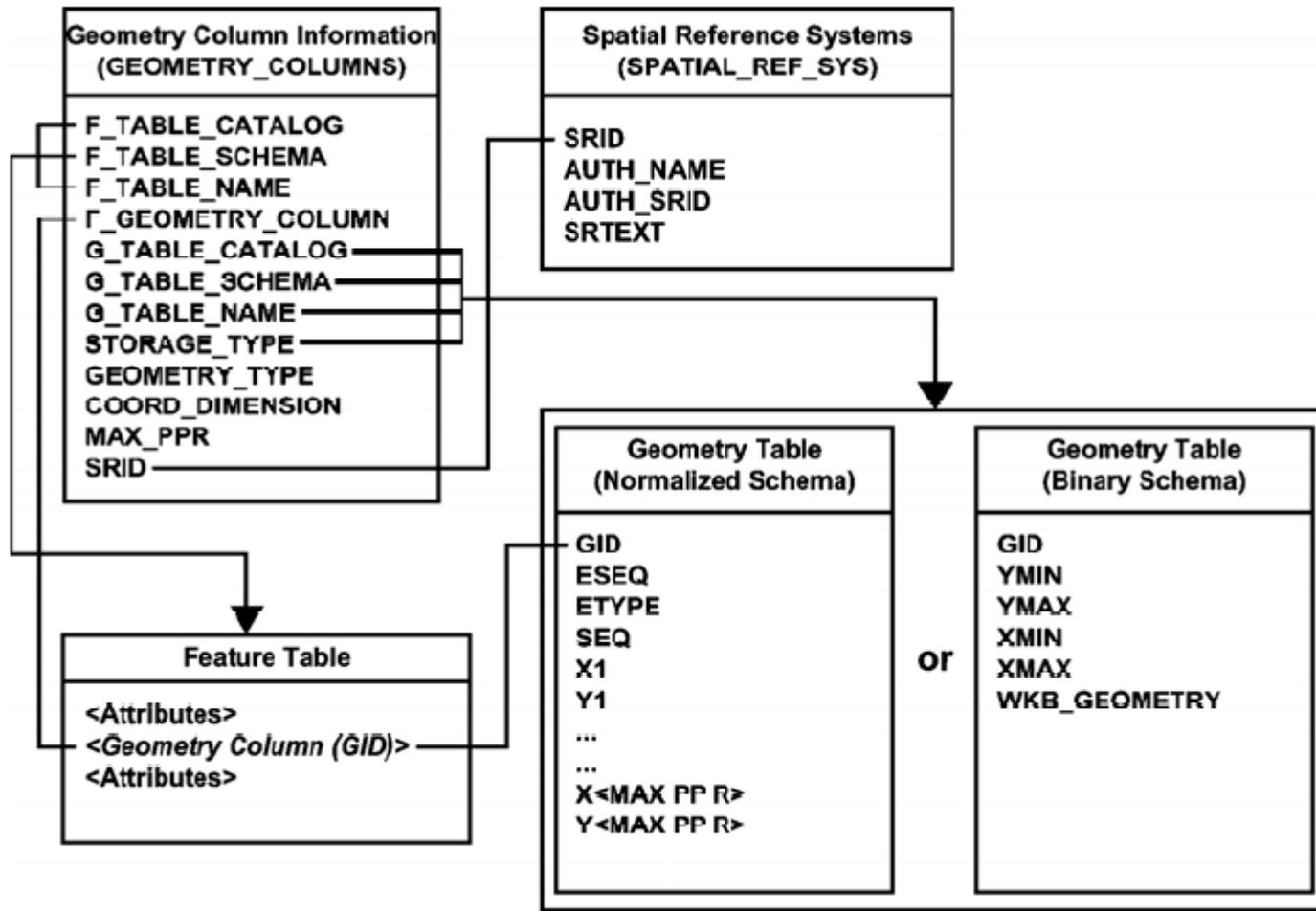
SFS – SQL

- SPATIAL_REF_SYS: Es el diccionario de códigos de sistemas de referencia espaciales (SRS). Solamente es necesaria para hacer operaciones de re-proyección, pero resulta útil para consulta durante el desarrollo. Dentro de esta tabla, interesan particularmente los campos SRID y STEXT.
- Ej. Queremos obtener el SRID de la proyección que corresponde al datum Yacaré (

```
SELECT srid FROM spatial_ref_sys WHERE srtext LIKE '%Yacare%';
```


srid = 4309

SFS – SQL



SFS – SQL

Table 4: Geometry type codes

Code	Geometry type	Coordinates
0	GEOMETRY	\\ IN X Y
1	POINT	\\ IN X Y
2	LINESTRING	\\ IN X Y
3	POLYGON	\\ IN X Y
4	MULTIPOINT	\\ IN X Y
5	MULTILINESTRING	\\ IN X Y
6	MULTIPOLYGON	\\ IN X Y
7	GEOMCOLLECTION	\\ IN X Y
13	CURVE	\\ IN X Y
14	SURFACE	\\ IN X Y
15	POLYHEDRALSURFACE	\\ IN X Y
1000	GEOMETRYZ	\\ IN X Y Z
1001	POINTZ	\\ IN X Y Z
1002	LINESTRINGZ	\\ IN X Y Z

Code	Geometry type	Coordinates
1003	POLYGONZ	\\ IN X Y Z
1004	MULTIPOINTZ	\\ IN X Y Z
1005	MULTILINESTRINGZ	\\ IN X Y Z
1006	MULTIPOLYGONZ	\\ IN X Y Z
1007	GEOMCOLLECTIONZ	\\ IN X Y Z
1013	CURVEZ	\\ IN X Y M
1014	SURFACEZ	\\ IN X Y M
1015	POLYHEDRALSURFACEZ	\\ IN X Y Z
2000	GEOMETRY	\\ IN X Y M
2001	POINTM	\\ IN X Y M
2002	LINESTRINGM	\\ IN X Y M
2003	POLYGONM	\\ IN X Y M
2004	MULTIPOINTM	\\ IN X Y M
2005	MULTILINESTRINGM	\\ IN X Y M
2006	MULTIPOLYGONM	\\ IN X Y M
2007	GEOMCOLLECTIONM	\\ IN X Y M
2013	CURVEM	\\ IN X Y M
2014	SURFACEM	\\ IN X Y M
2015	POLYHEDRALSURFACEM	\\ IN X Y M
3000	GEOMETRYZM	\\ IN X Y Z M
3001	POINTZM	\\ IN X Y Z M
3002	LINESTRINGZM	\\ IN X Y Z M

Ejemplos de Operadores Espaciales en SQL



- ▶ Utilizaremos las siguientes *feature tables*:
 - Ciudades(gid, código, nombre, población, geom), capa de polígonos.
 - Calles (gid, código, nombre, geom), capa de líneas.
 - Hoteles (gid, nombre, dirección, estrellas, capacidad, geom) capa de puntos.
 - Rios(gid, nombre, geom), capa de líneas
- ▶ Operadores en el SELECT
 - Obtener nombre, código y área de la ciudad de Montevideo:
SELECT c.nombre, c.codigo, ST_AREA(c.geom) AS area
FROM Ciudades c WHERE c.nombre='Montevideo';



Ejemplos de Operadores Espaciales en SQL

- Obtener nombre y longitud del río con codigo=223
SELECT r.nombre, **ST_LENGTH**(r.geom) AS longitud
FROM Rios r WHERE r.codigo=223;
- Listar las nombre y densidad de población de ciudades en orden decreciente de densidad:
SELECT c.nombre, c.poblacion/**ST_AREA**(c.geom) AS densidad
FROM Ciudades c;
ORDER BY densidad DESC;



Ejemplos de Operadores Espaciales en SQL

- ▶ Operadores en el WHERE (Join Espacial):
 - Obtener la cantidad de hoteles 4 estrellas en la ciudad de Colonia:
SELECT COUNT(h.nombre) FROM Hoteles h, Ciudades c
WHERE **ST_CONTAINS**(c.geom, h.geom)
AND c.nombre='Colonia' AND h.estrellas=4
 - Listar todas las ciudades que se encuentren a menos de 100 km del río Uruguay
SELECT c.nombre FROM Ciudades c, Rios r
WHERE **ST_OVERLAPS**(c.geom, **ST_BUFFER**(r.geom, 100))
AND r.nombre='Uruguay'



PostGIS

- ▶ Implementaciones SFS–SQL

- ▶ Libres:
 - PostgreSQL/PostGIS (desde: 2001)
 - MySQL Spatial Extensions (desde: 2003)

- ▶ Algunas comerciales:
 - Oracle Spatial (desde: 1998)
 - ESRI ArcSDE (desde: 1996)
 - SQL Server (desde: 2008)

PostGIS

- ▶ Extiende el DBMS PostgreSQL con nuevos tipos de datos y funciones almacenadas
- ▶ Incluye todos los tipos OGC SFS–SQL
- ▶ Almacena las geometrías en EWKB (Extended Well–known Binary).
- ▶ Soporta geometrías avanzadas y rasters.
- ▶ Posee un amplio conjunto de funciones (operadores espaciales, álgebra de mapas, operaciones raster/vectorial, transformaciones, etc.)
- ▶ Agrega tipo Geography: solo coordenadas geográficas. Ventaja: exactitud global (no hay proyecciones). Desventaja: Operaciones complejas (menos performance).
- ▶ Utiliza *Feature Tables* con GID y geometría y *Geometry_Columns* como una vista
- ▶ Utiliza índices espaciales (mediante árboles GiST).
- ▶ Operaciones avanzadas (ruteo, topologías de redes, geocodificación, objetos 3D, etc.)

Ejercicios

Preparación de la base de datos

- 1) Creamos una base de datos utilizando pgadmin
- 2) Agregamos la extensión geográfica: `CREATE EXTENSION postgis;`
- 3) Exportamos shapefiles a scripts SQL: `shp2pgsql -s 32721 -I -W LATIN1 shp1 ft1 | psql -U user -d postgis`
Se crea la tabla **ft1** a partir del shapefile **shp1**. Se especifica el SRID **32721** y la creación de índice espacial. Se indica que la codificación del dbf es **LATIN1**. Se ejecuta el script resultante en la base **postgis** con el usuario **user**.

Ejercicios

- Parámetros:
 - -s indica el SRID(sistema de referencia espacial)
 - -I indica que se cree un índice espacial (sobre el campo geometry)
 - -W indica que los atributos descriptivos del dbf sean convertidos desde la codificación especificada hacia UTF8. El SQL resultado contiene un comando SET CLIENT_ENCODING to UTF8, para que luego pueda ser reconvertido de UTF8 a la codificación configurada en la base de datos.
- ▶ Inversamente, se puede exportar una feature table a un shapefile

```
pgsql2shp -f shp1 -u user postgis ft1
```

Ejercicios

- ▶ La función `st_astext` nos permite ver la representación WKT (Well-known text) de una geometría.

```
select st_astext(the_geom) from ft_ejes where  
nom_calle='SORIANO';
```

- ▶ Existen métodos similares para pasar a otras representaciones (`st_asgml`, `st_askml`, etc.)
- ▶ Nota: Si la geometría es muy grande, la salida se muestra en blanco en pgAdminIII

Ejercicios

- ▶ Consultas con operadores espaciales
 1. Devolver los departamentos del Uruguay, en orden decreciente de área calculada.
 2. Encontrar las localidades con un área inferior a 10 millones de metros cuadrados.
 3. Devolver los ejes en formato WKT de la calle Magallanes.
 4. Devolver la longitud de la calle Magallanes.
 5. Devolver el nombre del departamento que contiene la calle Magallanes
 6. Devolver todos los nombres de las calles que cruzan la calle Magallanes.
 7. Devolver las manzanas que se encuentran en el cruce de Magallanes y Colonia.
 8. Encontrar las manzanas que tienen intersección con los ejes de calles.