



# Formation PostGIS

Vecteurs – analyse et structure / gestion des tables - Octobre 2020

Fabien Guerreiro

Supports sous Licence Ouverte Etalab

**Crédits** : détails des crédits dans les supports.

INSTITUT NATIONAL SUPÉRIEUR DES SCIENCES AGRONOMIQUES, DE L'ALIMENTATION ET DE L'ENVIRONNEMENT

MEMBRE DE



# Jointures attributaires

## Les requêtes

- ... WHERE  
NomTable1.NomColonne1 =  
NomTable2.NomColonne2
- SELECT <colonnes> FROM <table1>  
JOIN <table2> ON <condition de  
jointure>



#	Name	Type	Null	Default
0	PKLUID	INTEGER	Y	
1	Geometry	MULTIPOLYGON	Y	
2	ID_BDCARTO	INTEGER	Y	
3	NOM_COMM	TEXT(50)	Y	
4	INSEE_COMM	TEXT(5)	Y	
5	STATUT	TEXT(20)	Y	
6	X_COMMUNE	INTEGER	Y	
7	Y_COMMUNE	INTEGER	Y	
8	SUPERFICIE	INTEGER	Y	
9	POPULATION	INTEGER	Y	
10	INSEE_CANT	TEXT(2)	Y	
11	INSEE_ARR	TEXT(1)	Y	
12	NOM_DEPT	TEXT(30)	Y	
13	INSEE_DEPT	TEXT(2)	Y	
14	NOM_REGION	TEXT(30)	Y	
15	INSEE_REG	TEXT(2)	Y	

#	Name	Type	Null	Default
0	PKLUID	INTEGER	Y	
1	Geometry	MULTIPOLYGON	Y	
2	DepCom	TEXT(5)	Y	
3	Nom_Com	TEXT(40)	Y	
4	Iris	TEXT(4)	Y	
5	DcomIris	TEXT(9)	Y	
6	Nom_Iris	TEXT(40)	Y	
7	Typ_Iris	TEXT(1)	Y	
8	Origine	TEXT(1)	Y	

## Exercice

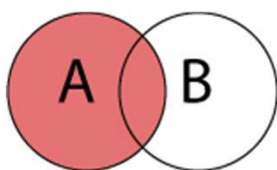
A l'aide du constructeur graphique de requête, rechercher les IRIS des communes simples, et trier le résultat par nom d'IRIS.

Doivent apparaître dans le résultat, la géométrie des IRIS, leur nom, le nom de leur commune, et le nom de leur département.

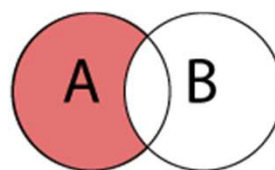
# Jointures attributaires

## Les requêtes

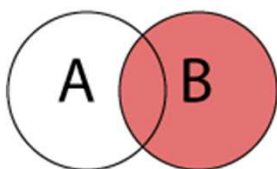
- SELECT <colonnes> FROM <table1> JOIN <table2> ON <condition de jointure>



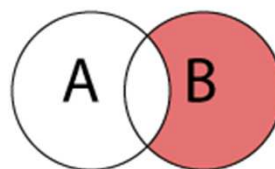
**LEFT JOIN**  
SELECT \*  
FROM A  
LEFT JOIN B ON A.key = B.key;



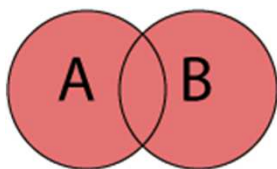
**LEFT JOIN moins intersection B**  
SELECT \*  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL;



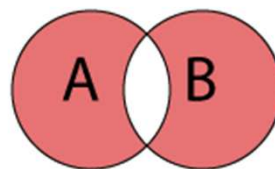
**RIGHT JOIN**  
SELECT \*  
FROM A  
RIGHT JOIN B ON A.key = B.key;



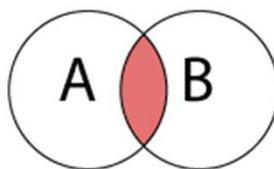
**RIGHT JOIN moins intersection A**  
SELECT \*  
FROM A  
RIGHT JOIN B ON A.key = B.key  
WHERE A.key IS NULL;



**FULL JOIN**  
SELECT \*  
FROM A  
FULL OUTER JOIN B ON A.key = B.key;



**FULL JOIN moins l'intersection**  
SELECT \*  
FROM A  
FULL OUTER JOIN B ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL;



**INNER JOIN**  
SELECT \*  
FROM A  
INNER JOIN B ON A.key = B.key;

# Jointures spatiales

## Comparaison

ST\_Equals(geometry A, geometry B)  
ST\_Intersects(geometry A, geometry B)  
ST\_Disjoint(geometry A, geometry B)  
ST\_Crosses(geometry A, geometry B)  
ST\_Overlaps(geometry A, geometry B)  
ST\_Touches(geometry A, geometry B)  
ST\_Within(geometry A, geometry B)  
ST\_Contains(geometry A, geometry B)

Les fonctions suivantes sont également intéressantes :

ST\_Dwithin(geometry A, geometry B, distance)  
ST\_Distance(geometry A, geometry B)

Exemple :

```
SELECT id_a, table_a.geom FROM table_a, table_b WHERE  
ST_Intersects(table_a.geom, table_b.geom);
```

Renvoie les objets de table\_a qui intersectent ceux de table\_b, sans les découper.

# Géométrie dans les agrégations

## GROUP BY

Toutes les colonnes en sortie (sauf celle du critère de regroupement) doivent faire l'objet d'une fonction d'agrégation, ce doit être également le cas pour la géométrie :

- **ST\_UNION(geom)**
- **ST\_Multi()**
- **ST\_Linemerge()**

QGIS demande un identifiant unique, à recréer avec la requête :

- **row\_number() over()**

## ST\_Intersection(geometry A, geometry B)

Opération sur les géométries

Exemple :

```
SELECT id_a, ST_Intersection(table_a.geom, table_b.geom) AS geom  
FROM table_a, table_b;
```

Découpe les objets de table\_a pour ne garder que ceux qui intersectent table\_b.

## Exercice pratique

1. Quels sont les ponctuels hydrographiques de la commune de La Flèche ?
2. Quelle est la longueur de la 'rivière le loir' dans chacune des communes intersectées par le cours d'eau ?
3. Sélectionner les 'PONCTUELS HYDROGRAPHIQUES' qui sont à moins de 5 km d'un établissement d'enseignement (couche ETABLISSEMENT)
4. Quel est l'établissement le plus proche du centroïde de la commune de la Flèche?

On utilisera les coordonnées X\_COMMUNE et Y\_COMMUNE et la fonction **ST\_Makepoint()**. Le SRID (Identifiant du Système de Référence Spatial) est 2154, mais on pourra le cas échéant généraliser la requête à tout SRID en utilisant la fonction **ST\_Srid()** qui récupère le srid d'une géométrie.

# Définition d'une table

**Gestionnaire de base de données**

Base de données | Schéma | Table

Import de couche/fichier | Exporter vers le fichier...

**Fournisseurs de données**

- GeoPackage
- Oracle Spatial
- PostGIS
  - PostgreSQL12\_local
    - public
      - geography\_columns
      - geometry\_columns
      - route\_xy
      - spatial\_ref\_sys
- SpatialLite
- Couches virtuelles
  - Couches du projet

**route\_xy**

**Informations générales**

Type de relation : Table  
 Propriétaire : postgres  
 Pages : 218  
 Lignes (estimation) : 3818  
 Privilèges : select, insert, update, ...

**PostGIS**

Colonne : geom  
 Géométrie : MULTILINESTRING  
 Dimension : 2  
 Réf. spatiale : RGF93 / Lambert-93 (2154)  
 Emprise estimée : 464325, 28125, 6734256, ...  
 Emprise : 464380, 85602, 6734312, ...

Aucun index spatial défini (en créer un)

**Champs**

#	Nom	Type
1	id	varchar
2	geom	geometry (MultiLineSt
3	prec_plani	float8
4	prec_alti	float8
5	nature	varchar (19)
6	numero	varchar (10)
7	nom_rue_g	varchar (100)
8	nom_rue_d	varchar (100)
9	importance	varchar (2)
10	d_admin	varchar (14)

pgAdmin 4

**pgAdmin** | Fichier | Objet | Outils | Aide

Objet : commune

**Informations générales**

Type : Table

**Champs**

#	Nom	Type
1	id	integer
2	geom	geometry (MultiPolygon, 2154)
3	id_bdcarto	bigint
4	nom_comm	character varying (50)
5	insee_comm	character varying (5)
6	statut	character varying (20)
7	x_commune	integer
8	y_commune	integer
9	superficie	bigint
10	population	integer
11	insee_cant	character varying (2)
12	insee_arr	character varying (1)
13	nom_dept	character varying (30)
14	insee_dept	character varying (2)
15	nom_region	character varying (30)
16	insee_reg	character varying (2)

**SQL**

```

1 -- Table: public.commune
2
3 -- DROP TABLE public.commune;
4
5 CREATE TABLE public.commune
6 (
7     id integer NOT NULL DEFAULT nextval('commune_id_seq'::regclass),
8     geom geometry(MultiPolygon,2154),
9     id_bdcarto bigint,
10    nom_comm character varying(50) COLLATE pg_catalog."default",
11    insee_comm character varying(5) COLLATE pg_catalog."default",
12    statut character varying(20) COLLATE pg_catalog."default",
13    x_commune integer,
14    y_commune integer,
15    superficie bigint,
16    population integer,
17    insee_cant character varying(2) COLLATE pg_catalog."default",
18    insee_arr character varying(1) COLLATE pg_catalog."default",
19    nom_dept character varying(30) COLLATE pg_catalog."default",
20    insee_dept character varying(2) COLLATE pg_catalog."default",
21    nom_region character varying(30) COLLATE pg_catalog."default",
22    insee_reg character varying(2) COLLATE pg_catalog."default",
23    CONSTRAINT commune_pkey PRIMARY KEY (id)
24 )
25
26 TABLESPACE pg_default;
27
28 ALTER TABLE public.commune
29     OWNER to postgres;
30 -- Index: sidx_commune_geom
31
32 -- DROP INDEX public.sidx_commune_geom;
33
34 CREATE INDEX sidx_commune_geom
35     ON public.commune USING gist
36     (geom)
37     TABLESPACE pg_default;
        
```

**Dbmanager (QGIS)**

**PgAdmin**

# Définition SQL d'une table

## Base

```
CREATE TABLE commune  
(id integer NOT NULL, id_bdcarto integer, [...] );
```

## Séquence

```
DEFAULT nextval('"COMMUNE_id_seq"'::regclass)
```

## Clé primaire

```
ALTER TABLE commune  
ADD CONSTRAINT "COMMUNE_pkey" PRIMARY KEY(id);
```

## Droits

```
ALTER TABLE commune OWNER TO postgres;
```

## Géométrie

```
ALTER TABLE commune ADD COLUMN geom geometry(MultiPolygon,2154);
```



# La géométrie d'une table

## La table spatial\_ref\_sys

Références EPSG

	srid [PK] integer	auth_name character varying (256)	auth_srid integer	srtext character varying (2048)	proj4text character varying (2048)
1	2000	EPSG	2000	PROJCS[Anguilla 1957 / Britis...	+proj=tmerc +lat_0=0 +lon_0=-6...
2	2001	EPSG	2001	PROJCS[Antigua 1943 / British...	+proj=tmerc +lat_0=0 +lon_0=-6...
3	2002	EPSG	2002	PROJCS[Dominica 1945 / Briti...	+proj=tmerc +lat_0=0 +lon_0=-6...
4	2003	EPSG	2003	PROJCS[Grenada 1953 / Britis...	+proj=tmerc +lat_0=0 +lon_0=-6...
5	2004	EPSG	2004	PROJCS[Montserrat 1958 / Bri...	+proj=tmerc +lat_0=0 +lon_0=-6...
6	2005	EPSG	2005	PROJCS[St. Kitts 1955 / British...	+proj=tmerc +lat_0=0 +lon_0=-6...
7	2006	EPSG	2006	PROJCS[St. Lucia 1955 / Britis...	+proj=tmerc +lat_0=0 +lon_0=-6...
8	2007	EPSG	2007	PROJCS[St. Vincent 45 / Britis...	+proj=tmerc +lat_0=0 +lon_0=-6...
9	2008	EPSG	2008	PROJCS[NAD27(CGQ77) / SCo...	+proj=tmerc +lat_0=0 +lon_0=-5...
10	2009	EPSG	2009	PROJCS[NAD27(CGQ77) / SCo...	+proj=tmerc +lat_0=0 +lon_0=-5...
11	2010	EPSG	2010	PROJCS[NAD27(CGQ77) / SCo...	+proj=tmerc +lat_0=0 +lon_0=-6...

## La vue geometry\_columns

Géométrie des tables

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	formation	public	route_xy	geom		2	2154 MULTILINESTRING
2	formation	public	commune	geom		2	2154 MULTIPOLYGON
3	formation	public	etablissement	geom		2	2154 POINT
4	formation	public	ponctuel_hydrogra...	geom		2	2154 MULTIPOINT
5	formation	public	troncon_hydrogra...	geom		2	2154 MULTILINESTRING
6	formation	public	iris_extrait72	geom		2	2154 MULTIPOLYGON

## L'index spatial

```
CREATE INDEX sidx_commune_geom
```

```
ON commune
```

```
USING gist
```

```
(geom);
```

## La vue geography\_columns (coordonnées sphériques)

# La projection d'une table

## **geometry\_columns**

Vue des géométries

## **Modifier la projection d'une table**

```
SELECT UpdateGeometrySrid ('NomTable','geometry' ,srid);
```

## **Reprojeter une table**

```
ST_Transform(geometry,srid)
```

## **Effacer la géométrie**

```
SELECT DropGeometryColumn ('NomSchema', 'NomTable','geom');
```

## **Effacer une table géométrique**

```
DROP TABLE NomTable;
```

## **Exercice**

Créer la table *commune\_wgs84*, à partir de la table *commune* reprojétée en WGS84

# Le schéma, la table et la vue

## Le schéma

```
CREATE SCHEMA nom_du_schema ;  
SELECT * FROM nom_schema.nom_table ;
```

## La table

```
CREATE TABLE nom_table AS [..]  
ALTER TABLE, INSERT INTO et DROP
```

## La vue

```
CREATE VIEW nom_vue AS [..]
```

## Exercice pratique

1. Charger les couches suivantes dans PostGIS comme dans l'exercice du module précédent :

/BD\_TOPO/I\_ZONE\_ACTIVITE/PAI\_SANTE.SHP

/BD\_TOPO/H\_ADMINISTRATIF/COMMUNE.SHP

/BD\_TOPO/E\_BATI/BATI\_INDUSTRIEL.SHP

/BD\_TOPO/F\_VEGETATION/ZONE\_VEGETATION.SHP

Créer une nouvelle vue **BATI\_INDUSTRIEL10** et la charger dans QGIS en sélectionnant dans la table BATI\_INDUSTRIEL les 'Bâtiment industriels' (attention à la majuscule!) dont la hauteur est d'au moins 10 m

2. Créer la vue **conifere** les 'Forêt fermée de conifères' de la commune de La Flèche. Ne pas oubliez de mettre une condition de jointure entre les deux couches... qui devra être ici spatiale.
3. Calculer la somme des surfaces des 'Forêt fermée de feuillus' de la commune de la Flèche en ha (1ha = 10 000 m<sup>2</sup>), en faisant attention à ne prendre en compte que les parties de surfaces des polygones réellement situées à l'intérieur de la commune.