Secrétariat général

INITIATION SQL

Septembre 2022

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE
DE LA TRANSITION
ÉCOLOGIQUE
ET SOLIDAIRE

MINISTÈRE
DE LA COHÉSION
DES TERRITOIRES
ET DES RELATIONS
AVEC LES
COLLECTIVITÉS
TERRITORIALES

CONCEPTION :

ALAIN FERRATON (SNUM/MSP/GSG)

HERVÉ LUCQ (SNUM/UNI/DRC)

Secrétariat général

PROGRAMME

1. Environnements SQL
2. Qu’est-ce que le SQL ?
3. La sélection d’objet : SELECT
4. La création de vues et vues matérialisées
5. Les jointures entre tables
6. Les fonctions géographiques
7. Création d’une table
8. La gestion des tables : ALTER - DROP - RENAME
9. Insertion d’enregistrements dans une table : INSERT
10. La mise à jour des données : UPDATE
11. Synthèse des principales fonctions

Secrétariat
général

ENVIRONNEMENTS SQL

Secrétariat général

Environnements SQL

Qu'est-ce qu'un SGBD : définition

Une base de données (database en anglais), permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité.

Celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles. Dans la très grande majorité des cas, ces informations sont très structurées, et la base est localisée dans un même lieu et sur un même support. Ce dernier est généralement informatisé. (source Wikipédia)

La base de données est au centre des dispositifs informatiques de collecte, mise en forme, stockage et utilisation d'informations.

Le dispositif comporte :

- Un système de gestion de base de données (abréviation : SGBD)
- Un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu.

Secrétariat général

Environnements SQL

Qu'est-ce qu'un SGBD : généralités

Une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés **tables**.

Selon le modèle relationnel, une base de données consiste en une ou plusieurs relations entre tables.

Les lignes de ces relations sont appelées des nuplets ou **enregistrements**.

Les colonnes sont appelées des **attributs**.

Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de base de données relationnels **SGBDR**.

Pratiquement tous les systèmes relationnels utilisent le **langage SQL** pour interroger les bases de données.

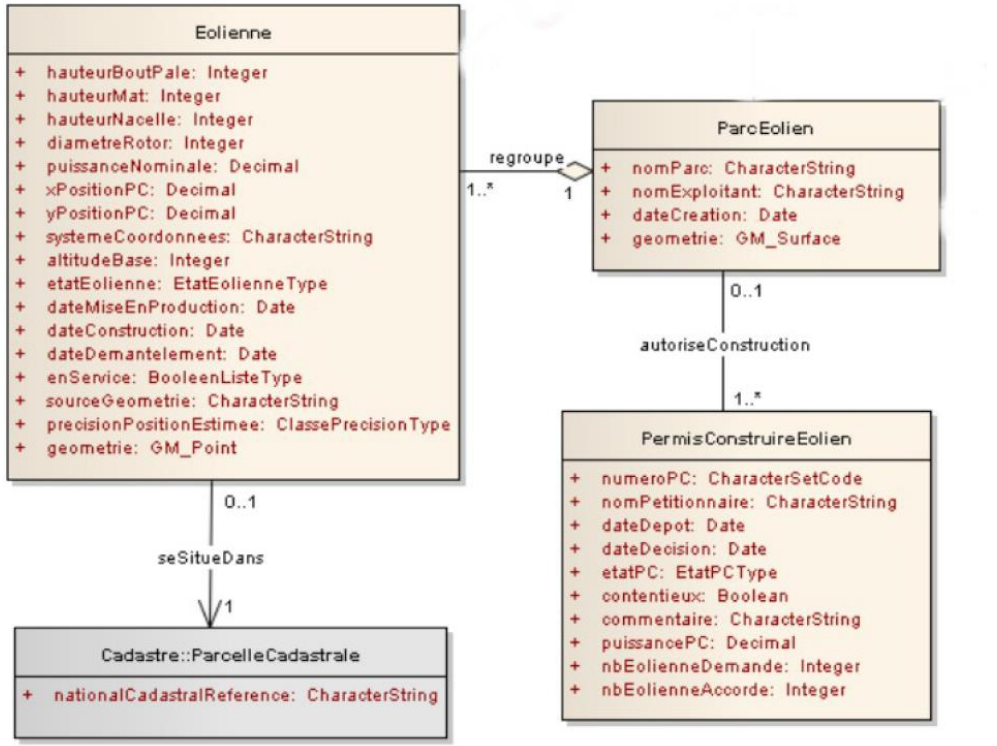
ID	ID_BDCARTO	NOM_COMM	INSEE_COMM	STATUT	X_COMMUNE	Y_COMMUNE	SUPERFICIE	POPULATION
2	720000282	SAINT-JEAN-DE-LA-MOTTE	72291	Commune simple	478935	6744018	3203	900
3	720000009	ARTHEZE	72009	Commune simple	466877	6748256	865	400
4	490000363	VAULANDRY	49380	Commune simple	472055	6726373	2765	300
5	490000100	CLEFS	49101	Commune simple	470066	6730106	2592	900
6	720000180	MAREIL-SUR-LOIR	72185	Commune simple	475371	6739051	1183	600
7	720000042	BOUSSE	72044	Commune simple	470515	6745247	1202	400
8	720000021	LE BAILLEUL	72022	Commune simple	462145	6746131	2746	1200
9	720000081	CLERMONT-CREANS	72084	Commune simple	473148	6741278	1782	1200
10	720000174	MALICORNE-SUR-SARTHE	72179	Chef-lieu de canton	469673	6750652	1513	2000
11	720000348	THORÉE-LES-PINS	72357	Commune simple	477876	6733984	2818	700
12	720000131	LA FONTAINE-SAINT-MARTIN	72135	Commune simple	479050	6747256	1372	600
13	720000149	LA FLECHE	72154	Sous-préfecture	470872	6737445	7421	15400
14	720000366	VILLAINES-SOUS-MALICORNE	72377	Commune simple	467557	6744178	1916	1000
15	720000104	CRE	72108	Commune simple	464444	6733839	1719	800
16	720000106	CROSMIERES	72110	Commune simple	463343	6741281	2045	900
17	490000301	SAINT-QUENTIN-LES-BEAUREPAIRE	49315	Commune simple	467128	6731077	751	300
18	720000024	BAZOUGES-SUR-LE-LOIR	72025	Commune simple	461769	6736584	2990	1200
19	720000096	COURCELLES-LA-FORET	72100	Commune simple	473803	6748536	1960	400
20	720000158	LIGRON	72163	Commune simple	474237	6745574	1348	500

Enregistrement

Secrétariat général

Environnements SQL

Qu'est-ce qu'un SGBD : exemple



Extrait du modèle relationnel du standard COVADIS de l'Éolien terrestre (formalisme UML)

Le langage SQL permet de gérer la base de données et également de l'interroger. Par exemples :

- *Avoir des tables cohérentes et organisées pour chaque thème (parc, éolienne, permis, parcelle) sans redondance d'information*
- *pour un parc éolien défini, calculer la puissance nominale de l'ensemble des éoliennes qui le compose*
- *Pour une parcelle donnée trouver les références du permis de construire du parc correspondant*

On peut considérer que la plus simple expression d'une base de donnée est une base composée d'une seule table

Secrétariat général

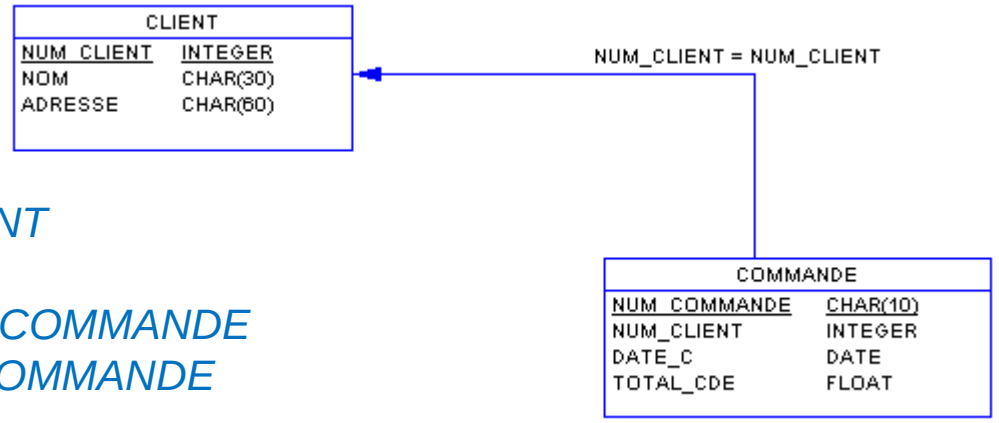
Environnements SQL

Qu'est-ce qu'un SGBD : clé primaire

Dans les modèles de données relationnels, la gestion des relations (non géographique) se fait principalement à l'aide de la notion de **clé primaire/clé étrangère**. Une clé primaire est un attribut dont la valeur est **unique pour chaque enregistrement** de la table (par exemple un numéro en séquence, auto-incrémenté). On parle de clé étrangère lors de l'utilisation d'une primaire d'une table dans une autre table (relation)

Cela permet d'identifier de façon unique chaque enregistrement de la table.

Une clé primaire doit toujours avoir une valeur renseignée (pas de valeur NULL ou vide).



NUM_CLIENT est la clé primaire de la table CLIENT

NUM_COMMANDE est la clé primaire de la table COMMANDE
NUM_CLIENT est une clé étrangère de la table COMMANDE

Secrétariat général

Environnements SQL

Des bases de données accessibles depuis différents clients

Différents clients (applications) permettent d'accéder à une base données mais ils n'offrent pas tous les mêmes possibilités

LibreOffice : clients permettant de visualiser, d'extraire et de saisir de la donnée (formulaire)

PgAdmin : client de postgresql permettant de :

- Gérer et administrer le serveur
- visualiser, extraire et saisir de la donnée

QGIS : client permettant de visualiser, extraire, saisir de la donnée et faire quelques manipulations de gestion des objets d'une base de donnée existante

Exploite notamment

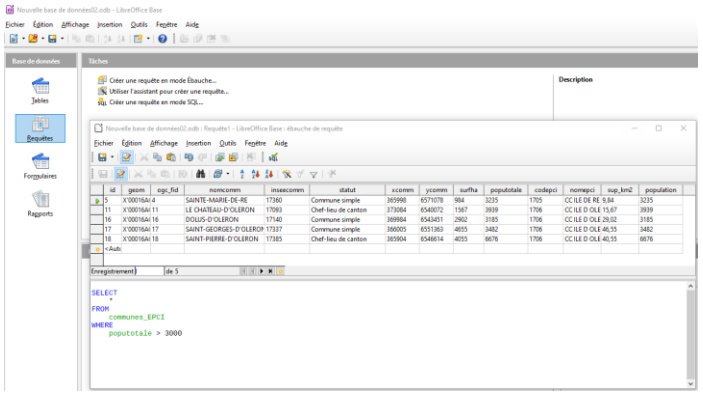
Secrétariat général

Environnements SQL

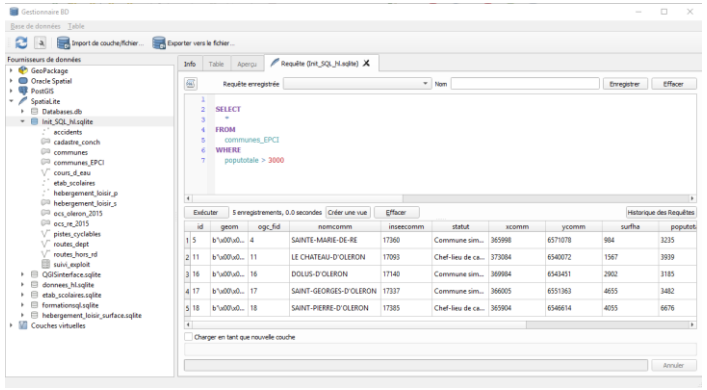
Un moteur de données accessibles depuis différents clients

Des clients pour exécuter du SQL (exemples)

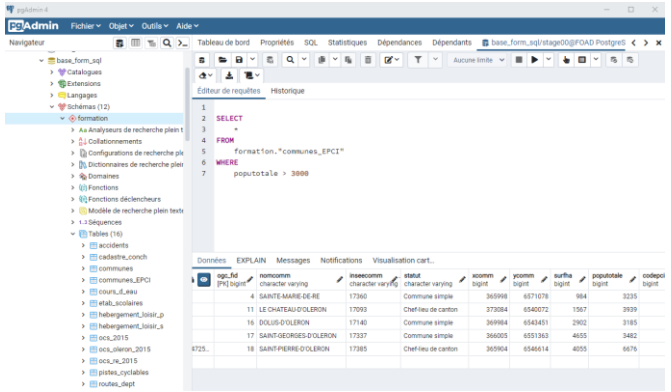
LibreOffice Base



PgAdmin



QGIS - dbmanager



qui exploitent notamment des bases de données

- PostgreSQL
- SQLite
- ..?.. SQLite

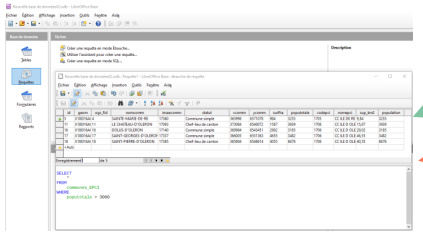
Secrétariat général

Environnements SQL

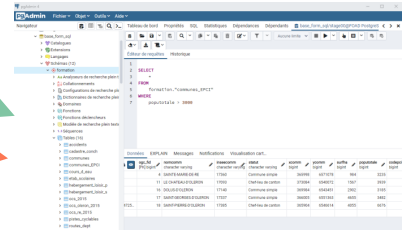
Un moteur de données accessibles depuis différents clients

Si la syntaxe SQL est structurée de la même façon, il existe des différences dans les fonctions suivant le type de base de donnée utilisée

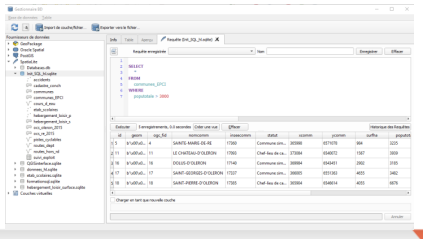
LibreOffice Base



QGIS - dbmanager



PgAdmin



Exemple pour concaténer des valeurs la fonction **||** fonctionne dans les 2 cas et **concat()** ne fonctionne qu'avec postgreSQL

Secrétariat général

Organisation du serveur

Principes généraux d'administration d'un serveur PostgreSQL

Utilisateur

Réseau

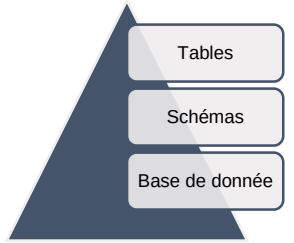
Administrateur réseau

Serveur PostgreSQL

Administrateur serveur

Bases de données

Administrateur base de donné



Secrétariat général

ENVIRONNEMENTS SQL

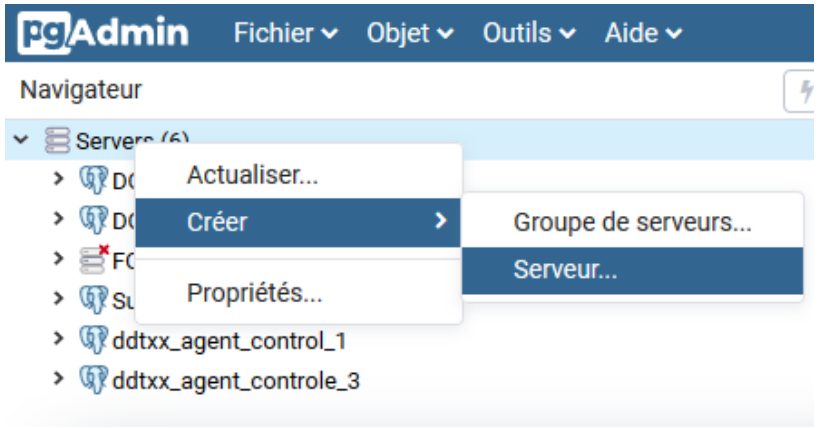
Connexion à une base PostgreSQL avec Pgadmin et zones de rédaction du SQL

Secrétariat général

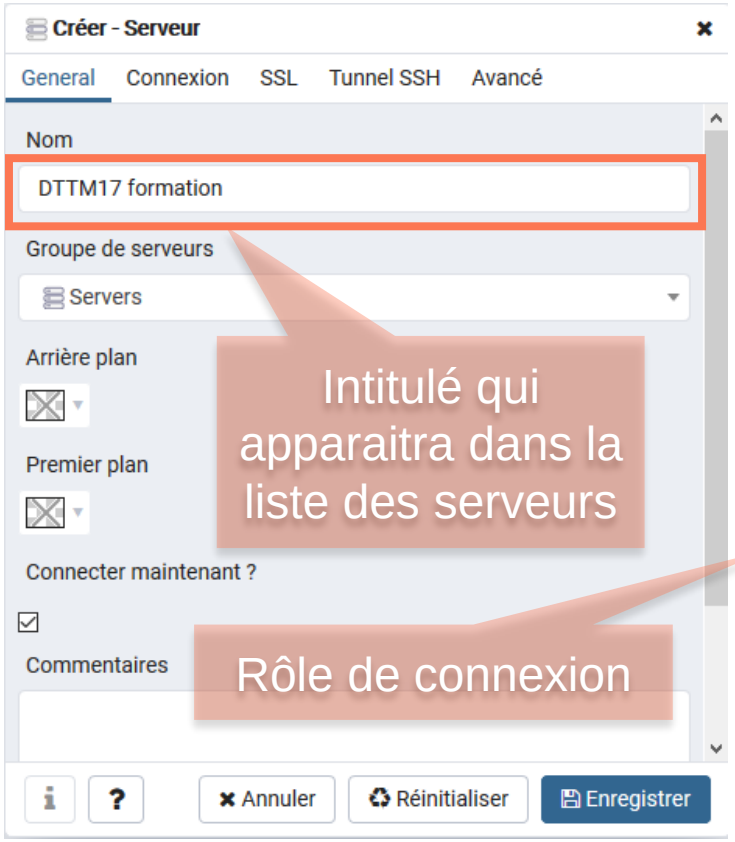
Exploitation des bases PostgreSQL

PgAdmin4 : se connecter à une base

Lancer PgAdmin4



Pour la formation nous utiliserons PgAdmin4 serveur

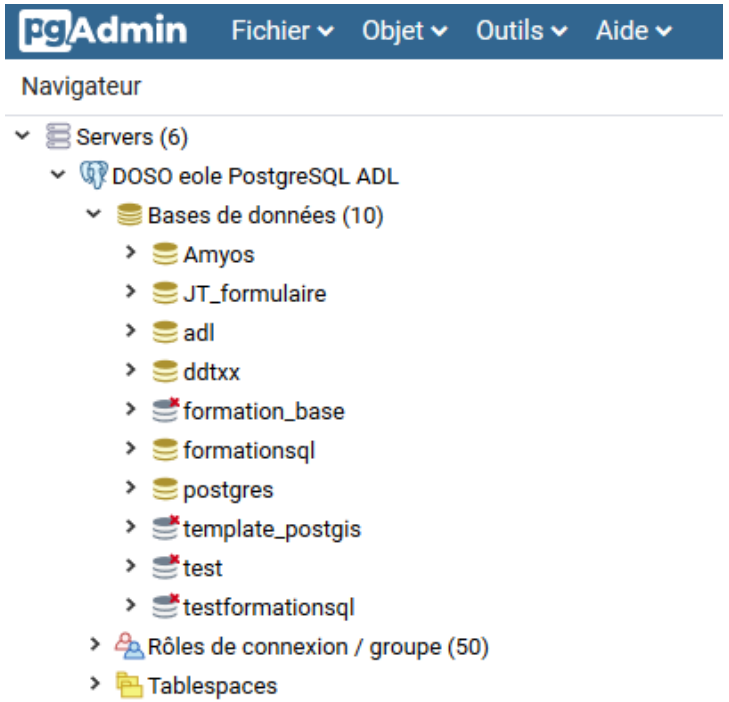


Secrétariat général

Exploitation des bases PostgreSQL

PgAdmin4 : se connecter à une base

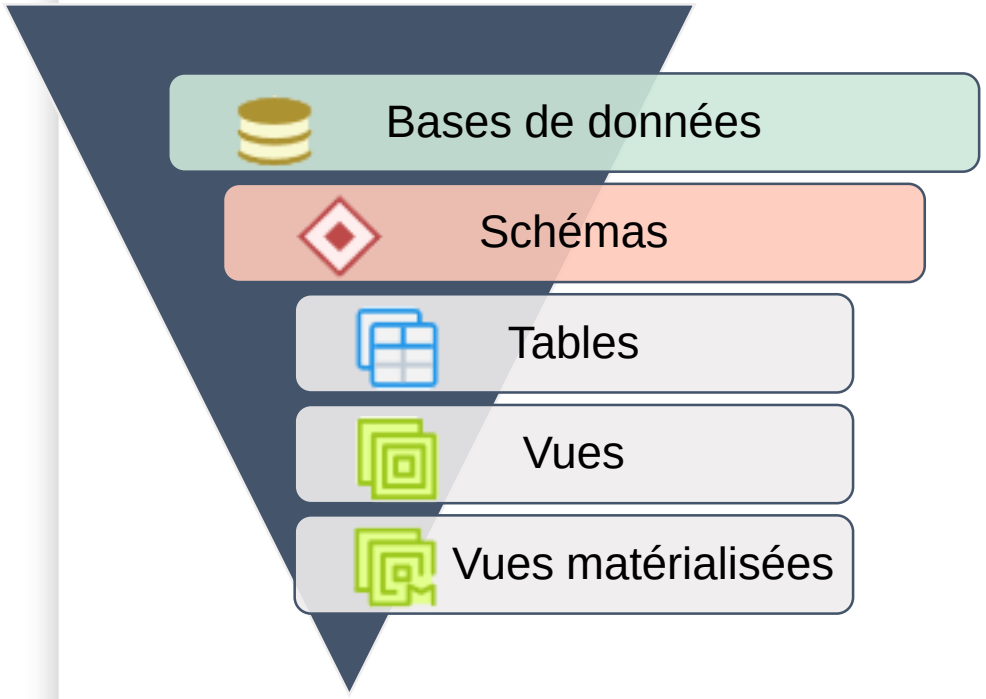
La connexion au serveur permet d'accéder aux bases de données de ce serveur

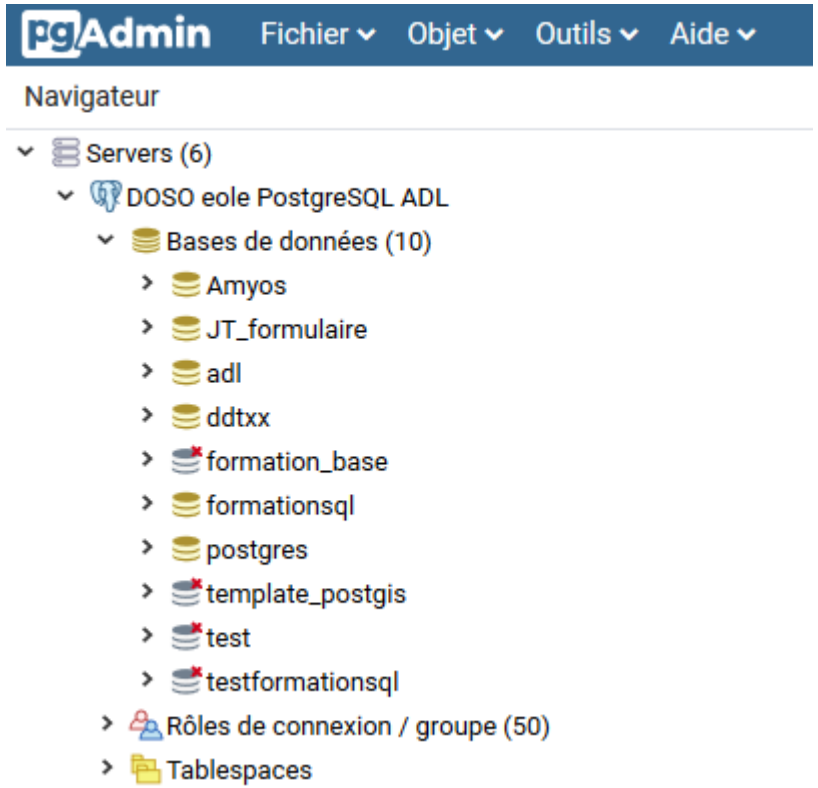


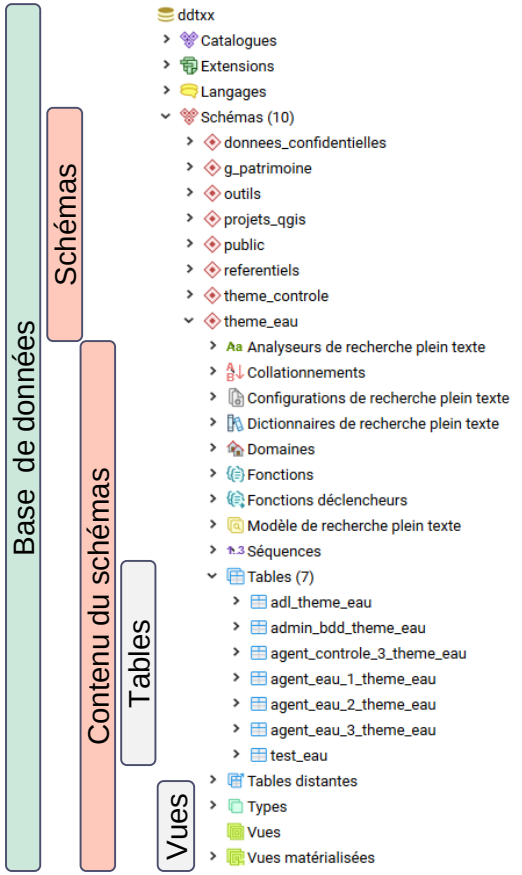
Secrétariat général

Organisation des données

Les différents niveaux d'organisation dans une base de donnée







Secrétariat général

Exploitation des bases PostgreSQL

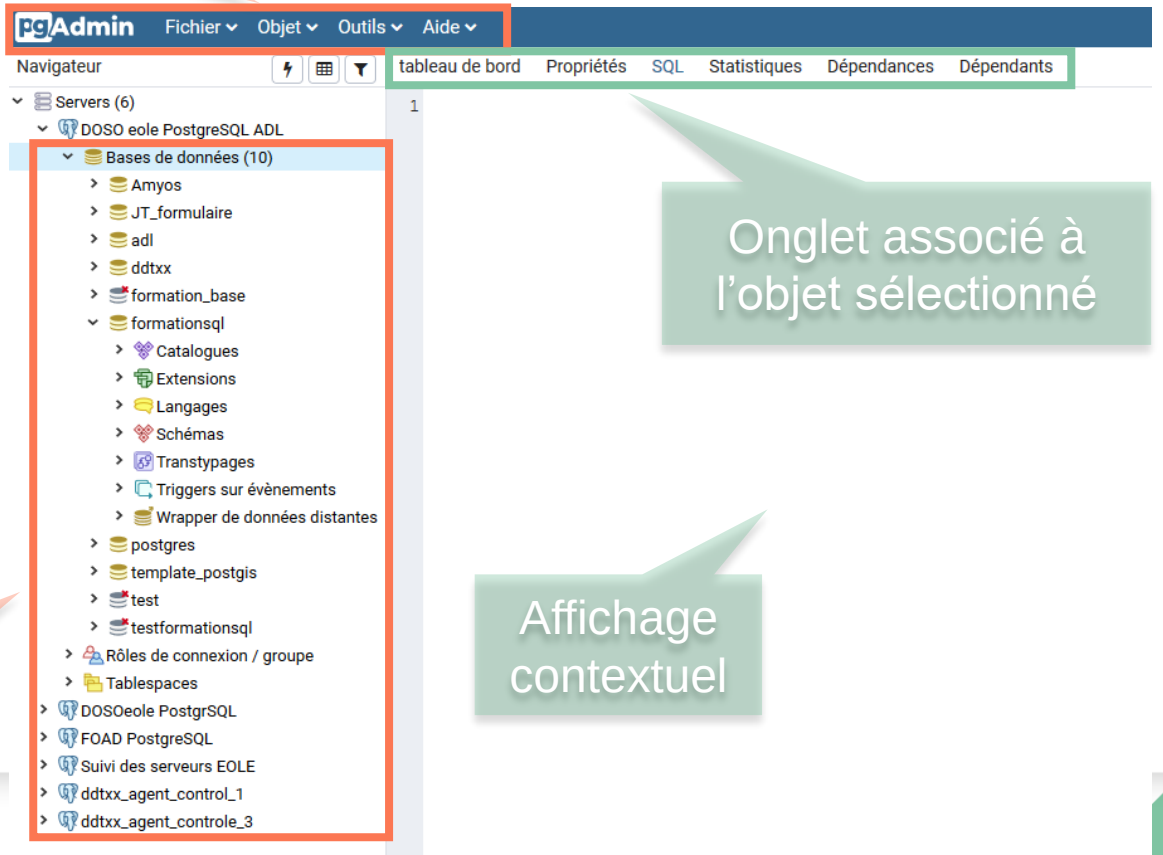
PgAdmin4 : interface de PgAdmin4

ou

Editeur de requête

Afficher les données

Menus PgAdmin



Onglet associé à l'objet sélectionné

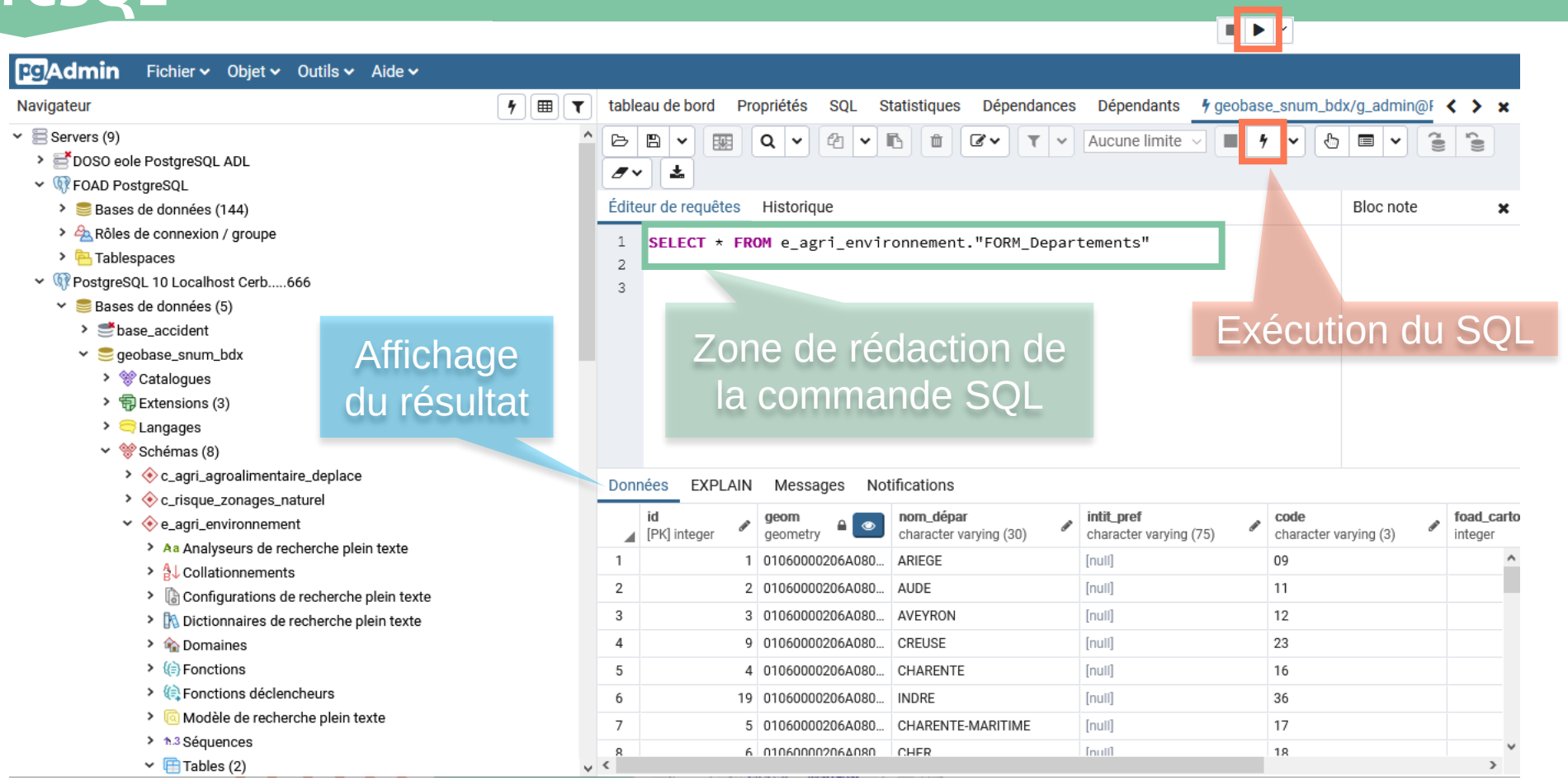
Objets du serveur

Affichage contextuel

Secrétariat général

Exploitation des bases PostgreSQL

PgAdmin4 : interface de PgAdmin4



Secrétariat général

Exercice 1

Se connecter au serveur de la formation

- Depuis votre navigateur internet, se connecter avec PgAdmin4 serveur :
  - Adresse : <adresse IP du serveur>/pgadmin4/browser
  - Utilisateur : <nom utilisateur fourni par le formateur>
  - Mot de passe : <mot de passe fourni par le formateur>
- Explorer les différents éléments dans la base de donnée **« base_form_sql »**

Secrétariat général

ENVIRONNEMENTS SQL

Connexion à une base PostgreSQL avec QGIS et DBmanager et zones de rédaction du SQL

Secrétariat général

Exploitation des bases PostgreSQL

QGIS : se connecter à une base

Dans QGIS, 2 possibilités dont une indispensable pour accéder au contenu d'une base de donnée :

1. Depuis le connecteur QGIS  (en 2.16) ou  (en 3.x)
   Il permet de définir les paramètres de connexion à **UNE** base PostgreSQL. Cette étape indispensable, permet :
   - ✓ D'accéder au contenu d'une base de donnée pour charger des tables ou vues
   - ✓ De configurer cette connexion pour l'explorateur et DbManager
2. Depuis DbManager, lorsque la connexion a été préalablement configurée

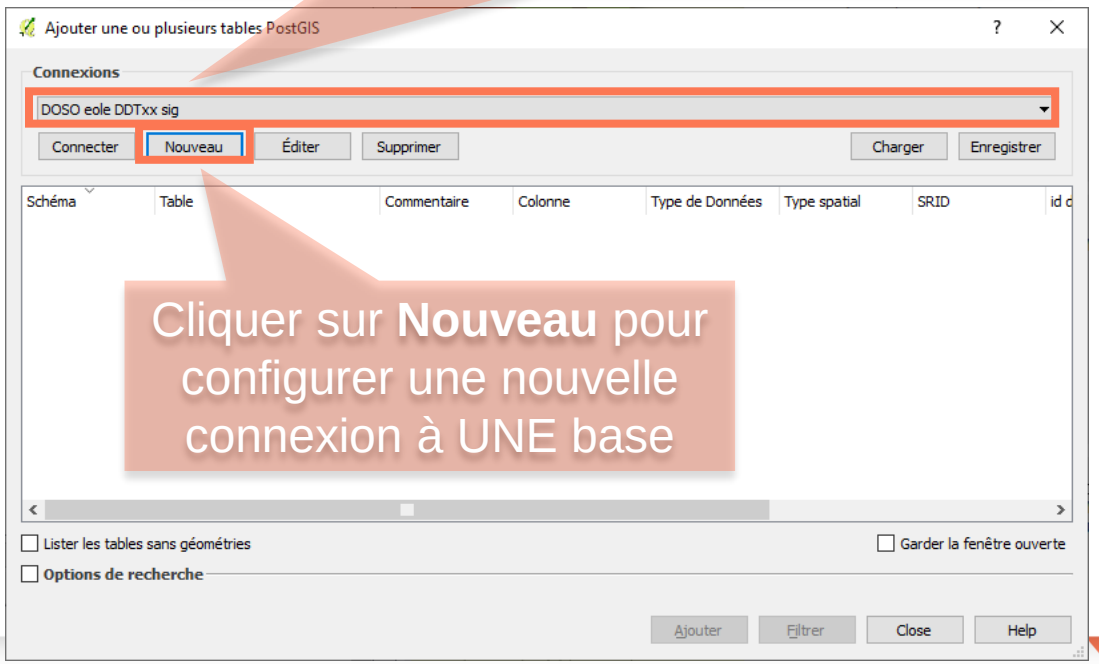
Secrétariat général

Exploitation des bases PostgreSQL

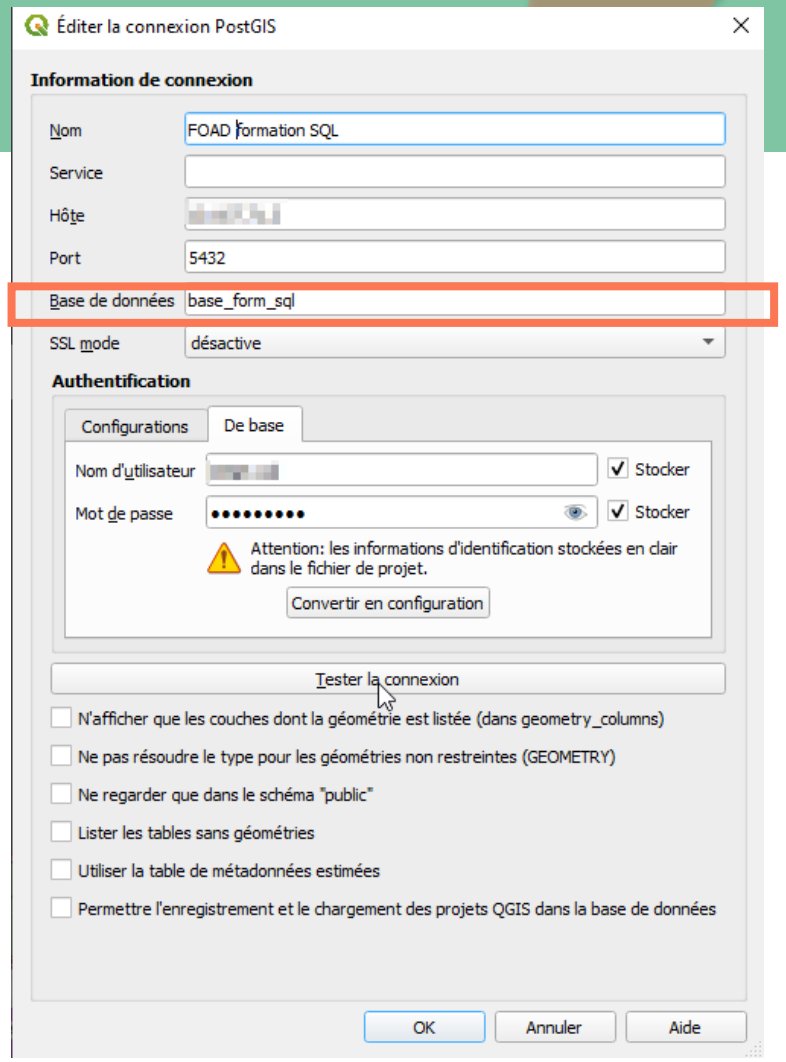
Configuration du connecteur

A la différence de PgAdmin4 on indique le nom de la base de donnée

Sélectionner une connexion existante, sinon configurer une nouvelle



Cliquer sur **Nouveau** pour configurer une nouvelle connexion à UNE base



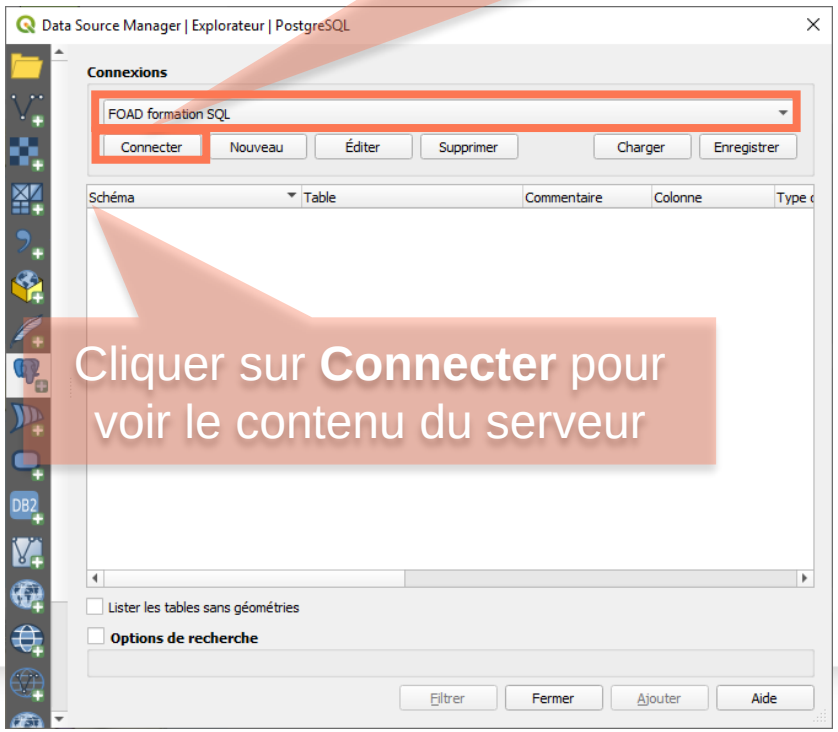
Secrétariat général

Exploitation des bases PostgreSQL

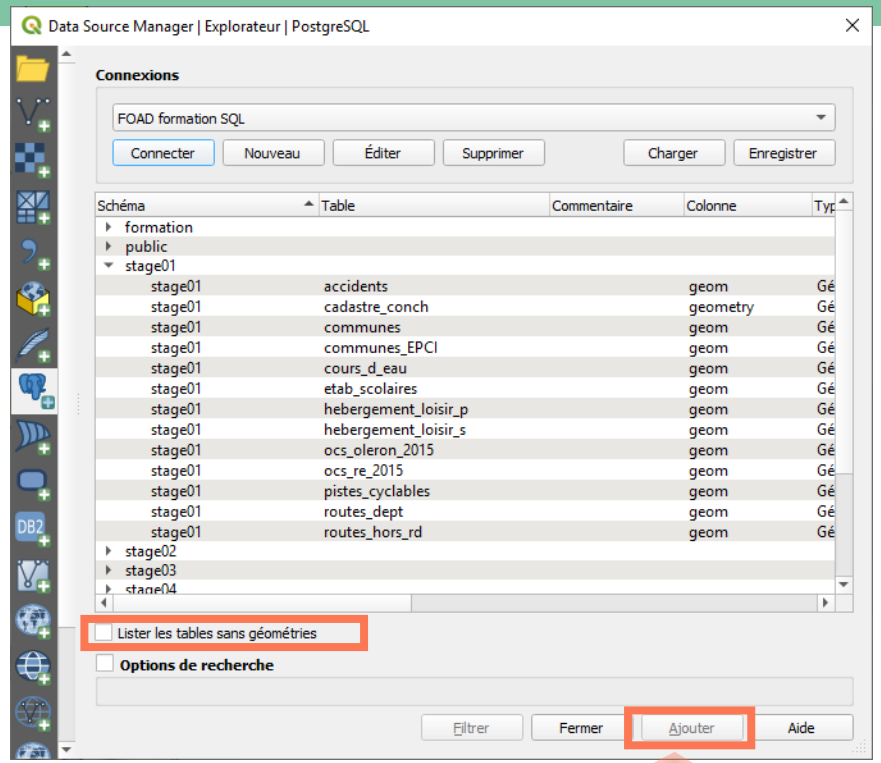
QGIS : se connecter à une base

Configuration du connecteur

Sélectionner une connexion



Cliquer sur **Connecter** pour voir le contenu du serveur



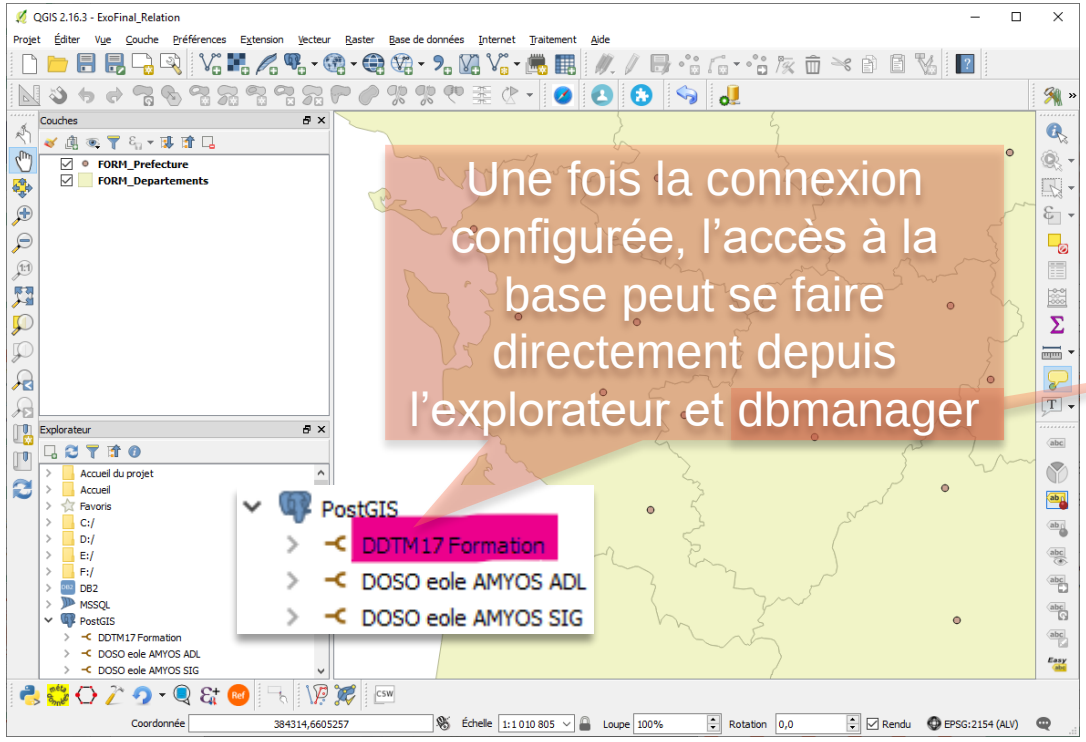
Cliquer sur **Ajouter** pour afficher les tables sélectionnées dans QGIS

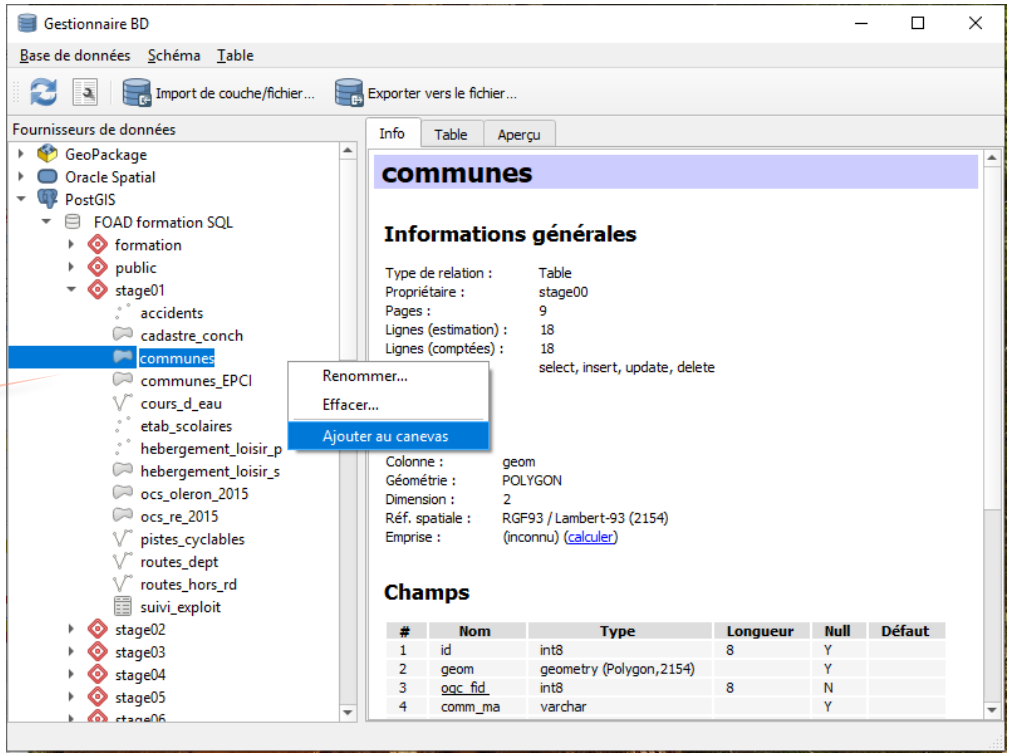
Secrétariat général

Exploitation des bases PostgreSQL

QGIS : se connecter à une base

Configuration du connecteur

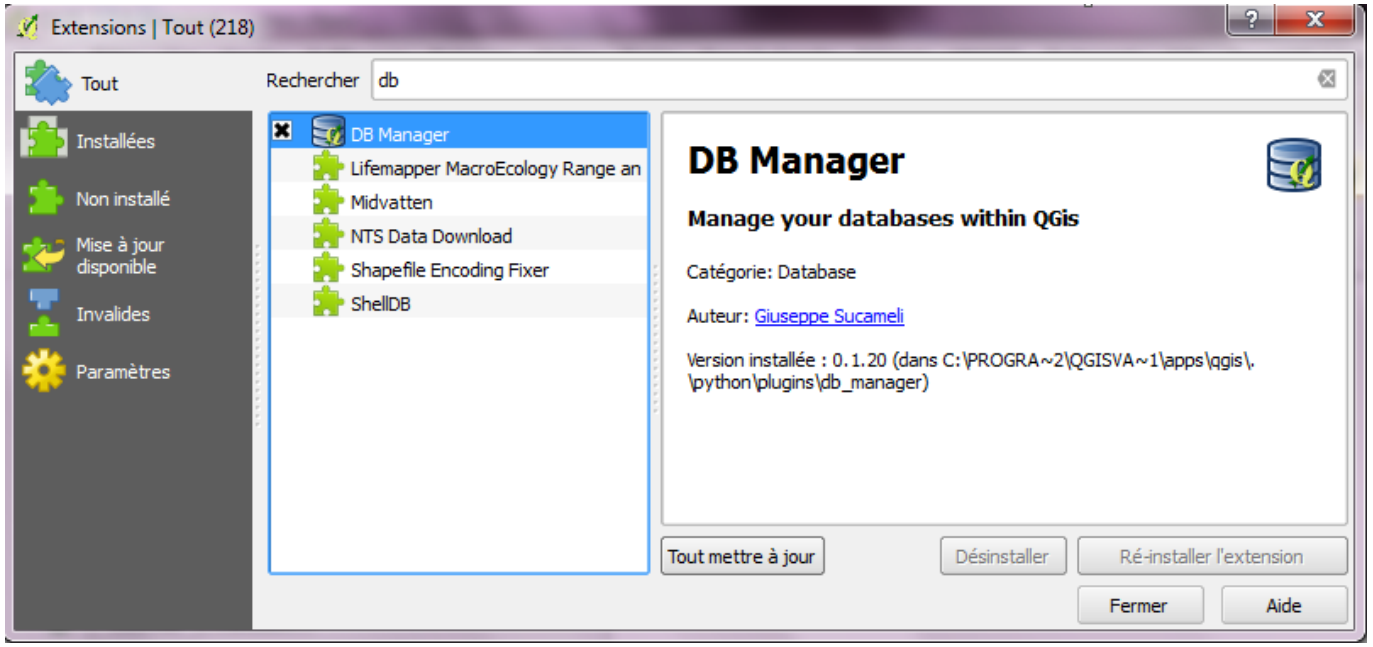


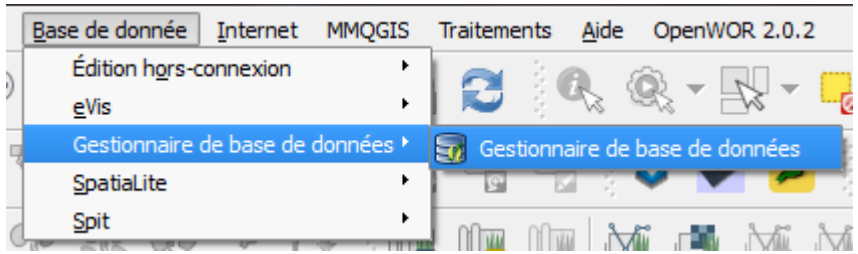


Secrétariat général

Exploitation des bases PostgreSQL

QGIS : DBManager





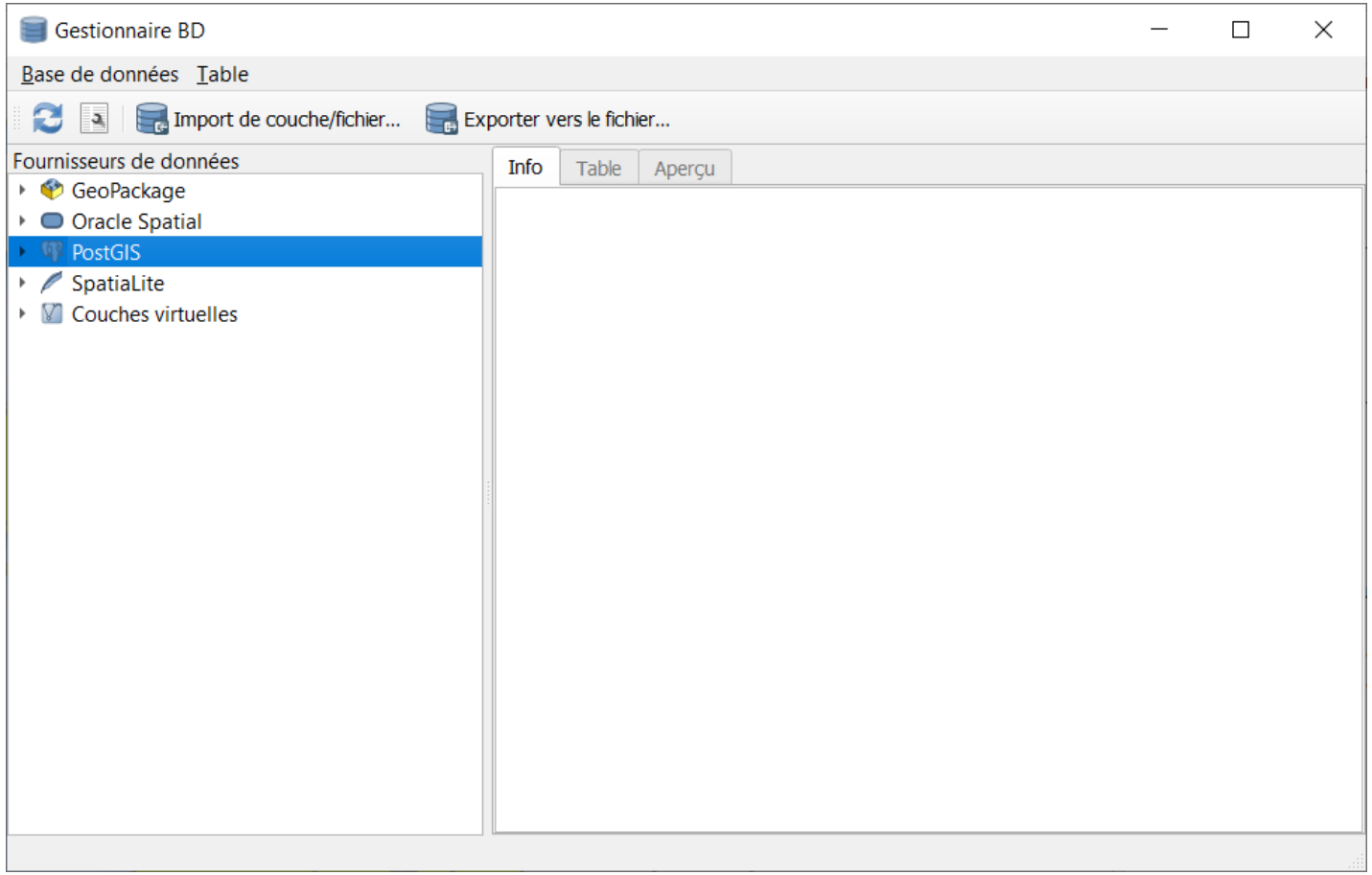
Secrétariat général

Exploitation des bases PostgreSQL

QGIS : DBManager

Permet de travailler avec :

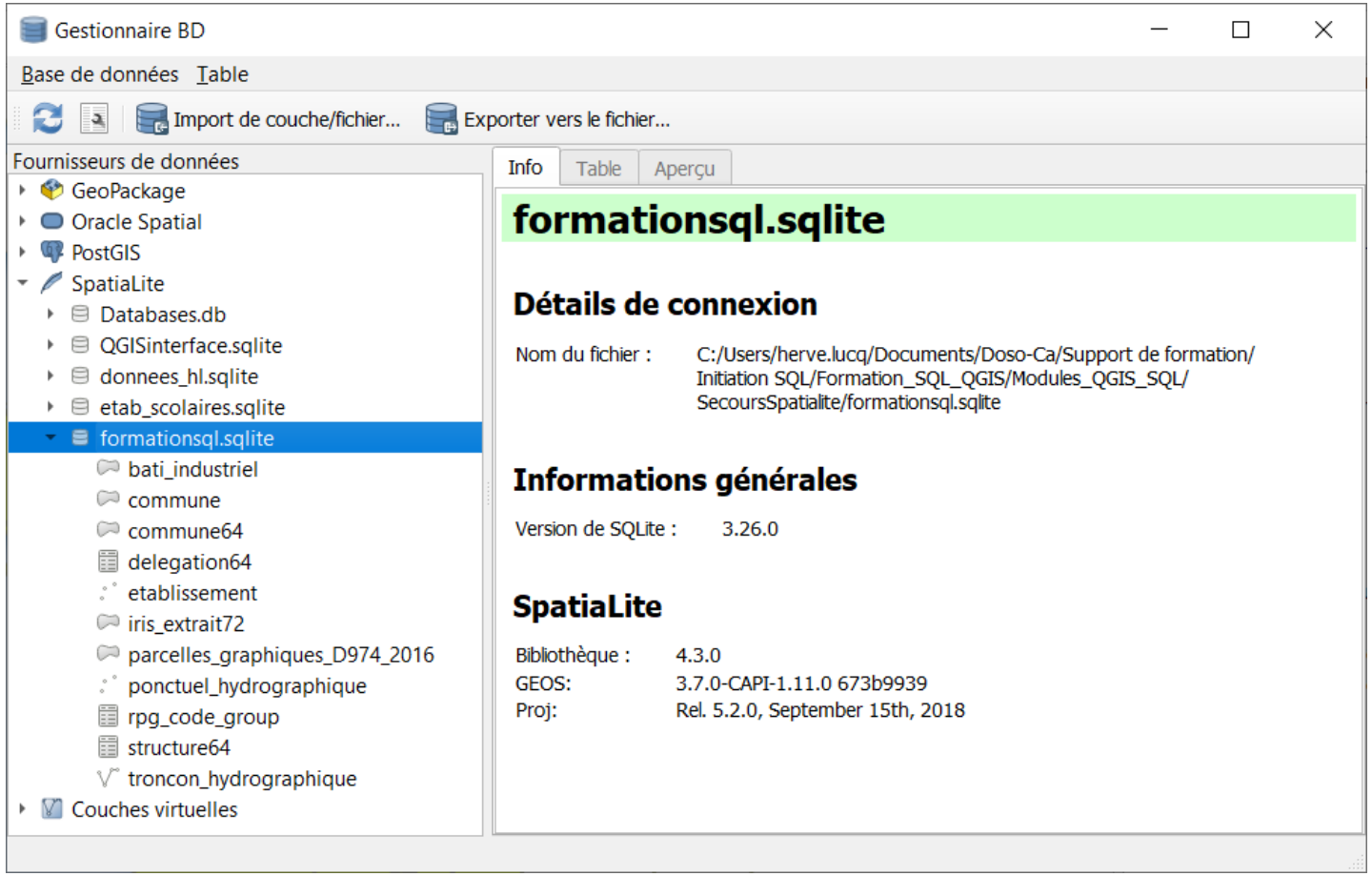
- Postgis
- Spatialite
- ….



Secrétariat général

Exploitation des bases PostgreSQL

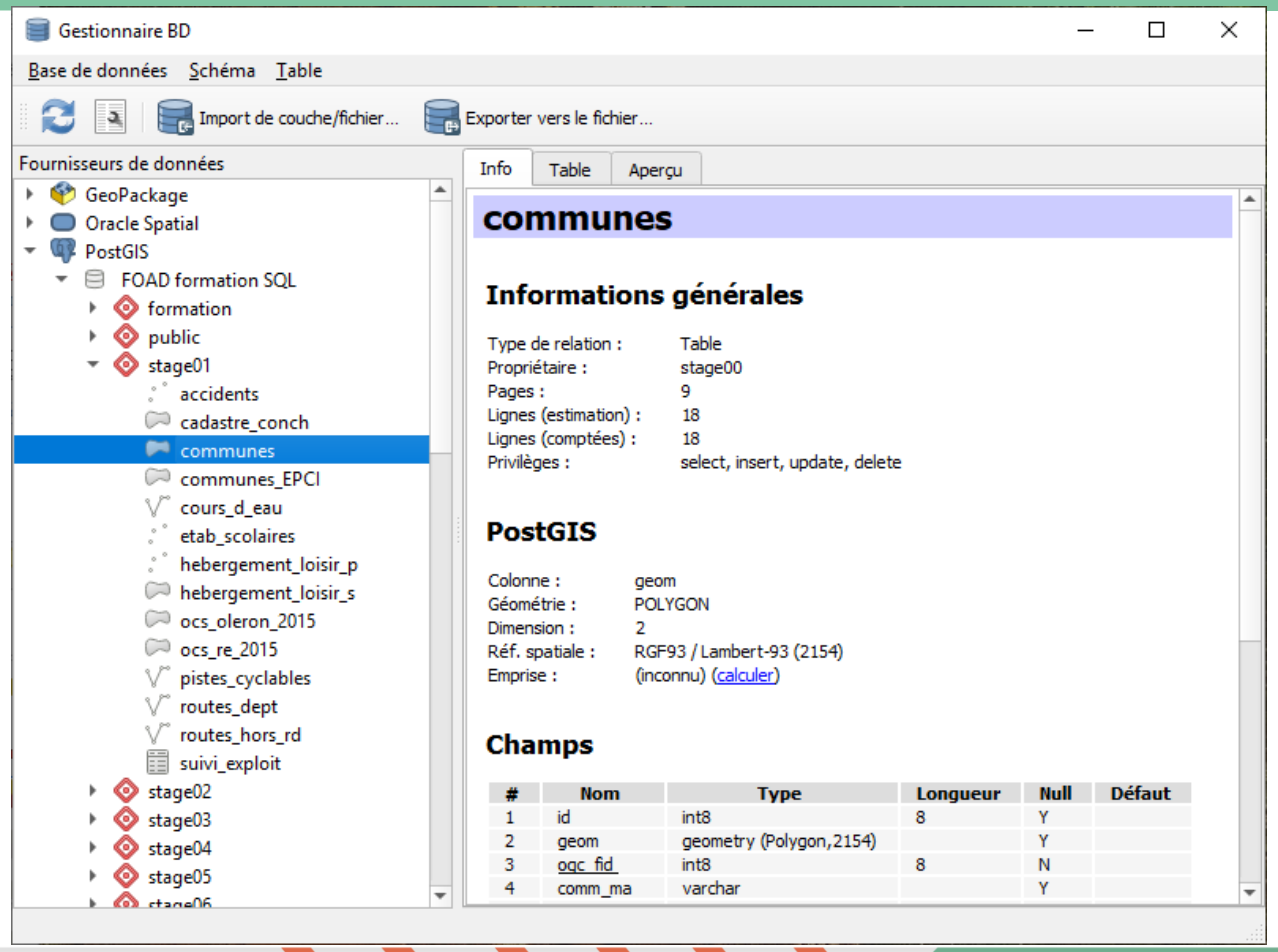
QGIS : DBManager



Secrétariat général

Exploitation des bases PostgreSQL

QGIS : DBManager



Secrétariat général

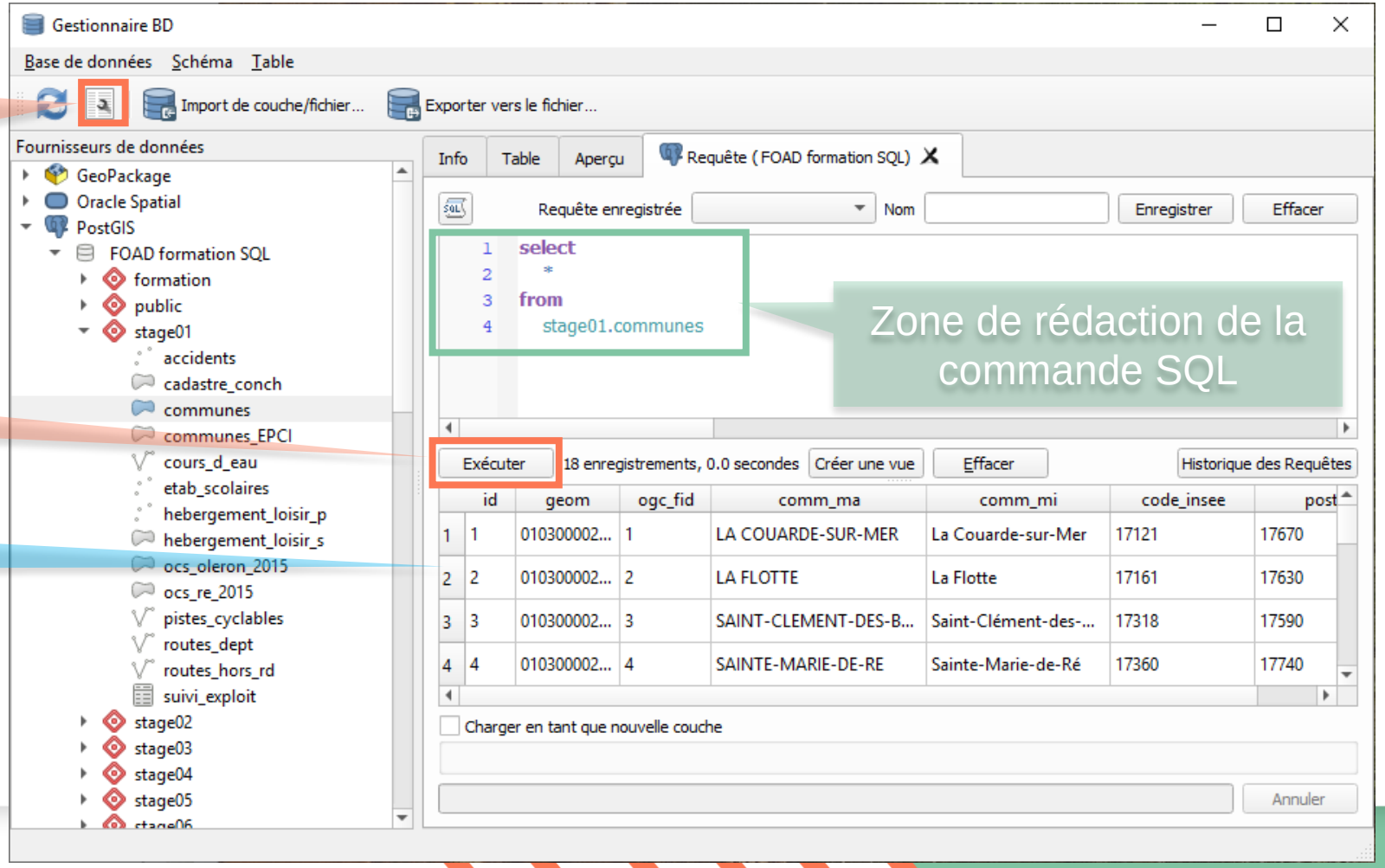
Exploitation des bases PostgreSQL

QGIS : DBManager

Lancement de l'interface SQL (attention à l'objet sélectionné)

Exécution de la commande SQL

Affichage du résultat



Secrétariat général

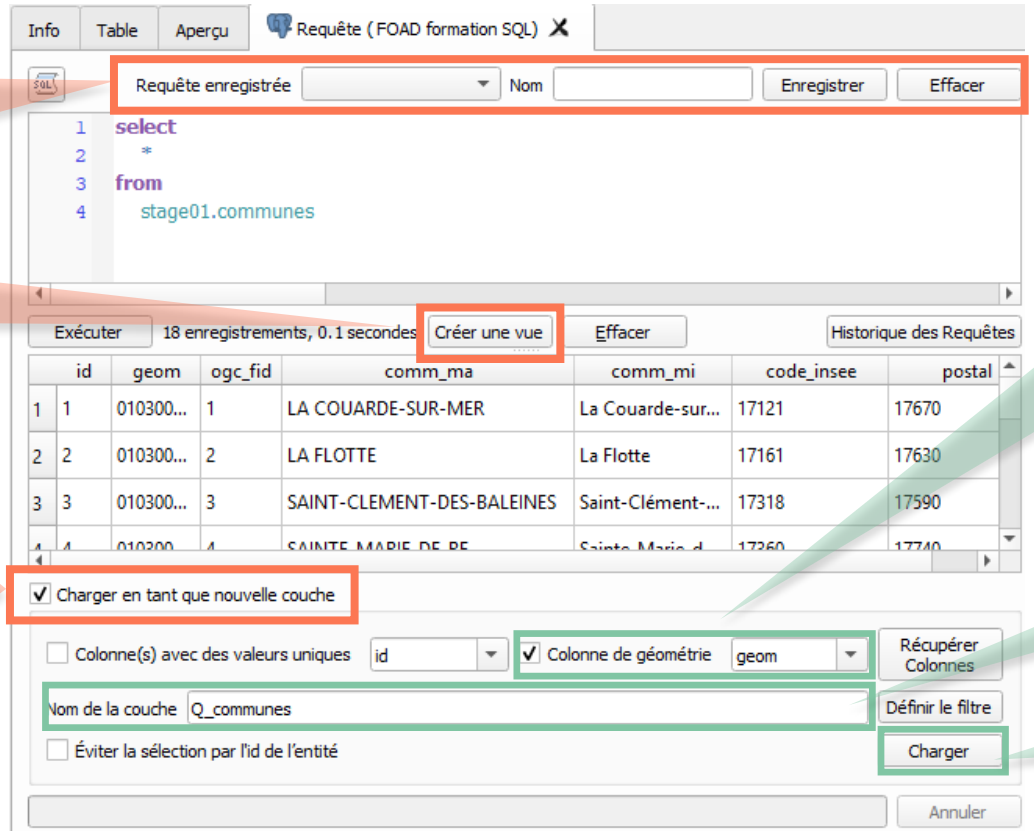
Exploitation des bases PostgreSQL

QGIS : DBManager

Enregistrer la requête pour la rappeler ultérieurement

Crée une vue dans la base de données

Cocher la case et renseigner les paramètres pour afficher la couche dans QGIS



Nom de la colonne qui contient la géométrie, si le paramètre n'est pas renseigné seul le tableau s'affichera

Intitulé de la couche résultat dans QGIS

Affiche la couche dans QGIS

Secrétariat général

Exercice 2

Se connecter à la base de la formation avec QGIS

- Se connecter avec QGIS à la base « formation » de la formation avec les paramètres suivants :
  - Intitulé du serveur : Libre (exemple : formation PostgreSQL)
  - Adresse : <adresse du serveur>
  - Port : 5432
  - Base : base_form_sql
  - Utilisateur : <fourni par le formateur>
  - Mot de passe : *******
- Explorer les différents éléments dans la base de donnée **« formation »**
- Charger la couche « communes » du schéma qui vous est alloué

Secrétariat
général

QU'EST-CE QUE LE SQL

Secrétariat général

Qu'est-ce que le SQL ? ...

Les sujets abordés :

- Création de table
  - ✓ Attributaire avec PgAdmin
  - ✓ Géographique avec QGIS
- Sélection d'objets d'une table
- Suppression de table
- Modification d'une table :
  - ✓ Ajouter un colonne
  - ✓ Mettre à jours des informations

Secrétariat général

Qu'est-ce que le SQL ?

Définitions et finalités

SQL = Structured Query Language est un langage de requêtes structuré qui est composé de 3 sous-ensembles :

- Langage de Définition de Données (**LDD**) : créer et supprimer des objets dans la base de données
- Langage de Contrôle de Données (**LCD**) : gérer les droits sur les objets
- Langage de Manipulation de Données (**LMD**) : pour la recherche, l'insertion, la mise à jour et la suppression de données

Il est utilisé dans PostgreSQL et dans QGIS

Secrétariat général

Qu'est-ce que le SQL ?

Définitions et finalités

LDD

- CREATE (créer)
- ALTER (modifier)
- RENAME (renommer)
- DROP (supprimer)

LCD

- GRANT (attribuer des droits)
- REVOKE (révoquer des droits)

LMD

- SELECT (extraire des données)
- INSERT (insertion des données)
- UPDATE (modifier des données)
- DELETE (supprimer des données)
- COPY (import-export de données)

LCT

- COMMIT (validation des transactions)
- ROLLBACK (annulation des transactions)

Secrétariat général

Qu'est-ce que le SQL ?

Points d'attention

Suivant la base de données utilisée (PostgreSQL, SQLite,…) la syntaxe varie pour appeler les objets (tables)

Définition d'une table :

- [Emplacement] *(schéma si base postrgreSQL)*
- Intitulé

Définition d'un champ:

- [Table] *(utilisation de plusieurs tables)*
- Intitulé
- Définition des attributs

Exemple PostgreSQL:

- il faut préfixer : **nom_du_schema.nom_de_la_table** (si non spécifié : schéma par défaut donc le schéma public)
- Il faut mettre les intitulés des objets entre « » s'ils contiennent des majuscules, espaces, commence par un chiffre,…

Exemples syntaxes :

PostgreSQL : *SELECT nom, "Population" FROM formation."communes_EPCI"*
SQLite : *SELECT nom, Population FROM communes_EPCI*

Plusieurs tables : *SELECT communes.nom, communes.Population, etablissement.intitule FROM communes, etablissement WHERE ...*

Secrétariat général

SYNTAXE SQL

- La sélection d'objet : SELECT
  - ✓ SELECT
  - ✓ Les alias
  - ✓ DISTINCT
  - ✓ WHERE
  - ✓ Opérateurs de comparaisons
  - ✓ Le transtypage
  - ✓ Les fonctions numériques
  - ✓ Les fonctions chaines de caractères
  - ✓ Les fonctions dates
  - ✓ La fonction conditionnelle CASE WHEN
  - ✓ ORDER BY
  - ✓ LIMIT
  - ✓ GROUP BY

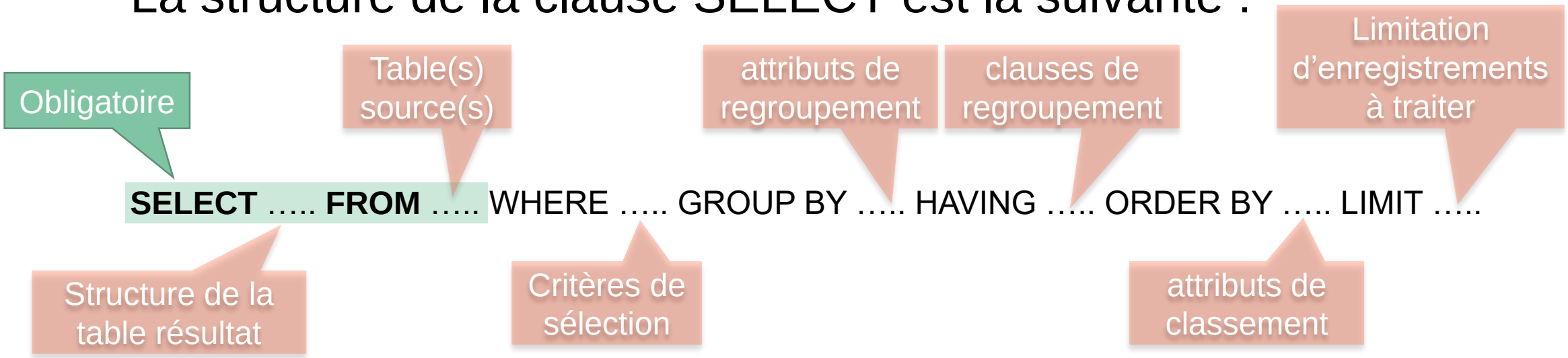
Secrétariat général

Syntaxe SQL

L'instruction SELECT

La clause SELECT sert à sélectionner des enregistrements à partir de certains critères

La structure de la clause SELECT est la suivante :



*Exemple : SELECT * FROM communes WHERE population > 1000*
Sélectionne tous les enregistrements de la table commune qui ont une valeur dans l'attribut population supérieur à 1000

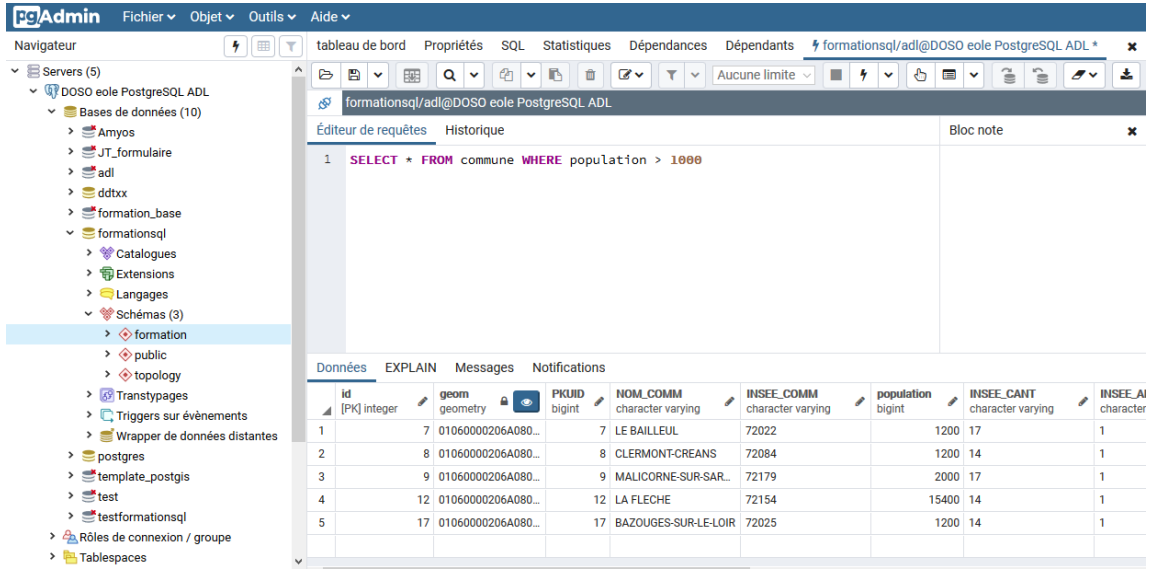
Secrétariat général

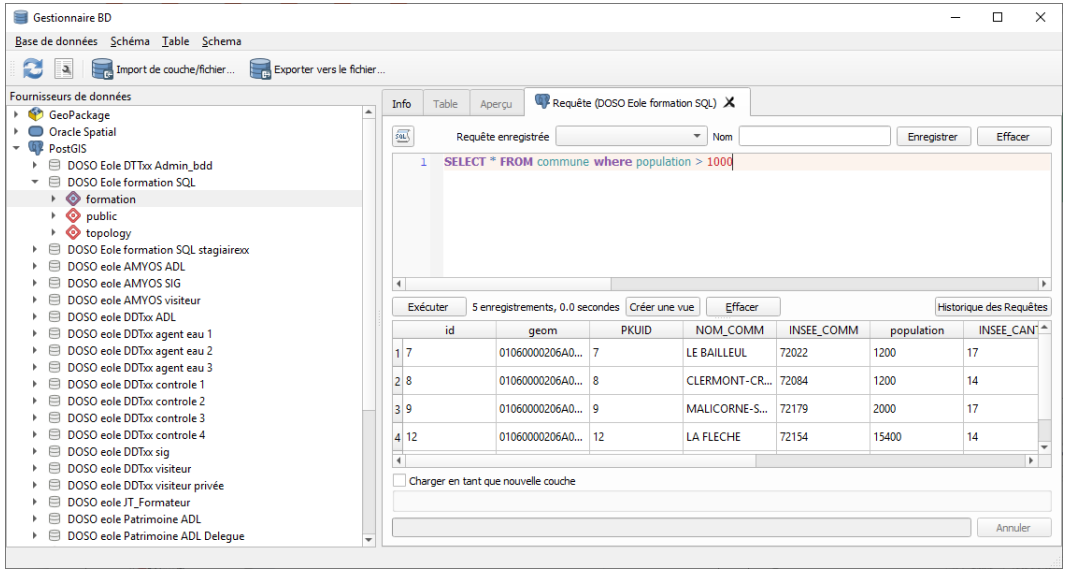
Syntaxe SQL

L'instruction SELECT

2 possibilités pour rédiger les requêtes :

- Dans Pgadmin4 (à privilégier)
- Dans Gestionnaire DB de QGIS





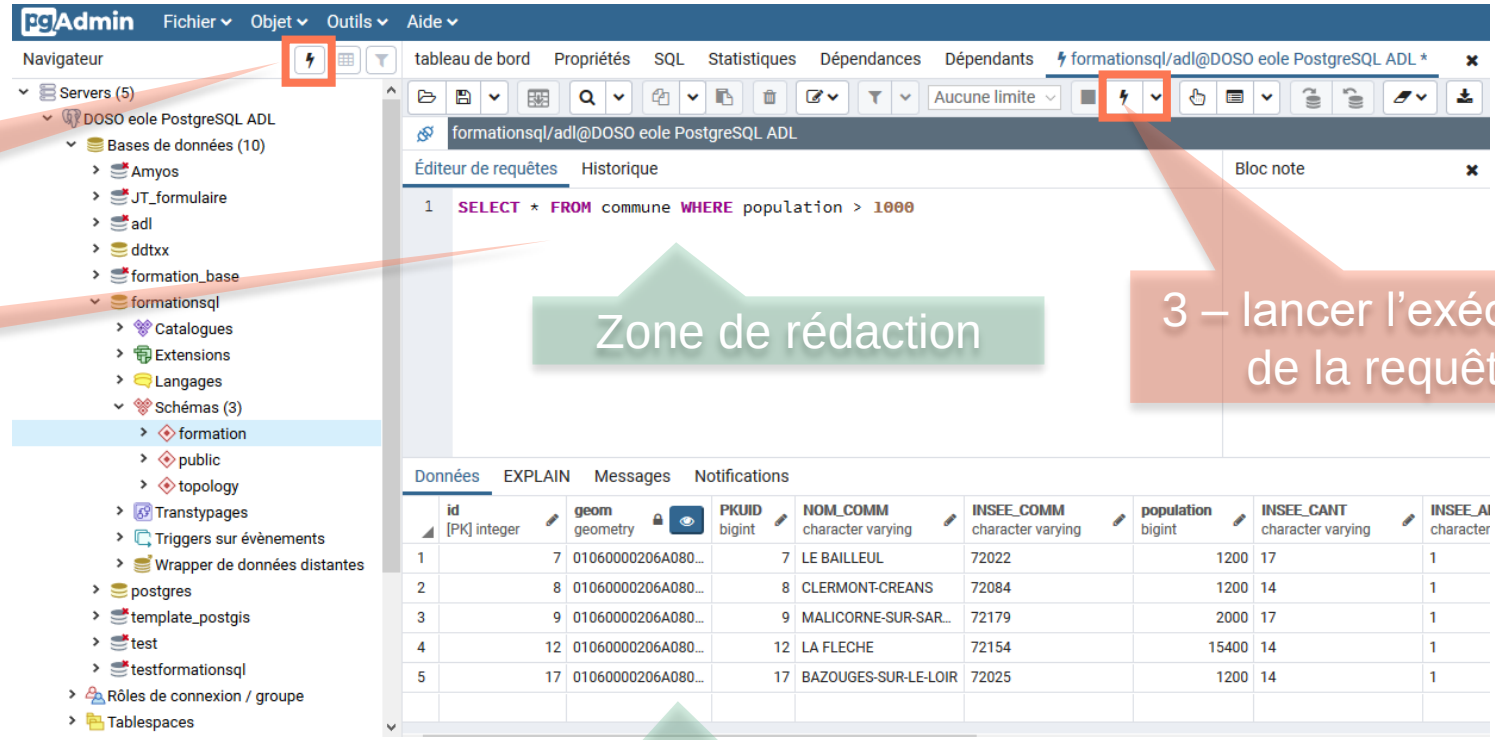
Secrétariat général

Syntaxe SQL

L'instruction SELECT : PgAdmin4

1 – Cliquer l'icone « éditeur de requêtes »

2 – Rédiger la requête



Zone de rédaction

3 – lancer l'exécution de la requête

Zone de résultat

Secrétariat général

Exploitation des bases PostgreSQL

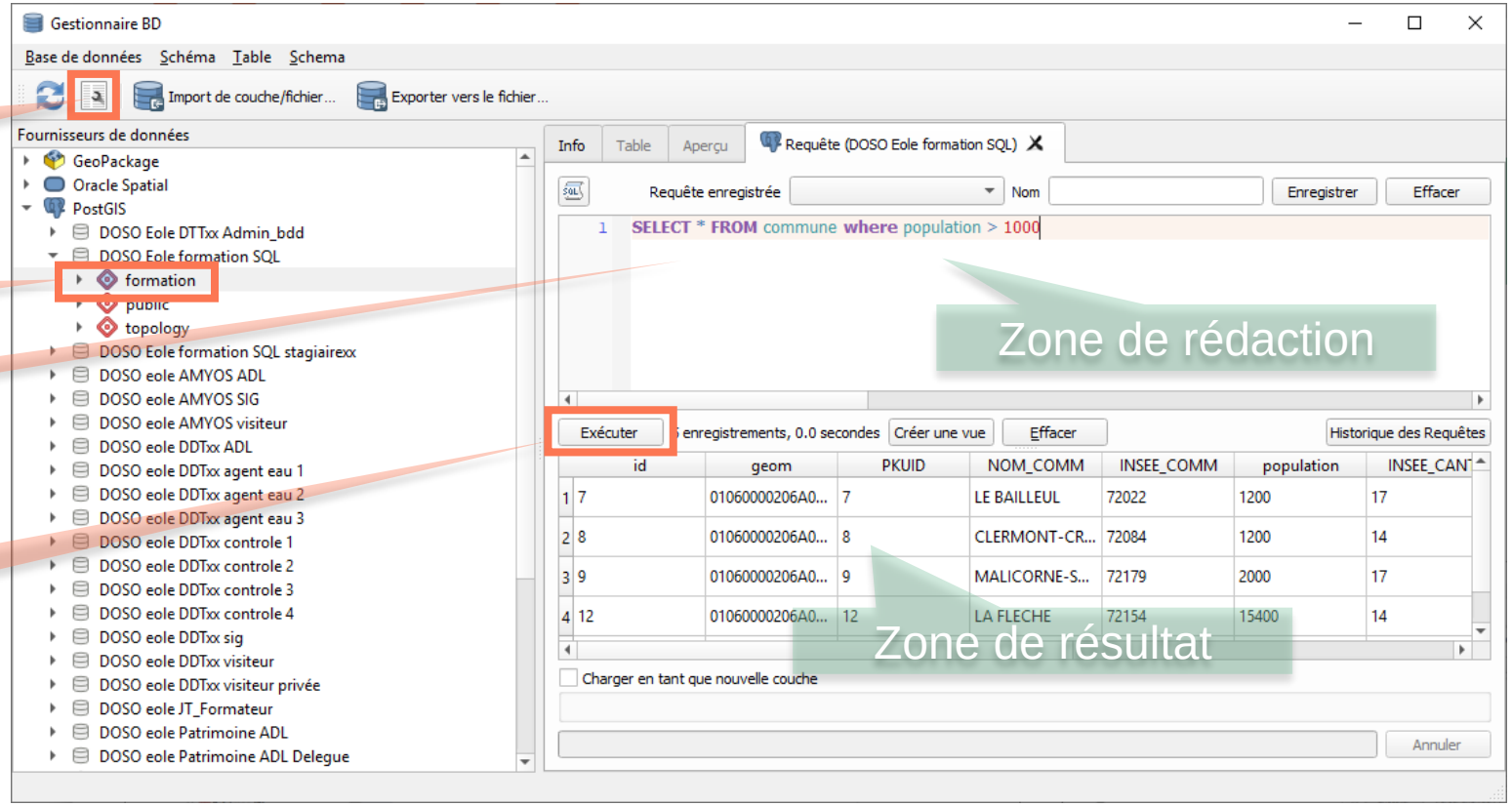
L'instruction SELECT : Gestionnaire BD de QGIS

2 – Cliquer l'icone « éditeur de requêtes »

1 – Sélectionner le schéma

3 – Rédiger la requête

4 – lancer l'exécution de la requête



Secrétariat général

Exploitation des bases PostgreSQL

L'instruction SELECT : Principe de rédaction

balise de **début** du bloc de commentaires

balise de **fin** du bloc de commentaires

```
/* ceci est un
Commentaire sur plusieurs lignes */
SELECT *
FROM formation.communes ;
```

Instruction **1** SQL

FIN de l'instruction **1** SQL

balise de **début** de la ligne de commentaires

```
-- commentaire sur une ligne
SELECT *
FROM formation.communes LIMIT 3 ;
```

Instruction 2 SQL

FIN de l'instruction **2** SQL

Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

SELECT (liste des attributs) **FROM** (liste des tables) **WHERE** (Conditions)

Points d'attention sur l'appel des objets (tables et attributs) :

- L'appel d'une table se fait en **indiquant son emplacement** (schéma). S'il n'est pas indiqué de schéma postgreSQL considère que la table est dans le schéma par défaut (public)
  *Exemple : SELECT * FROM **formation.**commune*

- PostgreSQL est sensible à la casse (**c**ommune ≠ **C**ommune)

- Si le nom de schéma, de table ou d'attribut contient une majuscule ou commence par un chiffre, il faut indiquer ce dernier entre des doubles quôtes :
  *Exemple : SELECT **"**INSEE_com**"**, nom, geom FROM formation.**"**Commune**"***

Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

SELECT (liste des attributs) **FROM** (liste des tables) **WHERE** (Conditions)

- **SELECT** : indique le sous-ensemble des attributs (colonnes) devant apparaître dans la réponse (SELECT * : sélectionne tous les champs)
- **FROM** : décrit les tables (au moins une) utilisées dans la requête
- **WHERE** : exprime les conditions (optionnel)

Exemple :
*SELECT ***** FROM communes WHERE population > 1000*
Sélectionne tous les enregistrements de la table commune qui ont une valeur dans l'attribut population supérieur à 1000

*SELECT **nom_com, insee_com, geom** FROM communes WHERE population > 1000*
Sélectionne tous les enregistrements de la table commune qui ont une valeur dans l'attribut population supérieur à 1000 et restitue le résultat avec une structure de table avec les attributs nom_com, insee_com, geom

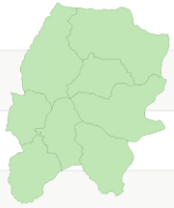
Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

SELECT nom_comm, insee_comm **FROM** travail.communes

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

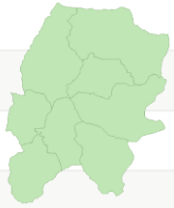
Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

SELECT nom_comm, insee_comm **FROM** travail.communes

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

nom_com	insee_com
SAINT-PIERRE-DU-PALAIS	17386
LA CLOTTE	17113
BOSCAMNANT	17055
LA GENETOUZE	17173
LA BARDE	17033
SAINT-MARTIN-DE-COUX	17366
LE FOUILLOUX	17167
SAINT-AIGULIN	17309

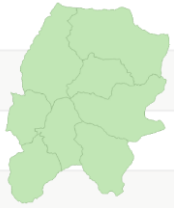
Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

```
SELECT nom_comm, insee_comm, geom
FROM travail.communes
```

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

```
SELECT nom_comm, insee_comm, geom
FROM travail.communes
```

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

nom_com	insee_com	geom
SAINT-PIERRE-DU-PALAIS	17386	
LA CLOTTE	17113	
BOSCAMNANT	17055	
LA GENETOUZE	17173	
LA BARDE	17033	
SAINT-MARTIN-DE-COUX	17366	
LE FOUILLOUX	17167	
SAINT-AIGULIN	17309	

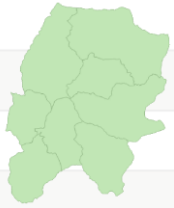
Secrétariat général

Syntaxe SQL

L'instruction SELECT : Rédaction

```
SELECT nom_comm,
        "Pop_0_25" + "Pop_25_260" + "Pop_sup_60"
FROM travail.communes
```

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

Secrétariat général

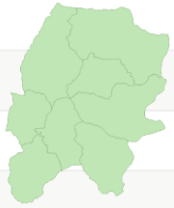
Syntaxe SQL

L'instruction SELECT : Rédaction

```
SELECT nom_comm,
       "Pop_0_25" + "Pop_25_260" + "Pop_sup_60"
FROM travail.communes
```

Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60
SAINT-PIERRE-DU-PALAIS	17386	100	190	90
LA CLOTTE	17113	205	300	190
BOSCAMNANT	17055	45	136	200
LA GENETOUZE	17173	47	94	79
LA BARDE	17033	140	205	128
SAINT-MARTIN-DE-COUX	17366	87	215	140
LE FOUILLOUX	17167	204	323	224
SAINT-AIGULIN	17309	411	676	836

geom


nom_com	Pop_0_25 + Pop_25_60 + Pop_sup_60
SAINT-PIERRE-DU-PALAIS	380
LA CLOTTE	695
BOSCAMNANT	381
LA GENETOUZE	220
LA BARDE	473
SAINT-MARTIN-DE-COUX	442
LE FOUILLOUX	751
SAINT-AIGULIN	1923

Secrétariat général

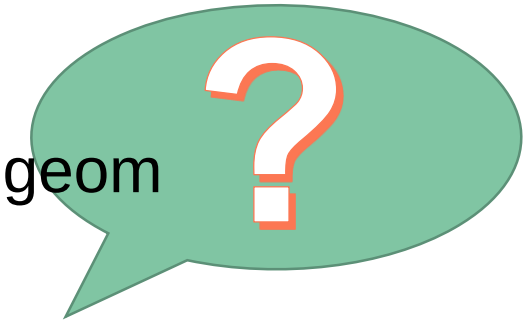
Syntaxe SQL

L'instruction SELECT : Rédaction

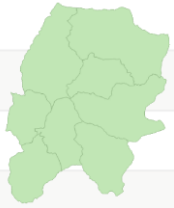
SELECT

nom_comm, “Pop_0_25“ + “Pop_25_260“ + “Pop_sup_60“, geom

FROM travail.communes



Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

Secrétariat général

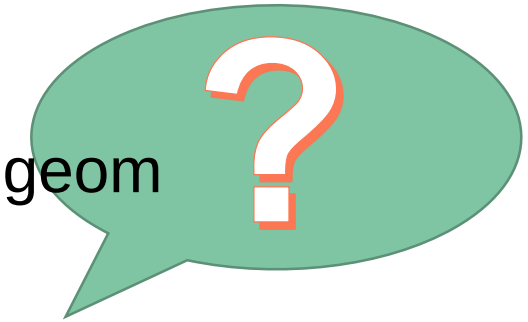
Syntaxe SQL

L'instruction SELECT : Rédaction

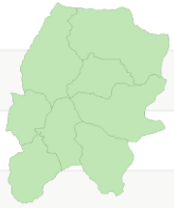
SELECT

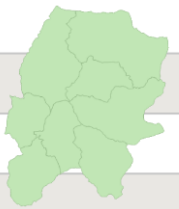
nom_comm, "Pop_0_25" + "Pop_25_260" + "Pop_sup_60", geom

FROM travail.communes



Tables : communes

nom_com	insee_com	Pop_0_25	Pop_25_60	Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	17386	100	190	90	
LA CLOTTE	17113	205	300	190	
BOSCAMNANT	17055	45	136	200	
LA GENETOUZE	17173	47	94	79	
LA BARDE	17033	140	205	128	
SAINT-MARTIN-DE-COUX	17366	87	215	140	
LE FOUILLOUX	17167	204	323	224	
SAINT-AIGULIN	17309	411	676	836	

nom_com	Pop_0_25 + Pop_25_60 + Pop_sup_60	geom
SAINT-PIERRE-DU-PALAIS	380	
LA CLOTTE	695	
BOSCAMNANT	381	
LA GENETOUZE	220	
LA BARDE	473	
SAINT-MARTIN-DE-COUX	442	
LE FOUILLOUX	751	
SAINT-AIGULIN	1923	

Secrétariat général

Syntaxe SQL

L'instruction SELECT : les Alias

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette possibilité est particulièrement utile pour faciliter la lecture et l'écriture des requêtes.

Principe de rédaction : objet **as** intitule_alias ou objet intitule_alias

Exemple :

```
SELECT
    insee_com  as insee,
    nom_com as "nom de la commune",
    geom
FROM
    formation.communes
```

ou

```
SELECT
    insee_com insee,
    nom_com as "nom de la commune",
    geom
FROM
    formation.communes
```

Alias Alias

Données | EXPLAIN | Messages | Notifications

	insee character varying	nom de la commune character varying
1	72291	SAINT-JEAN-DE-LA-MO…
2	72009	ARTHEZE
3	49380	VAULANDRY
4	49101	CLEFS

Jamais de création d'alias dans la clause WHERE

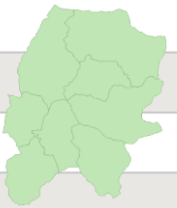
Secrétariat général

Syntaxe SQL

L'instruction Alias : Rédaction

```
SELECT
    nom_comm,
    "Pop_0_25" + "Pop_25_60" + "Pop_sup_60" as population,
    geom
FROM travail.communes
```

nom_com	population	geom
SAINT-PIERRE-DU-PALAIS	380	
BOSCAMNANT	381	
SAINT-MARTIN-DE-COUX	442	
LA CLOTTE	695	
LA BARDE	473	
SAINT-AIGULIN	1923	
LE FOUILLOUX	751	
LA GENETOUZE	220	



Secrétariat général

Exercice 4

Utilisation du SELECT

Dans QGIS dbmanager, à partir de la table communes_EPCI qui se trouve dans le schéma qui vous est assigné :

- Afficher les informations suivantes de la commune
  - Nom : nomcomm
  - Code INSEE : inseecomm (mettre un alias INSEE)
  - Population : poputotale
  - Superficie : surfha ou sup_km2 (mettre un alias Superficie en ….)
  - Densité de population : poputotale/ sup_km2 (mettre un alias Densité)
  - Les objets : geom
- Afficher
  - Afficher la couche dans QGIS et visualiser les données et la **carte**

Secrétariat général

Correction de l'exercice

Exercice 4

Utilisation du SELECT

Rédaction structurée

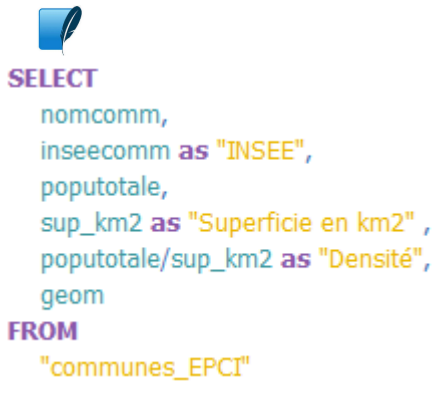
Bloc SELECT

Bloc FROM

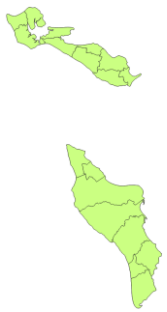
Alias

```
SELECT
    nomcomm,
    inseecomm as "INSEE",
    poputotale,
    sup_km2 as "Superficie en km2" ,
    poputotale/sup_km2 as "Densité",
    geom
FROM
    formation."communes_EPCI"
```

La structuration de la requête par bloc facilite la lecture mais surtout l'identification des erreurs



```
SELECT
    nomcomm,
    inseecomm as "INSEE",
    poputotale,
    sup_km2 as "Superficie en km2" ,
    poputotale/sup_km2 as "Densité",
    geom
FROM
    "communes_EPCI"
```



nomcomm	INSEE	poputotale	Superficie en m2	Densité
SAINT-CLEMENT-DES-BALEINES	17318	719	6,8	105,735294117647
SAINT-TROJAN-LES-BAINS	17411	1466	17,53	83,6280661722761
LE CHATEAU-D'OLERON	17093	3939	15,67	251,372048500319
LA BREE-LES-BAINS	17486	751	7,27	103,301237964237
LE GRAND-VILLAGE-PLAGE	17485	1026	6,05	169,586776859504
RIVEDOUX-PLAGE	17297	2292	4,52	507,079646017699

Secrétariat général

Syntaxe SQL

L'instruction SELECT DISTINCT

L'instruction SELECT DISTINCT permet d'éviter les redondances dans les résultats.

Principe de rédaction : SELECT DISTINCT *colonnes* FROM *schema.table*

lib15niv1 character varying	lib15niv2 character varying	lib15niv3 character varying
Forêts et milieux semi-naturels	Espaces ouverts, sans ou avec peu de végétation	Plages, dunes, sable
Forêts et milieux semi-naturels	Espaces ouverts, sans ou avec peu de végétation	Roches nues
Forêts et milieux semi-naturels	Forêts	Forêts de conifères
Forêts et milieux semi-naturels	Forêts	Forêts de feuillus
Forêts et milieux semi-naturels	Forêts	Forêts mélangées
Forêts et milieux semi-naturels	Milieux à végétation arbustive et/ou herbacée	Landes et broussailles
Surfaces en eau	Eaux continentales	Plans d'eau
Surfaces en eau	Eaux maritimes	Mers et océans
Territoires agricoles	Cultures permanentes	Vergers et petits fruits
Territoires agricoles	Cultures permanentes	Vignobles
Territoires agricoles	Prairies	Prairies
Territoires agricoles	Terres arables	Terres arables hors périmètres permanents d'irrigation
Territoires agricoles	Zones agricoles hétérogènes	Territoires principalement occupés par l'agriculture, avec
Territoires artificialisés	Espaces verts artificialisés non agricoles	Equipements sportifs et de loisirs
Territoires artificialisés	Espaces verts artificialisés non agricoles	Espaces verts urbains publics ou privés
Territoires artificialisés	Mines, décharges et carrières	Chantiers
Territoires artificialisés	Mines, décharges et carrières	Décharges
Territoires artificialisés	Mines, décharges et carrières	Extraction de matériaux

```
SELECT DISTINCT
    lib15niv1
FROM
    formation.ocs_re_2015
```

lib15niv1 character varying
Forêts et milieux semi-naturels
Surfaces en eau
Territoires agricoles
Territoires artificialisés
Zones humides

Secrétariat général

Syntaxe SQL

L'instruction SELECT DISTINCT

Si n colonnes sont citées dans la clause SELECT DISTINCT le résultat affichera les n-uplets sans redondance.

Principe de rédaction : SELECT DISTINCT col1,col2 FROM schema.table

lib15niv1 character varying	lib15niv2 character varying	lib15niv3 character varying
Forêts et milieux semi-naturels	Espaces ouverts, sans ou avec peu de végétation	Plages, dunes, sable
Forêts et milieux semi-naturels	Espaces ouverts, sans ou avec peu de végétation	Roches nues
Forêts et milieux semi-naturels	Forêts	Forêts de conifères
Forêts et milieux semi-naturels	Forêts	Forêts de feuillus
Forêts et milieux semi-naturels	Forêts	Forêts mélangées
Forêts et milieux semi-naturels	Milieux à végétation arbustive et/ou herbacée	Landes et broussailles
Surfaces en eau	Eaux continentales	Plans d'eau
Surfaces en eau	Eaux maritimes	Mers et océans
Territoires agricoles	Cultures permanentes	Vergers et petits fruits
Territoires agricoles	Cultures permanentes	Vignobles
Territoires agricoles	Prairies	Prairies
Territoires agricoles	Terres arables	Terres arables hors périmètres permanents d'irrigation
Territoires agricoles	Zones agricoles hétérogènes	Territoires principalement occupés par l'agriculture, avec
Territoires artificialisés	Espaces verts artificialisés non agricoles	Equipements sportifs et de loisirs
Territoires artificialisés	Espaces verts artificialisés non agricoles	Espaces verts urbains publics ou privés
Territoires artificialisés	Mines, décharges et carrières	Chantiers
Territoires artificialisés	Mines, décharges et carrières	Décharges
Territoires artificialisés	Mines, décharges et carrières	Extraction de matériaux

```
SELECT DISTINCT
    lib15niv1,
    lib15niv2
FROM
    formation.ocs_re_2015
```

lib15niv1 character varying	lib15niv2 character varying
Forêts et milieux semi-naturels	Espaces ouverts, sans ou avec peu de végétation
Forêts et milieux semi-naturels	Forêts
Forêts et milieux semi-naturels	Milieux à végétation arbustive et/ou herbacée
Surfaces en eau	Eaux continentales
Surfaces en eau	Eaux maritimes
Territoires agricoles	Cultures permanentes
Territoires agricoles	Prairies
Territoires agricoles	Terres arables
Territoires agricoles	Zones agricoles hétérogènes
Territoires artificialisés	Espaces verts artificialisés non agricoles
Territoires artificialisés	Mines, décharges et carrières
Territoires artificialisés	Zones industrielles ou commerciales et réseaux de communication
Territoires artificialisés	Zones urbanisées
Zones humides	Zones humides intérieures
Zones humides	Zones humides maritimes

Secrétariat général

Exercice 5

Utilisation du SELECT

Dans PgAdmin4, à partir de la table ocs_re_2015 qui se trouve dans le **schéma** qui vous est assigné :

- établir la liste des types d'occupation du sol (lib15niv1) sur l'ile de Ré :
  - Table : ocs_re_2015
  - Attribut : lib15niv1
- afficher le résultat de la colonne lib15niv1 avec l'alias « Nature Occupation Sol »

	Nature de l'occupation du sol character varying
1	Zones humides
2	Territoires artificialisés
3	Forêts et milieux semi-naturels
4	Surfaces en eau
5	Territoires agricoles

Secrétariat général

Correction de l'exercice

Exercice 5

Utilisation du SELECT

Utilisation de la clause DISTINCT pour éviter les doublons

Alias

Ne pas oublier de déclarer le schéma

```
SELECT DISTINCT lib15niv1 as "nature de l'occupation" from formation.ocs_2015
```

Le choix de l'alias n'est pas judicieux dans le cas d'une exploitation du résultat, mais permet une d'avoir un intitulé lisible : il faut juger en fonction de l'usage (mais en général il faut éviter les accents, blancs, caractères spéciaux, voire même les majuscules)

```
SELECT DISTINCT
    lib15niv1 as "Nature de l'occupation du sol"
FROM
    ocs_re_2015
```

	Nature de l'occupation du sol character varying
1	Zones humides
2	Territoires artificialisés
3	Forêts et milieux semi-naturels
4	Surfaces en eau
5	Territoires agricoles

Secrétariat général

LES SUPER DÉFIS

Les messages d'erreurs

```
1  SELECT
2      nomcomm,
3      inseecomm as "INSEE",
4      poptotale, sup_km2 as "Superficie en m2",
5      poputotale/sup_km2 as "Densité hab/km2",
6  FROM
7      formation."communes_EPCI"
```

Données EXPLAIN Messages Notifications

```
ERREUR : ERREUR:  erreur de syntaxe sur ou près de « FROM »
LINE 6: FROM
        ^

État SQL : 42601
Caractère : 126
```

Secrétariat général

LES SUPER DÉFIS

Les messages d'erreurs

```
1  SELECT
2      nomcomm,
3      inseecomm as "INSEE",
4      poptotale, sup_km2 as "Superficie en m2",
5      poputotale/sup_km2 as "Densité hab/km2",
6  FROM
7      formation."communes_EPCI"
```

Données EXPLAIN Messages Notifications

```
ERREUR : ERREUR:  erreur de syntaxe sur ou près de « FROM »
LINE 6: FROM
        ^


État SQL : 42601
Caractère : 126
```

Données EXPLAIN Messages Notifications

```
ERREUR : ERREUR:  la colonne « poptotale » n'existe pas
LINE 4:  poptotale, sup_km2 as "Superficie en m2",
         ^
HINT:  Peut-être que vous souhaitiez référencer la colonne « communes_EPCI.poputotale ».


État SQL : 42703
Caractère : 42
```

Secrétariat général

LES SUPER DÉFIS

Les messages d'erreurs

```
1  SELECT
2      nomcomm,
3      inseecomm as "INSEE",
4      poptotale, sup_km2 as "Superficie en m2",
5      poputotale/sup_km2 as "Densité hab/km2",
6  FROM
7      formation."communes_EPCI"
```

Données EXPLAIN Messages Notifications

```
ERREUR : ERREUR:  erreur de syntaxe sur ou près de « FROM »
LINE 6: FROM
        ^


État SQL : 42601
Caractère : 126
```

Données EXPLAIN Messages Notifications

```
ERREUR : ERREUR:  la colonne « poptotale » n'existe pas
LINE 4:  poptotale, sup_km2 as "Superficie en m2",
         ^
HINT:  Peut-être que vous souhaitiez référencer la colonne « communes_EPCI.poputotale ».


État SQL : 42703
Caractère : 42
```

Secrétariat général

Syntaxe SQL

L'instruction SELECT : la clause WHERE

La commande WHERE dans une requête SQL permet d'extraire des enregistrements d'une base de données qui respectent une condition

Principe de rédaction : SELECT …. FROM …… **WHERE** ...condition(s)…

Toute la difficulté réside dans la rédaction de la condition :

- Si la condition est vérifiée l'enregistrement est sélectionné
- Si la condition n'est pas vérifiée l'enregistrement n'est pas sélectionné

Pour rédiger une condition on utilise des opérateurs :

- de comparaisons (exemple tel attribut est égal à une valeur ou texte défini)
- de logique (permet de cumuler des conditions)

Secrétariat général

Syntaxe SQL

L'instruction SELECT : la clause WHERE les opérateurs de comparaisons

x	x	=	Égal	Col_num = 10 Col_car = *'vigne'*
x	x	<> ou !=	Différent	Col_num != 10 Col_car != 'vigne'
x	x	>	Supérieur	
x	x	<	Inférieur	
x	x	<=	Supérieur ou égal	
x	x	>=	Inférieur ou égal	
x	x	BETWEEN	Compris entre	Col_num **BETWEEN** 100 **AND** 500
x	x	IN	Fait parti de la liste de valeur	Col_num **IN(**10**,**15**,**40**)** Col_car **IN(**'vigne' **,** 'forêt'**)**
x	x	LIKE	Contient ou est comme	Col_Car **LIKE** *'%*forêt%'
x		soundex	«Sonne comme »	**Soundex**(col_car) = **Soundex**('cristel')

Secrétariat général

Syntaxe SQL

L'instruction SELECT : la clause WHERE, l'opérateur de comparaison LIKE

LIKE 'chaîne' (sensible à la casse)

ILIKE 'chaîne' (insensible à la casse) – uniquement pour PostgreSQL

Caractères joker :

- **%** correspond à plusieurs caractères quelconques
- _ correspond à un seul caractère quelconque

Exemples :

SELECT * FROM commune WHERE nom_commune **LIKE** 'M**%**'

→ Sélectionne toutes les communes dont le nom commence par M

SELECT * FROM commune WHERE nom_commune **ILIKE** '**%**SAINT**%**'

→ Sélectionne toutes les communes dont le nom contient la chaîne 'SAINT' (en lettres minuscules ou majuscules)

Secrétariat général

Syntaxe SQL

L'instruction SELECT : la clause WHERE, la clause NULL

NULL aucune valeur attribuée

Exemples :

SELECT * FROM communes WHERE nom_com **IS NULL**

→ Sélectionne toutes les enregistrements de la table communes qui n'ont pas de valeurs (nom de commune) dans l'attribut nom_com

SELECT * FROM commune WHERE nom_commune **IS NOT NULL**

→ Sélectionne toutes les enregistrements de la table communes qui ont une valeur (nom de commune) dans l'attribut nom_com

Remarque : rédaction aboutissant au même résultat que IS NOT NULL

SELECT * FROM commune WHERE nom_commune

Secrétariat général

Exercice 6

Utilisation du SELECT

Dans PgAdmin4, sélectionner dans le **schéma** qui vous est assigné les communes_EPCI qui ont une population supérieur à **1500 habitants**, Afficher le nom de la commune, son code INSEE et sa population:

- Table : communes_EPCI
- Attributs : nomcomm, inseecomm et poputotale

Auto-complétion : CTL + espace

	nomcomm character varying	inseecomm character varying	poputotale bigint
1	LA FLOTTE	17161	2861
2	SAINTE-MARIE-DE-RE	17360	3235
3	LE BOIS-PLAGE-EN-RE	17051	2356
4	LE CHATEAU-D'OLERON	17093	3939
5	SAINT-MARTIN-DE-RE	17369	2471
6	RIVEDOUX-PLAGE	17297	2292
7	DOLUS-D'OLERON	17140	3185
8	SAINT-GEORGES-D'OLE...	17337	3482
9	SAINT-PIERRE-D'OLER...	17385	6676

Secrétariat général

Correction de l'exercice

Exercice 6

Utilisation du SELECT

```
SELECT
  nomcomm,
  inseecomm,
  poputotale
FROM
  "communes_EPCI"
WHERE
  poputotale > 1500
```

Dans PgAdmin4, sélectionner dans le **schéma** qui vous est assigné les communes qui ont une population supérieur à **1500 habitants**, Afficher le nom de la commune, son code INSEE et sa population

```
SELECT nomcomm, inseecomm, poputotale FROM formation."communes_EPCI" WHERE poputotale > 1500
```

	nomcomm character varying	inseecomm character varying	poputotale bigint
1	LA FLOTTE	17161	2861
2	SAINTE-MARIE-DE-RE	17360	3235
3	LE BOIS-PLAGE-EN-RE	17051	2356
4	LE CHATEAU-D'OLERON	17093	3939
5	SAINT-MARTIN-DE-RE	17369	2471
6	RIVEDOUX-PLAGE	17297	2292
7	DOLUS-D'OLERON	17140	3185
8	SAINT-GEORGES-D'OLE...	17337	3482
9	SAINT-PIERRE-D'OLER...	17385	6676

Critère

Il faut mettre le nom de la table entre des doubles quôtes car il contient des majuscules

Pas besoin de mettre 1500 entre simples quôtes car il s'agit d'un constante de type numérique

Secrétariat général

Exercice 7

Utilisation du SELECT

Dans PgAdmin4, sélectionner dans le **schéma** qui vous est assigné les communes dont le nom contient « **PLAGE** » Afficher le nom de la commune, son code INSEE et sa population :

- Table : communes_EPCI
- Attributs : nomcomm, inseecomm et poputotale

Auto-complétion : CTL + espace

	nomcomm character varying	inseecomm character varying	poputotale bigint
1	LE GRAND-VILLAGE-PLAGE	17485	1026
2	LE BOIS-PLAGE-EN-RE	17051	2356
3	RIVEDOUX-PLAGE	17297	2292

Secrétariat général

Correction de l'exercice

Exercice 7

Utilisation du SELECT

```
SELECT
  nomcomm,
  inseecomm,
  poputotale
FROM
  "communes_EPCI"
WHERE
  nomcomm LIKE '%PLAGE%'
```

Communes dont le nom contient « **PLAGE** »

```
SELECT nomcomm, inseecomm,poputotale FROM formation."communes_EPCI"
WHERE nomcomm LIKE '%PLAGE%'
```

LIKE pour utiliser les caractères joker

% pour indiquer qu'il peut y avoir d'autres caractères

nomcomm character varying	inseecomm character varying	poputotale bigint
LE GRAND-VILLAGE-PL...	17485	1026
LE BOIS-PLAGE-EN-RE	17051	2356
RIVEDOUX-PLAGE	17297	2292

Secrétariat général

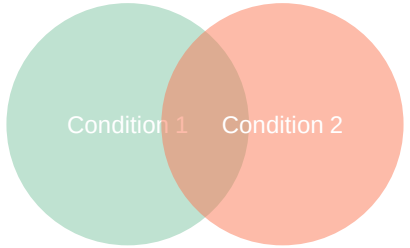
Syntaxe SQL

L'instruction SELECT : la clause WHERE les opérateurs logiques

Dans la clause WHERE il est possible de stipuler plusieurs conditions en utilisant un opérateur logique

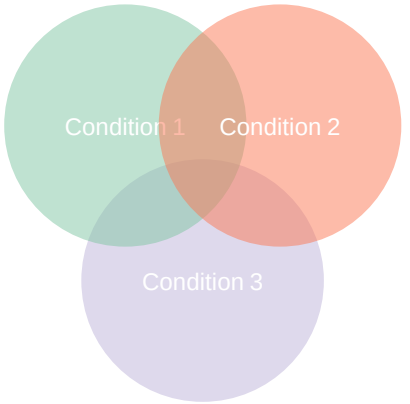
- **OR** (séparer 2 conditions dont au moins une doit être vérifiée)

  SELECT * FROM commune WHERE statut = 'commune simple' OR statut = 'chef-lieu de canton'
  → Sélectionne les communes pour lesquelles le statut est commune simple ou chef-lieu de canton.



- **AND** (séparer 2 conditions qui doivent être vérifiées simultanément)

  SELECT * FROM commune WHERE statut = 'commune simple' AND population >10000
  → Seules les communes simples de plus de 10 000 habitants sont sélectionnées.



- **NOT** (permet d'inverser une condition)

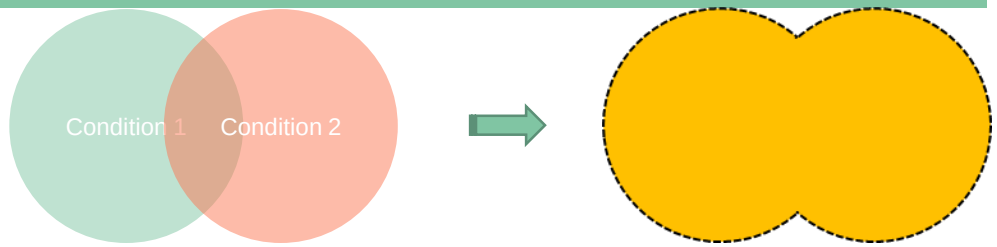
  SELECT * FROM commune WHERE NOT (statut = 'commune simple' OR statut = 'chef-lieu de canton')
  → Sélectionne les communes qui ne sont ni commune simple, ni chef-lieu de canton.

Secrétariat général

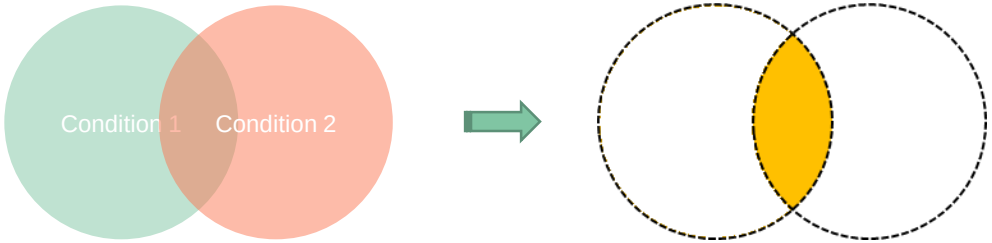
Syntaxe SQL

L'instruction SELECT : la clause WHERE les opérateurs logiques

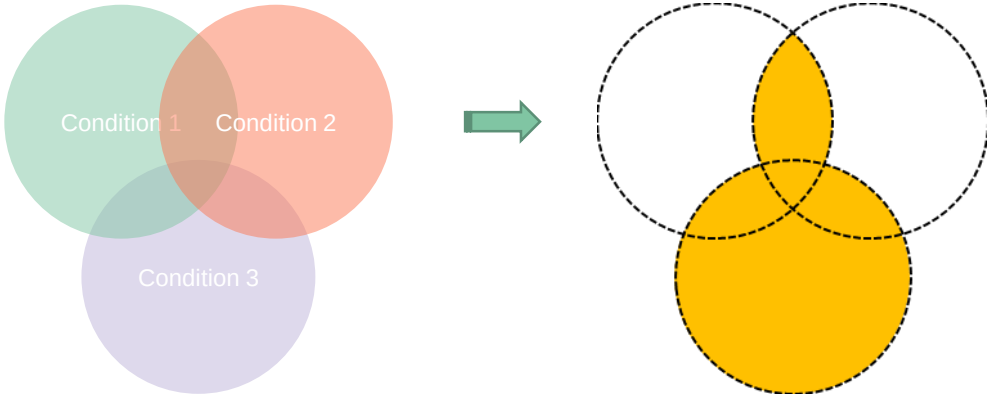
SELECT … FROM ….
WHERE cond1 **OR** cond2



SELECT … FROM ….
WHERE cond1 **AND** cond2



SELECT … FROM ….
WHERE cond1 **AND** cond2 **OR** cond3



Secrétariat général

Exercice 8

Utilisation du SELECT

Dans PgAdmin4, sélectionner dans le **schéma** qui vous est assigné les communes qui ont une population supérieur à **1500 habitants** et dont le nom contient « **PLAGE** »
Afficher le nom de la commune, son code INSEE et sa population :

- Table : communes_EPCI
- Attributs : nomcomm, inseecomm et poputotale

	nomcomm character varying	inseecomm character varying	poputotale bigint
1	LE BOIS-PLAGE-EN-RE	17051	2356
2	RIVEDOUX-PLAGE	17297	2292

Auto-complétion : CTL + espace

Secrétariat général

Correction de l'exercice

Exercice 8

Utilisation du SELECT

Communes qui ont une population supérieur à **1500 habitants** et dont le nom contient « **PLAGE** »

```
SELECT nomcomm, inseecomm, poputotale FROM formation."communes_EPCI"
WHERE poputotale > 1500 AND nomcomm LIKE '%PLAGE%'
```

Critère 1

AND car les 2 critères doivent être vérifiés

Critère 1

% pour indiquer qu'il peut y avoir d'autres caractères

```
SELECT
  nomcomm,
  inseecomm,
  poputotale
FROM
  "communes_EPCI"
WHERE
  nomcomm LIKE '%PLAGE%' AND poputotale > 1500
```

	nomcomm character varying	inseecomm character varying	poputotale bigint
1	LE BOIS-PLAGE-EN-RE	17051	2356
2	RIVEDOUX-PLAGE	17297	2292

Secrétariat général

Exercice 9

Utilisation du SELECT

Dans le gestionnaireBD de QGIS, à partir de la table ocs_re_2015 :

- Sélectionner à partir de l'attribut **lib15niv3** les décharges et les chantiers
- Afficher dans le résultat les attributs geom, gid, lib15niv3, surf_m2 et surf_ha

Pour en savoir plus sur le SQL et les fonctions:

- https://sql.sh/

	geom	gid	lib15niv3	surf_m2	surf_ha
1	01060000206A0...	49919	Chantiers	1645.18135901419	0.164518135901...
2	01060000206A0...	49920	Chantiers	1349.17534562685	0.134917534562...
3	01060000206A0...	49432	Décharges	9533.14714728215	0.953314714728...
4	01060000206A0...	49699	Décharges	7177.55685376456	0.717755685376...
5	01060000206A0...	48492	Décharges	1701.35688904986	0.170135688904...
6	01060000206A0...	48493	Décharges	1551.30950327528	0.155130950327...
7	01060000206A0...	48494	Décharges	1555.43905562694	0.155543905562...
8	01060000206A0...	48495	Décharges	2205.71980891013	0.220571980891...
9	01060000206A0...	48496	Décharges	1192.91735883792	0.119291735883...
10	01060000206A0...	48690	Décharges	1961.61034395724	0.196161034395...
11	01060000206A0...	48751	Décharges	3087.90643705473	0.308790643705...

Secrétariat général

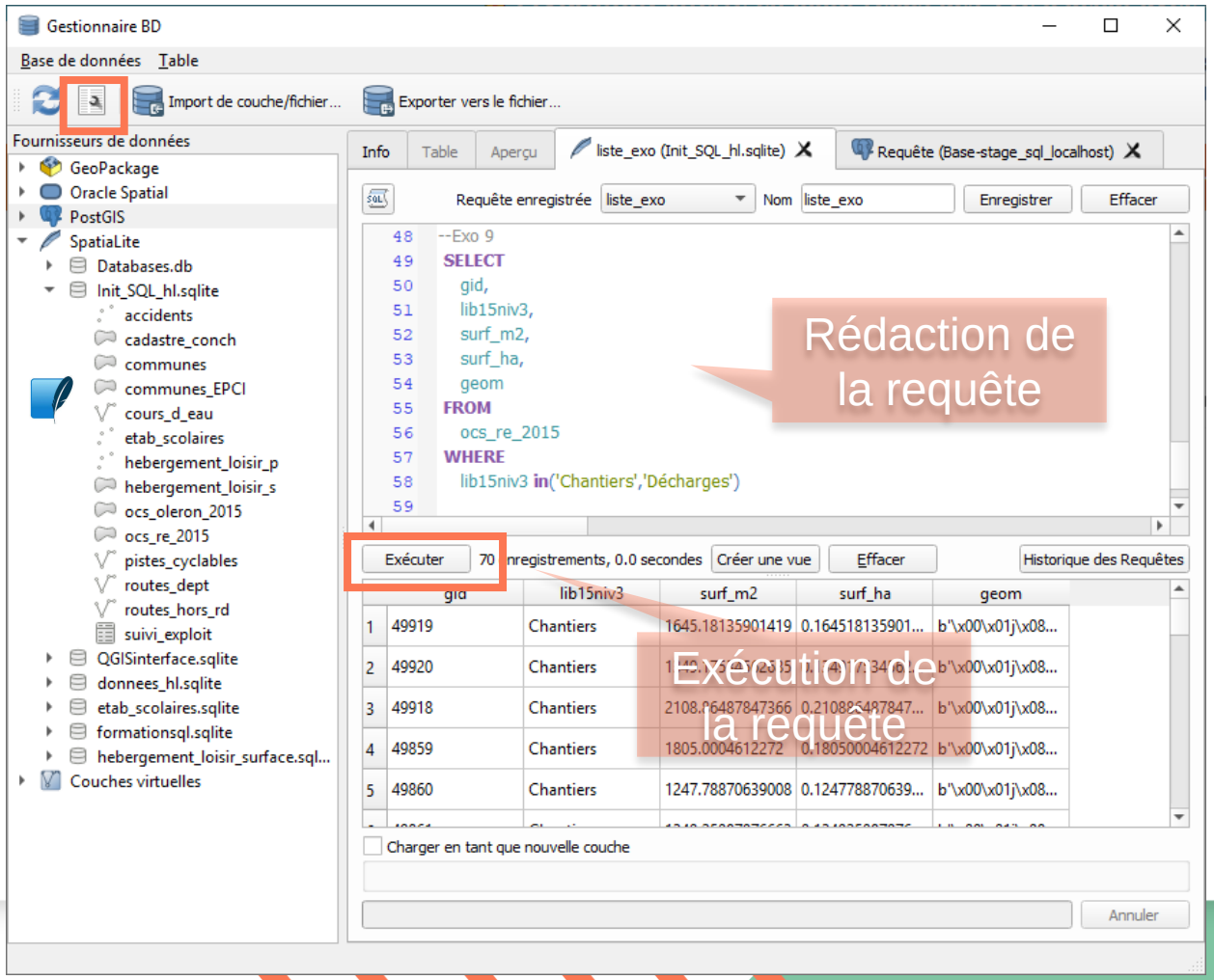
Correction de l'exercice

Exercice 9

Utilisation du SELECT

	lib15niv3
1	Plans d'eau
2	Tissu urbain discontinu
3	Marais maritimes
4	Roches nues
5	Terres arables hors périmèt...
6	Zones industrielles, comm...
7	Tissu urbain continu
8	Vignobles
9	Forêts de feuillus
10	Landes et broussailles
11	Zones intertidales
12	Zones portuaires
13	Espaces verts urbains publi...
14	Plages, dunes, sable
15	Forêts mélangées
16	Forêts de conifères
17	Equipements sportifs et de ...
18	Territoires principalement ...
19	Décharges
20	Vergers et petits fruits
21	Marais intérieurs
22	Mers et océans
23	Chantiers
24	Extraction de matériaux
25	Prairies
26	Réseaux routiers et ferroviai...

```sql
SELECT
   gid,
   lib15niv3,
   surf_m2,
   surf_ha,
   geom
FROM
   formation.ocs_re_2015
WHERE
   lib15niv3 in('Chantiers','Décharges')
```



Secrétariat général

Correction de l'exercice

Exercice 9

Utilisation du SELECT

	lib15niv3
1	Plans d'eau
2	Tissu urbain discontinu
3	Marais maritimes
4	Roches nues
5	Terres arables hors périmèt...
6	Zones industrielles, comm...
7	Tissu urbain continu
8	Vignobles
9	Forêts de feuillus
10	Landes et broussailles
11	Zones intertidales
12	Zones portuaires
13	Espaces verts urbains publi...
14	Plages, dunes, sable
15	Forêts mélangées
16	Forêts de conifères
17	Equipements sportifs et de ...
18	Territoires principalement ...
19	Décharges
20	Vergers et petits fruits
21	Marais intérieurs
22	Mers et océans
23	Chantiers
24	Extraction de matériaux
25	Prairies
26	Réseaux routiers et ferroviai...

```
SELECT
    geom,
    gid,
    lib15niv3,
    surf_m2,
    surf_ha
FROM
    formation.ocs_re_2015
WHERE
    lib15niv3 in('Décharges','Chantiers')
```

IN(….,….)
pour lister les valeurs recherchées

```
lib15niv3='Décharges' OR ib15niv3='Chantiers'
```

Secrétariat général

Exercice 10

Utilisation du SELECT

Dans le gestionnaireBD de QGIS, à partir de la table ocs_re_2015 :

- Sélectionner à partir de l'attribut **lib15niv3** les éléments qui ne sont ni des décharges ni des chantiers

Secrétariat général

Correction de l'exercice

Exercice 10

Utilisation du SELECT

```sql
SELECT
    geom,
    gid,
    lib15niv3,
    surf_m2,
    surf_ha
FROM
    formation.ocs_re_2015
WHERE
    NOT (lib15niv3 = 'Décharges' OR lib15niv3 = 'Chantiers')
```

NOT(….,….)
pour inverser le critère

```sql
SELECT
  gid,
  lib15niv3,
  surf_m2,
  surf_ha,
  geom
FROM
  ocs_re_2015
WHERE
  Not lib15niv3 in('Chantiers','Décharges')
```

Secrétariat général

Correction de l'exercice

Exercice 10

Utilisation du SELECT

```sql
SELECT
    geom,
    gid,
    lib15niv3,
    surf_m2,
    surf_ha
FROM
    formation.ocs_re_2015
WHERE
    NOT (lib15niv3 = 'Décharges' OR lib15niv3 = 'Chantiers')
```

NOT(….,….)
pour inverser le critère

```sql
SELECT
  gid,
  lib15niv3,
  surf_m2,
  surf_ha,
  geom
FROM
  ocs_re_2015
WHERE
  Not lib15niv3 in('Chantiers','Décharges')
```

```sql
(lib15niv3 IN ('Décharges','Chantiers'))
```

Secrétariat général

Syntaxe SQL

Les principaux type de données

CHARACTER (ou **CHAR**) : texte de longueur fixe

CHARACTER VARYING (ou **VARCHAR**) : texte de longueur maximale fixée

TEXT : suite longue de caractères (sans limite de taille)

NUMERIC (ou DECIMAL ou DEC) : décimal

INTEGER (ou INT) : entier long

REAL (ou **FLOAT**): réel à virgule flottante dont la représentation est binaire

BOOLEAN (ou LOGICAL) : vrai/faux

DATE : date du calendrier grégorien

Secrétariat général

Syntaxe SQL

L'utilisation du **booléen** dans les expressions

le type BOOLEAN peut prendre 3 états (Vrai / Faux / inconnu = NULL)

Les valeurs pour **vrai** sont :

TRUE ou '**true**'
'**T**' ou '**t**'
'**y**' ou '**yes**'
'**on**'
1

Les valeurs pour **faux** sont :

FALSE ou '**false**'
'**F**' ou '**f**'
'**n**' ou '**no**'
'**off**'
0

Secrétariat général

Syntaxe SQL

L'utilisation du **booléen** dans les expressions

Si **mon_champ** est de type booléen (ou logique) :

Requête avec le test à l'état « vrai » :

- SELECT * FROM ma_table WHERE **mon_champ**
- SELECT * FROM ma_table WHERE **mon_champ is true**
- SELECT * FROM ma_table WHERE **mon_champ =TRUE / 'TRUE' / 't' / 'true' / 'y' / 'yes' / '1'**

Requête avec le test à l'état « faux »:

- SELECT * FROM ma_table WHERE **NOT mon_champ**
- SELECT * FROM ma_table WHERE **mon_champ is false**
- SELECT * FROM ma_table WHERE **mon_champ =FALSE / 'FALSE' / 'f' / 'false' / 'n' / 'no' / '0'**

Secrétariat général

Syntaxe SQL

Le transtypage

Cast : fonction standard SQL permettant de convertir un type de données en un autre type

Exemples :

- convertir un champ INTEGER en TEXT
- convertir un champ INTEGER en FLOAT
- convertir un champ DATE en TEXT
- convertir un champ géométrique POLYGONE en POINT (cf diapo geom)

Syntaxe

CAST(**nom_champ** as **Nvx_TYPE**) ou

nom_champ :: Nvx_TYPE

Secrétariat général

Syntaxe SQL

Le transtypage

Une opération de transtypage est parfois nécessaire pour obtenir le résultat souhaité.

Exemple de calcul d'un indicateur (ratio de deux entiers) sachant que le résultat de la division de deux entiers est un entier :

SELECT surfha/100 AS surf_km2 FROM formation.«communes_EPCI»

surfha bigint	sans_t bigint	avec_t double prec
680	6	6.8
880	8	8.8
1175	11	11.75
1232	12	12.32

Pour obtenir un résultat satisfaisant il faut au minimum convertir le numérateur ou le dénominateur en flottant:

*SELECT **cast(**surfha **AS FLOAT)** /100 AS surf_km2 FROM formation.«communes_EPCI»*

Ou

*SELECT surfha**::FLOAT** /100 as surf_km2 FROM formation.«communes_EPCI»*

Secrétariat général

Syntaxe SQL

Fonctions de traitements des nombres

x	x	Pow(a,b)	« a » puissance « b »	*Pow(2,3)*	*8*
x	x	Sqrt(a)	Racine carrée de « a »	Sqr(16)	4
x	x	Round(a)	Arrondi « a » au plus proche entier	Round(42.8)	43
x	x	Round(a,b)	Arrondi « a » à la « b »ème décimale *Attention « a » de type « numéric »*	Round(42.7846, 3)	42.785
x	x	Pi()		Pi()	3.141592…..
x	-	Trunc(a)	Retire ce qu'il y a après zéro	Trunc(42.8)	42
x	-	Trunc(a,b)	Retire ce qu'il y a après « b »ème décimale *Attention « a » de type « numéric »*	Trunc(42.7846, 3)	42.784
x	-	Mod(a,b)	Modulo (reste de la division a/b)	Mod(10,4)	2
x	x	Radians(a)	Converti a (degré) en radian	Radians(180)	3.141592…..

Secrétariat général

Syntaxe SQL

Fonctions de traitements des nombres

Exemples d'utilisation :

SELECT ***Round(****(population / sup_km2)::NUMERIC****, 2)*** *AS densite_habkm2 FROM formation.«communes_EPCI»*

ou

SELECT ***Round(((****population / ****(cast(****surfha* ***AS FLOAT)*** */100))****::NUMERIC , 2)*** *AS densite_habkm2 FROM formation.«communes_EPCI»*

Secrétariat général

Exercice 11

Utilisation de fonctions mathématiques

Dans PgAdmin4, à partir de la table commune_EPCI qui se trouve dans le schéma qui vous est assigné :

- Afficher les informations suivantes de la commune
  - Nom : nomcomm
  - Code INSEE : inseecomm
  - Population : poputotale
  - Superficie en km2
  - Population
  - La densité arrondie à 2 chiffres après la virgule alias densite_hab_km2

Secrétariat général

Correction de l'exercice

Exercice 11

Utilisation de fonctions mathématiques

```
SELECT
  nomcomm,
  inseecomm as "INSEE",
  poputotale,
  sup_km2 as "Superficie en km2" ,
  round(poputotale/sup_km2,2) as "Densité",
  geom
FROM
  "communes_EPCI"
```

```
SELECT
        nomcomm,
        inseecomm,
        poputotale,
        sup_km2,
        Round((population / sup_km2)::NUMERIC, 2) AS densite_hab_km2

FROM
        formation.«communes_EPCI»
```

Secrétariat général

Exercice 12

Utilisation de fonctions mathématiques

Dans PgAdmin4, à partir de la table commune_EPCI qui se trouve dans le schéma qui vous est assigné :

- Afficher les informations suivantes de la commune **qui ont une densité supérieur à 200 hab/km2**
  - Nom : nomcomm
  - Code INSEE : inseecomm
  - Population : poputotale
  - Superficie en km2
  - Population
  - La densité arrondie à 2 chiffres après la virgule alias densite_hab_km2

Correction de l'exercice

Exercice 12

Utilisation de fonctions mathématiques

```
SELECT
  nomcomm,
  inseecomm as "INSEE",
  poputotale,
  sup_km2 as "Superficie en km2" ,
  round(poputotale/sup_km2,2) as "densite_hab_km2",
  geom
FROM
  "communes_EPCI"
WHERE
  "densite_hab_km2" > 200
```

```
SELECT
        nomcomm,
        inseecomm,
        populationtotale,
        sup_km2,
        Round(cast(population / sup_km2 as NUMERIC), 2) AS densite_hab_km2
FROM
        formation.«communes_EPCI»
WHERE
        (population / sup_km2) >= 200
```

Secrétariat général

Syntaxe SQL

Fonctions de traitements des chaines de caractères

Ces fonctions permettent de travailler avec les champs de type texte :

- Dans la clause SELECT elles permettent de modifier l'affichage

```
SELECT nomcomm, length(nomcomm), Initcap(nomcomm), Replace(nomcomm,'PLAGE','MER') from formation."communes_EPCI"
```

nomcomm character varying	length integer	initcap text	replace text
LE GRAND-VILLAGE-PLAGE	22	Le Grand-Village-Plage	LE GRAND-VILLAGE-MER
LE BOIS-PLAGE-EN-RE	19	Le Bois-Plage-En-Re	LE BOIS-MER-EN-RE
RIVEDOUX-PLAGE	14	Rivedoux-Plage	RIVEDOUX-MER

- Dans la clause WHERE elles permettent de modifier la référence utilisée

```
SELECT nomcomm, length(nomcomm) from formation."communes_EPCI" WHERE length(nomcomm)<20
```

nomcomm character varying	length integer
LE BOIS-PLAGE-EN-RE	19
RIVEDOUX-PLAGE	14

Syntaxe SQL

Fonctions de traitements des chaines de caractères

Comme toutes les fonctions celles-ci peuvent s'imbriquer

```
SELECT nomcomm, length(nomcomm), Initcap(Replace(nomcomm,'PLAGE','MER')) from formation."communes_EPCI"
```

nomcomm character varying	length integer	initcap text
LE GRAND-VILLAGE-PLAGE	22	Le Grand-Village-Mer
LE BOIS-PLAGE-EN-RE	19	Le Bois-Mer-En-Re
RIVEDOUX-PLAGE	14	Rivedoux-Mer

Secrétariat général

Syntaxe SQL

Fonctions de traitements des chaines de caractères

x	-	Concat(a,b)	Concatène a et b	*Concat('Rouge','Bleu')*	*RougeBleu*
x	x	||		*'Rouge'* || *'Bleu'*	
x	-	Concat_ws(a,b,..,n)	Concatène b n avec séparateur «a»	Concat_ws(',',*'Rge','Bleu')*	*Rouge,Bleu*
x	x	Length(a)	Compte combien il y a de caractères dans « a »	Length('Bonjour')	7
x	x	Upper(a)	Retourne « a » en majuscule	Upper('Bonjour à tous')	BONJOUR A TOUS
x	x	Lower(a)	Retourne « a » en minuscule	Lower('Bonjour PAUL')	bonjour paul
x	-	Initcap(a)	Retrourne « a » avec la première lettre de chaque mot en majuscule	Initcap('SALUT PAUL')	Salut Paul

Secrétariat général

Syntaxe SQL

Fonctions de traitements des chaines de caractères

x	x	Substr(a,b,[c])	Extraire une partie d'une chaine de caractère « a » à partir de la position « b » sur une longueur de « c »	Substr('Bonjour',3,4)	njou
x	-	Substring(a,b,[c])		Substring('Bonjour',3,4)	
-	x	Instr(a,b)	Renvoi la position de la chaine « b » dans la chaine « a »	Instr('Bonjour','jo')	4
x	-	Position(b in a)		Position('jo' in 'Bonjour')	
x	x	Replace(a,b,c)	Remplacer des caractères dans une chaîne de caractère «a» en remplaçant «b» par «c»	*Replace(« LE BOIS PLAGE EN RE, 'PLAGE', 'MER')*	*LE BOIS **MER** EN RE*
x	-	Left(a,b)	Récupère les « b » premiers caractères de « a »	Left('Bonjour',3)	Bon
x	-	Right(a,b)	Récupère les « b » derniers caractères de « a »	Right('Bonjour',3)	our

Secrétariat général

Syntaxe SQL

Fonctions de traitements des chaines de caractères

x	x	Trim(a)	Supprime les espaces de la chaine de caractère « a »	Trim('_Bonjour_à_tous_')	Bonjouràtous
x	x	Ltrim(a)	Supprime les espaces en **début** de la chaine de caractère « a »	Ltrim('_Bonjour à tous_')	Bonjour à tous_
x	x	Rtrim(a)	Supprime les espaces en **fin** de la chaine de caractère « a »	Rtrim('_Bonjour à tous_')	_Bonjour à tous
x	-	Lpad(a,b,c)	Ajouter le contenu spécifié « c » en **début** du chaine « a », jusqu'à atteindre la longueur désirée « b »	Lpad('Bonjour', 10, 'X')	XXXBonjour
x	-	Rpad(a,b,c)	Ajouter le contenu spécifié « c » en **fin** du chaine « a », jusqu'à atteindre la longueur désirée « b »	Lpad('Bonjour', 10, 'X')	BonjourXXX

Secrétariat général

Exercice 13

Fonctions de traitements des chaines de caractères

Dans PgAdmin4, à partir de la table commune_EPCI

- Afficher les informations suivantes des communes **qui contiennent le mot 'SAINT' dans la colonne nomcomm**
  - Nom : nomcomm
  - Code INSEE : inseecomm
  - Population : poputotale
  - L'intitulé abrégé du nom de la commune ST-…… à la place de SAINT-…….
- Option : Afficher en plus l'intitulé abrégé avec une Majuscule pour chaque mot

INITCAP ne fonctionne avec SQLite

nomcomm character varying	Nom abrégé text	Nom abrégé etiq text
SAINT-CLEMENT-DES-BALEINES	ST-CLEMENT-DES-BALEINES	St-Clement-Des-Baleines
SAINT-DENIS-D'OLERON	ST-DENIS-D'OLERON	St-Denis-D'Oleron
SAINTE-MARIE-DE-RE	STE-MARIE-DE-RE	Ste-Marie-De-Re
SAINT-MARTIN-DE-RE	ST-MARTIN-DE-RE	St-Martin-De-Re
SAINT-TROJAN-LES-BAINS	ST-TROJAN-LES-BAINS	St-Trojan-Les-Bains
SAINT-GEORGES-D'OLERON	ST-GEORGES-D'OLERON	St-Georges-D'Oleron
SAINT-PIERRE-D'OLERON	ST-PIERRE-D'OLERON	St-Pierre-D'Oleron

Secrétariat général

Correction de l'exercice

Exercice 13

Fonctions de traitements des chaines de caractères

```
SELECT
        nomcomm,
        REPLACE(nomcomm,'SAINT','ST') as "Nom abrégé",
        INITCAP(replace(nomcomm,'SAINT','ST')) as "Nom abrégé etiq"
FROM
        formation.«communes_EPCI»
WHERE
        nomcomm like '%SAINT%'
```

```
SELECT
  nomcomm,
  replace(nomcomm,'SAINT','ST') as "Nom abrégé",
  upper(Substr(nomcomm,1,1))||lower(Substr(replace(nomcomm,'SAINT','ST'),2))
FROM
  "communes_EPCI"
WHERE
  nomcomm LIKE '%SAINT%'
```

nomcomm character varying	Nom abrégé text	Nom abrégé etiq text
SAINT-CLEMENT-DES-BALEINES	ST-CLEMENT-DES-BALEINES	St-Clement-Des-Baleines
SAINT-DENIS-D'OLERON	ST-DENIS-D'OLERON	St-Denis-D'Oleron
SAINTE-MARIE-DE-RE	STE-MARIE-DE-RE	Ste-Marie-De-Re
SAINT-MARTIN-DE-RE	ST-MARTIN-DE-RE	St-Martin-De-Re
SAINT-TROJAN-LES-BAINS	ST-TROJAN-LES-BAINS	St-Trojan-Les-Bains
SAINT-GEORGES-D'OLERON	ST-GEORGES-D'OLERON	St-Georges-D'Oleron
SAINT-PIERRE-D'OLERON	ST-PIERRE-D'OLERON	St-Pierre-D'Oleron

Secrétariat général

Exercice 14

Fonctions de traitements des chaines de caractères

Dans PgAdmin4, à partir de la table suivi_exploit et seulement pour les opérations (d_lbl_oper) qui ne sont pas « renouvellement »

- Afficher les informations suivantes :
  - d_lbl_oper
  - Detenteur
- Construire une colonne intitulé **code** formatée sur 15 caractères qui soit constituée :
  - Des 2 premières lettres de la colonne d_lbl_oper en majuscule
  - D'un underscore « _ »
  - Du détenteur en majuscule
  - D'un underscore
  - De 'X' pour compléter le code afin qu'il fasse 15 caractères

SQLite : utiliser Substr au lieu de Rpad et Left

d_lbl_oper character varying	Detenteur character varying	rpad text
Substitution à un tiers	TE6Fr12	SU_TE6FR12_XXXX
Substitution à un tiers	CA8Ed13	SU_CA8ED13_XXXX
Substitution à un tiers	BR4To4	SU_BR4TO4_XXXXX
Echange	RI9Pi15	EC_RI9PI15_XXXX
Substitution à un tiers	BE7Fa6	SU_BE7FA6_XXXXX
Echange	RA6Je15	EC_RA6JE15_XXXX
Réduction (superficie / longueur)	GA8La18	RÉ_GA8LA18_XXXX

Secrétariat général

Correction de l'exercice

Exercice 14

Fonctions de traitements des chaines de caractères

```
SELECT
        d_lbl_oper,
        "Detenteur",
        Rpad(Concat(Upper(left(d_lbl_oper,2)
),'_',Upper("Detenteur"),'_'),15,'X')
FROM
        formation.«suivi_exploit»
WHERE
        d_lbl_oper <> 'Renouvellement'
```

```
SELECT
    d_lbl_oper, "Detenteur",
    Substr(Upper(Substr(d_lbl_oper,1,2))
        || '_' ||
        Upper("Detenteur")
        || '_'||
        'XXXXXXXXXXXXX'
        , 1 ,15) as code
FROM
    suivi_exploit
WHERE
    d_lbl_oper <> 'Renouvellement'
```

d_lbl_oper character varying	Detenteur character varying	rpad text
Substitution à un tiers	TE6Fr12	SU_TE6FR12_XXXX
Substitution à un tiers	CA8Ed13	SU_CA8ED13_XXXX
Substitution à un tiers	BR4To4	SU_BR4TO4_XXXXX
Echange	RI9Pi15	EC_RI9PI15_XXXX
Substitution à un tiers	BE7Fa6	SU_BE7FA6_XXXXX
Echange	RA6Je15	EC_RA6JE15_XXXX
Réduction (superficie / longueur)	GA8La18	RÉ_GA8LA18_XXXX

Secrétariat général

Syntaxe SQL

Fonctions de traitements des dates

x	-	Current_date	Date courante		
x	-	Current_time	Heure courante		
x	-	Now()	Date et heure courante		
x	x	Date('now')	Date courante		
x	-	Age(a,[b])	Soustrait à la date courante ou à la date « b »	Age('2021-05-27','2012-01-09')	9 years 4 mons 18 days
x	-	Date_part('day', a)	day : jour du mois	Date_part('day', '2020-02-25')	25
-	x	Strftime('%d', a)	%d : jour du mois	Strftime('%d', '2020-02-25')	25
x	-	Date_part('month', a)	month : numero du mois	Date_part('month', '2020-02-25')	2 (1 = Jan.)
-	x	Strftime('%m', a)	%m : mois de l'année	Strftime('%m', '2020-02-25')	22 (1 = Jan.)

Secrétariat général

Syntaxe SQL

Fonctions de traitements des dates

x	-	Date_part('month', a)	month : numero du mois	Date_part('month', '2020-02-25')	2 (1 = Jan.)
-	x	Strftime('%m', a)	%m : mois de l'année	Strftime('%m', '2020-02-25')	02 (1 = Jan.)
x	-	Date_part('year', a)	year : Année	Date_part('year', '2020-02-25')	2020
-	x	Strftime('%Y', a)	%Y : Année	Strftime('%Y', '2020-02-25')	2020
x	-	Date_part('dow', a)	Dow: jour de la semaine (dimanche =0)	Date_part('dow', '2020-021-25')	4 (0 = Dim.)
-	x	Strftime('%w', a)	%w: jour de la semaine (dimanche =0)	Strftime('%w','2020-021-25')	4 (0 = Dim.)
x	-	Date_part('doy', a)	Doy : jour de l'année	Date_part('doy', '2020-02-25')	56 (1 – 365)
x	-	Date_part('week', a)	Week : numero de la semaine	Date_part('week', '2020-02-25')	8 (1 – 52)

Secrétariat général

Exercice 15

Fonctions de traitements des chaines de caractères

Dans PgAdmin4, à partir de la table accidents

- Sélectionner les accidents qui ont eu lieu un **samedi (6) ou un dimanche (0)** :
  - Date_acc

    (utiliser la fonction date_part('dow', ….) qui donne le type de jour (0 : dimanche, 1 : lundi,……, 6 : samedi)
- Afficher les informations suivantes :
  - Type d'accident (attribut Type_accident)
  - Date de l'accident (attribut date_acc)
  - Type de jour (attribut date_acc) alias code_jour

Type_accident character varying	date_acc date	code_jour double precision
Accident grave non mortel	2019-02-23	6
Accident grave non mortel	2019-08-25	0
Accident grave non mortel	2019-02-23	6
Accident grave non mortel	2019-12-29	0
Accident Léger	2019-05-19	0
Accident Léger	2019-12-15	0
Accident mortel	2019-06-29	6
Accident mortel	2019-08-11	0

Secrétariat général

Correction de l'exercice

Exercice 15

Fonctions de traitements des chaines de caractères

```
SELECT
        "Type_accident",
        date_acc,
        date_part('dow' , date_acc) as code_jour
FROM
        formation.accidents
WHERE
        date_part('dow' , date_acc)  in( 0,6)
```

```
SELECT
  "Type_accident",
  date_acc,
  strftime('%w',date_acc) as "code_jour"
FROM
  accidents
WHERE
  code_jour in('0','6')
```

Secrétariat général

Syntaxe SQL

Expression conditionnelle CASE WHEN

Il peut être intéressant d'afficher une informations en fonction d'un résultat.

« Si telle valeur, afficher alors … »

Par exemple afficher le jour de l'accident et non le code du jour.

Si l'accident s'est produit le week-end afficher l'intitulé du jour (samedi ou dimanche) sinon afficher qu'il s'agit d'un jours de la semaine

Type_accident character varying	date_acc date	code_jour double precis	le_jour text
Accident grave non mortel	2019-02-23	6	Samedi
Accident grave non mortel	2019-05-09	4	jours de la semaine
Accident grave non mortel	2019-05-29	3	jours de la semaine
Accident grave non mortel	2019-07-01	1	jours de la semaine
Accident grave non mortel	2019-07-03	3	jours de la semaine
Accident grave non mortel	2019-08-06	2	jours de la semaine
Accident grave non mortel	2019-08-14	3	jours de la semaine
Accident grave non mortel	2019-08-25	0	Dimanche
Accident grave non mortel	2019-02-12	2	jours de la semaine
Accident grave non mortel	2019-02-23	6	Samedi
Accident grave non mortel	2019-09-03	2	jours de la semaine

Secrétariat général

CASE *expression* WHEN *valeur* THEN *résultat* [WHEN ...] [ELSE *résultat*] END

Syntaxe SQL

Expression conditionnelle CASE WHEN

L'expression conditionnelle **CASE WHEN** permet de définir les valeurs en fonction du résultat de l'expression :

CASE *expression*
WHEN *valeur1* THEN *résultat*
WHEN *valeur1* THEN *résultat*
[ELSE *résultat*]
END

Il peut être intéressant d'utiliser cette expression dans la clause SELECT pour afficher des valeurs en fonction du résultat. Par exemple afficher le jour de l'accident si l'accident s'est produit le week-end sinon afficher qu'il s'agit d'un jours de la semaine

```
SELECT
        date_acc,
        date_part('dow', date_acc) as code_jour,
        Age( date_acc),
        CASE date_part('dow', date_acc)
                WHEN 0 THEN 'Dimanche'
                WHEN 6 THEN 'Samedi'
                ELSE 'jours de la semaine'
        END as le_jour,
        geom
FROM formation.accidents
```

Secrétariat général

Exercice 16

Fonctions de condition

Dans PgAdmin4, à partir de sélection accidents

- Afficher les informations suivantes :
  - Type d'accident (attribut Type_accident)
  - Date de l'accident (attribut date_acc)
  - Type de jour (attribut date_acc) alias code_jour
  - Intitulé du jour (attribut date_acc) alias le_jour : Afficher :
    - ✓ Samedi (valeur 6)
    - ✓ Dimanche (valeur 0)
    - ✓ Jours de la semaine (valeurs 1,2,3,4,5)

Type_accident character varying	date_acc date	code_jour double précis	le_jour text
Accident grave non mortel	2019-02-23	6	Samedi
Accident grave non mortel	2019-05-09	4	jours de la semaine
Accident grave non mortel	2019-05-29	3	jours de la semaine
Accident grave non mortel	2019-07-01	1	jours de la semaine
Accident grave non mortel	2019-07-03	3	jours de la semaine
Accident grave non mortel	2019-08-06	2	jours de la semaine
Accident grave non mortel	2019-08-14	3	jours de la semaine
Accident grave non mortel	2019-08-25	0	Dimanche
Accident grave non mortel	2019-02-12	2	jours de la semaine
Accident grave non mortel	2019-02-23	6	Samedi
Accident grave non mortel	2019-09-03	2	jours de la semaine

Secrétariat général

Correction de l'exercice

Exercice 16

Fonctions de condition

```
SELECT
  "Type_accident",
  date_acc,
  strftime('%w', date_acc) as "code_jour",
  CASE strftime('%w', date_acc)
     WHEN '0' THEN 'Dimanche'
     WHEN '6' THEN 'Samedi'
     ELSE 'jours de la semaine'
  END as le_jour,
  geom
FROM
  accidents
```

```
SELECT
        "type_accident",
        date_acc,
        date_part('dow', date_acc) as code_jour,
        CASE date_part('dow', date_acc)
                WHEN 0 THEN 'Dimanche'
                WHEN 6 THEN 'Samedi'
                ELSE 'jours de la semaine'
        END as le_jour,
        geom
FROM formation.accidents
```

date_acc character varying	code_jour double precisio	jour_acc text
Accident grave non mortel	6	Samedi
Accident grave non mortel	4	jours de la semaine
Accident grave non mortel	3	jours de la semaine
Accident grave non mortel	1	jours de la semaine
Accident grave non mortel	3	jours de la semaine
Accident grave non mortel	2	jours de la semaine
Accident grave non mortel	3	jours de la semaine
Accident grave non mortel	0	Dimanche
Accident grave non mortel	2	jours de la semaine
Accident grave non mortel	6	Samedi

Secrétariat général

Syntaxe SQL

Clause ORDER BY

La clause **ORDER BY** suivi d'une liste de champs permet d'ordonner le résultat de la requête en fonction de ces champs

*SELECT * FROM commune **ORDER BY nom_comm***

- classer le résultat par nom de commune

Le tri décroissant peut-être obtenu en ajoutant **DESC**.

*SELECT * FROM commune ORDER BY nom_comm **DESC***

Il est possible d'utiliser l'alias d'une colonne spécifié dans la clause SELECT

SELECT nom_comm, population/superficie AS **densite** FROM commune ORDER BY **densite**

Secrétariat général

Syntaxe SQL

Les clauses LIMIT et OFFSET

LIMIT : permet de limiter le nombre de réponses renvoyées
Intéressant pour tester une requête sur une table contenant beaucoup d'objets
Il est également possible d'utiliser la clause OFFSET pour décaler le nombre de lignes à obtenir

Exemples :
SELECT * FROM commune **LIMIT 1** → renvoie le premier objet
SELECT * FROM commune **LIMIT 10 OFFSET 5** → *renvoie les enregistrements de 6 à 15*

Pour obtenir des classements, on associe **ORDER BY** avec **LIMIT**
SELECT * FROM commune ORDER BY population **DESC LIMIT 5**
→ renvoie les **5 communes les plus peuplées**

Secrétariat
général

LES JOINTURES ENTRE TABLES

Secrétariat général

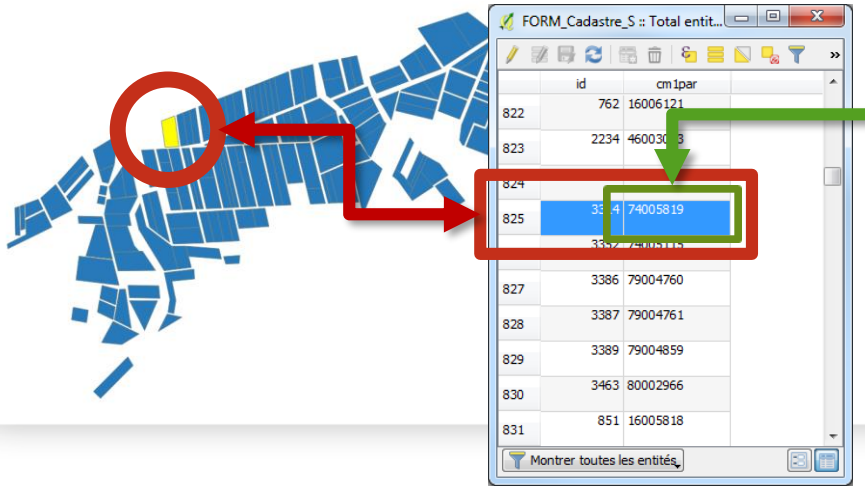
Syntaxe SQL

Jointures attributaires

Mettre en relation 2 tables (ou plus) afin de combiner leurs colonnes

- La plupart des jointures attributaires se font en imposant l'égalité d'une colonne d'une table à une colonne d'une autre table

Exemple :



id	Quartier	Detenteur	Concession	Commune	Surface	Longueur	D_Decision	D_expiration	Lbl_operation	Famille	Lbl_nature	Lbl_espece	Civilite	Nom
2039	LR	19744020	16006018		3000	0	18/10/93	26/03/25	Renouvellement	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2040	LR	19744020	13007624	ST MARTIN DE RE	500	0	19/08/14	26/03/25	Renouvellement	Dépot	Dépot Surélevé	Huître Creuse	M.	TEXIER
2041	LR	19744020	13007924		500	0	30/10/95	26/03/25	Création	Dépot	Dépot Surélevé	Huître Creuse	M.	TEXIER
2042	LR	19744020	13008624		640	0	01/09/89	30/04/20	Régularisation cadastrale	Dépot	Dépot Surélevé	Huître Creuse	M.	TEXIER
2043	LR	19744020	13008834		1595	0	24/06/97	26/03/25	Renouvellement	Dépot	Dépot Surélevé	Huître Creuse	M.	TEXIER
2044	LR	19744020	020 555	LA FLOTTE	2400	0	18/05/11	01/06/18	Substitution à un tiers	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2045	LR	19744020	020 425		2500	0	05/07/00	24/11/24	Agrandissement (superficie/ longueur)	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2046	LR	19744020	09 30	STE MARIE DE RE	750	0	03/08/12	26/03/25	Création	Captage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2047	LR	19744020	160 121	LOIX	2500	0	01/08/08	26/03/25	Renouvellement	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2048	LR	19744020	16005425	LOIX	2496	0	17/05/06	26/08/34	Substitution à un tiers	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	TEXIER
2049	LR	19744036	74005819		4140	0	13/07/98	13/07/24	Création	Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2050	LR	19744036	74005115	ARS EN RE	4800	0	17/05/06	30/11/25	Echange	Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2051	LR	19744036	46003063	CHATELAILLON PLA	1856	0	07/04/05	04/08/24	Renouvellement	Captage	A Plat Terrain Découvrant	Huître Creuse	M.	GUIGNET
2052	LR	19744036	79004760		2400	0	25/11/02	25/11/24	Création	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2053	LR	19744036	80002966	ARS EN RE	1500	0	17/05/06	13/02/28	Echange	Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2054	LR	19744036	52004233	FOURAS	1325	0	24/05/11	04/08/24	Substitution partielle à des tiers	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2055	LR	19744036	79004761		2400	0	25/11/02	25/11/24	Création	Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2056	LR	19744036	79004859		2400	0	25/11/02	25/11/24	Création	Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	GUIGNET
2057	LR	19744144	16005818	LOIX	4000	0	22/06/12	04/03/17	Renouvellement	Captage/Elevage	En Surélevé Terrain Découvrant	Huître Creuse	M.	NEAU

Secrétariat général

Syntaxe SQL

Jointures attributaires

Une jointure peut être réalisée suivant 2 principes :

- En déclarant celle-ci au niveau de la clause WHERE (le résultat ne contient que les enregistrements communs)

SELECT * FROM tab1, tab2 **WHERE tab1.lien1 = tab2.lien2**

- En déclarant explicitement la jointure (JOIN, INNER JOIN, LEFT JOIN,…) : ce principe est recommandé car dans ce cas on maitrise le type de résultat (seulement les enregistrements communs, tous les enregistrements de la table tab1, …)

SELECT * **FROM tab1 JOIN tab2 ON lien1 =lien2**

Dans les 2 cas, l'attribut lien1 de la table tab1 correspond à l'attribut lien2 de la table tab2

Secrétariat général

Syntaxe SQL

Jointures attributaires

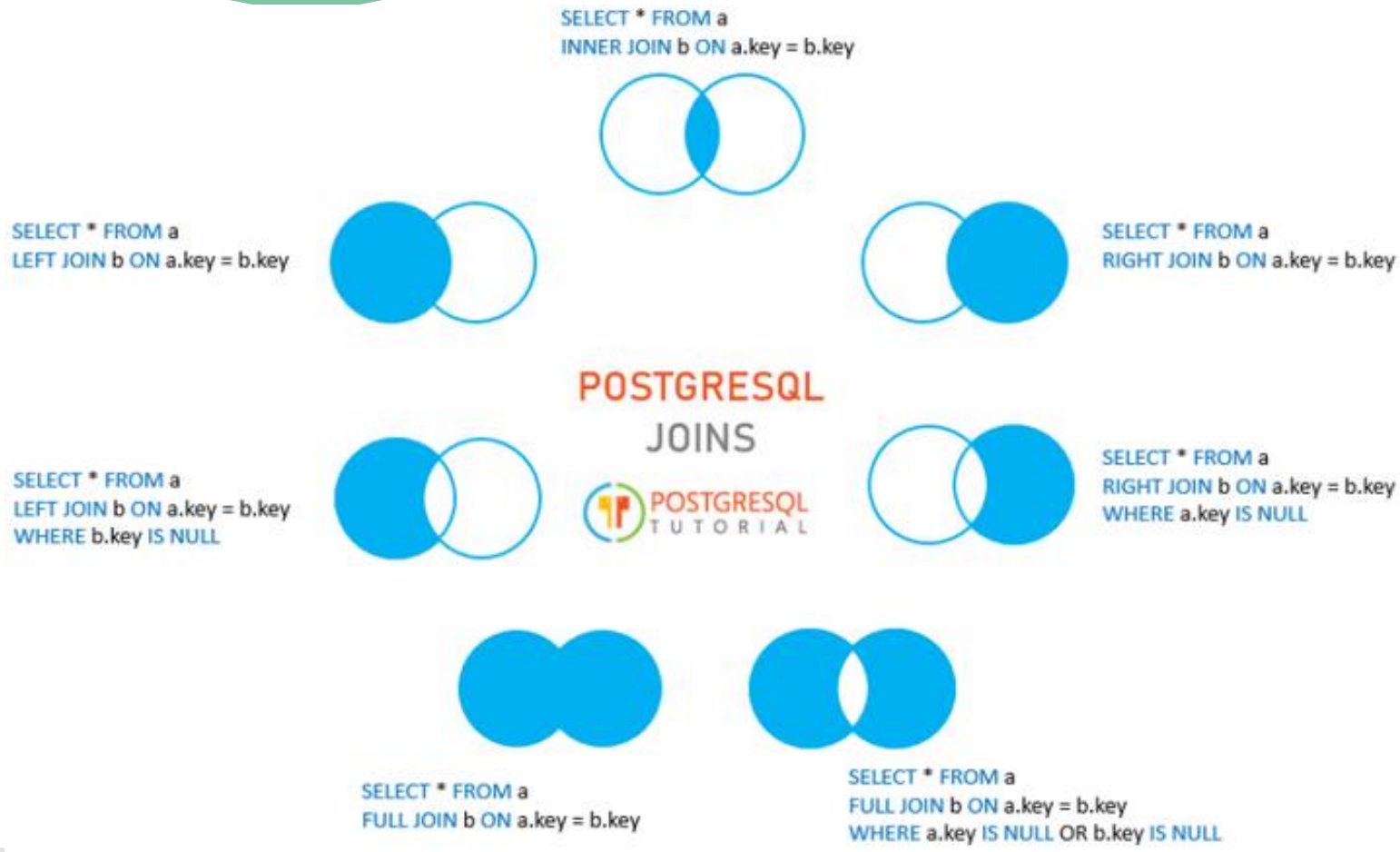
Principe de rédaction :

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

JOIN ou INNER JOIN : les enregistrements communs
LEFT JOIN : tous les enregistrements de la table de gauche
RIGHT JOIN : tous les enregistre
FULL JOIN : tous les enregistrements des 2 tables

Syntaxe SQL

Jointures attributaires



Tables en entrée	Type de jointure	Requête SQL	Résultat
Table 1 (id, nom) : 1 a ; 2 b ; 3 c Table 2 (id, valeur) : 1 xxx ; 3 yy ; 3 yyyy ; 5 zzz	**Croisée** CROSS JOIN Pas de critère de jointure	**SELECT *** **FROM** table1 **AS** t1 **CROSS JOIN** table2 **AS** t2 ;	t1.id, t1.nom, t2.id, t2.valeur 1 a 1 xxx 1 a 3 yy 1 a 3 yyyy 1 a 5 zzz 2 b 1 xxx 2 b 3 yy 2 b 3 yyyy 2 b 5 zzz 3 c 1 xxx 3 c 3 yy 3 c 3 yyyy 3 c 5 zzz
	Complète FULL JOIN Nécessité d'avoir un critère de jointure	**SELECT *** **FROM** table1 **AS** t1 **FULL JOIN** table2 **AS** t2 **ON** t1.id = t2.id ;	t1.id, t1.nom, t2.id, t2.valeur 1 a 1 xxx 2 b 3 c 3 yy 3 c 3 yyyy 5 zzz
	À gauche LEFT JOIN Nécessité d'avoir un critère de jointure	**SELECT *** **FROM** table1 **AS** t1 **LEFT JOIN** table2 **AS** t2 **ON** t1.id = t2.id ;	t1.id, t1.nom, t2.id, t2.valeur 1 a 1 xxx 2 b 3 c 3 yy 3 c 3 yyyy
	À droite RIGHT JOIN Nécessité d'avoir un critère de jointure	**SELECT *** **FROM** table1 **AS** t1 **RIGHT JOIN** table2 **AS** t2 **ON** t1.id = t2.id ;	t1.id, t1.nom, t2.id, t2.valeur 1 a 1 xxx 3 c 3 yy 3 c 3 yyyy 5 zzz
	Naturelle JOIN Nécessité d'avoir un critère de jointure	**SELECT *** **FROM** table1 **AS** t1 **JOIN** table2 **AS** t2 **ON** t1.id = t2.id ;	t1.id, t1.nom, t2.id, t2.valeur 1 a 1 xxx 3 c 3 yy 3 c 3 yyyy

Secrétariat général

Exercice 21

Réaliser une jointure attributaire

Réaliser une jointure afin de venir compléter les données de la table cadastre_conch avec les informations contenues dans la table suivi_exploit

- Attributs de jointure :
  - ✓ cm1par pour la table cadastre_conch
  - ✓ Concession pour la table suivi_exploit

Résultat 1423 lignes (et non 1008)

	id	cm1par
2	859	16006480
3	842	16007855
4	841	16007960
5	885	16006880
6	884	16006894
7	887	16004825
8	886	16005858

	id	d_lbl_oper	d_lbl_espe	Detenteur	Concession	Commune	Surface	Longueur	la con	Points	C_operation	Lbl_operation	C_Nature	Famille	Lbl_nature	C_espece	Lbl_especе
4	318	Renouvellement	Huître Creuse	TU5Lo17	[illegible]	RIVEDOUX PLAGE	750	0	C	1500	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
5	344	Renouvellement	Huître Creuse	CH7Mi7	08007032	RIVEDOUX PLAGE	750	0	C	1500	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
6	320	Renouvellement	Huître Creuse	BE7Er17	08005818		450	0	C	225	20	Renouvellement	67	Dépot	Dépot Surélevé	53020	Huître Creuse
7	300	Renouvellement	Huître Creuse	BE7Fa6	08004839	RIVEDOUX PLAGE	332	0	C	664	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
8	299	Mise à la dispos...	Huître Creuse	PO7Fr16	08004752	RIVEDOUX PLAGE	825	0	C	1650	95	Mise à la dispos...	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
9	302	Mutation après ...	Huître Creuse	LA6Lo4	08004951	RIVEDOUX PLAGE	825	0	C	1650	75	Mutation après ...	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
10	301	Renouvellement	Huître Creuse	BE7Fa6	08004920	RIVEDOUX PLAGE	720	0	C	360	20	Renouvellement	67	Dépot	Dépot Surélevé	53020	Huître Creuse
11	295	Renouvellement	Huître Creuse	BE7Li6	08003860	RIVEDOUX PLAGE	750	0	C	1500	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
12	293	Renouvellement	Huître Creuse	BE7AI10	08003722		295	0	C	147	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse
13	298	Renouvellement	Huître Creuse	FO7Fr15	08004733	RIVEDOUX PLAGE	600	0	C	600	20	Renouvellement	23	Captage/Elevage	En Surélevé Terr...	53020	Huître Creuse

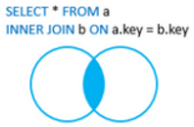
Secrétariat général

Correction de l'exercice

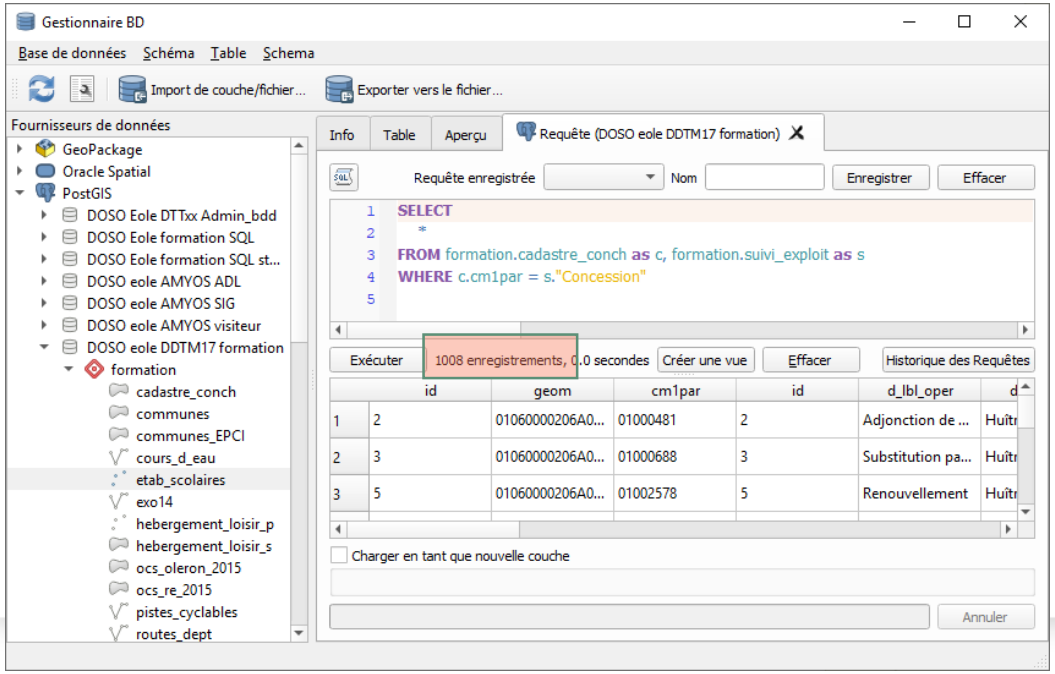
Exercice 21

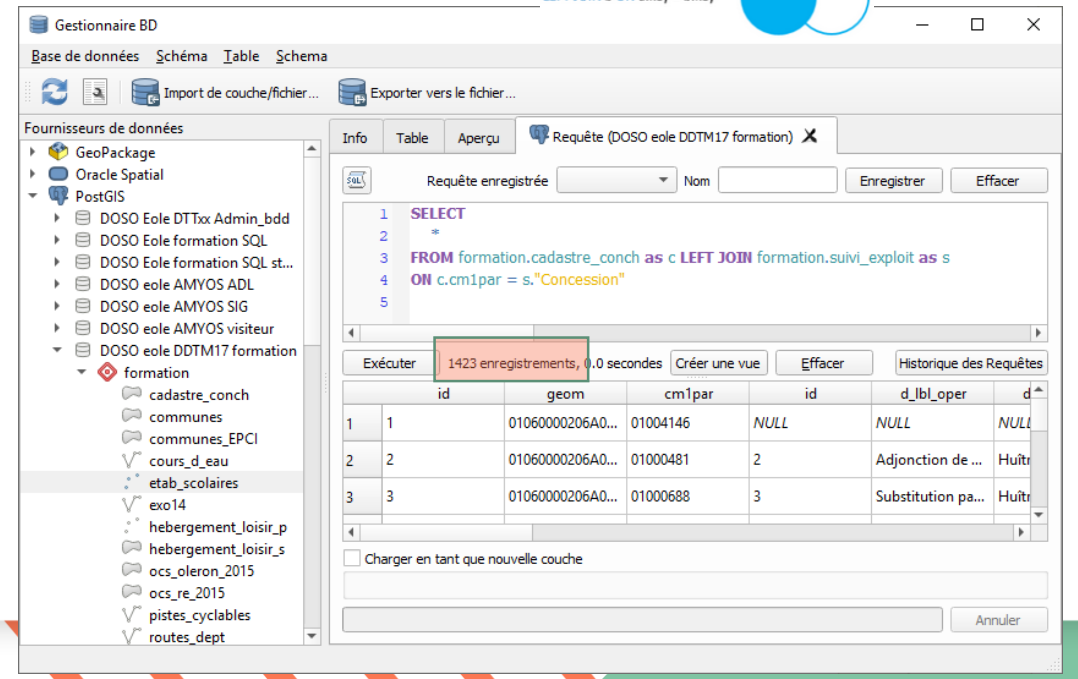
Réaliser une jointure attributaire

Il faut dans le cas de cet exercice utiliser un LEFT JOIN




```
SELECT *
FROM formation.cadastre_conch as c LEFT JOIN formation.suivi_exploit as s
ON c.cm1par = s."Concession"
```





Secrétariat général

Syntaxe SQL

Jointures géographiques

Tout comme les jointures attributaires, les jointures géographiques peuvent être réalisées suivant 2 principes :

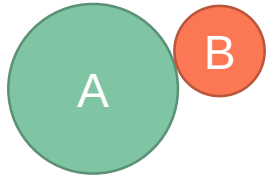
- En déclarant celle-ci au niveau de la clause WHERE (le résultat ne contient que les enregistrements communs)

SELECT …… , **xxx.geom** FROM tab1, tab2 **WHERE** *<Condition de jointure entre **tab1.geom et tab2.geom**>*

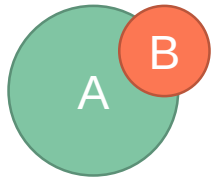
- En déclarant explicitement la jointure (JOIN, INNER JOIN, LEFT JOIN,…) : ce principe est recommandé car dans ce cas on maitrise le type de résultat (seulement les enregistrements communs, tous les enregistrements de la table tab1, …)

SELECT …… , **xxx.geom** FROM **tab1 JOIN tab2 ON** *<Condition de jointure entre **tab1.geom et tab2.geom**>*

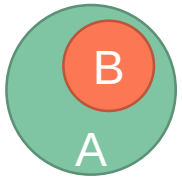
touche



Intersect



contient



Et bien d'autres……

Secrétariat général

Syntaxe SQL

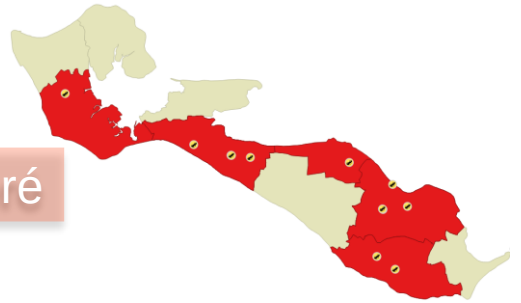
Jointures géographiques

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes, accidents
Where
   st_contains(communes.geom,accidents.geom)
```

Objet récupéré

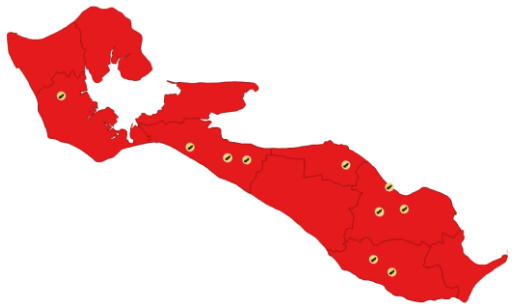
Opérateur géographique

Intitulé de la colonne géographique



Type_accident	date_acc	code_insee	comm_mi
Accident mortel	2019-09-19	17121	La Couarde-sur-Mer
Accident grave non mortel	2019-09-11	17161	La Flotte
Accident Léger	2019-07-17	17161	La Flotte
Accident Léger	2019-07-17	17161	La Flotte
Accident mortel	2019-08-06	17161	La Flotte
Accident grave non mortel	2019-08-14	17360	Sainte-Marie-de-Ré
Accident grave non mortel	2019-10-09	17360	Sainte-Marie-de-Ré
Accident grave non mortel	2019-12-04	17369	Saint-Martin-de-Ré
Accident grave non mortel	2019-09-20	17019	Ars-en-Ré
Accident grave non mortel	2019-07-03	17121	La Couarde-sur-Mer
Accident grave non mortel	2019-08-25	17121	La Couarde-sur-Mer
Accident grave non mortel	2019-09-17	17121	La Couarde-sur-Mer

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes LEFT JOIN accidents
ON
   st_contains(communes.geom,accidents.geom)
```



Type_accident	date_acc	code_insee	comm_mi
Accident mortel	2019-09-19	17121	La Couarde-sur-Mer
Accident grave non mortel	2019-09-11	17161	La Flotte
Accident Léger	2019-07-17	17161	La Flotte
Accident Léger	2019-07-17	17161	La Flotte
Accident mortel	2019-08-06	17161	La Flotte
		17318	Saint-Clément-des-Baleines
Accident grave non mortel	2019-08-14	17360	Sainte-Marie-de-Ré
Accident grave non mortel	2019-10-09	17360	Sainte-Marie-de-Ré
		17051	Le Bois-Plage-en-Ré
		17207	Loix
Accident grave non mortel	2019-12-04	17369	Saint-Martin-de-Ré
		17286	Les Portes-en-Ré
Accident grave non mortel	2019-07-03	17121	La Couarde-sur-Mer
Accident grave non mortel	2019-09-20	17019	Ars-en-Ré
Accident grave non mortel	2019-08-25	17121	La Couarde-sur-Mer
		17297	Rivedoux-Plage
Accident grave non mortel	2019-09-17	17121	La Couarde-sur-Mer

Secrétariat général

Syntaxe SQL

Jointures géographiques

Définition de la jointure

where

Jointure uniquement

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes, accidents
Where
   st_contains(communes.geom,accidents.geom)
```

Jointure + critères

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes,accidents
Where
   st_contains(communes.geom,accidents.geom) AND  communes.code_insee in('17286','17318','17019',
   '17207','17121','17369','17051','17161','17360','17297')
```

Join

Jointure uniquement

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes LEFT JOIN accidents
ON
   st_contains(communes.geom,accidents.geom)
```

Jointure + critères

```
SELECT
   accidents."Type_accident", accidents.date_acc,
   communes.code_insee, communes.comm_mi, communes.geom
FROM
   communes LEFT JOIN accidents
ON
   st_contains(communes.geom,accidents.geom)
WHERE
   communes.code_insee in('17286','17318','17019',
   '17207','17121','17369','17051','17161','17360','17297')
```

Secrétariat général

Syntaxe SQL

Jointures géographiques : principales fonctions spatiales

ST_Intersects(geometry A, geometry B) au moins un point commun.

ST_Within(geometry A, geometry B) le premier objet est complétement dans le deuxième.

ST_Contains(geometry A, geometry B) le deuxième objet est complétement dans le premier.

ST_Overlaps(geometry A, geometry B) le premier objet est partiellement contenu dans le deuxième objet

ST_Touches(geometry A, geometry B) les contours ont au moins un point commun mais pas leurs intérieurs

ST_Dwithin(geometry A, geometry B, distance) retourne les objets si la distance la plus courte entre A et B est inférieure ou égale à distance.

ST_Distance(geometry A, geometry B) <= distance retourne les objets si la distance la plus courte entre A et B est inférieure ou égale à distance.

Secrétariat général

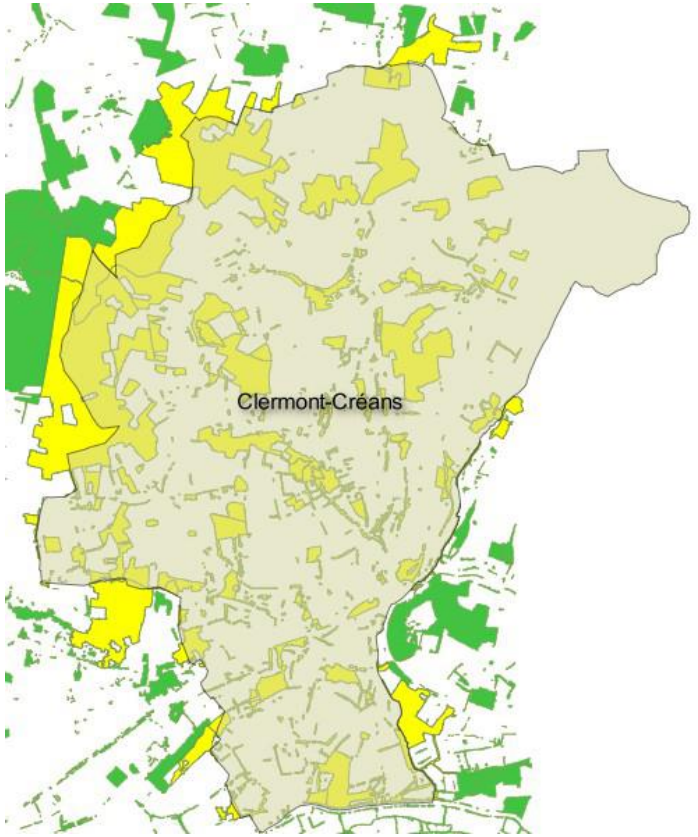
Syntaxe SQL

Jointures géographiques : ST_Intersects

Intersecte

Les objets de la couche végétation sélectionnés (polygones jaunes) ont au moins un point commun avec le polygone (grisé) de la commune

ST_Intersects(geometry A, geometry B)



Secrétariat général

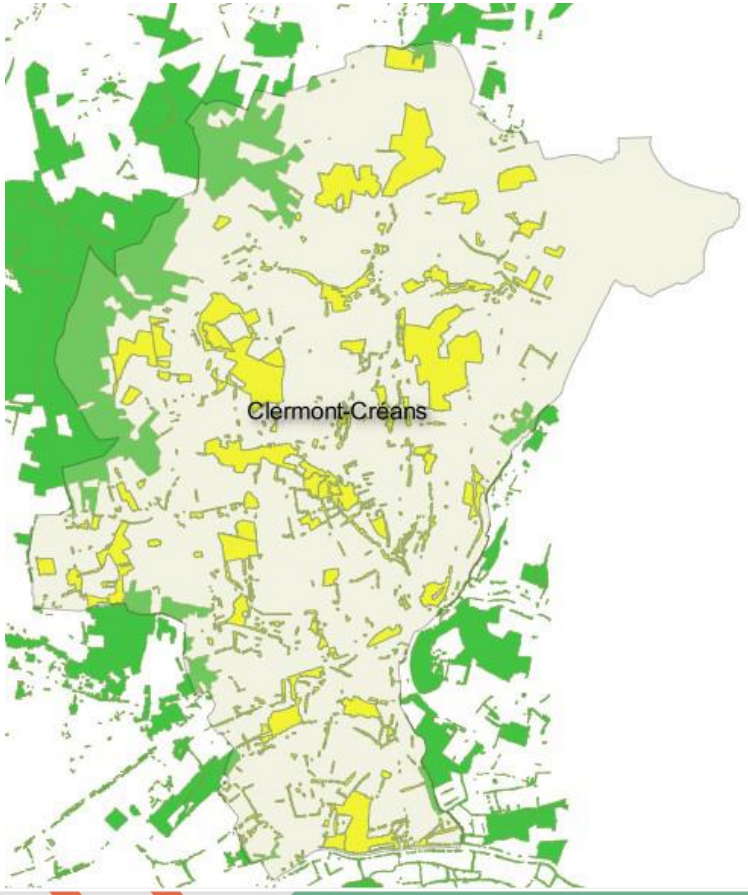
Syntaxe SQL

Jointures géographiques : ST_Within

A l'intérieur de

Les objets de la couche végétation sélectionnés (polygones jaunes) sont contenus entièrement dans le polygone (grisé) de la commune

ST_Within(geometry A, geometry B)



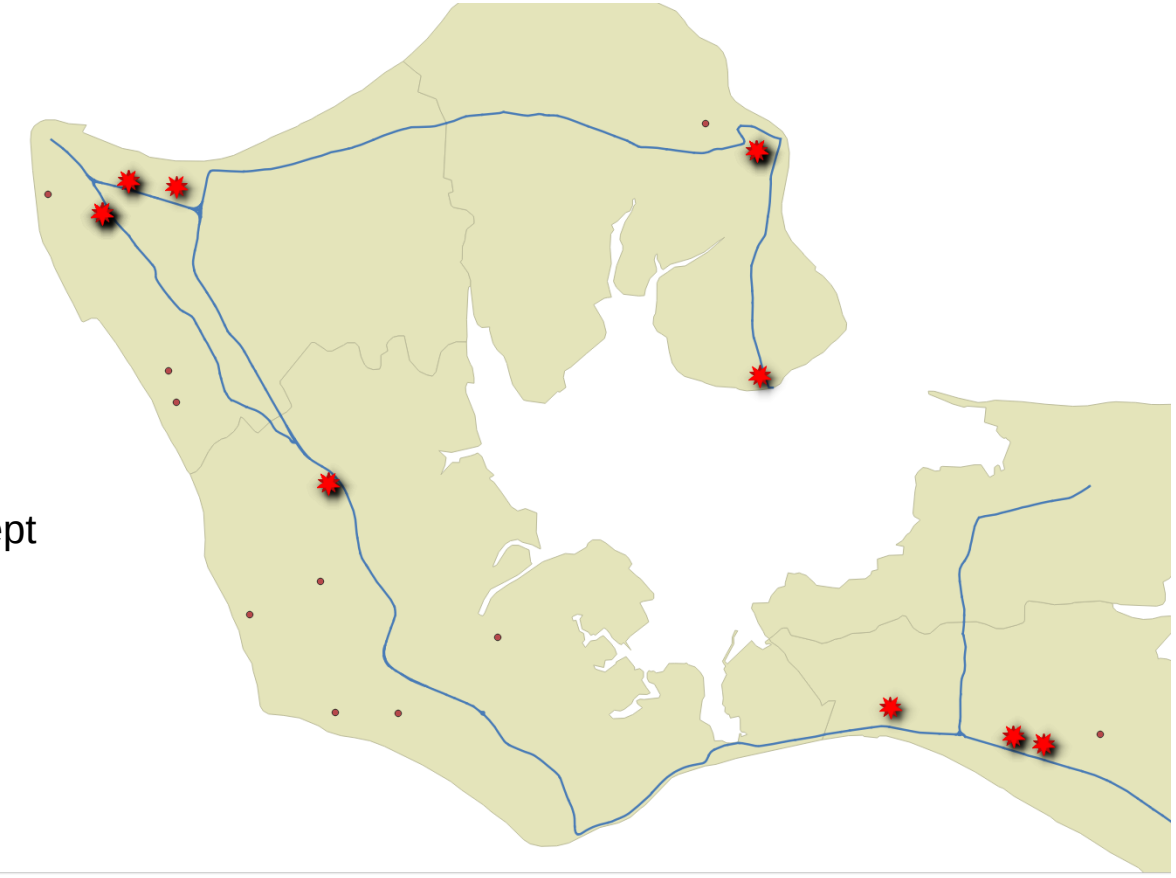
Secrétariat général

Syntaxe SQL

Jointures géographiques : ST_DWithin

A une distance de moins de

Les objets de la hébergement se trouvent à moins de 200m des objets de la couche des routes



```
SELECT hebergement_loisir_p.geom as geom
FROM formation.hebergement_loisir_" , formation.routes_dept
WHERE st_dwithin("hebergement_loisir_p".geom,
routes_dept.geom,200)
```

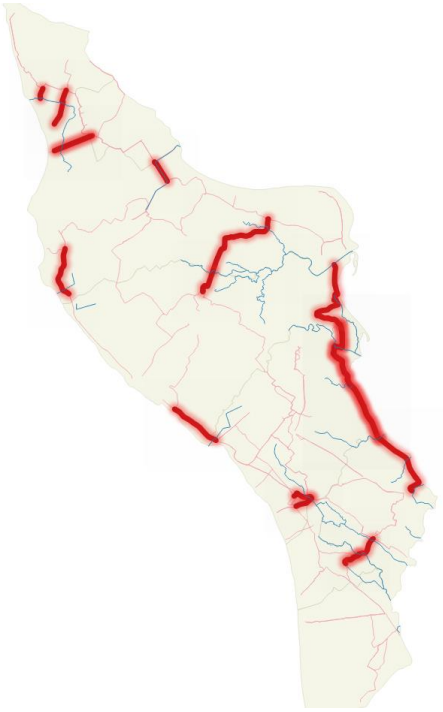
Secrétariat général

Exercice 22

Jointures géographiques

Sélectionner les pistes cyclables qui traversent un cours d'eau :

- Tables :
  - ✓ cours_d_eau
  - ✓ pistes_cyclables



Secrétariat général

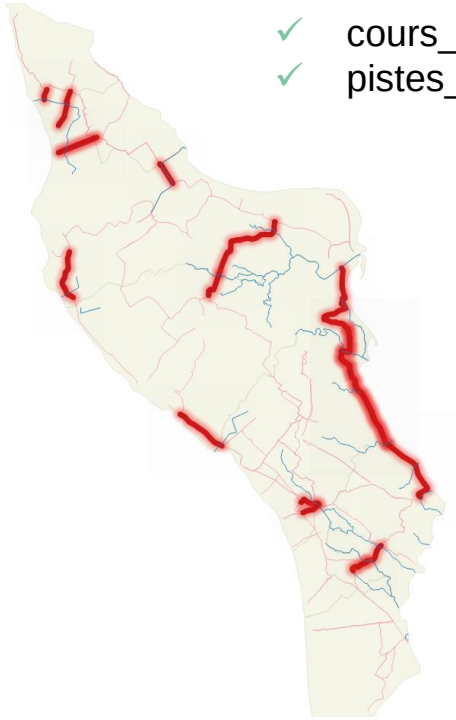
Correction de l'exercice

Exercice 22

Jointures géographiques

Sélectionner les pistes cyclables qui traversent un cours d'eau :

- Tables :
  - ✓ cours_d_eau
  - ✓ pistes_cyclables



```
SELECT
   pc.*
FROM
   formation.pistes_cyclables pc,
   formation.cours_d_eau ce
WHERE
   st_intersects(pc.geom, ce.geom)
```

```
SELECT
   pc.*
FROM
   pistes_cyclables pc,
   cours_d_eau ce
WHERE
   st_intersects(pc.geom, ce.geom)
```

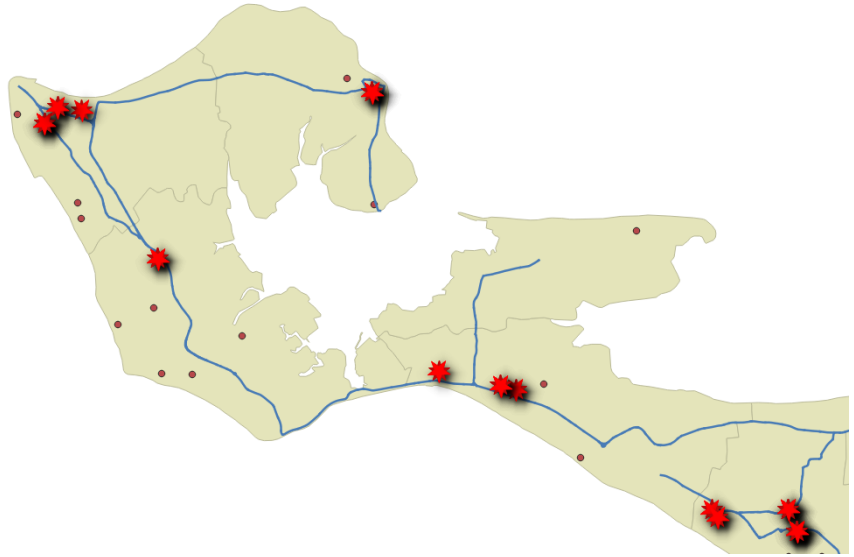
Secrétariat général

Exercice 23

Jointures géographiques

- Sélectionner les hébergements (points) qui sont des « village de vacances » ou des « campings …. » et qui se trouvent à une distance de moins de 200m d'une route départementale
  - Tables (attributs)
    - ✓ hebergement_loisir_p (typestruct)
    - ✓ routes_dept

- Réflexion autour du filtrage de la couche des hébergements :
  - ✓ Avantages et inconvénients de faire cette requêtes en plusieurs étapes
  - ✓ Quelles différences entre « Utiliser une vue » et « Utiliser un filtre QGIS » avant de faire la jointure



Secrétariat général

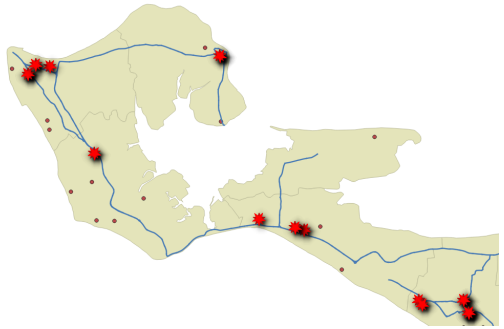
Correction de l'exercice

Exercice 23

Jointures géographiques

```
SELECT
  h.*
FROM
  hebergement_loisir_p h,
  routes_dept r
WHERE
  st_distance(h.geom, r.geom) < 200
  AND
    (typestruct ='village de vacances'
    OR typestruct like 'camping%')
```

- hébergements qui sont des « villages de vacances » ou des « campings …. » à de moins de 200m d'une route départementale
  - Tables (attributs)
    - ✓ hebergement_loisir_p (typestruct)
    - ✓ routes_dept



- Réflexion :
  - ✓ Avantages et inconvénients de faire cette requêtes en plusieurs étapes : plus facile
  - ✓ Le filtre QGIS n'est pas pris en compte dans la requête
    celle-ci se fait sur la table source

```
SELECT
    h.*
FROM
    formation.hebergement_loisir_p h,
    formation.routes_dept r
WHERE
    St_dwithin(h.geom,r.geom,200)
AND
    (typestruct='village de vacances'
     OR
     typestruct ilike 'Camping%')
```

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N

Tables sources

tab1

A	a
B	b
C	c

tab2

A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Dans le cas d'une jointure attributaire la relation est de type **1 à N** si un objet de la table 1 est mis en correspondance avec plusieurs objets de la table 2

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N

Tables sources

tab1

A	a
B	b
C	c

tab2

A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Dans le cas d'une jointure attributaire la relation est de type **1 à N** si un objet de la table 1 est mis en correspondance avec plusieurs objets de la table 2

Dans le cas d'une jointure géographique la relation est de type **1 à N** si un objet de la table 1 est mis en relation par l'opérateur géographique avec plusieurs objets de la table 2

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N

Tables sources

tab1

A	a
B	b
C	c

tab2

A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Résultat jointure

A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Exploitation dans QGIS

QGIS impose de disposer d'une colonne d'identification unique pour pouvoir afficher le résultat.

2 possibilités

Dans le cas d'une jointure attributaire la relation est de type **1 à N** si un objet de la table 1 est mis en correspondance avec plusieurs objets de la table 2

Dans le cas d'une jointure géographique la relation est de type **1 à N** si un objet de la table 1 est mis en relation par l'opérateur géographique avec plusieurs objets de la table 2

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N

Tables sources

tab1

A	a
B	b
C	c

tab2

A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Résultat jointure

A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Dans le cas d'une jointure attributaire la relation est de type **1 à N** si un objet de la table 1 est mis en correspondance avec plusieurs objets de la table 2

Dans le cas d'une jointure géographique la relation est de type **1 à N** si un objet de la table 1 est mis en relation par l'opérateur géographique avec plusieurs objets de la table 2

Exploitation dans QGIS

QGIS impose de disposer d'une colonne d'identification unique pour pouvoir afficher le résultat.

2 possibilités

- *Distinct on (col)*

A	a	A	?
B	b	B	1
C	c	C	?

- *Row_number() Over()*

1	A	a	A	1
2	A	a	A	2
3	B	b	B	1
4	C	c	C	1
5	C	c	C	2
6	C	c	C	3

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **DISTINCT ON**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

*Résultat jointure **sans** Distinct On*

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Si rien n'est spécifié, alors le regroupement ne gardera que la première valeur correspondante de la table jointe

```
SELECT DISTINCT ON (col_1)
      col_1,
      col_2,
      col_x,
      col_y
FROM tab1, tab2
WHERE tab1.lien1 = tab2.lien2
```

*Résultat jointure **avec** Distinct On*

A	a	A	1
B	b	B	1
C	c	C	1

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **GROUP BY**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

*Résultat jointure **sans** Group By*

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Remarque : une colonne géométrique est une colonne classique

*Il faut utiliser les **fonctions d'agrégations** pour gérer le contenu des colonnes et donc utiliser la clause **GROUP BY**.*

Les colonnes sans fonction d'agrégation sont citées obligatoirement dans le group by

```
SELECT
      col_1,
      col_2,
      string_agg(col_y , ' - '),
      max(col_y)
FROM tab1, tab2
WHERE tab1.lien1 = tab2.lien2
GROUP BY col_1,col_2
```

*Résultat jointure **avec** Group By*

A	a	1 - 2	2
B	b	1	1
C	c	1 - 2 - 3	3

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **GROUP BY et clé primaire**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

*Résultat jointure **sans** Group By*

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Remarque : une colonne géométrique est une colonne classique

Si tab1 possède une clé primaire (notion qui sera vue par suite), on peut se passer de fonction d'agrégation sur les colonnes de cette table.

```
SELECT
      col_1,
      col_2,
      string_agg(col_y , ' - '),
      max(col_y)
FROM tab1, tab2
WHERE tab1.lien1 = tab2.lien2
GROUP BY col_1
```

*Résultat jointure **avec** Group By*

A	a	1 - 2	2
B	b	1	1
C	c	1 - 2 - 3	3

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **Row_number() Over()**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Résultat jointure

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

Dans le cas le résultat de la jointure les ligne de la table 1 sont **dupliquées N** fois si un objet de la table 1 est mis en correspondance avec N plusieurs objets de la table 2 **(y compris les objets géométriques)**

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **Row_number() Over()**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Résultat jointure

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

```
SELECT
      Row_number () over() as id_qgis
      col_1,
      col_2,
      col_x,
      col_y
FROM tab1, tab2
WHERE tab1.lien1 = tab2.lien2
```

Dans le cas le résultat de la jointure les ligne de la table 1 sont **dupliquées N** fois si un objet de la table 1 est mis en correspondance avec N plusieurs objets de la table 2 **(y compris les objets géométriques)**

Citer **Row_number () over()** dans la clause SELECT permet de générer une colonne d'identification unique (numérotation automatique).

Secrétariat général

Syntaxe SQL

Jointures cas des relations 1 à N : **Row_number() Over()**

Tables sources

tab1

col_1	col_2
A	a
B	b
C	c

tab2

col_x	col_y
A	1
A	2
B	1
C	1
C	2
C	3

Jointure

1 à n

(inner, left, right,…)

Résultat jointure

col_1	col_2	col_x	col_y
A	a	A	1
A	a	A	2
B	b	B	1
C	c	C	1
C	c	C	2
C	c	C	3

```
SELECT
      Row_number () over() as id_qgis
      col_1,
      col_2,
      col_x,
      col_y
FROM tab1, tab2
WHERE tab1.lien1 = tab2.lien2
```

Dans le cas le résultat de la jointure les ligne de la table 1 sont **dupliquées N** fois si un objet de la table 1 est mis en correspondance avec N plusieurs objets de la table 2 **(y compris les objets géométriques)**

Citer **Row_number () over()** dans la clause SELECT permet de générer une colonne d'identification unique (numérotation automatique).

1	A	a	A	1
2	A	a	A	2
3	B	b	B	1
4	C	c	C	1
5	C	c	C	2
6	C	c	C	3

Exercice 24

Jointures géographiques

A partir des tables ocs_re_2015 et communes, sélectionner les « Marais » (attribut lib15niv3) qui sont sur les communes (tables communes attribut comm_ma) de :

- RIVEDOUX-PLAGE
- LOIX
- ARS-EN-RE

Afficher le résultat dans QGIS

Attention un marais peux être sur plusieurs communes

Secrétariat général

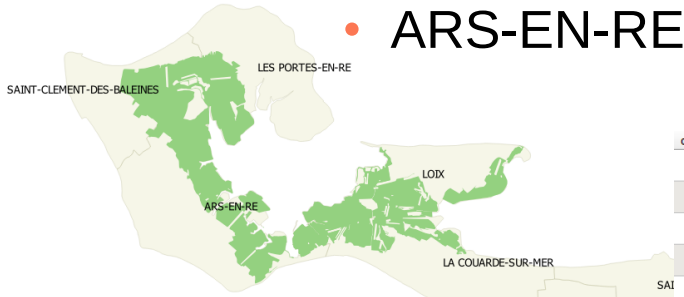
Correction de l'exercice

Exercice 24

Jointures géographiques

A partir des tables ocs_re_2015 et communes, sélectionner les « Marais » (attribut lib15niv3) qui sont sur les communes (tables communes attribut comm_ma) de :

- RIVEDOUX-PLAGE
- LOIX
- ARS-EN-RE



ogc_fid	code15niv3	lib15niv3	surf_ha	surf_m2	liste_com
2739	411	Marais intérieurs	1,83700039503961	18370,0039503962	RIVEDOUX-PLAGE
2740	411	Marais intérieurs	1,99728087251501	19972,8087251501	RIVEDOUX-PLAGE
2742	421	Marais maritimes	0,036805615378...	368,05615378368	ARS-EN-RE
2743	421	Marais maritimes	3,62618978103394	36261,8978103394	LOIX
2745	421	Marais maritimes	0,008481954000...	84,8195400035784	ARS-EN-RE
2753	421	Marais maritimes	6,389543880972	63895,43880972	LOIX
2754	421	Marais maritimes	63,7907653636759	637907,653636759	LOIX
2755	421	Marais maritimes	121,525778362168	1215257,78362168	ARS-EN-RE
2756	421	Marais maritimes	200,3764498387	2003764,498387	LOIX
2757	421	Marais maritimes	1,36301605353624	13630,1605353624	ARS-EN-RE
2758	421	Marais maritimes	50,5654978468862	505654,978468862	ARS-EN-RE
2759	421	Marais maritimes	220,464443488144	2204644,43488144	LOIX / ARS-EN-RE
2760	421	Marais maritimes	599,170166585928	5991701,66585928	ARS-EN-RE

Avec clé primaire

```sql
select
    o.ogc_fid,
    o.code15niv3,
    o.lib15niv3,
    o.surf_ha,
    o.surf_m2,
    string_agg(c.comm_ma ,'/' ),
    o.geom
FROM
    formation.ocs_re_2015 as o
    JOIN
    formation.communes as c
    ON
    st_intersects(o.geom, c.geom)
WHERE
    o.lib15niv3 like '%Marais%' AND
    c.comm_ma in ('RIVEDOUX-PLAGE','LOIX','ARS-EN-RE')
group by
    o.ogc_fid
```

Secrétariat général

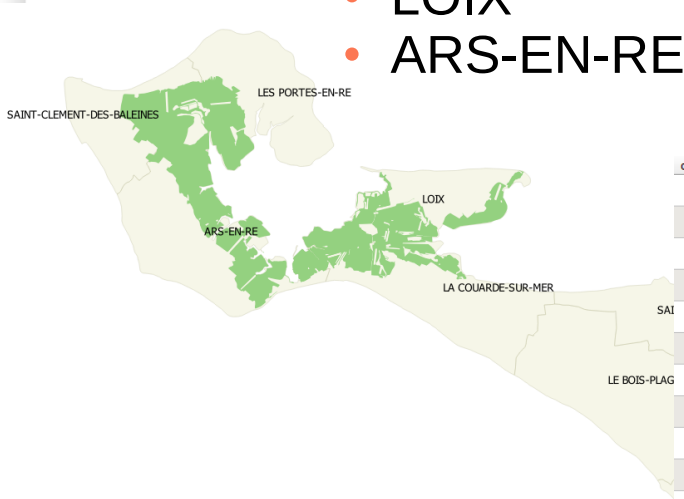
Correction de l'exercice

Exercice 24

Jointures géographiques

A partir des tables ocs_re_2015 et communes, sélectionner les « Marais » (attribut lib15niv3) qui sont sur les communes (tables communes attribut comm_ma) de :

- RIVEDOUX-PLAGE
- LOIX
- ARS-EN-RE



ogc_fid	code15niv3	lib15niv3	surf_ha	surf_m2	liste_com
2739	411	Marais intérieurs	1,83700039503961	18370,0039503962	RIVEDOUX-PLAGE
2740	411	Marais intérieurs	1,99728087251501	19972,8087251501	RIVEDOUX-PLAGE
2742	421	Marais maritimes	0,036805615378...	368,05615378368	ARS-EN-RE
2743	421	Marais maritimes	3,62618978103394	36261,8978103394	LOIX
2745	421	Marais maritimes	0,008481954000...	84,8195400035784	ARS-EN-RE
2753	421	Marais maritimes	6,389543880972	63895,43880972	LOIX
2754	421	Marais maritimes	63,7907653636759	637907,653636759	LOIX
2755	421	Marais maritimes	121,525778362168	1215257,78362168	ARS-EN-RE
2756	421	Marais maritimes	200,3764498387	2003764,498387	LOIX
2757	421	Marais maritimes	1,36301605353624	13630,1605353624	ARS-EN-RE
2758	421	Marais maritimes	50,5654978468862	505654,978468862	ARS-EN-RE
2759	421	Marais maritimes	220,464443488144	2204644,43488144	LOIX / ARS-EN-RE
2760	421	Marais maritimes	599,170166585928	5991701,66585928	ARS-EN-RE

Sans clé primaire

```sql
select
    o.ogc_fid,
    o.code15niv3,
    o.lib15niv3,
    o.surf_ha,
    o.surf_m2,
    string_agg(c.comm_ma ,'/' ),
    o.geom
FROM
    formation.ocs_re_2015 as o
    JOIN
    formation.communes as c
    ON
    st_intersects(o.geom, c.geom)
WHERE
    o.lib15niv3 like '%Marais%' AND
    c.comm_ma in ('RIVEDOUX-PLAGE','LOIX','ARS-EN-RE')
group by
    o.ogc_fid,o.ogc_fid,o.code15niv3,
    o.lib15niv3,o.surf_ha,o.surf_m2,o.geom
```

Secrétariat général

LES FONCTIONS GÉOGRAPHIQUES

Secrétariat général

Syntaxe SQL

Fonctions de traitements géographiques

Les mesures

x	x	St_area(geom)	Retourne l'aire d'une géométrie dans l'unité du système de référence spatiale	*Num*
x	x	St_length(geom)	Retourne la longueur de la géométrie dans l'unité du système de référence spatiale.	*Num*
x	x	St_perimeter(geom)	Retourne le périmètre de la géométrie dans l'unité du système de référence spatiale.	*Num*
x	x	St_nrings(geom)	renvoie le nombre d'anneaux d'une géométrie polygon	*Num*
x	x	St_npoints(geom)	Retourne le nombre de sommets d'une géométrie	*Num*
x	x	St_distance(geom1 , geom2)	retourne la distance cartésienne en 2 dimensions minimum entre deux géométries	*Num*

Secrétariat général

Geometry type :
1 POINT
2 LINE
3 POLYGON
4 MULTIPOINT
5 MULTILINES
6 MULTIPOLYGON

Syntaxe SQL

Fonctions de traitements géographiques

Les objets

x	x	St_Centroid(geom)	Retourne le centroid	*objet*
x	x	St_X(geom)	Retourne la coordonnée X d'un objet POINT dans l'unité du système de référence spatiale	*Num*
x	x	St_Y(geom)	Retourne la coordonnée Y d'un objet POINT dans l'unité du système de référence spatiale	*Num*
x	x	St_PointOnSurface(geom)	retourne un point qui est obligatoirement dans l'entité.	*objet*
x	x	St_SetSRID(geom,srid)	Met à jour le SRID d'une géométrie	*objet*
x	x	St_Buffer(geom, distance)	Retourne une zone tampon dont le contour est à une distance donnée de la géométrie d'origine	*objet*
x	x	St_Union(geom)	Fusionne les objets lors de l'utilisation d'un Group By	*objet*
x	x	Geometry(type, epsg)	Déclare une colonne géographique(Point,Linestring,Polygon)	*décl.*

Secrétariat général

Exercice 25

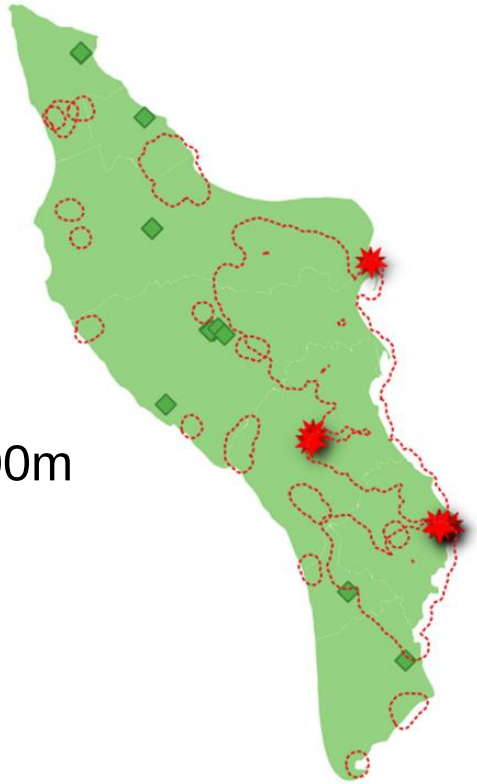
Utiliser des fonctions géographiques

Sélectionner les établissements scolaires qui sont à moins de 300 d'un marais et les afficher.

- etab_scolaires
- ocs_oleron_2015 attribut lib15niv3

Afficher dans QGIS la Carte avec :

- Les limites des Communes
- Requête n°1
  - Les établissements scolaires étant dans le périmètres des 300m et afficher dans le résultat la distance entre l'établissement scolaire et le marais)
- Requête n°2
  - Limites des 300m autour du marais



Secrétariat général

Correction de l'exercice

Exercice 25

Utiliser des fonctions géographiques PostgreSQL

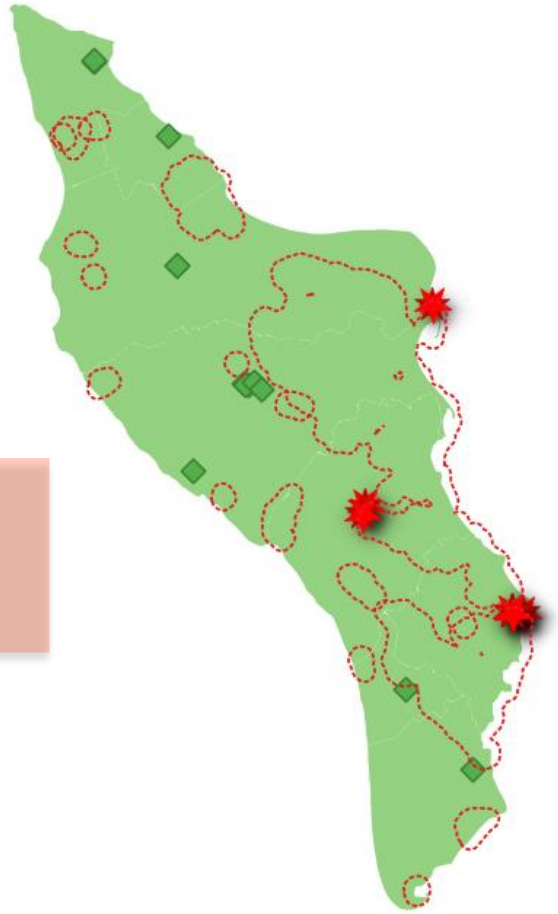
Calculer la distance entre l'établissement et le marais

```
SELECT
    e.*,
    st_distance(m.geom,e.geom) as "Distance Marais"
FROM
    formation.ocs_oleron_2015 as m,
    formation.etab_scolaires as e
WHERE
    m.lib15niv3 like '%Marais%' AND st_dwithin(e.geom,m.geom,300)
```

Jointure entre les marais et les établissements

Fusionner les objets

```
select
lib15niv3,
ST_union(st_buffer(geom, 300)) as geom
from
   formation.ocs_oleron_2015 as m
where
   m.lib15niv3 like '%Marais%'
   Group by lib15niv3
```



Secrétariat général

Correction de l'exercice

Exercice 25

Utiliser des fonctions géographiques Spatialite

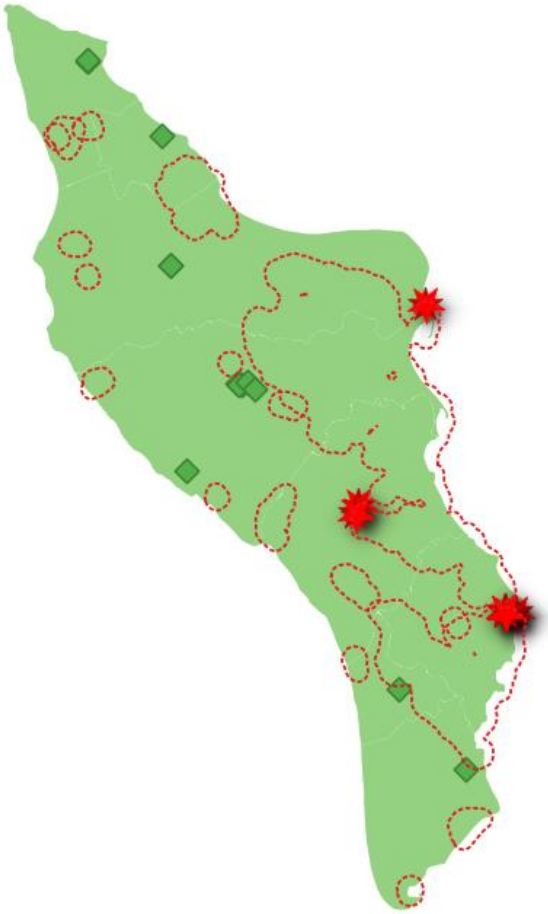
Calculer la distance entre l'établissement et le marais

```
SELECT
  e.*,
  st_distance(m.geom, e.geom) as "Distance Marais"
FROM
  ocs_oleron_2015 m,
  etab_scolaires e
WHERE
  m.lib15niv3 like 'Marais%'
  AND
  st_distance(e.geom,m.geom) < 300
```

Jointure entre les marais et les établissements

Fusionner les objets

```
SELECT
  lib15niv3,
  st_union(st_buffer(geom,300)) as geom
FROM
  ocs_oleron_2015
WHERE
  lib15niv3 like 'Marais%'
GROUP BY
  lib15niv3
```



Secrétariat général

EXPLOITATION DE LA BASE DE DONNÉES

- Qu'est-ce que le SQL ?
- La sélection d'objet : SELECT
- La création de vues
- Les jointures entre tables
- Les fonctions géographiques
- Création d'une table

Secrétariat général

Syntaxe SQL

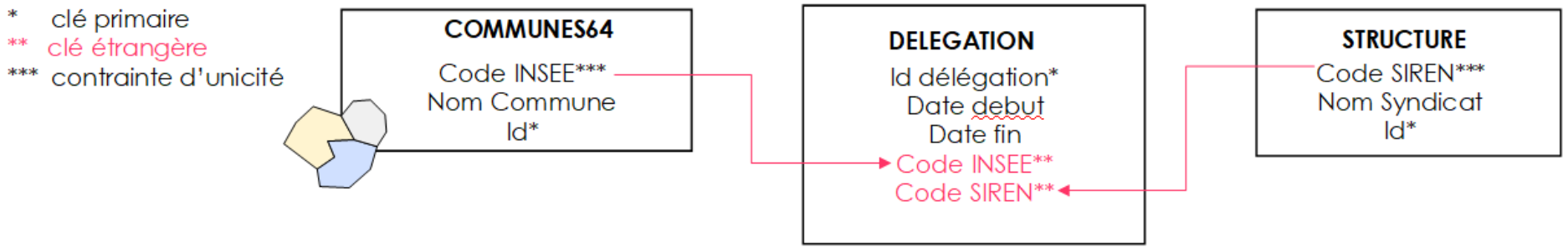
PgAdmin4 : création d'une table CONTRAINTES

Clé primaire (pk) – Clé étrangère (fk) :

Clé primaire :

- identifiant unique d'un enregistrement dans une table (de préférence Integer ou serial).
- Permet de construire l'intégrité référentielle d'une base de donnée

Clé étrangère: référence d'un enregistrement d'une autre table (clé primaire d'une autre table)



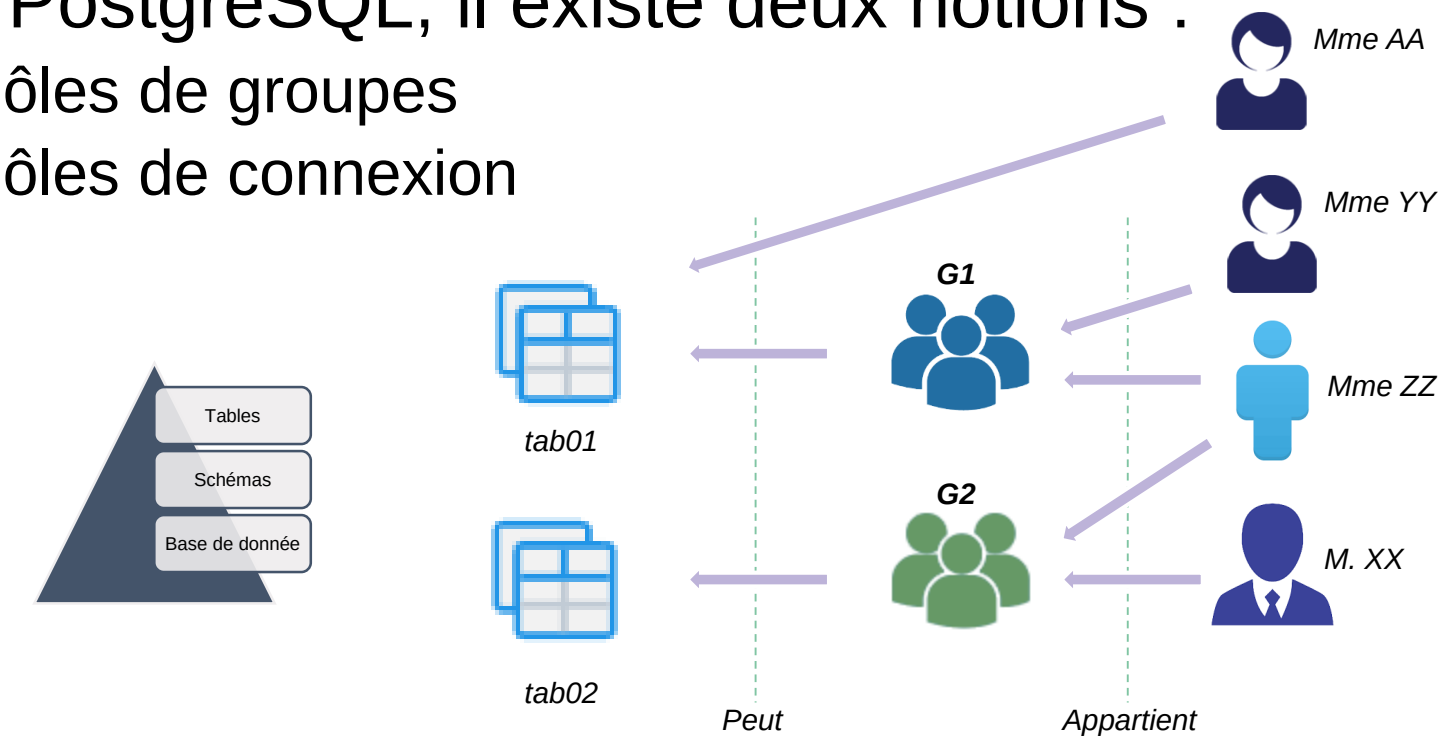
Secrétariat général

Syntaxe SQL

PgAdmin4 : création d'une table SÉCURITÉ

Dans PostgreSQL, il existe deux notions :

- Rôles de groupes
- Rôles de connexion



La création des rôles de groupe et des rôles de connexion dépend de l'administrateur du serveur ou de l'administrateur de la base de donnée (si autorisation). Cette notion n'est pas abordée dans cette formation.

Secrétariat général

Syntaxe SQL

création d'une table attributaire

Principe de rédaction :

```
CREATE TABLE schema.table
   ( id serial,
     champ1 character(15),
     champ2 integer,
     CONSTRAINT pk_id PRIMARY KEY (id) )
   ;
```

Nom et emplacement de la table

Début du bloc

Structure de la table

fin du bloc

Fin d'une instruction SQL

```
GRANT ALL ON TABLE schema.table TO role1;
GRANT SELECT ON TABLE schema.table TO role2 ;
```

Affectation des droits

Secrétariat général

Syntaxe SQL

création d'une table géographique

Principe de rédaction postgresql :

```
CREATE TABLE schema.table
   ( id serial NOT NULL,
     champ1 character (15),
     champ2 integer,
     geom geometry(Point,2154),
     CONSTRAINT matable_geo_pkey PRIMARY KEY (id) )
;

GRANT ALL ON TABLE schema.table TO role1;
GRANT SELECT ON TABLE schema.table TO role2;
```

Définition d'un attribut de type géométrique

Principe de rédaction spatialite :

```
CREATE TABLE table
   ( id serial NOT NULL,
     champ1 character (15),
     champ2 integer,
     CONSTRAINT matable_geo_pkey PRIMARY KEY (id) ) ;

SELECT AddgeometryColumn('table','geom',2154,'polygon',2)
```

Définition d'un attribut de type géométrique

Nom de la table

Intitulé de la colonne géométrique

Projection

Type de géométrie

2D

Secrétariat général

Syntaxe SQL

création d'une table à partir d'une table

Il est possible de créer une table à partir d'une autre table. La table créée aura la même structure que la table source (résultat du SELECT) et contiendra les enregistrements de la table source.

Principe de rédaction postgresql :

CREATE TABLE schema.matable **AS SELECT** …

Exemple : création de la table etab_scol2 dans le schéma formation à partir de la table etab_scolaires

```
CREATE TABLE
  formation.etabscol2
AS
  SELECT * FROM formation.etab_scolaires
```

Ou si seulement besoin de récupérer la structure

```
CREATE TABLE
  formation.etabscol2
AS
  SELECT * FROM formation.etab_scolaires LIMIT 0
```

Principe de rédaction spatialite :

CREATE TABLE matable **AS SELECT** …

INSERT INTO

geometry_Columns (f_table_name, f_geometry_column, geometry type, coord_dimension,srdid,spatial_index_enabled)

VALUES

(nom_de_la_table,
intitule_colonne_geometrique,
type_de_geometrie,
projection,
index)

Secrétariat
général

LA GESTION DES TABLES

Uniquement abordé dans l'environnement postgreSQL

Secrétariat général

Syntaxe SQL

L'instruction ALTER

ALTER TABLE permet de modifier la définition d'une table.

Syntaxe : **ALTER TABLE** nom_table **(+instruction)**

Pour ajouter une nouvelle colonne :

ALTER TABLE nom_table **ADD** nom_colonne datatype

Pour supprimer une colonne :

ALTER TABLE nom_table **DROP COLUMN** nom_colonne

Exemple 1 : Ajouter une colonne « nb_hab » à la table commune

ATLER TABLE commune **ADD** nb_hab integer

Exemple 2 : Supprimer la colonne « nb_hab » à la table commune

ATLER TABLE commune **DROP COLUMN** nb_hab

Secrétariat général

Syntaxe SQL

L'instruction ALTER dans PgAdmin

ALTER TABLE …ALTER COLUMN permet la modification du type de la colonne :

- sans l'instruction **USING**.

ALTER TABLE nom_table **ALTER COLUMN** nom_colonne **TYPE** datatype

- avec l'instruction **USING**. La clause optionnelle USING précise comment calculer la nouvelle valeur de la colonne à partir de l'ancienne

ALTER TABLE nom_table **ALTER COLUMN** nom_colonne **TYPE** datatype **USING** …..

Secrétariat général

Syntaxe SQL

L'instruction ALTER dans PgAdmin

Quelques exemples : **ALTER TABLE** nom_table **ALTER COLUMN** nom_colonne **TYPE** datatype **USING** …..

- Changer la géométrie d'une colonne géographique de POINT en POLYGON

```
ALTER TABLE formation.etab_scolaires ALTER COLUMN geom TYPE geometry(POLYGON,2154) USING st_Buffer(geom,300);
```

La géométrie source est de type POINT

On veut une géométrie de type POLYGON

Il faut transformer les points en polygones

- Changer un attribut numérique en un attribut sur 4 caractères et précéder par des « 0 »

```
ALTER TABLE formation.etab_scolaire ALTER COLUMN effec14_15 TYPE VARCHAR(4) USING lpad(effec14_15::varchar,4,'0');
```

effec14_15 bigint
116
93
46
44
67
140

effec14_15 character varying (4)
0116
0093
0046
0044
0067
0140

Compléter avec des « 0 »

Transcryptage obligatoire car lpad est une fonction chaine de caractère

Secrétariat général

Syntaxe SQL

L'instruction ALTER dans PgAdmin

ALTER TABLE …ADD CONSTRAINT permet la création d'une clé primaire.

La syntaxe pour ajouter une clé primaire est la suivante:

ALTER TABLE nom_table **ADD CONSTRAINT** **nom_contrainte** **PRIMARY KEY** **(valeur_clé)**

Nom de la clé primaire à créer

Attribut servant de clé primaire. Cet attribut doit être obligatoirement renseigné et sans doublon

Ou

ALTER TABLE nom_table **ADD PRIMARY KEY** **(valeur_clé)**

Pour alimenter une table depuis QGIS celle-ci doit avoir au moins une clé primaire déclarée

Secrétariat général

Syntaxe SQL

L'instruction DROP et RENAME

DROP permet de supprimer des objets
DROP TABLE supprime la tables de la base de données

- Syntaxe : **DROP TABLE** schema.table

Seul son propriétaire (ou un superuser) peut détruire une table
Dans QGIS GestionnaireDB, clic droit sur une table → Effacer

RENAME Permet de renommer des objets

- Syntaxe : **ALTER TABLE** schema.tablesource **RENAME TO** nouveau_nom

Dans QGIS GestionnaireDB, clic droit sur une table renommer.

Secrétariat général

Exercice 26

Modifier la structure d'une table

Dans le schéma qui vous est assigné, Dupliquer la table cours_d_eau en une nouvelle table cours_d_eau_v2 dans le même schéma, puis ajouter la clef primaire ogc_fid.

- Supprimer les attributs :
  - date_destr
  - date_app
  - date_conf
- Ajouter un attribut
  - longueur type float

Correction de l'exercice

Exercice 26

Modifier la structure d'une table

Dans le schéma qui vous est assigné, table cours_d_eau :

- Dupliquer la table cours_d_eau et ajouter ogc_fid comme clef primaire
  - CREATE TABLE formation.cours_d_eau_v2 as (SELECT * FROM formations.cours_d_eau)
  - ALTER TABLE formation.cours_d_eau_v2 ADD PRIMARY KEY (ogc_fid)
- Supprimer les attributs :
  - date_destr **ALTER TABLE** formation.cours_d_eau_v2 **DROP COLUMN** date_destr
  - date_app **ALTER TABLE** formation.cours_d_eau_v2 **DROP COLUMN** date_app
  - date_conf **ALTER TABLE** formation.cours_d_eau_v2 **DROP COLUMN** date_conf
- Ajouter un attribut
  - longueur type integer **ALTER TABLE** formation.cours_d_eau_v2 **ADD** longueur integer

Secrétariat
général

INSERTION D'ENREGISTREMENTS

Uniquement abordé dans l'environnement postgreSQL

Secrétariat général

Syntaxe SQL

L'instruction INSERT et DELETE

L'instruction INSERT permet d'ajouter (si l'on dispose des droits) un ou des nouveaux enregistrements dans une table :

Principes de rédactions :

- d'insertion d'une ligne à la fois, avec tous les attributs:

INSERT INTO <nom_table> **VALUES** ('valeur1', 'valeur2', …)

- d'insertion d'une ligne à la fois, en définissant les attributs :

INSERT INTO <nom_table> (col1, col2, ….) **VALUES** ('valeur1', 'valeur2', …)

Secrétariat général

Syntaxe SQL

L'instruction INSERT

L'instruction INSERT permet d'ajouter un ou des nouveaux enregistrements dans une table :

Principes de rédactions :

- d'insertion de plusieurs ligne à la fois

```
INSERT INTO client (prenom, nom, ville, age)  VALUES
        ('Arthur', 'Bupont', 'Pau',24) ,
        ('Alain', 'Boisjoli', 'La Rochelle',56),
        ('Martin','Pécheur', 'Bordeaux', 28)
```

Secrétariat général

Syntaxe SQL

L'instruction INSERT

L'instruction INSERT permet d'ajouter un ou des nouveaux enregistrements dans une table :

Principes de rédactions :

- d'insertion de plusieurs ligne à la fois à partir d'une autre table

INSERT INTO <nom_table> (valeur1, valeur2,valeur3)
SELECT (col1, col2, col3) **FROM** table where ….

Rédiger un SELECT restituant les attributs devant être insérés dans la table

Secrétariat général

Exercice 27

Insérer des enregistrements

Créer dans le schéma qui vous est assigné la table ocs_2015 qui regroupe les enregistrements des tables ocs_re_2015 et ocs_oleron_2015 :

- Etape 1 créer une table à partir de ocs_oleron_2015
- Etape 2 insérer les enregistrements de la table ocs_re_2015 dans la table nouvellement créée
- Etape 3 finaliser le paramétrage de la couche (clé primaire-droits,…)

Secrétariat général

Correction de l'exercice

Exercice 27

Insérer des enregistrements

ocs_2015

- Etape 1 créer à partir de ocs_oleron_2015

```
CREATE TABLE formation.ocs_2015 as SELECT * FROM formation.ocs_oleron_2015
```

- Etape 2 insérer ocs_re_2015

```
INSERT INTO formation.ocs_2015
    (geom, id, ogc_fid,
     code15niv1, lib15niv1,
     code15niv2, lib15niv2,
     code15niv3, lib15niv3,
     code15niv4, lib15niv4,
     surf_m2, surf_ha, perimetre)
SELECT
    geom, id, ogc_fid,
     code15niv1, lib15niv1,
     code15niv2, lib15niv2,
     code15niv3, lib15niv3,
     code15niv4, lib15niv4,
     surf_m2, surf_ha, perimetre
FROM formation.ocs_re_2015
```

Secrétariat général

Syntaxe SQL

L'instruction INSERT et DELETE

L'instruction DELETE permet de supprimer (si l'on dispose des droits) un ou des enregistrements dans une table :

Principes de rédactions :

- Suppression de tous les enregistrements d'une table

  DELETE FROM <nom_table>
- Suppression des enregistrements d'une table qui réponde à un critère

  DELETE FROM <nom_table> **WHERE** condition
- Suppression des enregistrements d'une table qui réponde à un critère lié à une autre table

  DELETE FROM <nom_table> **USING** <table2> **WHERE** jointure +condition

Exemple :

```
DELETE FROM films USING producteurs WHERE id_producteur = producteurs.id AND producteurs.nom = 'Eastwood'
```

Table dans laquelle les enregistrements seront suprimés

Table liée

Id du producteur de la table films

Id du producteur de la table producteurs

Condition

Secrétariat
général

MISE À JOURS DES DONNÉES

Uniquement abordé dans l'environnement postgreSQL

Secrétariat général

Syntaxe SQL

L'instruction UPDATE

L'instruction UPDATE permet de mettre à jour des données :

- Dans Pgadmin4 et GestionnaireDB au moyen d'une instruction SQL
- Directement avec les fonctionnalités de QGIS quand la couche est mise en mode édition

Très souvent cette commande est utilisée avec la clause WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

Secrétariat général

Syntaxe SQL

L'instruction UPDATE : PgAdmin4

Dans PgAdmin4 et GestionnaireDB :

Principe de rédaction (mise à jour d'une seule colonne sans condition) :

UPDATE schema.table **SET** colonne = Valeur

Table (ou vue) à modifier

Colonne concernée

Valeur à affecter, il peut s'agir d'une constante, d'une colonne, d'une formule

Secrétariat général

Syntaxe SQL

L'instruction UPDATE : PgAdmin4

Dans PgAdmin4 et GestionnaireDB :

Principe de rédaction (mise à jour de plusieurs colonnes sans condition) :

UPDATE schema.table **SET** colonne1 = Valeur1, colonne2 = Valeur2

- Table (ou vue) à modifier
- Première Colonne concernée
- Séparateur
- Seconde Colonne concernée

Secrétariat général

Syntaxe SQL

L'instruction UPDATE : PgAdmin4

Dans PgAdmin4 et GestionnaireDB :

Principe de rédaction (mise à jour de plusieurs colonnes **avec condition**) :

Rédaction de la condition

```
UPDATE schema.table SET col1 = Val1, col2 = Val2 WHERE condition
```

*Exemple : mise à jour d'un intitulé de l'ocs (remplacer Décharge par Décharge**s**)*

```
UPDATE formation.ocs_re_2015 SET lib15niv4 = 'Décharges' WHERE lib15niv4 = 'Décharge'
```

Ou encore

```
UPDATE formation.ocs_re_2015 SET lib15niv4 = lib15niv4||'s' WHERE lib15niv4 = 'Décharge'
```

Secrétariat général

Exercice 28

Mettre à jour des données : SQL

Dans PgAdmin4 mettre à jour la table **ocs_2015** précédemment créer en remplaçant l'intitulé **Décharge** de l'attribut **lib15niv4** par **Décharges**

Secrétariat général

Correction de l'exercice

Exercice 28

Mettre à jour des données : SQL

Dans PgAdmin4 mettre à jour la table **ocs_2015** précédemment créer en remplaçant l'intitulé **Décharge** de l'attribut **lib15niv4** par **Décharges**

```
UPDATE formation.ocs_re_2015 SET lib15niv4 = 'Décharges' WHERE lib15niv4 = 'Décharge'
```

Ou encore

```
UPDATE formation.ocs_re_2015 SET lib15niv4 = lib15niv4||'s' WHERE lib15niv4 = 'Décharge'
```

Secrétariat général

Syntaxe SQL

L'instruction BEGIN – COMMIT et ROLL BACK

BEGIN initie un bloc de transaction, c'est-à-dire que toutes les instructions apparaissant après la commande BEGIN sont exécutées dans une seule transaction jusqu'à ce qu'un COMMIT ou ROLLBACK

COMMIT enregistre

ROLLBACK annule

Syntaxe :

```
BEGIN ;
        Instruction (exemple UPDATE ….)
COMMIT;
```

Secrétariat général

Syntaxe SQL

L'instruction UPDATE : QGIS

Dans QGIS, la commande UPDATE (comme la commande INSERT) est encapsulé lors de l'édition de couche la couche

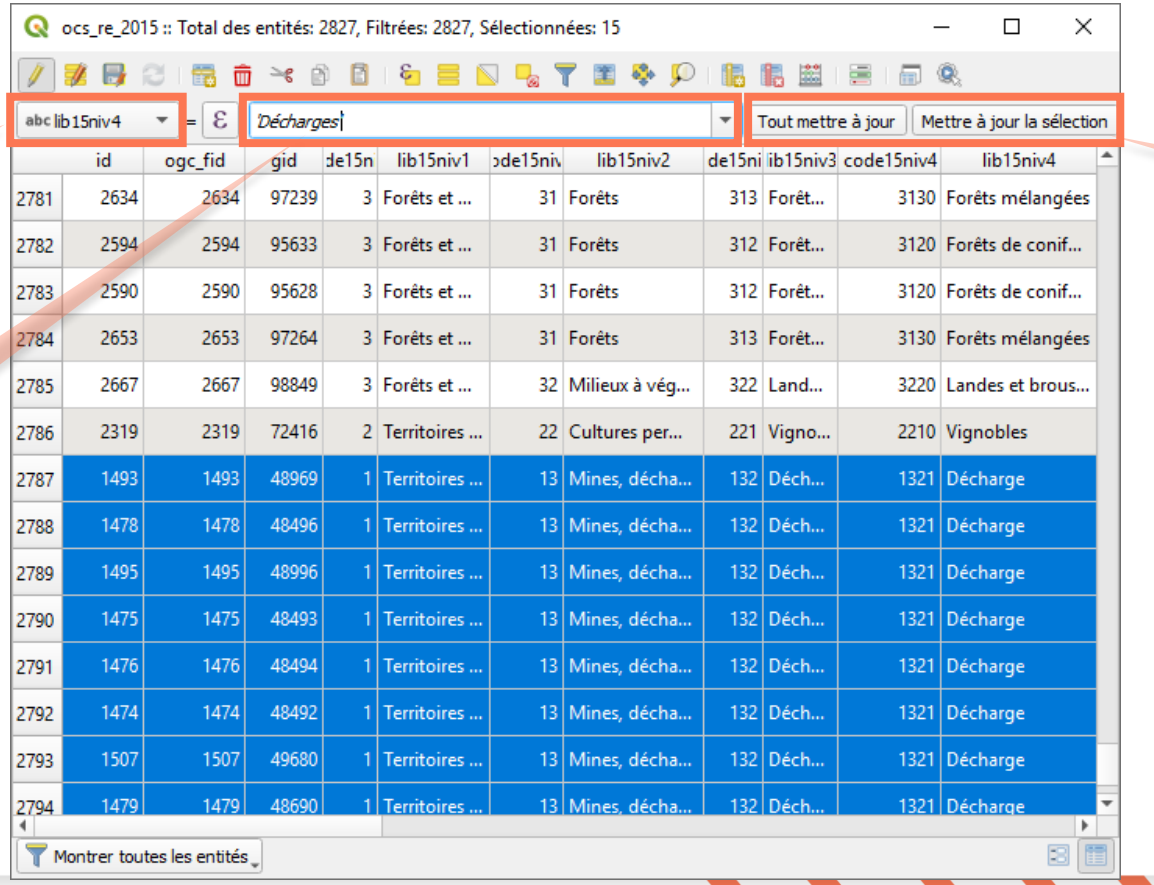
Il suffit donc de rendre une table postgreSQL éditable :

- Clic droit sur la couche et menu « basculer en mode édition ou cliquer sur l'icone 
- Modifier des objets :
  - ✓ Géométries
  - ✓ Attributs
- Enregistrer les modifications 

Secrétariat général

Syntaxe SQL

L'instruction UPDATE : QGIS



	id	ogc_fid	gid	de15n	lib15niv1	ode15niv	lib15niv2	de15ni	ib15niv3	code15niv4	lib15niv4
2781	2634	2634	97239	3	Forêts et ...	31	Forêts	313	Forêt...	3130	Forêts mélangées
2782	2594	2594	95633	3	Forêts et ...	31	Forêts	312	Forêt...	3120	Forêts de conif...
2783	2590	2590	95628	3	Forêts et ...	31	Forêts	312	Forêt...	3120	Forêts de conif...
2784	2653	2653	97264	3	Forêts et ...	31	Forêts	313	Forêt...	3130	Forêts mélangées
2785	2667	2667	98849	3	Forêts et ...	32	Milieux à vég...	322	Land...	3220	Landes et brous...
2786	2319	2319	72416	2	Territoires ...	22	Cultures per...	221	Vigno...	2210	Vignobles
2787	1493	1493	48969	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2788	1478	1478	48496	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2789	1495	1495	48996	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2790	1475	1475	48493	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2791	1476	1476	48494	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2792	1474	1474	48492	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2793	1507	1507	49680	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge
2794	1479	1479	48690	1	Territoires ...	13	Mines, décha...	132	Déch...	1321	Décharge

Attribut à mettre à jour

Nouvelle valeur

Choisir entre tous les enregistrements ou la sélection

Secrétariat général

Correction de l'exercice

Exercice 29

Mettre à jours des données : QGIS

Dans QGIS, mettre à jour l'attribut 'longueur' de la table cours_d_eau_v2 avec la longueur du tronçon (utiliser la fonction $lenght)