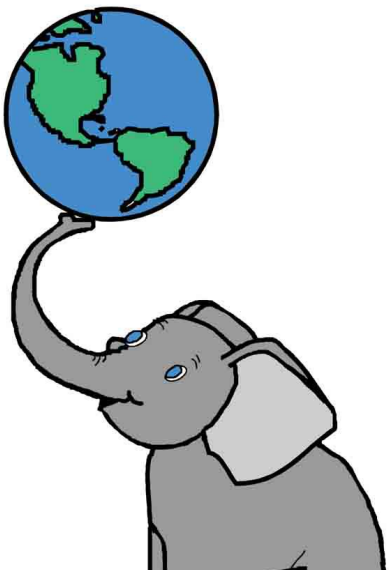
Gestion d'une base et de ses objets



Version 1.4

Ministère de la Transistion Ecologique
Licence ETALAB

Janvier 2022

Table des matières

Objectifs

Les objectifs du module sont de :

- savoir gérer les bases de données et leurs objets ;
- réaliser des imports de données ;
- réaliser des exports de données.

Introduction

Le temps d'apprentissage de ce module consacré à la gestion d'une base PostgreSQL/PostGIS et de ses objets est estimé à 4 heures.

Il comporte un exercice tutoré de création d'une base relationnelle et un exercice d'import et export auto-corrigé.

I

Gestion des bases, schémas et tables

A. Création d'une base "template" de données spatiale

La création d'une base de données n'est pas possible à partir de DBManager sous QGIS.

Nous avons abordé rapidement la création d'une base de données lors de *l'exercice guidé sur la gestion des droits* - p.37. Retenons que le propriétaire par défaut est le créateur de la base.

Complément

Collation : permet de préciser les fonctionnalités de régionalisation (ou localisation). Pour *en savoir plus*[1]...

Nous avons utilisé la fois précédente la possibilité de créer une base à partir d'une base *template* qui avait été créée préalablement *template_sig*. C'est effectivement une bonne pratique.

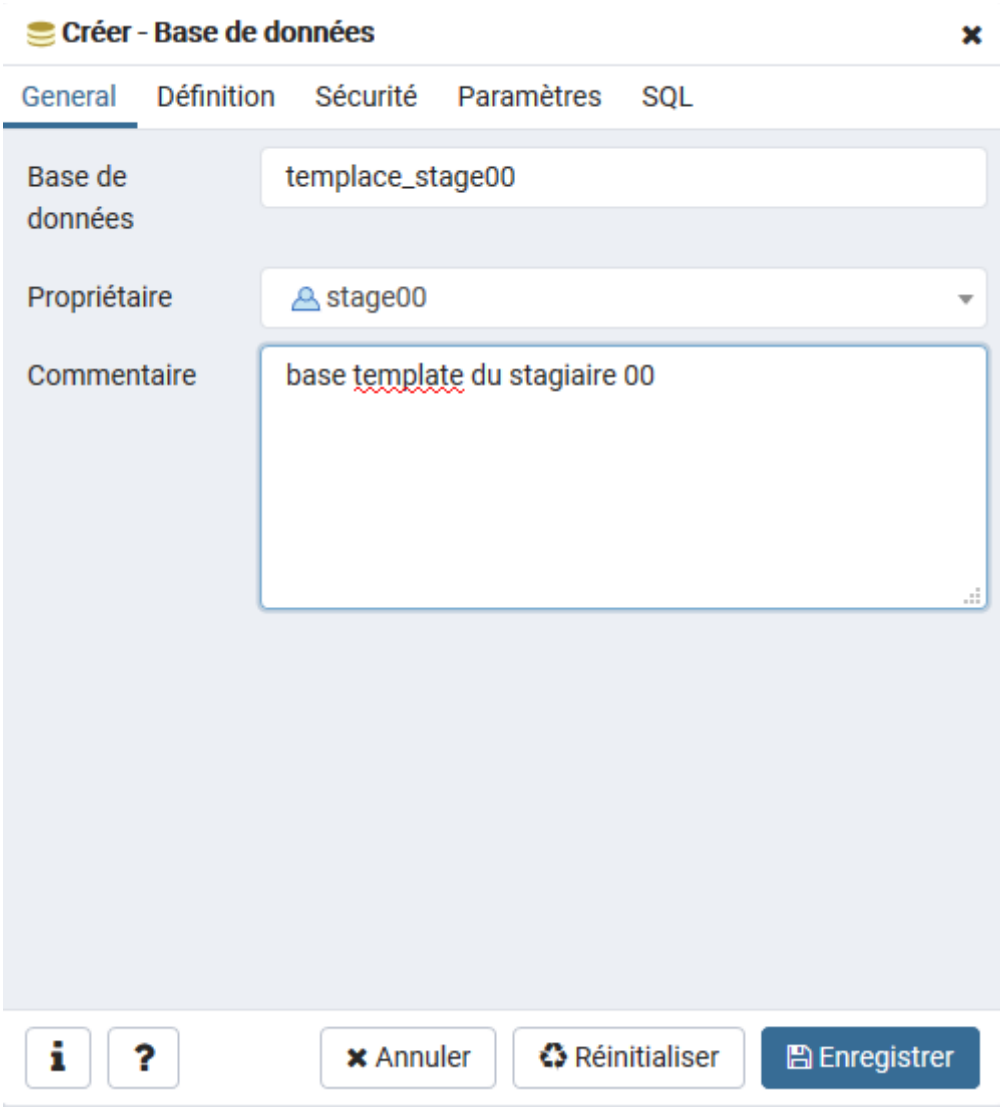
Méthode : Pas à pas : création d'une base "template"

Voyons comment créer une base 'template' au cas ou elle n'aurait pas été créée lors de l'installation de PostgreSQL / PostGIS.

Avec le compte *stageXX* (remplaçer XX par votre numéro), créer une nouvelle base *template_stageXX*.

(le propriétaire est *stageXX*)

1 - http://docs.postgresql.fr/current/charset.html



Se positionner dans cette base et utiliser  pour lancer l'éditeur SQL.

tapez

```
CREATE EXTENSION postgis;
```

lancer la requête avec le bouton 

Vous devriez voir apparaître un message ressemblant à :

« ***requêteexécutée avec succès en 863 ms.*** »

Constater que :

- une table *spatial_ref_sys* a été créée dans le schéma public
- quatre vues ont été créées (dont *geography_columns*)
- des fonctions ont été ajoutées.

Vérifier que l'extension est bien installée en exécutant l'ordre SQL suivant :

```
SELECT postgis_full_version();
```

La réponse doit être quelque chose comme :

"POSTGIS="2.3.1 r15264" GEOS="3.5.1-CAPI-1.9.1 r4246" PROJ="Rel. 4.9.3, 15 August 2016" GDAL="GDAL 2.1.2, released 2016/10/24" LIBXML="2.9.4" LIBJSON="0.12.1" RASTER"

Ici nous avons donc installé PostGIS 2.3.1 avec GDAL 2.1.2

Il est important de connaître la version de PostGIS installée, car certains problèmes sont parfois résolus dans des versions ultérieures. Voir par exemple : *http://postgis.net/docs/release_notes.html*[2]

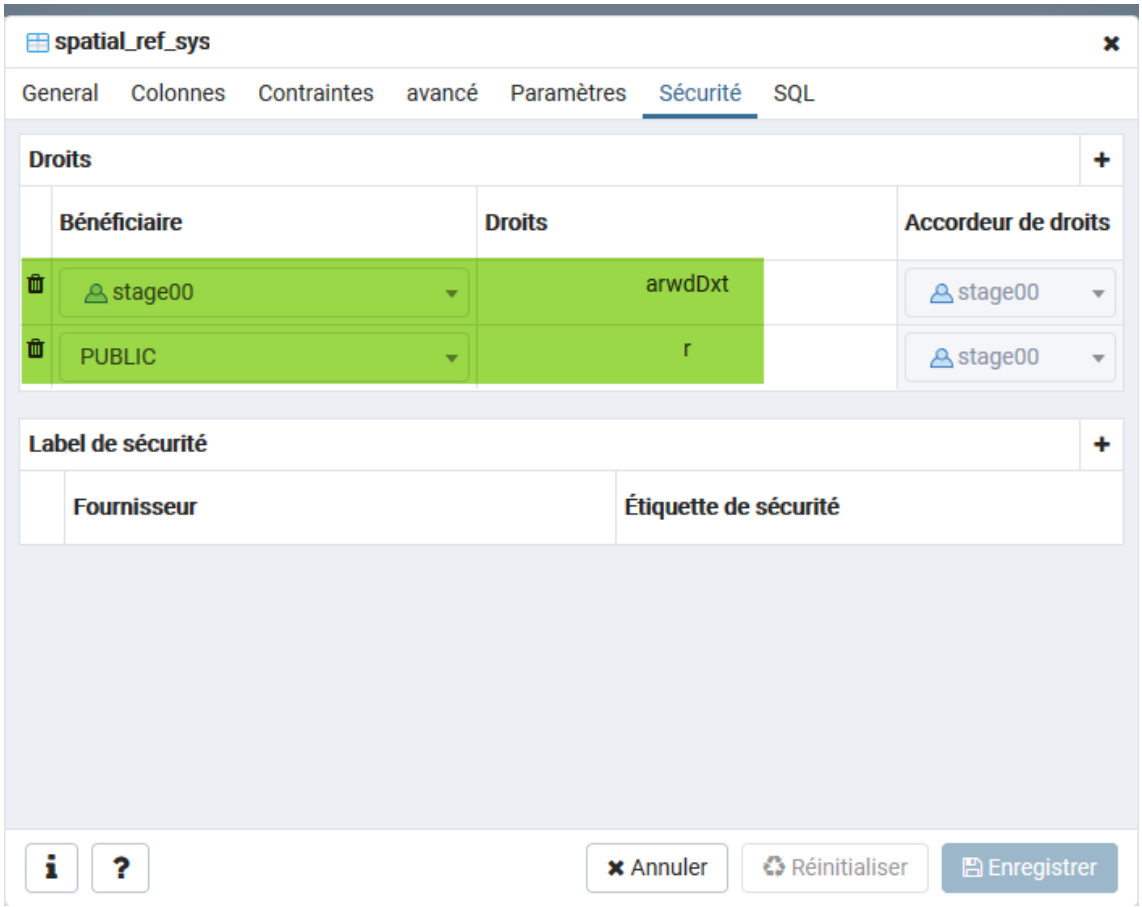
(de même avec les versions de gdal).

Remarque

Les droits pour le rôle public sur la table *spatial_sys_ref* et les vues PostGIS peuvent être positionnés dans le template, ce qui permettra de ne pas avoir à les modifier pour chaque nouvelle base créée.

On fera attention que le propriétaire de *spatial_sys_ref* est le rôle avec lequel on a créée l'extension postgis (ici *stageXX*).

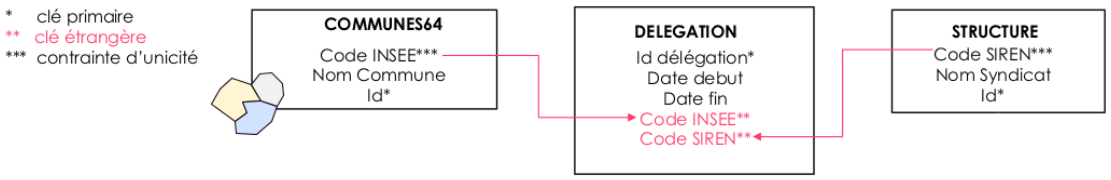
Exemple :



2 - http://postgis.net/docs/release_notes.html

B. Création d'une base relationnelle, schéma, table.

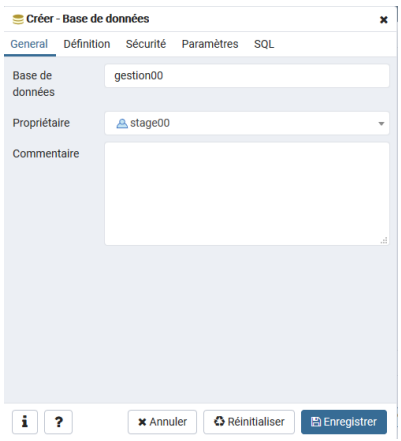
L'objectif est de construire la base de données suivante (cf module1 de la formation).



Seule la table *communes64* a une représentation géométrique.

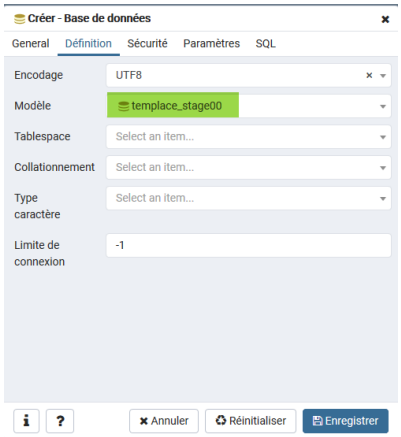
1. Création d'une base avec pgAdmin

Méthode : Pas à pas : créer une base avec pgAdmin



Étape 1 : Créer la base *gestionXX* avec le rôle *stageXX* en utilisant PgAdmin et le *template_stageXX* préalablement créé.

nb : il faut se déconnecter de la base *template_stageXX* pour pouvoir l'utiliser comme modèle de création d'une nouvelle base.



Créer un schéma *travail*.

Donner les droits au rôle *ecrivains00*

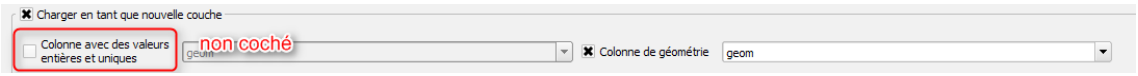
(vous savez le faire ! ne pas oublier de donner les droits de connexion sur la base)

Complément

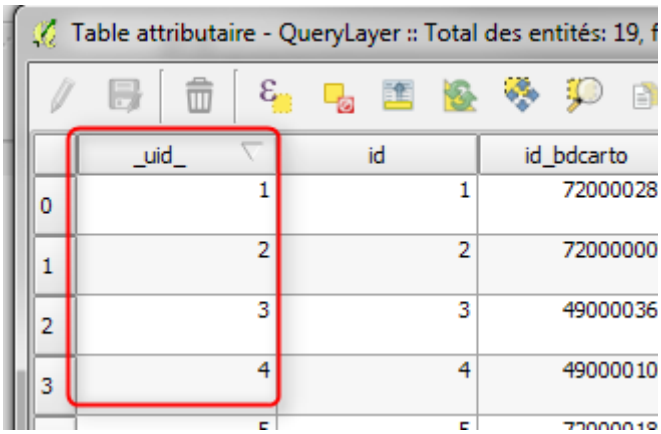
Les *OID*[3] sont des identifiants d'objets utilisé en interne par PostgreSQL.

Si une table n'a pas d'identifiant évident, il est recommandé d'ajouter un champ que l'on peut nommer 'id' de type serial non nul qui servira de clef primaire.

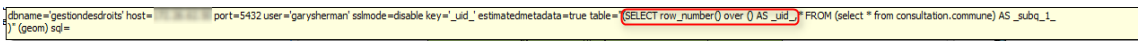
À noter qu'avant la version 1.20 de DBManager sous QGIS il était obligatoire de disposer d'un identifiant numérique unique pour pouvoir charger le résultat d'une requête sous forme de table dans QGIS. Ce n'est plus le cas, car cet identifiant est automatiquement créé comme un champ supplémentaire si on ne désigne pas un champ existant.



Création d'un champ supplémentaire _uid_



et l'ajout à la requête créée automatique par DBManager :




```
select row_number() over as _uid_
```

Complément : voir l'activité du serveur

Il est possible de configurer pdadmin pour surveiller l'activité du serveur.

Fichier > Préférences > Tableaux de bord > affichage

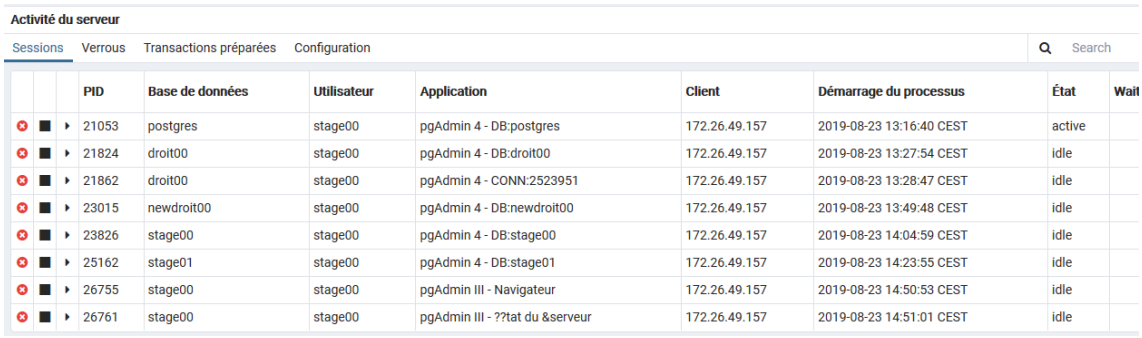
sélectionner : Afficher l'activité.

en vous positionnant sur un objet dans le navigateur (exemple une base de données) vous pouvez voir l'activité du serveur.

L'onglet 'Sessions' permet en particulier d'arrêter ou de tuer des processus en cours.

A utiliser, bien sûr, avec précaution !

3 - http://docs.postgresql.fr/9.4/datatype-oid.html



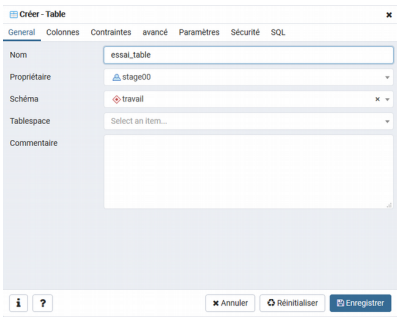
	PID	Base de données	Utilisateur	Application	Client	Démarrage du processus	État	Wait
	21053	postgres	stage00	pgAdmin 4 - DB:postgres	172.26.49.157	2019-08-23 13:16:40 CEST	active	
	21824	droit00	stage00	pgAdmin 4 - DB:droit00	172.26.49.157	2019-08-23 13:27:54 CEST	idle	
	21862	droit00	stage00	pgAdmin 4 - CONN:2523951	172.26.49.157	2019-08-23 13:28:47 CEST	idle	
	23015	newdroit00	stage00	pgAdmin 4 - DB:newdroit00	172.26.49.157	2019-08-23 13:49:48 CEST	idle	
	23826	stage00	stage00	pgAdmin 4 - DB:stage00	172.26.49.157	2019-08-23 14:04:59 CEST	idle	
	25162	stage01	stage00	pgAdmin 4 - DB:stage01	172.26.49.157	2019-08-23 14:23:55 CEST	idle	
	26755	stage00	stage00	pgAdmin III - Navigateur	172.26.49.157	2019-08-23 14:50:53 CEST	idle	
	26761	stage00	stage00	pgAdmin III - ??tat du &serveur	172.26.49.157	2019-08-23 14:51:01 CEST	idle	

2. Création d'une table avec PgAdmin

Avec le rôle *stageXX*, dans le schéma travail de la base *gestionXX*, nous allons créer la table *essai_table* avec les propriétés suivantes :

- **Propriétés** :
  - Nom de la table : *essai_table*
  - Propriétaire : *stageXX* (remplaçer XX par votre numéro de stagiaire)
  - Schéma : *travail*
- **Définition** :
  - Tablespace : *<tablespace par défaut>*
- **Colonnes** :
  - Id : *Serial Not NULL (clef primaire)*
  - Essai_table : *character (25)*
  - Insee : *character (20)*

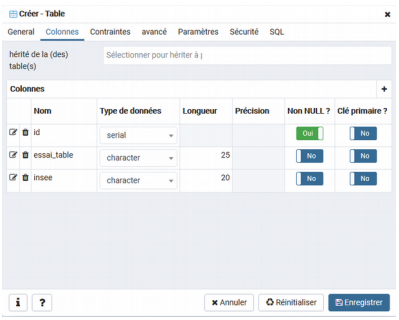
Méthode : Pas à pas : créer une table avec pgAdmin



Créer la table sous PgAdmin :

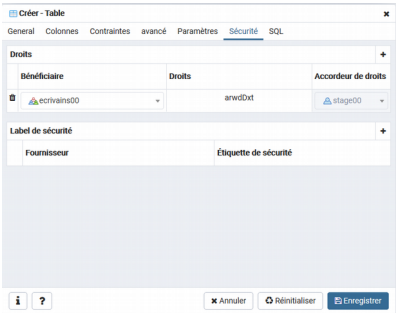
Dans le schéma *travail* de la base *gestionXX*, faire `clic droit` → `Créer` → `Table...`.

Remplir les onglets pour respecter les spécifications données ci-contre.

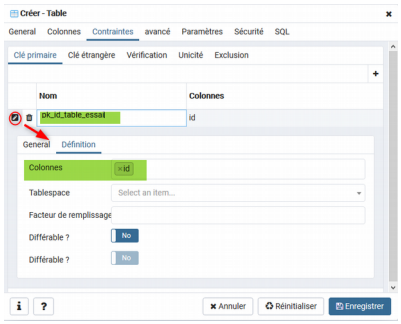


Définition des colonnes :

Ne pas oublier d'activer *NOT NULL* dans l'onglet '*définition*' pour la colonne *id*.



Ajouter les droits pour le groupe *ecrivains* dans l'onglet *Droits* :



Ajouter la contrainte de clef primaire dans l'onglet *Contraintes*.

Lui donner comme nom : *pk_id_table_essai*

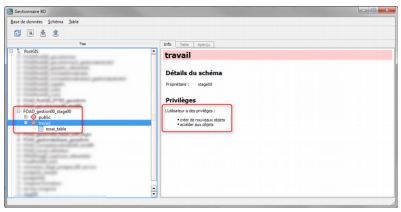
vérifier le script SQL dans l'onglet *SQL* et valider.

3. Création d'une table avec DBManager

Nous allons maintenant créer la table *essai_table_QGIS* avec exactement les mêmes spécifications attributaires mais en ajoutant une géométrie de type *POLYGON,* depuis *DBManager* sous QGIS.

Méthode : Pas à pas : créer une table avec DBManager

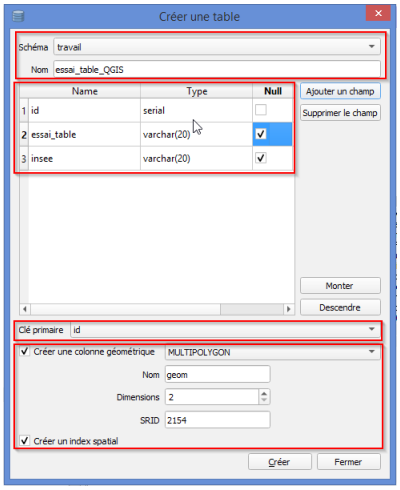
Sous QGIS, créer la liaison avec la base de données *gestionXX* avec le rôle *stageXX* :



Lançer *DBManager* ;

Menu → Base de données → gestionnaire BD

et vérifier que vous avez bien les droits sur le schéma *travail*

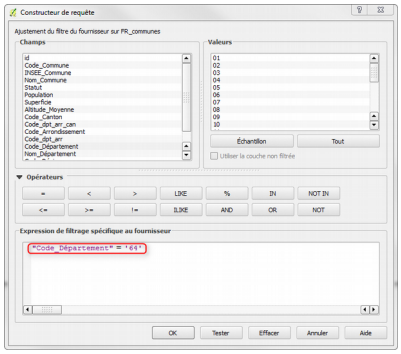


Puis créer la nouvelle table en passant par **Table → créer une table**.

La table doit maintenant apparaître dans la liste des tables du schéma travail :

4. Importation d'une table depuis QGIS

Méthode : Pas à pas : importer une table depuis QGIS



Nous allons maintenant travailler sur la base *stageXX* (remplacer XX par votre numéro de stagiaire, exemple : *stage00*)

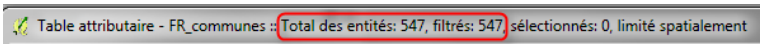
Sous QGIS, se connecter à la base *stageXX* avec le rôle *stageXX*

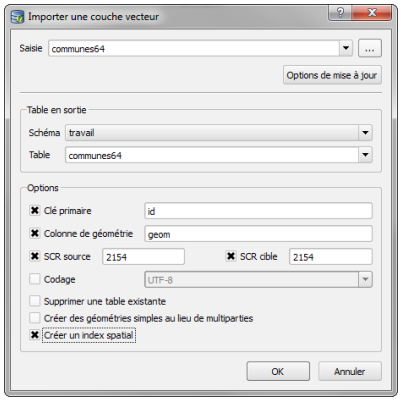
Sous QGIS, charger la couche *FR_communes* qui vous est fournie dans le schéma *travail* de la base *stageXX*, en filtrant sur *code_département=64*

(en utilisant le bouton `Filtrer` en bas de la boîte de dialogue du gestionnaire des sources de données PostgreSQL)

(nb : Si jamais il vous manque cette couche, elle est mise à disposition en bas de cette *page*[4])

Vérifier avec la table des attributs (ou avec le bouton 'tester') que vous avez bien 547 entités :





Avec DBManager, avec le rôle *stage00*, importer la couche filtrée dans le schéma *travail* de la base *stage00* et nommer la table *communes64*.

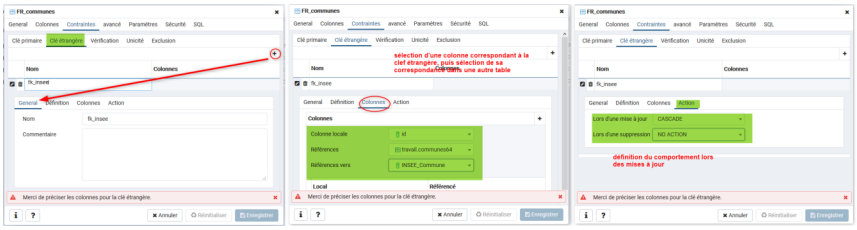
Dans DBManager : **`Table → Importer une couche ou un fichier`**

Si un filtre est actif, seuls les objets retenus sont créés dans la base de données.

Visualiser le résultat dans QGIS.

5. Création d'une clef étrangère avec PgAdmin

Il est possible de définir une clef étrangère (importation d'une clef primaire d'une autre table).



4 - http://www.geoinformations.developpement-durable.gouv.fr/postgis-support-pedagogique-a3347.html

Lorsque les données des colonnes référencées sont modifiées, des actions sont réalisées sur les données de la table référençant :

- La clause **ON DELETE** (onglet **action → lors d'une suppression**) spécifie l'action à réaliser lorsqu'une ligne référencée de la table de référence est supprimée.
- De la même façon, la clause **ON UPDATE** (lors d'une mise à jour) spécifie l'action à réaliser lorsqu'une colonne référencée est mise à jour.
- Si la ligne est mise à jour sans que la valeur de la colonne référencée ne soit modifiée, aucune action n'est réalisée.

Les actions suivantes sont possibles pour chaque clause (voir *CREATE TABLE*[5]) :

NO ACTION	Une erreur est produite pour indiquer que la suppression ou la mise à jour entraîne une violation de la contrainte de clé étrangère. C'est le comportement par défaut.
RESTRICT	Une erreur est produite pour indiquer que la suppression ou la mise à jour entraîne une violation de la contrainte de clé étrangère. Ce comportement est identique à NO ACTION, si ce n'est que la vérification n'est pas décalable dans le temps.
CASCADE	La mise à jour ou la suppression de la ligne de référence est propagée à l'ensemble des lignes qui la référencent, qui sont, respectivement, mises à jour ou supprimées.
SET NULL	La valeur de la colonne qui référence est positionnée à NULL.
SET DEFAULT	La valeur de la colonne qui référence est positionnée à celle par défaut.

Si les colonnes référencées sont modifiées fréquemment, il est conseillé d'ajouter un index sur la colonne de clé étrangère de façon à accélérer les actions référentielles associées à la colonne de clé étrangère.

Complément : Convention de nommage

Il est bon d'adopter une convention de nommage, par exemple :

- **clef primaire** : *pk_nom_champ_nom_de_la_table* (pk pour primary key), la clef primaire est souvent un champ nommé *id*.
- **clef étrangère** : *fk_nom_champ_nom_table* (fk pour foreign key)

C. 03 - (tutoré) création d'une base relationnelle

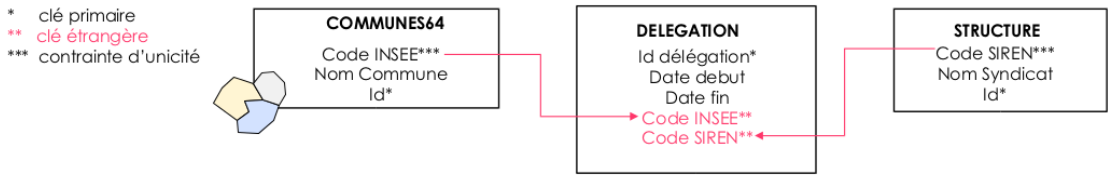
03 - création d'une base relationnelle

Cet exercice va vous permettre de vérifier l'intérêt de la gestion de l'intégrité référentielle dans un SGBD

5 - http://docs.postgresql.fr/9.4/sql-createtable.html

Question

Toujours dans le schéma *travail* (base stageXX), l'objectif est de créer les tables *Delegation* et *Structure* à partir du modèle de données ci-dessous :



Les fichiers *liste structure 64.dbf* et *liste delegation 64.dbf* sont fournis.

- Les importer sous PostgreSQL en passant par QGIS et DBbManager, puis compléter avec les contraintes du modèle de données (clef primaire et contraintes d'unicité).
- Pour les clefs étrangères, on utilisera l'action CASCADE en mise à jour et suppression.
- Afficher sous QGIS les tables *communes64* (géométrique) et les tables attributaires *structure* et *délégation*.
- Vérifier dans la table *délégation* l'existence de délégations avec le code siren 200026409 en utilisant un filtre d'expression.
- Dans la table *structure* supprimer le SIVOM de LAGOR (code siren = 200026409) et enregistrer.
- Vérifier à nouveau la table délégation.
- Conclure sur l'intérêt des clefs étrangères et du modèle relationnel et indiquer des exemples dans votre patrimoine de données qui pourraient bénéficier du modèle relationnel et la raison.

Envoyer un message à la BAL partagée pour indiquer aux tuteurs que l'exercice est fait et pour apporter votre conclusion

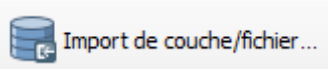
sur l'intérêt du modèle relationnel avec exemples.

II Import et export de données

A. Import de données

1. Import via DBManager

Nous avons déjà vu que DBManager permet de charger des fichiers dans une base PostGIS connectée sous QGIS, par glisser / lâcher de fichiers à partir du navigateur QGIS, ou par le menu **`Table -> Importer une couche ou un fichier`** pour les couches déjà chargée dans QGIS.

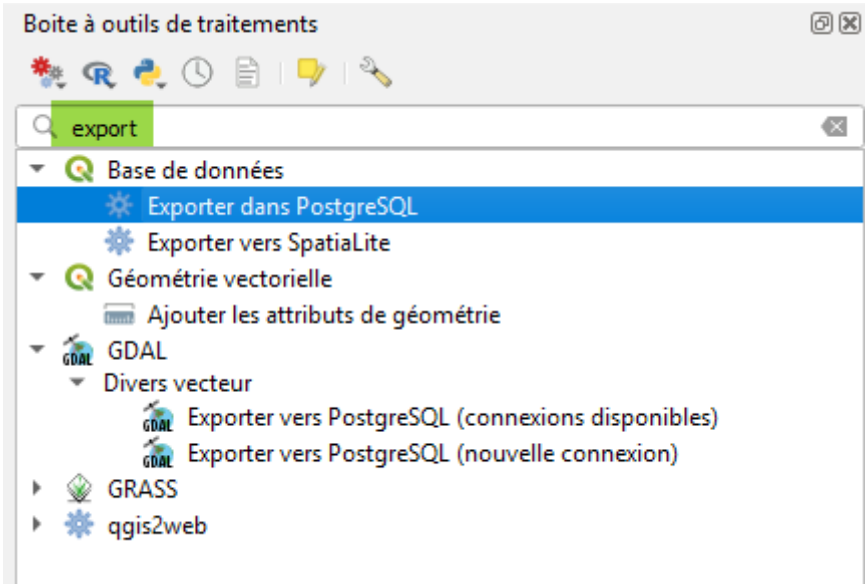
Ce dernier menu est également accessible par le bouton 

Vous pouvez également transférer des tables entre bases de données par un simple glisser-déposer ou par les algorithmes d'import du menu '*traitement*'

A noter que DBManager ne propose pas l'intégration de plusieurs couches en même temps (batch) et que les performances sont moindre que les algorithmes d'import du module '*traitement*'.

2. Import via Processing (menu traitement)

Un autre moyen d'importer une couche dans PostGIS est d'utiliser les algorithmes d'import vers PostGIS disponibles dans *Processing* (Boîte à outils de traitements)

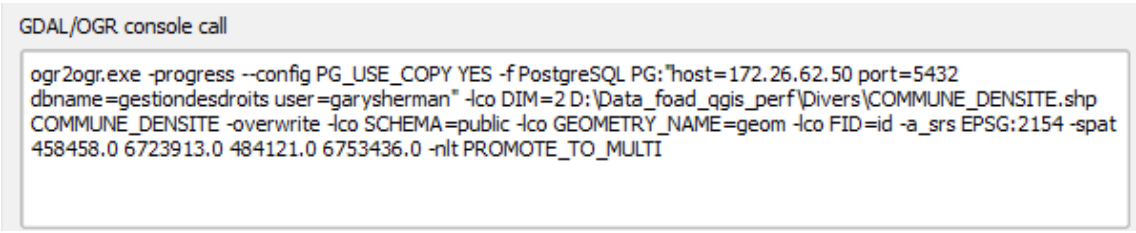


Par exemple, **`exporter vers PostgreSQL(connections disponibles)`** permet d'exporter dans PostGIS avec une connexion ouverte.

(nb : dans certains versions de QGIS le terme utilisé est *Importer* au lieu de *Exporter*)

Il s'agit en fait d'une aide à la rédaction d'une commande **ogr2ogr** qui apparaît en bas de la boîte de dialogue.

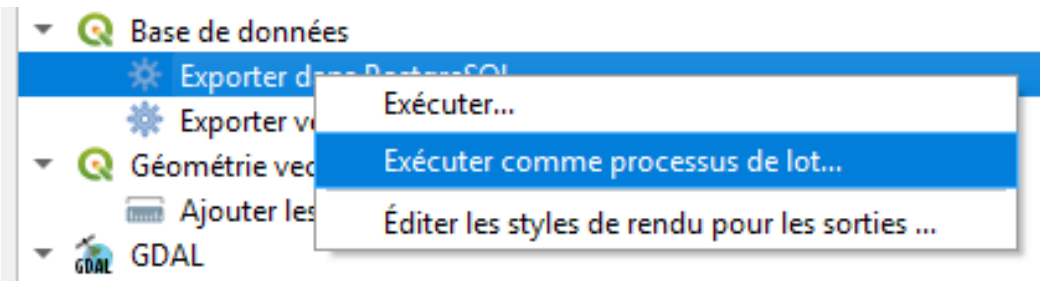
Exemple :



De nombreux paramètres sont réglables.

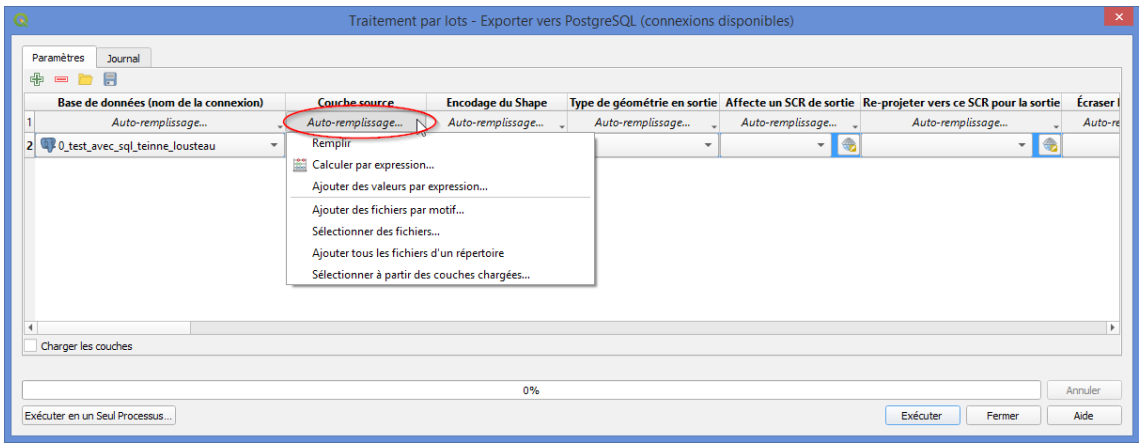
Ogr2ogr a été choisi par Faunalia pour réaliser cet algorithme. On trouvera *ici*[6] une justification en termes de performances par rapport à *shp2pgsql*.

À noter que cette méthode permet d'importer en lot. En effet un algorithme peut être lancé par **`clic droit → exécuter comme processus de lot`**.



6 - https://faunaliagis.wordpress.com/2014/11/24/a-new-qgis-tool-based-on-ogr2ogr-to-import-vectors-in-postgis-the-fast-way/

Le choix de plusieurs couches avec le bouton ... permet d'alimenter automatiquement autant de lignes que de couches à importer et de préciser ensuite les paramètres pour chaque couche :



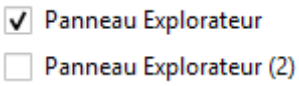
L'import via ogr2ogr par cette méthode est considéré comme beaucoup plus performant en termes de rapidité que l'import par DBManager.

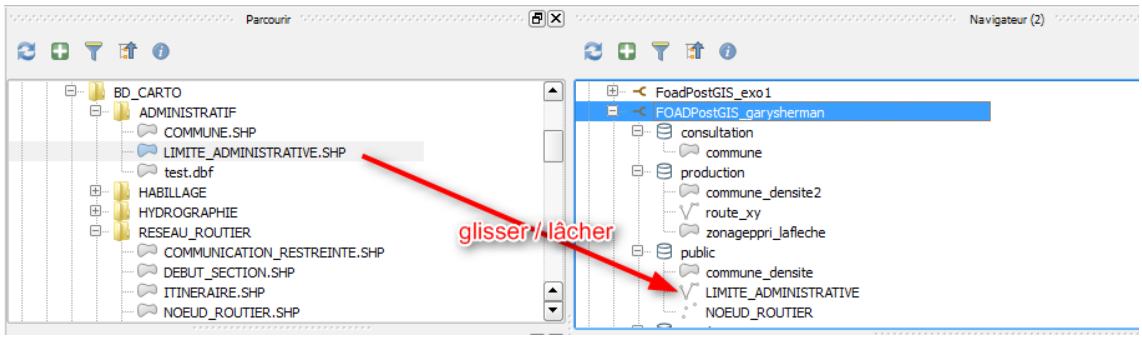
3. Import par Glisser/Lâcher via les navigateurs de QGIS

Il est possible sous QGIS d'utiliser le navigateur et même un deuxième navigateur pour faire des glisser / lâcher.

Fire `clic droit` dans une barre d'outils et activer Parcourir et Navigateur(2).

Ou Avec QGIS 3 les panneaux explorateur et explorateur (2).





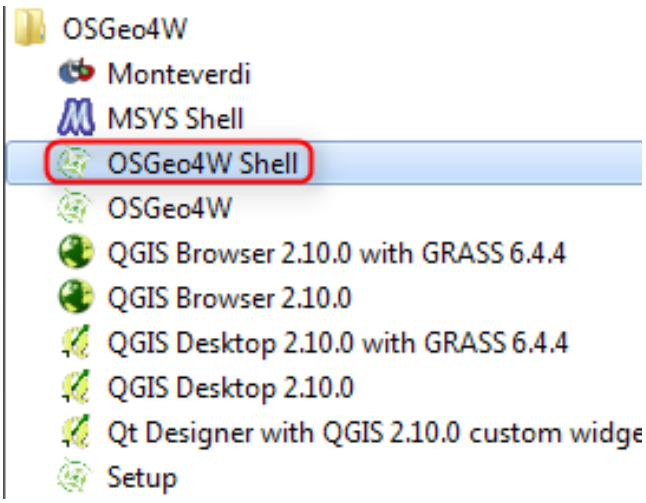
Attention

Avec le système du glissé/lâché on ne contrôle pas les problèmes d'encodage.

4. Import via ogr2ogr

GDAL (Geospatial Data Abstraction Library) est une bibliothèque libre permettant de lire et de traiter un très grand nombre de format d'images géographiques. Un sous-ensemble de cette bibliothèque est la bibliothèque OGR permettant d'accéder à la plupart des formats courants de données vectorielles. On trouvera une documentation en français sur *ce site*[7]. La documentation de référence sur ogr2ogr est *disponible ici.*[8]

Une installation de QGIS par OSGeo4W installe sur le poste local une console Shell permettant de lancer ogr2ogr (c'est également le cas pour les packages Ministère).



Si vous ne disposez pas du raccourci de lancement dans le menu vous pouvez lancer :

`C:\Program Files\QGIS 3.16\OSGeo4W.bat` (au besoin créer un raccourci sur le bureau).

On utilisera `ogr2ogr --version` pour récupérer le numéro de version installé :

```
C:\>ogr2ogr --version
GDAL 3.1.4, released 2020/10/20
```

Fermer la fenêtre shell en tapant `exit`.

Dans la pratique on utilisera surtout ogr2ogr dans des scripts sur le serveur (en poste local autant passer par les algorithmes de processing -menu traitement- qui offrent une interface sous forme de boîte de dialogue).

Dans ce cas on utilisera le ogr2ogr installé avec les packages du serveur (*exemple pour Debian*[9]).

7 - http://gdal.gloobe.org/
8 - http://www.gdal.org/ogr2ogr.html
9 - https://packages.debian.org/stable/gdal-bin

Il est possible de taper `ogr2ogr - - help` pour avoir une aide sur les paramètres.

Dans les options de ogr2ogr qui sont détaillées ici on notera en particulier :

- **`-f`** : format de sortie
- **`-overwrite`** : écrase les anciennes valeurs si existantes
- **`-append -update`** : ajoute des données sans écraser d'autres
- **`-nln`** : affecte nouveau nom à la table (si rien alors nom du fichier pour nom table)
- **`-s_srs`** : projection de la source
- **`-a_srs`** : assigne valeur de la projection en sortie
- **`-t_srs`** : transforme la projection (reprojection)
- **`-skipfailures`** : continue après un échec, ignorant l'objet en échec (en particulier si géométrie invalide).
- **`-nlt`** : exemple `-nlt MULTIPOLYGONE` pour imposer le type de géométrie.

Pour PostgreSQL on trouvera *ici la description*[10] des options spécifiques en particulier les Layer Creation Option (`-lco`).

Noter également l'importance de l'option `-config PG_USER_COPY YES` qui impose d'utiliser l'ordre `COPY` au lieu de `INSERT` et accélère beaucoup les traitements.

Exemple :

```
ogr2ogr -f "PostgreSQL" -append -nln "MaTable" -nlt MULTIPOLYGON -lco
GEOMETRY_NAME=the_geom    -lco    FID=gid    -lco    OVERWRITE=no    -lco
SCHEMA=MonSchema  -a_srs  EPSG:2154  PG:"dbname='MaBase'  host='MonHote'
port='PortDeHote'     user='MonIdentifiant'     password='MonMotDePasse'"
MonFichierSource
```

Pour un exemple plus concret, mettons que l'on veuille importer la table "COMMUNE_DENSITE.shp (fournie dans le jeu de données) sous le nom commune_densite

et que cette table soit disponible sous i:\

La commande sera :

```
ogr2ogr -f "PostgreSQL" -nln "commune_densite" -nlt MULTIPOLYGON -lco
GEOMETRY_NAME=geom     -lco     SCHEMA=travail     -a_srs     EPSG:2154
PG:"dbname='stage00'   host='10.167.71.3'   port='5432'   user='stage00'
password='stage00'" i:/COMMUNE_DENSITE.SHP
```

5. Import via Shp2pgsql

Shp2pgsql est un module de conversion pour les fichiers shape uniquement, qui est intégré à PostGIS (et donc disponible uniquement sur les machines ou PostGIS est installé). On l'utilisera donc plutôt pour réaliser des scripts sur le serveur.

La syntaxe est :

shp2pgsql [<options>] <shapefile> [<schema>.]<table>

Les principales options sont :

- **`-s`** précise le système de projection (SRS)
- **`-I`** génère un index spatial (Gist)
- **`-S`** utilise des géométries simples au lieu de géométrie multiple (défaut géométries MULTI).

10 - http://www.gdal.org/drv_pg.html

- **-d** Supprime la table avant de la récréer avec les nouvelles données.
- **-a** ajoute les données du shape dans la table.
- **-c** créé la table et l'alimente avec les données (option par défaut)
- **-p** crée la table sans insertion des données.
- **-D** Utilise des DUMP PostgreSQL
- **-W** Encodage des données attributaires (exemple : -W LATIN1)
- **-N** Gestion des géométries nulles
- **--help** aide en ligne de la commande

À noter également l'option **-m** qui depuis la version 2.2 de PostGIS permet de spécifier une table de conversion des noms de colonnes du fichier SHP vers des nouveaux noms de colonnes dans la table PostGIS. Les colonnes non spécifiées conservent leur nom.

La commande générique dans un shell pour importer sur un serveur distant est :

```
shp2pgsql [<options>] <shapefile> [<schema>.]<table> | psql.exe -h <serveur> -p <port> -U <user> -d <base>
```

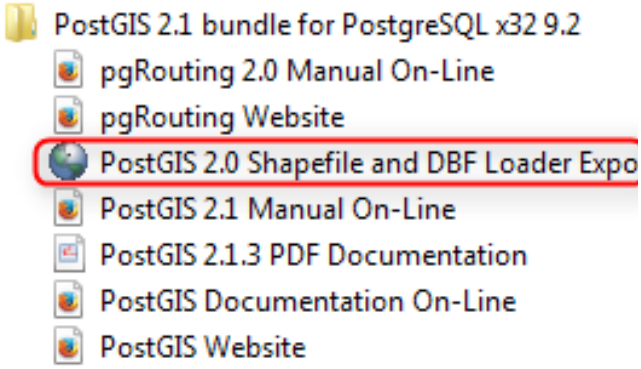
(on utilise un pipe, de symbole | , pour exécuter la sortie du shp2pgsql qui est une liste de commandes sql sur le serveur distant)

Aide-mémoire des options[11] (en anglais).

Pour le choix entre shp2pgsql vs ogr2ogr, on pourra se reporter à cette *discussion*[12] (en anglais) sur l'importation de grand ensemble de données. À noter cependant que la création d'index en import (qui est coûteuse) peut-être désactivée dans *ogr2ogr* avec l'option `-lco SPATIAL_INDEX=NO`. L'index spatial pouvant être créé après coup sur le serveur (voir *conseils*[13]).

Remarque

Pour une utilisation de PostgreSQL/PosGIS en mode bureautique, il existe une interface *shp2pgsql-gui* qui est installée par défaut si vous avez installé PostGIS sous windows avec *stackbuilder* (entrepriseDB). Cette interface est lancée par *PostGIS ShapeFile and DBF loader exporter*.



Son utilisation, assez intuitive, est décrite dans le *module SQL de la FOAD QGIS perfectionnement*[14]. Elle permet également l'export en utilisant en arrière plan *pgsql2shp.exe*.

11 - http://www.bostongis.com/pgsql2shp_shp2pgsql_quickguide.bqg
12 - https://doublebyteblog.wordpress.com/2014/08/06/importing-large-spatial-dataset-into-postgis/
13 - http://docs.postgresql.fr/current/populate.html
14 - http://piece-jointe-carto.developpement-durable.gouv.fr/NAT002/QGIS/formations/FOAD_PERF_QGIS34/pdf/M04_SQL_BDD_papier.pdf

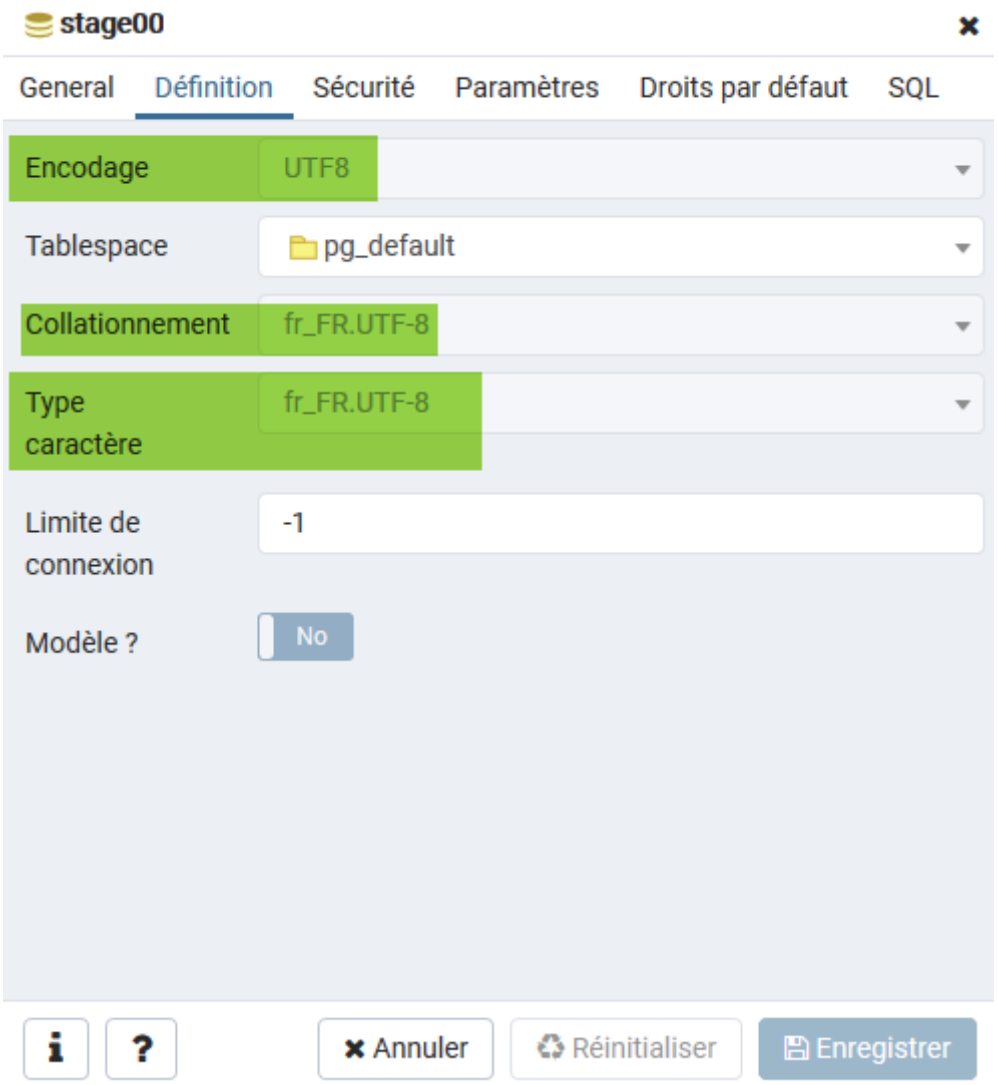
6. Les points d'attention en import de données

Conseil : L'encodage des caractères

Par défaut shp2pgsql et ogr2ogr ignorent l'encodage des fichiers et ne font pas de conversions.

Dans PostgreSQL chaque base peut avoir son jeu de caractères (éventuellement différent de celui qui a été attribué à la création du cluster de bases de données qui est généralement choisi comme étant UTF-8).

On peut voir l'encodage d'une base sans ses propriétés sous PgAdmin :



Que l'on peut retrouver dans la table base pg_database de la base postgres (base Postgres > catalogues > tables) :

Données | EXPLAIN | Messages | Notifications

	oid oid	datname name	datdba oid	encoding integer	datcollate name	datctype name	datistemplate boolean	datallowconn boolean
1	124...	postgres	10	6	fr_FR.UTF-8	fr_FR.UTF-8	false	true
2	134...	stage01	16473	6	fr_FR.UTF-8	fr_FR.UTF-8	false	true
3	134...	stage02	16474	6	fr_FR.UTF-8	fr_FR.UTF-8	false	true
4	124...	template0	10	6	fr_FR.UTF-8	fr_FR.UTF-8	true	false
5	134...	stage03	16475	6	fr_FR.UTF-8	fr_FR.UTF-8	false	true
6	134...	stage04	16476	6	fr_FR.UTF-8	fr_FR.UTF-8	false	true

Une conversion automatique entre le client et le serveur est possible si on déclare explicitement le jeu de code du client, par l'une des méthodes indiquées sur *cette page*[15]. En particulier `SET CLIENT_ENCODING TO 'valeur'` (par exemple `SET CLIENT ENCODING TO 'LATIN1'`, à taper dans un requêteur SQL).

Dans un shell on pourra utiliser la variable PGCLIENTENCODING.

- Exemple sous windows : `set PGCLIENTENCODING=latin1`
- Exemple sous Unix : `export PGCLIENTENCODING=latin1`

On pourra également consulter *cette page du site geoinfo*[16].

Une erreur typique d'un problème d'encodage sera un message du type :

« ERROR 1: INSERT command for new feature failed. »

« ERREUR: s├®quence d'octets invalide pour l'encodage ┬½ UTF8 ┬╗ : 0xe96475 »

Complément : Encodage de fichiers SHP

Il arrive de rencontrer des problèmes lors de l'alimentation de bases PostGIS en UTF8 à partir de *shapefiles* encodés en WIN1252.

Le problème provient de la version 1.9.0 d'OGR qui interprète l'encodage du SHP (dans le LDID ou le .cpg) pour faire la conversion et renvoyer le flux en UTF-8.

Si on déclare que l'encodage est du WIN1252, il y a donc une erreur puisque le flux est en UTF8.

La solution est de dire à OGR de "ne pas détecter l'encodage" du shapefile source.

Ceci ce fait en mettant la variable d'environnement *SHAPE_ENCODING* à ""

Il existe une autre façon avec les commande GDAL/OGR de positionner une variable d'environnement à l'exécution, c'est en précisant :

--config SHAPE_ENCODING ""

On peut préciser dans un script d'import les 2 variables d'environnement :

SET SHAPE_ENCODING="" (ou export SHAPE_ENCODING="" sous LINUX)

SET PGCLIENTENCODING="WIN1252" (ou export PGCLIENTENCODING="WIN1252" sous LINUX)

SHAPE_ENCODING="" : permet de dire à ogr de ne pas tenter de déterminer l'encodage des données attributaires des shapefiles.

PGCLIENTENCODING="WIN1252" : permet de dire à PostgreSQL qu'on va lui fournir

15 - http://docs.postgresql.fr/9.6/multibyte.html

16 - http://www.geoinformations.developpement-durable.gouv.fr/encodage-utf-8-avec-ogr2ogr-sous-windows-a2941.html

des flux en WIN1252. C'est PostgreSQL qui fera lui-même la conversion de ce flux vers l'encodage de la base.

Exemple :

```
I:\>SET PGCLIENTENCODING="WIN1252"

I:\>SET SHAPE_ENCODING=""

I:\>ogr2ogr --config PG_USE_COPY YES -f PostgreSQL PG:"host=172.[illegible] port=54
32 dbname=gestiondesbases password=[illegible] user=geoadmin" i:\COMMUNE.SHP -lco S
CHEMA=tableregles -nln commune12 -nlt PROMOTE_TO_MULTI
```

Conseil : Les géométries invalides

Il arrive que en cas de géométrie invalide au sens de l'OGC, un import, par exemple par ogr2ogr vers PostgreSQL échoue. Dans ce cas il peut être utile d'utiliser l'option **-skipfailure** qui permet de continuer le traitement après erreur. Voir plus loin, le chapitre sur la correction des erreurs de géométrie dans module sur les compléments SQL.

Conseil : Le type de géométrie

Il peut arriver que l'on cherche à insérer un fichier ayant des géométries hétérogènes, typiquement POLYGON et MULTIPOLYGON.

Dans ce cas si la table existe déjà dans PostGIS il faut changer la contrainte sur la géométrie

exemple :.. CONSTRAINT enforce_geotype_the_geom CHECK (geometrytype(the_geom) = 'MULTIPOLYGON'::text OR the_geom IS NULL), avant PostGIS 2.0

A partir de PostGIS 2.0, on peut par exemple changer la définition du champs the_geom en `geometry(geometry, 2154)` qui permettra d'accepter n'importe quelle géométrie.

Autre exemple pour passer une table (y compris les entités déjà existantes) en géométrie MULTIPOLYGON :

```
ALTER TABLE ads49.parcelle
ALTER COLUMN the_geom
SET DATA TYPE geometry(MULTIPOLYGON,2154) USING ST_Multi(the_geom)
```

Il peut également arriver que les données soient importées en 4D ou 3D (import Bdtopo par exemple) alors que l'on souhaite les exploiter comme des données 2D.

Dans ce cas on pourra ramener les géométries en 2D avec un `ST_Force_2D()`, exemple :

```
ALTER TABLE police_eau.troncons_bdp
ALTER COLUMN the_geom TYPE geometry(MultiLineString) USING
ST_Force_2D(the_geom);
```

Conseil : Les projections IGNF

Les projections **IGNF** ne sont pas présentes pas défaut dans PostGIS. Le site de l'*IGN*[17] ne propose plus (2022) de script au format sql pour les créer dans

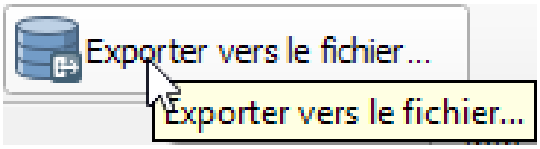
17 - http://geodesie.ign.fr/index.php?p=61&page=documentation#titre3

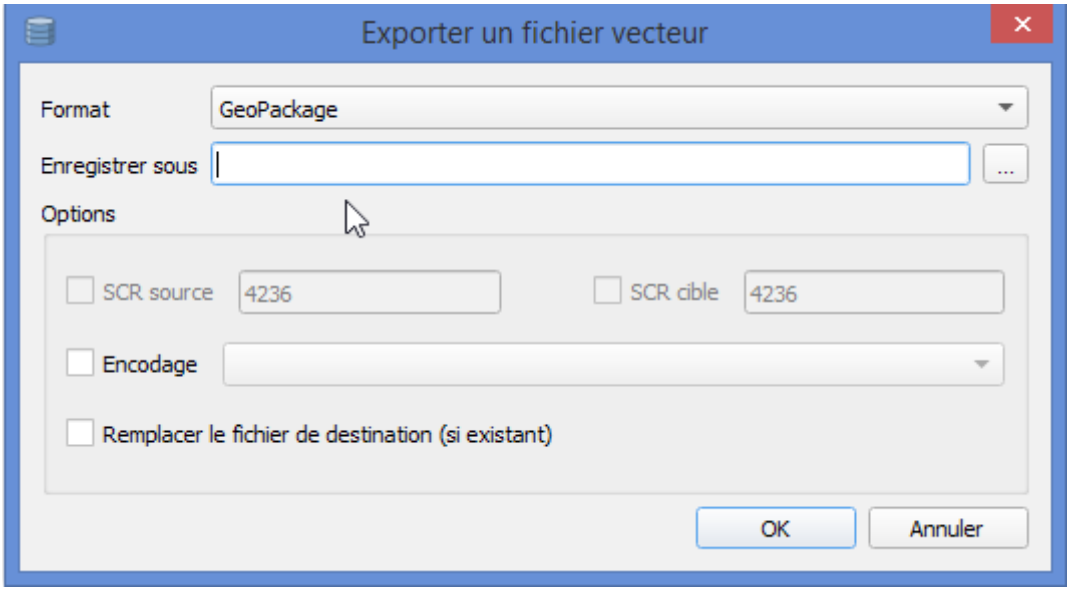
avant import.

B. Export de données

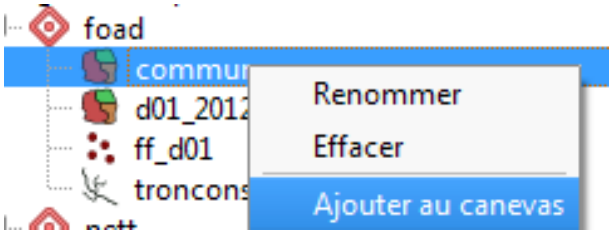
Méthode : Export de données avec DBManager

DBManager dispose d'une fonction d'export



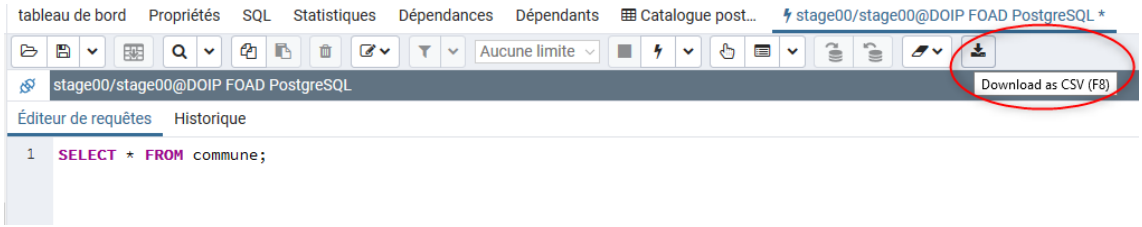


Il est également possible dans DBManager de charger une couche dans le canevas de QGIS par **clic droit** sur la couche → **Ajouter au canevas.**



Méthode : Exporter le résultat d'une requête SQL

Sous PgAdmin il est possible, après l'exécution d'une requête SQL, d'utiliser le bouton de téléchargement en CSV :



Conseil

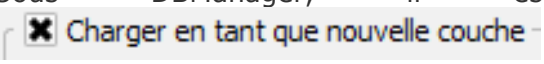
Si les données exportées contiennent des nombres à virgule, le séparateur décimal est par défaut un point. Exemple : **3.14159**

Ceci est très pratique pour importer les données dans les systèmes de gestion de bases de données, car la majorité d'entre eux fonctionne avec ce séparateur.

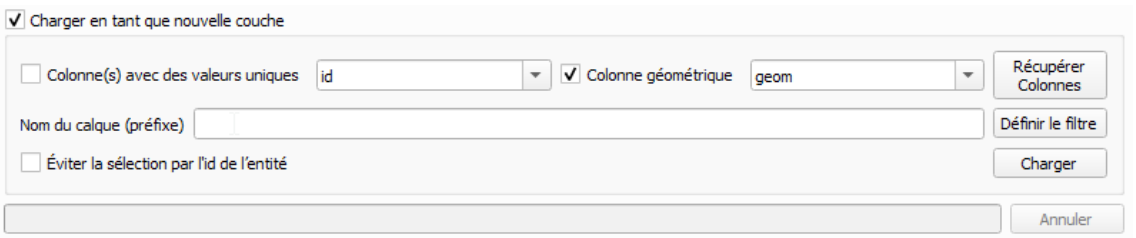
En revanche, l'import dans Calc peut poser des problèmes, car le séparateur décimal est la virgule. Pour contourner cet obstacle, il faut choisir le type de données *Anglais US* à l'ouverture du CSV.

Méthode

Sous DBManager, il est possible d'utiliser l'option



il faut alors préciser un certain nombre de paramètres :



Le bouton **`Récupérer Colonnes`** permet de récupérer les colonnes existantes.

Il faut une colonne avec des valeurs entières et unique. Si on ne coche pas cette case, DBManager rajoutera une colonne *--uuid--* calculée automatique avec l'expression `row_number() over()`

```
(SELECT row_number() over () AS _uid_,* FROM (select * from consultation.commune)
```

Complément : Export de données (autres)

pgsql2shp est le pendant de *shp2pgsql* pour exporter des données en shape à partir de PostgreSQL

Il est également possible d'utiliser des scripts avec ogr2ogr pour faire des exports depuis le serveur (par exemple pour une conversion automatique des données en

format SHP).

A noter également que ogr2ogr permet d'exporter en une seule commande toutes les tables ou une liste de table :

```
ogr2ogr -f "ESRI Shapefile" mondossier PG:"host=myhost user=myloginname dbname=mabase password=mypassword"
```

va exporter toutes les tables de la base `mabase` dans le repertoire `mondossier`

On pourrait également utiliser :

```
ogr2ogr -f "ESRI Shapefile" mondossier PG:"host=myhost user=myloginname dbname=mabase password=mypassword" table1 table2
```

pour n'exporter que les tables `table1` et `table2`

A partir de gdal 1.7 on peut *surcharger*[18] le schéma par défaut en utilisant `active_schema = mon schema` pour éviter de préfixer les tables à exporter.

(nb : il peut être nécessaire de mettre mylogin, mabase,... entre '' (quotes simples) s'ils contiennent des espaces)

C. 04 - Import et export de fichiers SHP

Cet exercice va vous permettre de mettre en œuvre un import et export de fichiers SHP

Question

[*Solution n°1 p 35*]

Importer dans le schéma *travail* de la base *stageXX* les tables :

PONCTUEL_HYDROGRAPHIQUE

SURFACE_HYDROGRAPHIQUE

TRONCON_HYDROGRAPHIQUE

puis exporter toutes les tables de la base `stageXX` dans le format SHP dans un dossier `\masauvegarde` sur votre PC.

Indice :

On pourra, par exemple, utiliser pour méthode le glisser-lâcher avec parcourir et navigateur(2) pour l'import

et une commande ogr2ogr pour exporter au format SHP

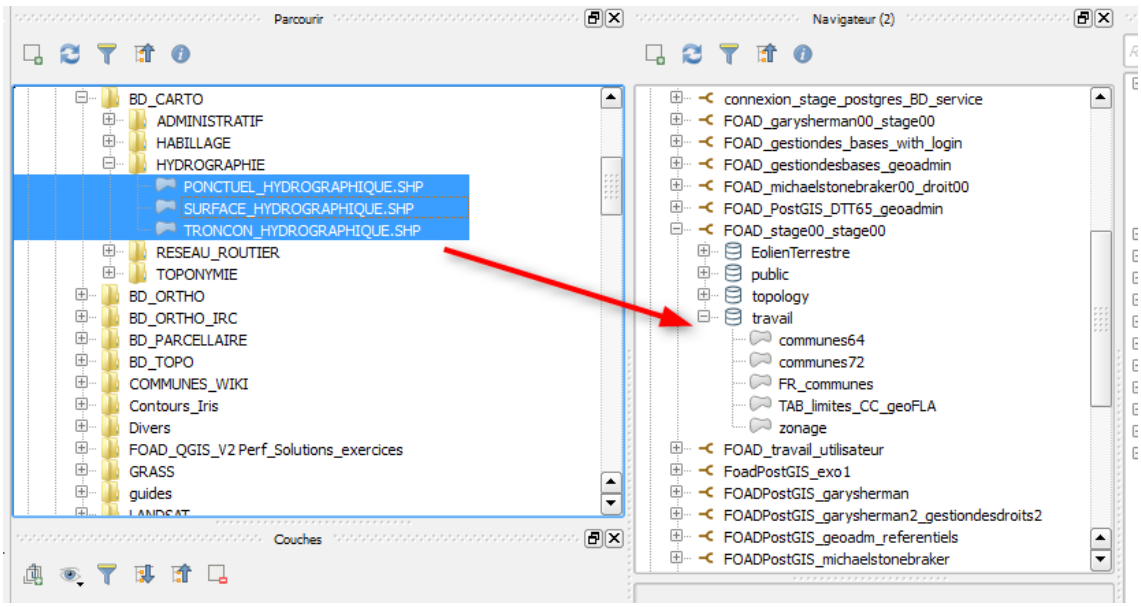
(on pourra utiliser la console `osgeo4w.bat` disponible sous `c:\program...\qgis`)

18 - http://www.gdal.org/drv_pg_advanced.html

Solution des exercices

> Solution n°1 *(exercice p. 34)*

Pour importer en une seule fois des couches qui sont sur le 'système de fichier' on peut, par exemple, utiliser la méthode par glisser-laché



Pour l'export, il faut lancer osgeo4w.bat sous c:\program...\qgis (à voir en fonction de l'installation de QGIS sur votre poste)

puis taper la commande :

```
ogr2ogr.exe -f "ESRI Shapefile" "i\masauvegarde" "PG:dbname='stageXX' host=XXX.XX.XX.XX port=5432 user='stageXX' password='XXXXX' sslmode=disable"
```

remplacer les XX par les bonnes valeurs...

Ceci sauvegarde toutes les tables de la base stage00.

```
C:\>ogr2ogr -f "ESRI Shapefile" "i:\masauvegarde" "PG:dbname='stage00' host=172.26.62.50 port=5432 user='stage00' password='stage00' sslmode=disable"
```

Des alertes peuvent indiquer que des noms de champs ont été tronqués à 10 caractères qui est le maximum dans le format SHP.

vérifier sous QGIS quelques fichiers sauvegardés.

Complément

Utilisation de ogr2ogr dans un fichier de commande (.bat si windows).

On peut automatiser les exports (et imports) dans un fichier de commande. Ci-dessous un exemple simple :

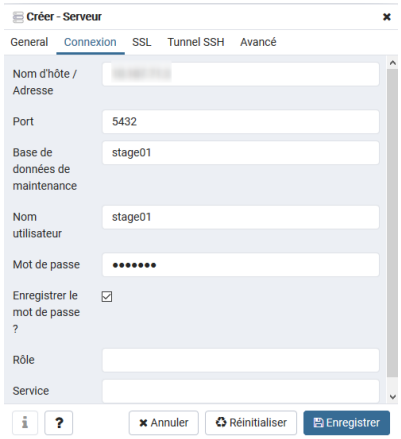
```
SET monpathexe=C:\Programmes\QGIS_Lyon\bin\
SET monformat="ESRI Shapefile"
SET monrepsauv="d:\divers\did"
SET mabase="stage00"
SET monserveurip=172.28.9.122
SET monserveurport=5632
SET monuser='stage00'
SET monpassword='stage00'
cls
%monpathexe%ogr2ogr --version
%monpathexe%ogr2ogr --help
%monpathexe%ogr2ogr.exe -f %monformat% %monrepsauv% "PG:dbname=%mabase% host=%monserveurip% port=%monserveurport% user=%monuser% password= %monpassword% sslmode=disable"
```

Contenus annexes

- Création de la base et des schémas

Lancer PgAdmin

Méthode : Créer la base et les schémas



Faire **Objet-> Créer un serveur** (ou utiliser le lien rapide de l'onglet 'tableau de bord' : ajouter un nouveau serveur) et utiliser les paramètres (hôte, port) de connexion qui vous ont été fournis par les organisateurs.

Vous avez du fournir l'adresse IP de votre poste de formation aux organisateurs qui en retour vous ont également retourné un login (de type stageXX) et un mot de passe.

Pour la formation pilote ces paramètres sont :

- **Nom** : `Serveur_formation - stageXX` (remplacer XX par le numéro qui vous est attribué)
- **Hôte** : `10.167.71.3`
- **port** : `5432`
- **Base maintenance** : `stageXX` (remplacer XX par le numéro qui vous est attribué)
- **Nom utilisateur** : `stageXX`
- **mot de passe** : `stageXX`
- **Couleur** : choisir éventuellement une autre couleur an arrière plan et/ou en premier plan. Ceci permettra de mettre en exergue que cette connexion est faite avec un rôle *superuser* (privilège accordé par l'administrateur système).

On peut « ***Enregistrer le mot de passe*** » dans l'onglet connexion**.**

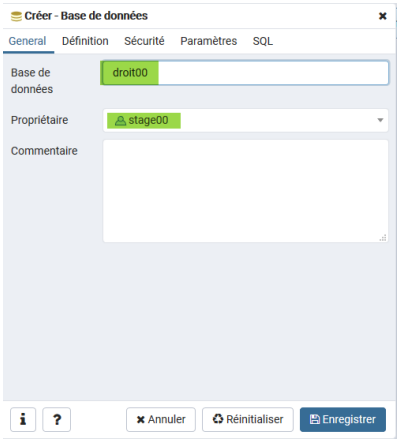
Dans la suite de la formation remplacer XX ou 00 par le numéro qui vous est attribué (ex : *droit01* à la place de *droitXX* ou de *droit00*)

Le serveur pour la formation est un serveur centralisé basé (Janvier 2019) sur la version suivante de PostgreSQL :

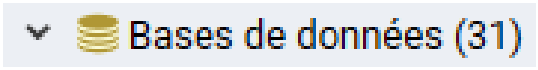
*"**PostgreSQL 9.6.11** on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516, 64-bit"*

et sur la version de PostGIS suivante :

*"**POSTGIS="2.3.1** r15264" GEOS="3.5.1-CAPI-1.9.1 r4246" PROJ="Rel. 4.9.3, 15 August 2016" GDAL="GDAL 2.1.2, released 2016/10/24" LIBXML="2.9.4" LIBJSON="0.12.1" RASTER"*

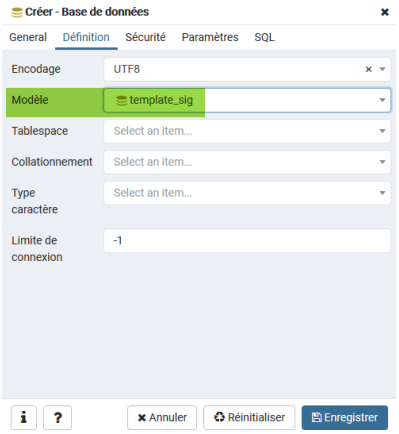


Faire un clic droit sur *Bases de données*



et `créer` **`une base de données`** (nous reviendrons plus tard sur les paramètres de création des bases de données).

Remplir l'onglet *Propriétés* comme indiqué ci-contre (remplacer le 00 de droit00 par votre numéro de stagiaire).



Remplir l'onglet *Définition* comme indiqué ci-contre (ne pas oublier de remplir dans l'onglet `Définition` le modèle : `template_sig`).

Vérifier que l'on a bien dans l'onglet *SQL*:

```
CREATE DATABASE droit00
WITH
OWNER = stage00
TEMPLATE = template_sig
ENCODING = 'UTF8'
CONNECTION LIMIT = -1;
```

(remplacer stage00 par stageXX, XX étant votre numéro de stagiaire) puis valider.

Double cliquer sur la base de données *droit00* (remplacer 00 par votre numéro)

Par défaut cette base contient au moins un schéma *public*.

Comme déjà indiqué il n'est pas conseillé de travailler dans le schéma *public*.

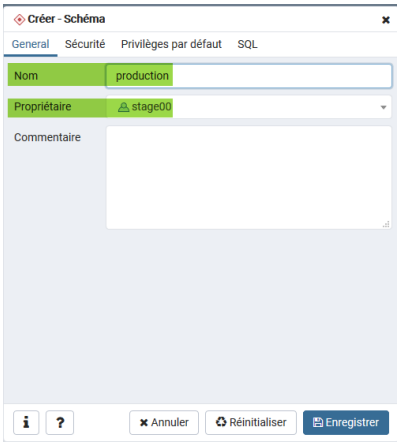
Nous allons créer les schémas *production* et *consultation* :

- Pour le schéma *production,* nous aurons deux types d'utilisateurs :
  - les *écrivainsXX* qui auront les droits de modifier les tables existantes du schéma *production* de la base *droitXX*
  - les *lecteursXX* qui n'auront que les droits de les visualiser.
- Pour le schéma *consultation,* tous les utilisateurs n'auront que les droits de lecture.

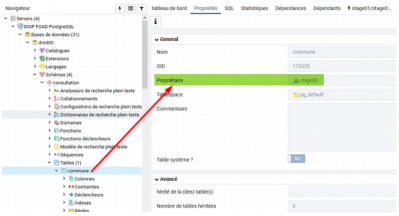
Pour créer le schéma *production* :

- faire un clic droit sur la base *droitXX* → **`Créer → schéma...`**
- Nous rajouterons les droits plus tard. Faire OK.

Puis, de même, créer le schéma *consultation*.



Rappel : jeux de données à télécharger



Pour remplir ces schémas par des tables nous allons restaurer un backup (nous reverrons plus tard, plus en détail, la problématique de sauvegarde et de restauration des bases).

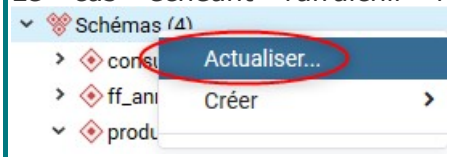
- Se positionner sur la base *droitXX*, puis clic droit 'restaurer' :
- Indiquer le fichier *gestiondesdroits.backup*.
- Dans l'onglet *Options de restauration* cocher dans *sections : pre-data* et data et dans *ne pas enregistrer : propriétaire*.
- Lancer la restauration.

Une fenêtre apparaît en bas à droite et indique la progression de la restauration...

Après un certain temps (un peu plus de 1 mn), la fenêtre indique 'Echec (code de sortie 1). Si on affiche les détails on se rend compte qu'il ne s'agit que de 'WARNING' du au fait que les schémas *production* et *consultation* existent déjà dans la base.

La base doit maintenant contenir la table *commune* dans le schéma *consultation* et les tables *route_xy* et *zonageppri_lafleche* dans le schéma *production*.

Le cas échéant rafraîchir l'affichage par un clic droit puis '*actualiser...*'



Vérifier dans l'onglet Propriétés que le propriétaire des tables (par exemple commune dans le schéma consultation) est bien stageXX (XX étant votre numéro de stagiaire).

Si ce n'est pas le cas c'est que vous n'avez pas coché ne pas sauvegarder propriétaire au moment de la restauration.

La base doit maintenant ressembler à ceci :