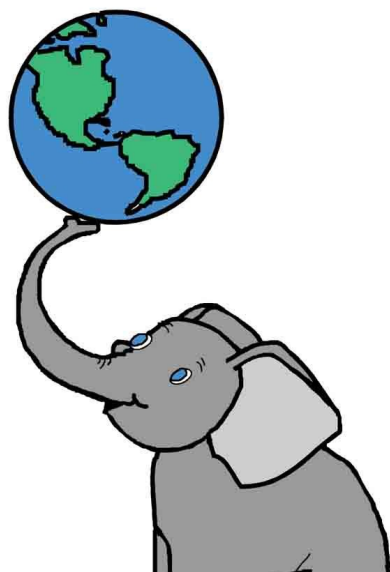


Administration de PostgreSQL/Post GIS



Version 1.4

Ministère de la Transition Ecologique
Licence ETALAB

Janvier 2022



Table des matières

Objectifs	5
Introduction	7
I - Les clients et les connexions	9
A. Fichiers de configuration (postgresql.conf et pg_hba.conf).....	9
B. Les outils clients.....	10
1. Les principaux outils clients.....	10
2. pgAdmin.....	11
3. Psql.....	17
4. DBManager.....	18
C. Organisation du serveur de formation.....	19
II - La gestion des droits	23
A. Les rôles.....	23
B. Propriétaire d'un objet.....	25
C. Droits sur les objets.....	26
D. Pas-à-pas sur un exemple.....	29
1. Création de la base et des schémas.....	30
2. Création des rôles.....	33
3. Utilisation avec DBManager sous QGIS.....	36
E. 01 (tutoré) - gestion des droits.....	49
F. ASGARD.....	49
III - Sauvegarde et restauration	53

A. Sauvegarde logique (dump).....	54
B. Sauvegarder le serveur (pg_dumpall).....	60
C. Restauration logique.....	60
D. 02 - sauvegarde et restauration.....	64
E. Utilisation d'une console PSQL.....	64
F. Sauvegarde au niveau système de fichiers.....	65
G. Archivage continu.....	66
IV - Maintenance	67
A. Gestion de l'espace disque.....	67
B. Opérations de maintenance sur les bases.....	69
Solution des exercices	73





Objectifs

Les objectifs du module sont de :

- connaître les principaux outils clients pour l'administration de PostgreSQL / PostGIS
- apprendre à gérer les droits
- savoir réaliser des sauvegardes et des restaurations
- découvrir les opérations de maintenance.



Introduction

Le temps d'apprentissage de ce module consacré à l'administration d'une base PostgreSQL/PostGIS est estimé à 3 heures, plus une heure pour la réalisation de l'exercice tutoré.

Il comporte :

- un exercice guidé sous forme de pas-à-pas ;
- un exercice auto-corrigé ;
- un exercice "tutoré" dont vous devrez communiquer le résultat à vos tuteurs.

Les clients et les connexions

I

Fichiers de configuration (postgresql.conf et pg_hba.conf)	9
Les outils clients	10
Organisation du serveur de formation	19

A. Fichiers de configuration (postgresql.conf et pg_hba.conf)

PostgreSQL.conf

Ce paragraphe concerne l'administrateur système et peut donc être sauté par les autres profils (voir acteurs et rôles dans le module d'introduction).

PostgreSQL attend des connexions depuis des logiciels clients (nous en verrons prochainement, mais QGIS en est un exemple).

Le fichier de configuration propre à chaque instance PostgreSQL se nomme *postgresql.conf*.

Ce fichier rassemble les directives de *configurations*¹ permettant d'adapter le comportement du serveur au matériel. Nous ne les détaillerons pas ici, mais notons qu'il est possible en particulier de définir les adresses TCP/IP que le serveur écoute ainsi que les ports d'écoute (défaut 5432). Ce fichier est géré par l'administrateur système.

Dans le cas d'une ouverture vers le Web il est conseillé de changer le port d'écoute par défaut.

pg_hba.conf

Les droits de **connexion** aux bases sont définies dans le fichier *pg_hba.conf* (type de connexion, ex : host, nom de la base de données, nom de l'utilisateur, paramètres réseaux, méthode d'authentification exemple : md5 pour demander un mot de passe).

Ce fichier est présent dans le répertoire initialisé avec l'instance, par exemple dans les systèmes Debian sous *etc/postgresql/9.3/main/pg_hba.conf*.

Une connexion ne concerne qu'une seule base de données et doit être réalisée avec un compte utilisateur.

1 - <http://docs.postgresql.fr/12/runtime-config.html#config-setting>

Ce fichier est décrit dans la *documentation de PostgreSQL*². Il y a beaucoup d'options réglables, par exemple le processus d'autovacuum est activé (ou non) par le paramètre *-autovacuum*.

Il faut redémarrer PostgreSQL pour prendre en compte les modifications (commande *restart*).

(en fait il est possible de faire un simple reload du fichier de conf si on a modifié des paramètres qui ne nécessitent pas un redémarrage complet, voir *pg_ctl*³ -reload)

Retenons qu'il permet un filtrage des utilisateurs par adresse IP de leur poste.



Exemple : Filtrage des utilisateurs par adresse IP

exemple : `host all all 172.26.0.0/24 md5`

Autorise tous les utilisateurs (2ème *all*) à se connecter à toutes les bases (1er *all*) depuis n'importe quelle hôte d'un réseau local en fournissant un mot de passe (*md5*) pour les adresses de 172.26.0.0 à 172.26.0.255 (l'option */24* est le *masque CIDR*⁴). On peut trouver sur Internet des *calculateurs*⁵ de masque CIDR.

Il aurait été possible de préciser plusieurs bases de données en les séparant par des virgules à la place du premier *all*.

Exemple : `host postgres, service_test all 172.26.0.0 /24 md5`

Ci-dessous un extrait du fichier *pg_hba.conf* du serveur de formation qui montre que les droits d'accès aux bases sont filtrés par adresse IP des stagiaires :

```
# stagiaires
host stage01,droit01,newdroit01,template_stage01,gestion01 stage01,gary01,michael01 172.26.0.0/32 md5
host stage02,droit02,newdroit02,template_stage02,gestion02 stage02,gary02,michael02 172.26.0.0/32 md5
host stage03,droit03,newdroit03,template_stage03,gestion03 stage03,gary03,michael03 172.26.0.0/32 md5
host stage04,droit04,newdroit04,template_stage04,gestion04 stage04,gary04,michael04 172.26.0.0/32 md5
host stage05,droit05,newdroit05,template_stage05,gestion05 stage05,gary05,michael05 172.26.0.0/32 md5
host stage07,droit07,newdroit07,template_stage07,gestion07 stage07,gary07,michael07 172.26.0.0/32 md5
host stage08,droit08,newdroit08,template_stage08,gestion08 stage08,gary08,michael08 172.26.0.0/32 md5
```

Les droits d'accès aux contenus des bases sont définis dans la base de données à plusieurs niveaux (base, schéma, table, colonne). Nous les examinons un peu plus loin.



Conseil : Astuce

Une tactique possible est d'ouvrir assez largement les droits de **connexion** aux utilisateurs du service dans le fichier *pg_hba.conf* et de contrôler plus finement la gestion des droits d'**accès** par base dans la gestion des droits et rôles de PostgreSQL, ce qui évite de solliciter trop souvent l'administrateur système et permet de reporter la gestion des droits courants, autre que le filtrage IP et éventuelles bases stratégiques (ex : base de sauvegarde), sur l'administrateur des bases.

Cependant il faut être conscient que même s'il ne peut accéder aux bases pour lesquels il n'a pas les droits, l'utilisateur peut parcourir l'arborescence des bases, ce qui n'est pas toujours souhaitable.

2 - <http://docs.postgresql.fr/12/client-authentication.html#auth-pg-hba-conf>

3 - <http://docs.postgresql.fr/12/app-pg-ctl.html>

4 - <https://fr.wikipedia.org/wiki/Sous-r%C3%A9seau>

5 - <http://cric.grenoble.cnrs.fr/Administrateurs/Outils/CalculMasque/>

B. Les outils clients

1. Les principaux outils clients

Outils	Description
psql	psql est le premier frontal pour PostgreSQL et est une interface en ligne de commande permettant la saisie de requêtes SQL, directement ou par l'utilisation de procédures stockées.
pgAdmin	pgAdmin est un outil d'administration graphique pour PostgreSQL distribué selon les termes de la licence PostgreSQL.
phpPgAdmin	phpPgAdmin est une interface web d'administration pour PostgreSQL. L'interface s'appuie sur des scripts PHP et sur la base de données PostgreSQL pour favoriser les diverses tâches d'administration. La version 7.13 date du 7 novembre 2020 et supporte PostgreSQL 13.
QGIS	QGIS est client de PostgreSQL.
dBeaver	Parmi les nombreux autres clients SQL du marché, dBeaver est un outil apprécié par un grand nombre en particulier pour son support multi-bases de données. C'est un outil libre diffusé sous licence Apache 2.0 dans sa version standard. Il existe une version Entreprise payante qui apporte <i>des fonctionnalités en plus</i> ⁶ . Son installation et son utilisation sont cependant plus délicate que pgAdmin.

Tableau 1 Quelques outils clients

Nous n'aborderons que superficiellement **psql** qui est plutôt réservé à l'administrateur système et aux administrateurs de base de données expérimentés pour nous concentrer sur deux clients que sont **PgAdmin** et **QGIS**.

nb : Il existe en réalité *beaucoup d'outils*⁷ clients de PostgreSQL. On trouvera *ici*⁸ un comparatif des 'meilleurs' clients graphique pour PostgreSQL en 2021.

2. pgAdmin

*pgAdmin*⁹ est le principal outil de gestion open source des bases de données PostgreSQL. Il peut-être exécuté en mode serveur web (web application) ou en mode bureau (desktop runtime). Il fournit une interface graphique pour la création, la maintenance et l'utilisation d'objets de base de données.

Si les formateurs vous ont donné une URL de la forme `172.26.XX.XX/pgadmin4` pour utiliser pgadmin en version serveur, utilisez-la dans votre navigateur.

6 - <https://dbeaver.com/edition/>

7 - https://wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools

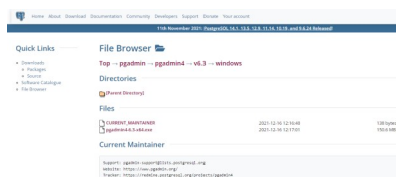
8 - <https://scalegrid.io/blog/which-is-the-best-postgresql-gui-2019-comparison/>

9 - http://www.pgadmin.org/?lang=fr_FR

Il vous faudra alors vous connecter avec le login et le mot de passe fourni par les formateurs, Exemple :



Méthode : Installation



Si les formateurs ne vous proposent pas d'utiliser pgadmin en version serveur :

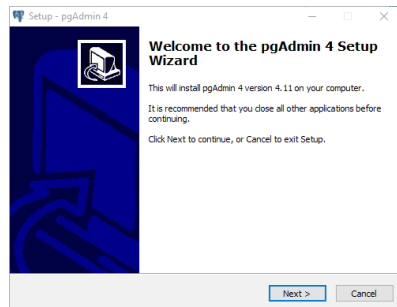
- Si vous avez installé PostgreSQL sur votre poste de travail, il est possible que vous disposiez déjà

pgAdmin installé. Dans ce cas vous pouvez le lancer à partir de la barre de programme.

- Si vous ne disposez pas encore de pgAdmin sur votre poste de travail et que les formateurs ne vous ont pas précisé d'utiliser pgadmin en version serveur vous devez télécharger un installateur à partir de <https://www.pgadmin.org/download/pgadmin-4-windows/>¹⁰).

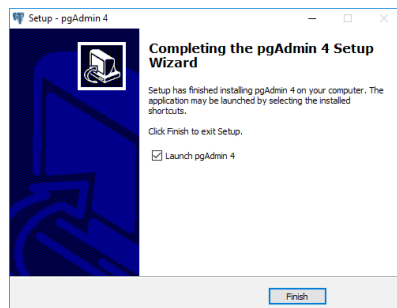
10 - <https://www.pgadmin.org/download/pgadmin-4-windows/>

Les clients et les connexions



Après téléchargement lancer l'exécutable (vous devez disposer de droits suffisants sur votre poste de travail).

Acceptez les conditions de la licence, puis poursuivre par défaut. Il peut y avoir un temps assez long dans la préparation de l'installation.

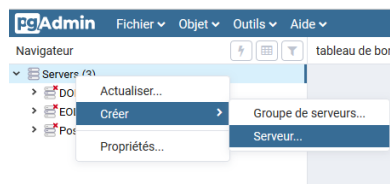


Si tout se passe bien vous devez arriver à cet écran :

Lancer pgAdmin 4. Il se lance dans votre navigateur par défaut ou en mode autonome (Desktop runtime) à partir de la version 5.

se connecter à la base de formation selon les paramètres fourni par l'organisateur.

Ajouter un serveur : Menu Objets > créer serveur (ou clic droit sur le mot 'servers' dans le navigateur), ou encore en utilisant le bouton 'Ajouter un nouveau serveur'

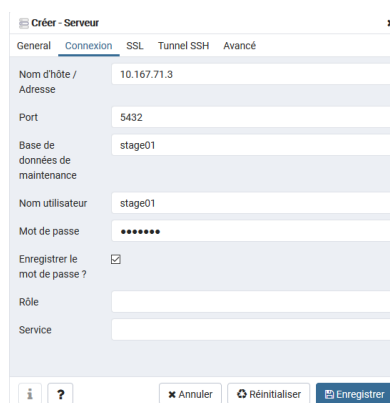


dans les liens rapides du tableau de bord :



Ajouter un nouveau serveur

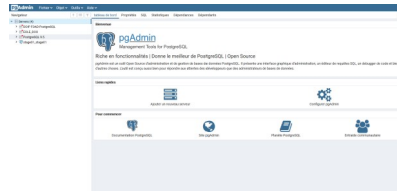
Saisir les paramètres du serveur



Dans le mode Bureau Il est possible d'enregistrer les mots de passe dans une base sqlite cryptée par un mot de passe maître que vous devez créer la première fois et qui vous sera ensuite demandé pour déverrouiller l'ensemble de vos connexions. Il est fortement recommandé d'utiliser un mot de passe maître si on enregistre les mots de passe des connexions. Si vous oubliez le mot de passe maître on peut utiliser le bouton 'réinitialiser le mot de passe maître' (ce qui supprime tous les mots de passe enregistrés et ferme toutes les connexions établies).

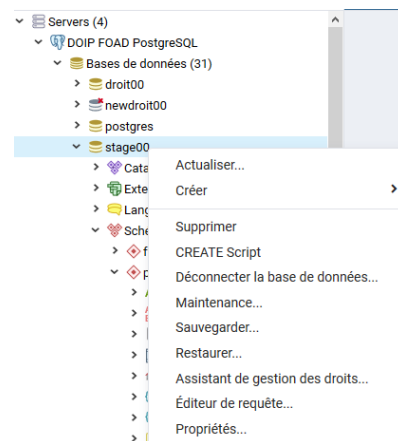
Nous verrons au fur et à mesure les différentes fonctions de pgAdmin.

Présentation




L'interface de pgAdmin comporte une barre de menus et une fenêtre divisée en deux volets : l'arborescence du navigateur dans le volet gauche et un navigateur à onglets dans le volet droit.

Pour chaque objet dans l'arborescence un menu contextuel est disponible.



Lors de la connexion au serveur existant, trois groupes d'éléments sont disponibles dans le navigateur d'objets (volet de gauche) :

- les **bases de données** existantes : La base postgres installée est la base de référence, qu'il ne faut jamais modifier ; elle sert de modèle pour la création de toutes les autres bases de données.
- les **tablespaces** : L'espace de stockage des tables et index. Il indique le répertoire d'écriture des fichiers. Cette partie est gérée par l'administrateur système (ou éventuellement l'administrateur de la base de données).
- les **rôles** : Les rôles de connexions correspondent aux différents utilisateurs, et les rôles groupe, aux groupes d'utilisateurs (nous y reviendrons). Cette partie est également gérée par l'administrateur de la base de donnée. Par défaut, il y a un seul rôle de connexion, celui de l'utilisateur « postgres » qui a tous les droits. Généralement cet utilisateur est désactivé pour éviter les problèmes de sécurité et l'administrateur système créé un rôle de connexion pour l'administrateur de la base.

Une barre d'outils donne accès à des fonctions fréquemment utilisées : (outils de requête SQL, affichage des données, et filtrage de ligne) 

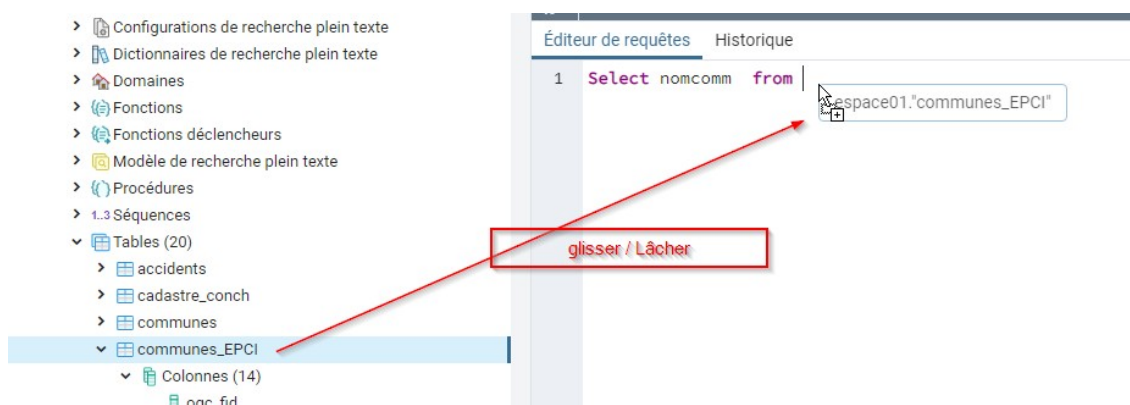
On peut éventuellement avoir des administrateurs délégués pour gérer une ou

plusieurs bases spécifiques (gestion des droits, maintenance,...) mais il est conseillé de centraliser le droit de création de nouvelles bases.

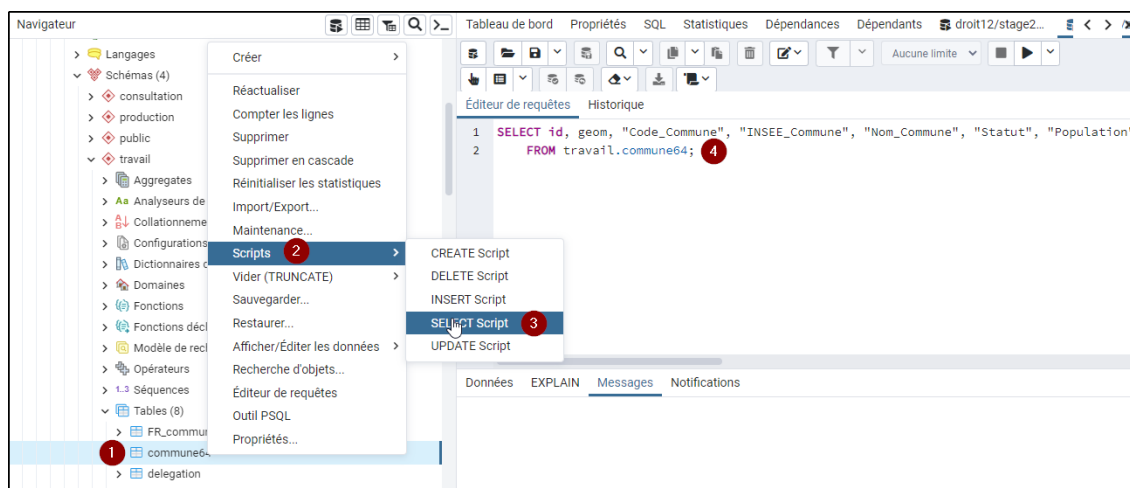


Complément : pgadmin4 : quelques astuces bien utiles...

Glisser/lâcher d'une table ou d'un champ lors de la rédaction d'une requête dans l'éditeur de requêtes pour éviter des erreurs de syntaxe (préfixé par le nom du schéma et double quotes si nécessaire).



Un clic droit sur les objets propose de pré-générer certains scripts...



CTRL + espace permet l'auto-complétion pour les noms de schémas, de tables, de champs (dans certaines conditions) et même de jointures.

Éditeur de requêtes Historique

```

1 SELECT *
2 FROM travail.delegations
3 JOIN travail.

```

CTRL + Espace

- "TRONCON_HYDROGRAPHIQUE"
- "vue_com64"
- communes64
 - communes64 ON communes64."INSEE_Commune" = delegations.code_insee
- delegations
- essai
- invalidgeometry
- makevalidgeometry
- population_commune64
- structures
 - structures ON structures.code_siren = delegations.code_siren

re-utiliser le code SQL de création des objets

La fenêtre principale de PGAdmin comporte un onglet SQL qui affiche le SQL nécessaire à la création de l'objet sur lequel vous êtes positionnés dans l'arborescence de votre base, et également, en commentaire, la commande permettant de le supprimer.

En cas de coquille dans la création d'un objet, il est facile de copier / coller ce code dans un éditeur SQL pour supprimer l'objet et le recréer après rectification du SQL pour qu'il soit conforme.

C'est aussi un bon moyen d'avoir un exemple de code SQL lorsque l'on a oublié la syntaxe

Navigateur

- Collationnements
- Configurations de recherche plein texte
- Dictionnaires de recherche plein texte
- Domaines
- Fonctions
- Fonctions déclencheurs
- Modèle de recherche plein texte
- Opérateurs
- 1.3 Séquences
- Tables (11)
 - FR_communes
 - PONCTUEL_HYDROGRAPHIQUE
 - SURFACE_HYDROGRAPHIQUE
 - TRONCON_HYDROGRAPHIQUE
 - communes64
 - delegations
 - Colonnes (2)
 - Contraintes (2)
 - fk_insee_delegation_commune
 - fk_siren_delegation_structure
 - Déclencheurs
 - Indexes
 - Politiques RLS

Tableau de bord Propriétés SQL 2 Statistiques Dépendances Dépendants droit12/stage2...

```

1 -- Constraint: fk_insee_delegation_commune
2
3 ALTER TABLE IF EXISTS travail.delegations DROP CONSTRAINT IF EXISTS fk_insee_delegation
4
5 ALTER TABLE IF EXISTS travail.delegations
6 ADD CONSTRAINT fk_insee_delegation_commune FOREIGN KEY (code_insee)
7 REFERENCES travail.communes64 ("INSEE_Commune") MATCH SIMPLE
8 ON UPDATE CASCADE
9 ON DELETE CASCADE;
10


```


On peut modifier rapidement dans ce code les erreurs (nom, action sur UPDATE et DELETE, ou les types de champs si je suis sur une table



Complément : Le gestionnaire de stockage de pgAdmin version server

PgAdmin en version server propose un gestionnaire de stockage qui est spécifique à cette version.

Le gestionnaire de stockage se lance depuis le menu Outils. Il est possible de créer des sous-répertoires dans cet espace grâce au bouton .

Le bouton  permet de téléverser un fichier du poste de travail vers l'espace de stockage. Le fichier doit faire moins de 50 Mo par défaut.

Il est possible de changer la taille maximum en allant dans Fichier -> Préférences -> Stockage options :

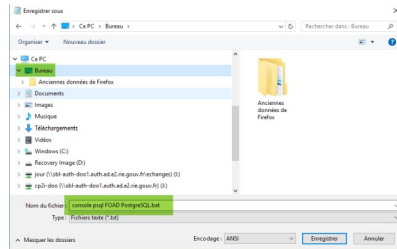
Préférences	
> Chemins	
> Debugger	
> Divers	
> Navigateur	
> Schéma Diff	
✓ Stockage	
Options	
> Tableaux de bord	
> Éditeur de diagramme entité-associ	
> Éditeur de requêtes	
Afficher les dossiers et fichiers cachés ?	<input type="checkbox"/> Faux
Dernier dossier visité	/
Format d'affichage des fichiers	Liste
Taille maximale des fichiers téléversés - upload (Mo)	800

Attention toutefois à ne pas saturer les réseaux, surtout en position de télétravail avec des fichiers trop volumineux.

Le gestionnaire de stockage permet de récupérer des sauvegardes sur le poste local ou au contraire de télécharger des sauvegardes ou fichiers à importer dans la base de données. Nous verrons des exemples dans cette formation.

3. Psql

psql est une interface en mode texte pour PostgreSQL. Il permet de saisir des requêtes de façon interactive, de les exécuter sur PostgreSQL et de voir les résultats de ces requêtes. Alternativement, les entrées peuvent être lues à partir d'un fichier. De plus, il fournit un certain nombre de méta-commandes et plusieurs fonctionnalités pour faciliter l'écriture des scripts et automatiser un nombre varié de tâches.



psql.exe est livré avec pgadmin4 sous C:\Program Files\pgAdmin 4\v6\runtime
Il est conseillé de se créer un .bat de lancement sur le bureau de votre PC qui permettra de lancer la console directement dans l'environnement du serveur de formation.

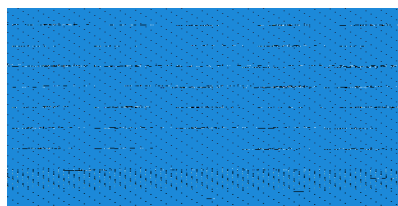
Avec un éditeur de texte (ex : *notepad++*¹¹) ou le bloc note de windows, créer un nouveau fichier et copier les lignes suivantes :

```
cls
@echo *****
@echo * Console psql - serveur de formation FOAD PostgreSQL *
@echo *****
@echo off
SET psql="C:\Program Files\pgAdmin 4\v6\runtime\psql.exe"
set user=stage01
set mdp=stage01
set dbname=stage01
set serveur=10.167.71.3
set port=5432
%psql% -h %serveur% -U %user% -p %port% -d %dbname%
@echo.
```

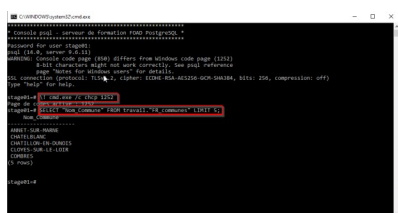
Veillez à remplacer dans les lignes ci-dessus 'stage01' par votre numéro d'utilisateur pour la formation (ex : stage12)

Puis enregistrer ce fichier sur votre bureau en le nommant 'console psql FOAD PostgreSQL.bat'

11 - <https://notepad-plus-plus.org/>



Au lancement du . bat, la console vous demande le mot de passe :



Taper votre mot de passe...
 Noter alors le message d'alerte sur le 'windows code page <1252>'.
 Pour changer cela exécuter les instructions décrites dans les notes pour les utilisateurs Windows en bas de la

page¹²

La syntaxe est : `! cmd.exe /c chcp 1252`

On a également ci-dessous un exemple d'une requête qui peut être rentrée directement dans la console psql (terminée par ;) :

```
select "Nom_Commune" from travail."FR_communes" LIMIT 5;
```

nb : CTRL C permet d'interrompre une requête.



Complément

Notez l'utilisation de LIMIT (pour ne traiter que les n premières occurrences)

Nous ne développerons pas beaucoup l'usage de psql dans ce cours.

psql est documenté ici : <https://docs.postgresql.fr/current/app-psql.html>¹³

Il existe des méta-commandes commençant par \.

ex : `\i nomfichier`

Lit l'entrée à partir du fichier nomfichier et l'exécute comme si elle avait été saisie sur le clavier. Permet par exemple d'exécuter un script sql.

À noter que l'on peut également utiliser pour ce faire la commande:

```
$ psql -f monfichier.sql
```

4. DBManager

DBManager (*gestionnaire BD* dans le menu *Base de données*) est le plugin à privilégier sous QGIS pour accéder aux bases de données. Son utilisation est détaillée dans la formation *QGIS perfectionnement – Module SQL*¹⁴ (PDF).

Les requêtes SQL peuvent être réalisées dans le requêteur SQL de Pgadmin ou dans DBManager sous QGIS.

12 - <https://docs.postgresql.fr/12/app-psql.html>

13 - <https://docs.postgresql.fr/current/app-psql.html>

14 - <http://www.geoinformations.developpement-durable.gouv.fr/perfectionnement-a-qgis-a2904.html>

Le choix dépend un peu des habitudes de chacun. Les personnes familiarisées avec PostgreSQL préféreront en général PgAdmin, les utilisateurs travaillant essentiellement sous QGIS pourront se passer de l'apprentissage de PgAdmin et travailler dans DBManager qui dans ses dernières versions dispose d'un assistant de requête SQL et d'une coloration syntaxique.

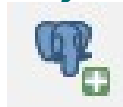
Dans tous les cas la visualisation des couches géométriques se fera dans QGIS



Méthode : Gestion des connexions

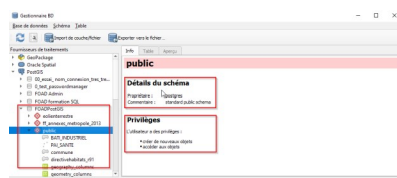
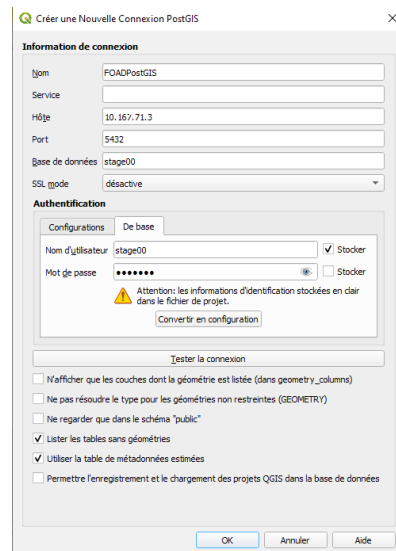
DBManager ne gère pas directement les connexions aux bases de données. Pour se connecter à une base de données PostGIS il faut passer par « **Ajouter des**

couches PostGIS »



puis

« **Nouveau** »



le bouton 'Connecter' permet ensuite de réaliser la connexion.

Dès lors la connexion apparaît dans le compartiment PostGIS de DBManager

Les métadonnées de chaque schéma

sont disponibles dans l'onglet 'info' de la fenêtre de droite.



Conseil

Il est possible d'utiliser DBManager pour créer des nouveaux schémas, mais pour la gestion fine (droit, rôle,...) il est conseillé d'utiliser PgAdmin (ou psql pour ceux qui maîtrisent cet outil).



Complément : Documentation

On trouvera plus d'informations sur DBManager dans la *documentation de QGIS*¹⁵.

C. Organisation du serveur de formation

Pour la formation vous devrez vous connecter au serveur selon les paramètres qui vous ont été fournis.

Il est possible qu'un filtrage IP soit actif pour les stagiaires, dans ce cas, les tuteurs vous demanderont de fournir l'adresse IP de votre poste.

Si vous ne pouvez pas accéder au serveur de formation, il est éventuellement possible de suivre cette formation en installant une instance locale de PostgreSQL/PostGIS, par exemple

à partir des packages de l'OsGeo : <http://download.osgeo.org/postgis/windows/>¹⁶

Alternativement il est possible de télécharger les sources à partir du *site officiel* ¹⁷, (installateur de EnterpriseDB pour Windows).

A titre d'information on peut se reporter *aux documents d'installation*¹⁸ de la société Dalibo.

Demander aux tuteurs quelle est la version recommandée pour la formation.

Pour la formation, le serveur mis en place est le suivant :

- Nom : celui que vous voulez
- **Hôte : 10.167.71.3**
- **Port TCP : 5432**
- Service : laissez vide
- **Base maintenance : stageXX**
- **Nom utilisateur : stageXX**
- **Mot de passe : stageXX**
- **XX** doit être remplacé par le numéro de stagiaire qui vous a été attribué (ex : stage01).

16 - <http://download.osgeo.org/postgis/windows/>

17 - <https://www.postgresql.org/download/>

18 - <https://public.dalibo.com/exports/formation/manuels/modules/b/b.handout.html>

Créer - Serveur
✕

General
Connexion
SSL
Tunnel SSH
Avancé

Nom d'hôte / Adresse

Port

Base de données de maintenance

Nom utilisateur

Mot de passe

Enregistrer le mot de passe ?

Rôle

Service

✕ Annuler
 Réinitialiser
📄 Enregistrer



Méthode

La mise en place d'un serveur PostgreSQL doit être accompagnée d'une réflexion sur :

- la gestion des bases (organisation des bases, schémas, tables,...)
- Les méthodes de chargement des données
- La création de vues permettant de mettre en forme la donnée pour les utilisateurs.
- la gestion des utilisateurs et des droits
- les opérations de maintenance (sauvegardes, indexation,...)
- Une analyse des risques, voir par exemple *les critères DICT*¹⁹

19 - https://fr.wikipedia.org/wiki/S%C3%A9curit%C3%A9_des_syst%C3%A8mes_d%27information

La gestion des droits

Les rôles	23
Propriétaire d'un objet	25
Droits sur les objets	26
Pas-à-pas sur un exemple	29
01 (tutoré) - gestion des droits	49
ASGARD	49

A. Les rôles

PostgreSQL gère les droits d'accès aux bases de données en utilisant le concept de rôles.

Bien que depuis la version 8.1 de PostgreSQL les deux concepts soient confondus, il est utile de distinguer :

- les **rôles de connexion** : permettant de se connecter aux serveurs
- les **rôles de groupes** permettant de gérer les droits d'accès aux bases et à leurs objets.



Conseil

Comme nous allons le voir, une bonne démarche est d'affecter un rôle de connexion à 1 ou plusieurs rôles de groupe qui définissent ses droits.



Seuls les rôles disposant de l'attribut **LOGIN** peuvent être utilisés comme rôle de connexion :

Pour créer un rôle de connexion on utilisera en SQL :

```
CREATE ROLE nom LOGIN ;
```

(dans les anciennes versions on utilisait `CREATE USER nom;` qui utilise LOGIN par défaut).

On peut, plus simplement, utiliser Rôle de connexion clic droit 'Ajouter un rôle de connexion' dans pgAdmin.

nb : La saisie d'un mot de passe est obligatoire pour définir un rôle de connexion (onglet 'définition' sous Pgadmin).

Nous verrons un peu plus loin comment les ordres SQL, que l'on utilise ici à titre pédagogique, peuvent être construits avec les assistants de l'interface de pgAdmin. Il n'est donc pas utile de mémoriser précisément la syntaxe SQL. Nous mettrons en pratique ces notions avec un 'pas à pas' à l'aide de pgAdmin... ne soyez donc pas effrayé par l'apparente complexité de cette étape de présentation des concepts... la mise en pratique sur des cas relativement simple vous permettra d'y voir plus clair !



Conseil

Comme indiqué au début de ce module il est nécessaire que le rôle de connexion dispose des *droits de connexion* qui sont contrôlés dans le fichier `pg_hba.conf`, sinon il ne pourra se connecter physiquement sur le serveur.



Méthode : Statut de superutilisateur

Les **superutilisateurs** ne sont pas pris en compte dans les vérifications des droits (ils outrepassent le système de droits), sauf pour le droit de connexion ou pour initier une réplication. C'est donc un droit dangereux. Il est préférable de faire la grande majorité du travail, avec un rôle qui n'est pas superutilisateur et de ne pas attribuer ce droit aux autres utilisateurs.

Pour donner un droit superutilisateur on utilisera en SQL :

```
CREATE ROLE nom SUPERUSER;
```

Il faut avoir le droit SUPERUSER pour créer un nouveau superutilisateur.



Méthode : Droit de création de base de données

Les **droits de création de bases** doivent être explicitement données à un rôle (à l'exception des super-utilisateurs qui passent au travers de toute vérification de droits). Pour créer un tel rôle, utilisez en SQL :

```
CREATE ROLE nom_utilisateur CREATEDB
```



Méthode : Droit de création de rôle

Un rôle doit se voir explicitement donné le **droit de créer d'autres rôles** (à l'exception des super-utilisateurs qui passent au travers de toute vérification de droits). Pour affecter un tel droit, utilisez :

```
CREATE ROLE nom CREATEROLE
```

Un rôle disposant du droit **CREATEROLE** peut aussi modifier et supprimer d'autres rôles, ainsi que donner ou supprimer l'appartenance à ces rôles.

C'est donc un droit important, néanmoins, pour créer, modifier, supprimer ou changer l'appartenance à un rôle superutilisateur, le statut de superutilisateur est requis.



Méthode : Droit d'initier une réplication en flux

Un rôle doit se voir explicitement donné le **droit d'initier une réplication en flux** (sauf pour les superutilisateurs, puisqu'ils ne sont pas soumis aux vérifications de permissions). Un rôle utilisé pour la réplication en flux doit toujours avoir le droit LOGIN. Pour affecter un tel droit, utilisez :

```
CREATE ROLE nom REPLICATION LOGIN.
```

En pratique on n'utilisera généralement pas ce droit (lié à la réplication entre bases) pour les besoins courant des services.



Méthode : Mot de passe

Un **mot de passe** est significatif si la méthode d'authentification du client exige que le client fournisse un mot de passe quand il se connecte à la base (voir le paragraphe 'Connexions' en début de ce module et la *gestion du fichier pg_hba.conf*). En SQL, on peut indiquer un mot de passe lors de la création d'un rôle avec :

```
CREATE ROLE nom_utilisateur PASSWORD 'le_mot_de_passe'
```



Conseil : Bonne pratique pour la création des rôles

Une bonne pratique est de créer un rôle qui dispose des droits **CREATEDB** (création de base) et **CREATEROLE** (création de nouveaux rôles) mais qui n'est pas un superutilisateur, et d'utiliser ce rôle pour toute la gestion des bases de données et des rôles. Cette approche évite les dangers encourus en travaillant en tant que superutilisateur pour des tâches qui n'ont pas besoin de cet état.



Méthode : Appartenance d'un rôle

Un **rôle de groupe** est un rôle sans attribut **LOGIN**

(en réalité un rôle qui a le privilège **LOGIN** peut aussi être un groupe, mais nous n'utiliserons pas cette possibilité dans ce cours).

exemple :

```
CREATE ROLE producteur ;
```

On peut lui ajouter des membres en utilisant **GRANT** et **REVOKE** :

Exemple :

```
GRANT producteur to user1 ;
```

Les rôles membres (ici user1) qui ont l'attribut **INHERIT** peuvent utiliser automatiquement les droits des rôles dont ils sont membres.

Ici si user1 a été créé par la commande `CREATE ROLE user1 LOGIN INHERIT ;` héritera des droits du rôle producteur dont il est devenu membre par l'ordre **GRANT**.

Attention :

Si un utilisateur n'est pas créé avec l'attribut **INHERIT**, il n'hérite pas par défaut des privilèges des groupes dont il est membre. L'ordre **SET ROLE** permet de lui accorder pour la session en cours les privilèges d'un rôle de groupe dont il est membre.

Exemple : si user1 est membre de *groupeproducteur*, il disposera des privilèges du groupe pour la session en cours avec :

```
SET ROLE groupeproducteur ;
```

On peut révoquer les droits avec **REVOKE**.

```
REVOKE producteur FROM user1 ;
```



Complément : Documentation

La notion de rôle est très souple, on consultera *la documentation*²⁰ pour voir les détails des différentes possibilités de mise en œuvre.

B. Propriétaire d'un objet

Propriétaire d'un objet et droits d'accès

Quand un objet est créé, il se voit affecter un propriétaire. Le propriétaire est normalement le rôle qui a exécuté la requête de création. Pour la plupart des objets, l'état initial est que seul le propriétaire (et les superutilisateurs) peuvent faire quelque chose avec cet objet. Pour permettre aux autres rôles de l'utiliser, des droits doivent être donnés.

- Il n'est pas nécessaire d'accorder des droits au propriétaire d'un objet car, par défaut, le propriétaire possède tous les droits.
- Les droits applicables à un objet particulier varient selon le type d'objet (table, fonction...).
- Le droit de modifier ou de détruire un objet est le privilège du seul propriétaire.

Un objet peut se voir affecter un nouveau propriétaire avec la commande **ALTER** correspondant à l'objet, par exemple ²¹. Les superutilisateurs peuvent toujours le faire. Les rôles ordinaires peuvent seulement le faire s'ils sont le propriétaire actuel de l'objet (ou un membre du rôle propriétaire, puisqu'un rôle de groupe peut être propriétaire d'un objet).

C. Droits sur les objets

Droits sur les bases de données

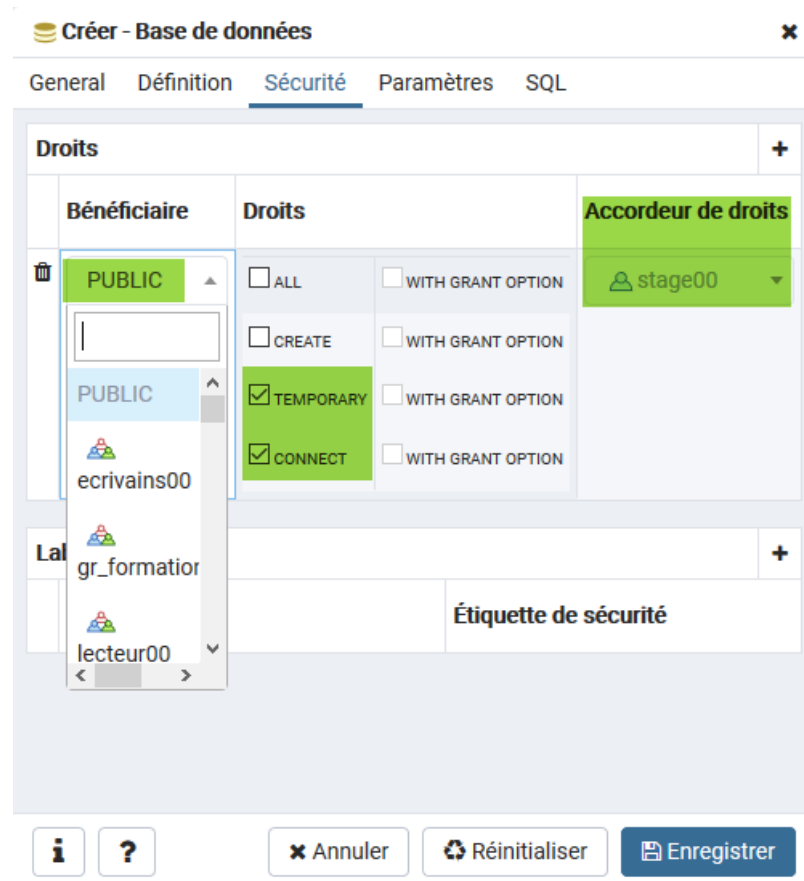
Lors de la création d'une base on peut indiquer des **droits globaux sur la base**.

Sous pgAdmin cela peut être indiqué en création de base de données dans l'onglet *sécurité*.

20 - <http://docs.postgresql.fr/12/role-membership.html>

21 - <http://docs.postgresql.fr/12/sql-altertable.html>

Ci-dessous le rôle `public` (voir compléments à la fin de ce paragraphe) se voit accorder les droits **TEMPORARY** et **CONNECT** par `stage00` :



Les droits globaux sur les bases sont :

- **CREATE** (`CREATE ON DATABASE nom_base`) : autorise la création de nouveaux schémas dans la base de données.
- **TEMPORARY** (`TEMPORARY ON DATABASE nom_base`) : autorise la création de tables temporaires.
- **CONNECT** (`CONNECT ON DATABASE nom_base`) : autorise la connexion à la base.

Les droits accordés au rôle `public` le sont pour tous les utilisateurs.

Dans l'exemple ci-dessus, tous les utilisateurs peuvent se connecter à la base en cours de création et créer des tables temporaires (sélection des options **TEMP** et **CONNECT** pour le groupe `public`).

En SQL on utilise la commande **GRANT**.

Exemple :

```
GRANT CREATE ON DATABASE nom_base TO public ;
```

donne le droit de création de schémas à tous les utilisateurs.

L'assistant de création de PgAdmin crée automatiquement le SQL en fonction de l'onglet de définition des droits. Cette syntaxe est visible dans le dernier onglet :

```

1 CREATE DATABASE demo
2 WITH
3 OWNER = stage00
4 ENCODING = 'UTF8'
5 CONNECTION LIMIT = -1;
6
7 GRANT TEMPORARY, CONNECT ON DATABASE demo TO PUBLIC;

```

Par défaut les droits **CONNECT** et **TEMP** sont accordés à *public*.

L'option `WITH GRANT OPTION` permet d'autoriser le rôle à lui-même accorder ce droit à d'autres (*possibilité de transmission*²²).

(A utiliser avec précaution).

Droits sur les schémas

Les droits sur les schémas sont :

- **USAGE** : autorise l'accès aux objets contenus dans le schéma.
- **CREATE** : autorise la création de nouveaux objets dans les schémas.

```

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
ON SCHEMA nom_schéma [, ...]
TO { [ GROUP ] nom_rôle | PUBLIC } [, ...] [ WITH GRANT OPTION ]

```

Droits sur les objets d'un schéma

Il est possible de gérer les droits sur les objets d'un schéma (en particulier les TABLES).

Nom	Description
ALL	Autorise tous les privilèges (selon le type de l'objet)

22 - <http://docs.postgresql.fr/12/ddl-priv.html>

Nom	La gestion des droits
	Description
SELECT	Permet la sélection sur tout ou partie des colonnes d'une table (requête SELECT)
INSERT	Permet d'insérer un enregistrement dans une table
UPDATE	Permet la mise à jour des champs d'une table.
DELETE	Permet d'effacer un enregistrement de la table
REFERENCE S	Droit requis pour création de clé étrangère (où il faut faire référence à une table tierce)
TRUNCATE	Permet d'effacer une table ou un ensemble de tables (récupération immédiate de l'espace disque sans VACCUUM) Truncate = Delete+Vacuum pour faire simple.
TRIGGER	Permet de créer de nouveaux déclencheurs associés à la table
TEMPORARY	Permet de créer des tables temporaires
EXECUTE	Permet d'exécuter une fonction (précise)

Nota bene : il est inutile de donner des droits sur des tables aux utilisateurs si vous ne donnez pas le droit USAGE sur le schéma contenant lesdites tables aux utilisateurs.

Par défaut le droit EXECUTE sur les fonctions est accordé au rôle *public*.

Pour plus de détails, on se rapportera à la documentation de la *commande GRANT*²³.



Fondamental : Étude préalable

On le voit, on peut aller très loin dans la gestion des droits. Dans la plupart des cas on se contentera d'une gestion relativement simple (gestion au niveau des bases et schémas avec droit en lecture et/ou écriture).

Il faut cependant retenir que la stratégie de gestion des droits est fondamentale et doit faire l'objet d'une étude par l'administrateur.



Conseil : Superutilisateur et administrateur

On différenciera généralement un superutilisateur (ex : superuser) et un administrateur (geoadmin).

- **Superutilisateur** : Il possède tous les droits. Cet utilisateur ne devrait être utilisé qu'en cas de nécessité. Il est donc conseillé de créer un rôle d'administrateur des bases de données qui n'aura pas ce droit de superutilisateur.
- **Administrateur des bases et utilisateurs (geoadm ou geoadmin)** : il a les droits, CREATEDB, CREATEROLE, INHERIT.

Exemple en SQL :

```
CREATE ROLE geoadm1 LOGIN
CREATEDB CREATEROLE
VALID UNTIL 'infinity';
```

Rappel : on identifie qu'il s'agit d'un rôle de connexion grâce au mot clef LOGIN



Remarque

Si le niveau des droits proposés peut être très fin, la pratique montre que le nombre de profils reste en général réduit. On pourra retenir en première approche :

- les administrateurs (superuser et geoadmin)
- les lecteurs
- les producteurs

*ASGARD*²⁴ qui est système simplifié de gestion des droits développé par le *SNUM*²⁵, et sur lequel nous reviendrons rapidement, ajoute un rôle 'editeur' permettant de **modifier** des données sans pouvoir créer ou supprimer des objets (tables, vues,...).



Complément : Pseudo rôle PUBLIC et schéma public

PUBLIC peut être considéré comme un rôle défini implicitement auquel tous les autres rôles appartiennent.

Tout rôle particulier aura la somme des privilèges qui lui sont accordés directement, des privilèges accordés à tout rôle dont il est membre et des privilèges accordés à PUBLIC (dont il est membre par défaut).

Comme indiqué plus haut, PUBLIC a des privilèges par défaut :

- CONNECT et CREATE TEMP TABLE pour les bases de données;
- EXECUTE, privilège pour les fonctions;
- USAGE , privilège pour les langues.

Ces privilèges peuvent être révoqués :

```
REVOKE ALL ON DATABASE MaBase FROM public;
```

Le schéma public est un schéma qui est créé par défaut lors de la création d'une base. PostgreSQL alloue des privilèges par défaut au rôle PUBLIC sur le schéma public, ce qui fait que tous les rôles en héritent. Ces droits peuvent également être révoqués.

D. Pas-à-pas sur un exemple

Il est maintenant temps de mettre en pratique les notions apprises précédemment pour :

- créer une base, des schémas, importer des tables ;
- accorder des droits ;
- vérifier à partir de QGIS.

Nous allons voir que l'utilisation de PgAdmin permet de ne pas mémoriser la syntaxe exacte des commandes SQL, les assistants générant les requêtes pour vous.

24 - https://snum.scenari-community.org/Asgard/Documentation/#SEC_Principes

25 - https://www.ecologie.gouv.fr/secretariat-general#scroll-nav__8

1. Création de la base et des schémas

Lancer PgAdmin



pgAdmin

Management Tools for PostgreSQL



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.

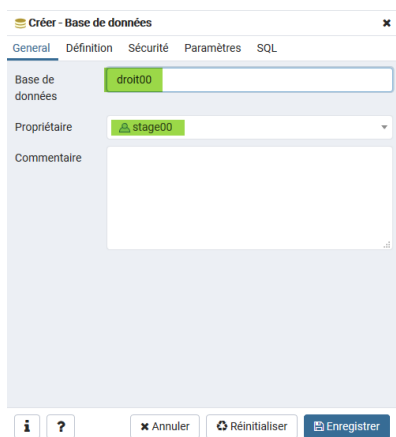
Dans la suite de la formation remplacer XX ou 00 par le numéro qui vous est attribué (ex : *droit01* à la place de *droitXX* ou de *droit00*)

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"

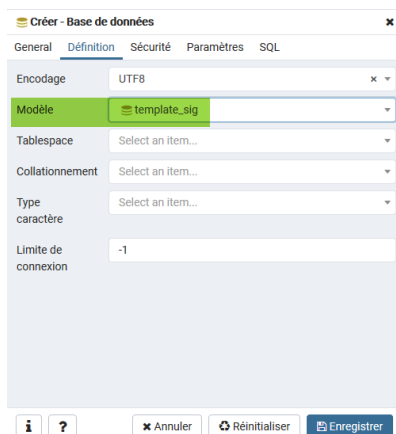


Faire un clic droit sur *Bases de données*



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* :

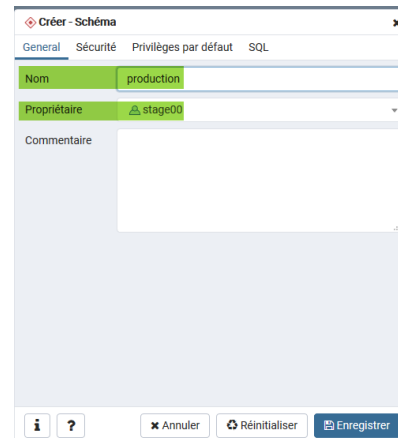
- Pour le schéma *production*, nous aurons deux types d'utilisateurs :
 - les *écrivainsXX* qui auront les droits de modifier les tables existantes du schéma *production* de la base *droitXX*
 - les *lecteursXX* qui n'auront que les droits de les visualiser.
- Pour le schéma *consultation*, tous les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

La gestion des droits

Pour créer le schéma *production* :

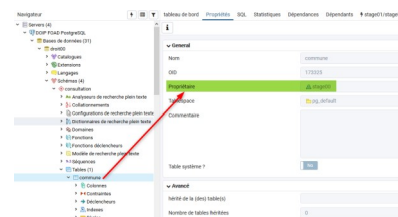
- faire un clic droit sur la base *droitXX* → **Créer** → **schéma...**
- Nous rajouterons les droits plus tard. Faire OK.

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).



Si vous utilisez pgadmin en version server il est nécessaire de téléverser au préalable le fichier *gestiondesdroits.backup* sur le serveur en utilisant le *gestionnaire de stockage* de pgadmin.

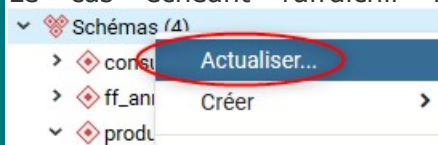
- Se positionner sur la base *droitXX*, puis clic droit 'restaurer' :
- Indiquer le fichier *gestiondesdroits.backup*.
- Dans l'onglet *Options de restauration* cocher dans *sections* : *pre-data* et *data* et dans *ne pas enregistrer* : *propriétaire*.
- Lancer la restauration.

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*.

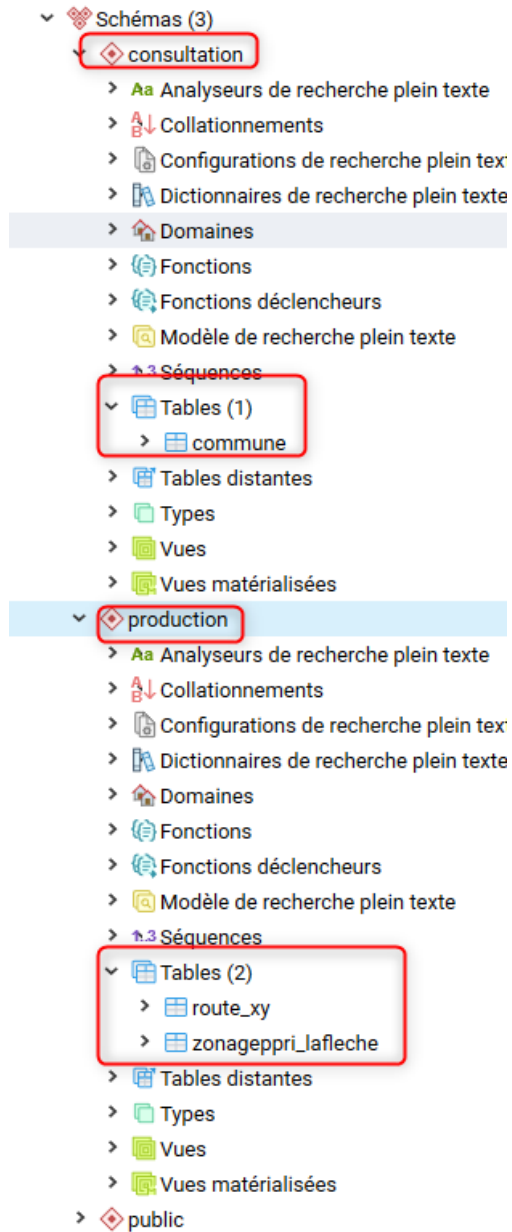
Le cas échéant rafraîchir l'affichage par un clic droit puis 'actualiser...'



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.

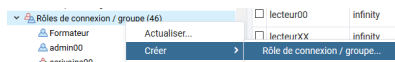
La base doit maintenant ressembler à ceci :



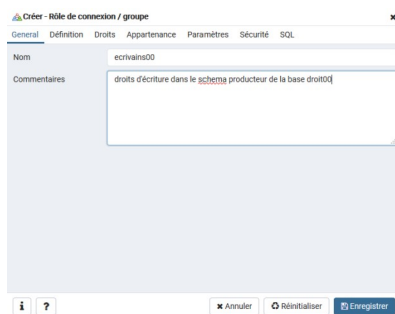
2. Création des rôles



Méthode : Nous allons maintenant créer les deux rôles de groupe



groupe



Ajouter le rôle de groupe *ecrivainsXX* :

- clic droit sur *Rôles de connexion /*

- puis **Ajouter un rôle de groupe...**

Rappel : Un rôle de connexion ne se distingue d'un rôle de groupe que par l'activation de la possibilité 'connexion autorisé' (LOGIN en SQL)

Vérifier dans l'onglet *SQL* que vous avez (remplacer *XX* par votre numéro de stagiaire) :

```
CREATE ROLE ecrivainsXX WITH
NOLOGIN
NOSUPERUSER
NOCREATEDB
NOCREATEROLE
INHERIT
NOREPLICATION
CONNECTION LIMIT -1;
COMMENT ON ROLE ecrivains00 IS 'droits d''écriture dans le schema
producteur de la base droit00';
IS 'droit d''écriture dans le schema producteur';
```

On utilise ici très peu les options de commande **CREATE ROLE** que l'on pourra quand même relire au passage...

De même, créer le rôle *lecteurXX* :

```
CREATE ROLE lecteurXX WITH
NOLOGIN
NOSUPERUSER
NOCREATEDB
NOCREATEROLE
INHERIT
NOREPLICATION
CONNECTION LIMIT -1;
```

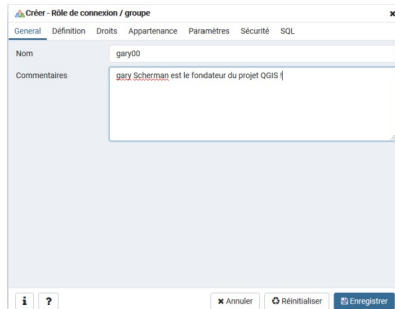


Méthode : Rôles de connexion

Pour l'instant les rôles *lecteurXX* et *ecrivainsXX* sont identiques et n'accordent aucun droit spécifique.

Nous allons maintenant créer deux rôles de connexion :

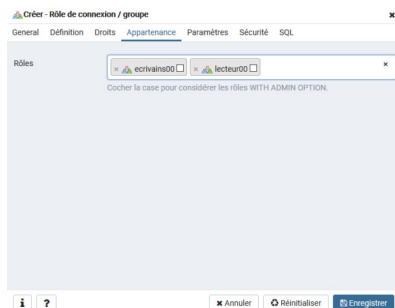
- *garyXX* qui appartiendra au groupe *lecteurXX* et *ecrivainsXX*,
- *michaelXX* qui appartiendra au groupe *lecteurXX* (rappel : remplacer XX par le numéro qui vous a été attribué)



- Faire clic droit sur rôle de connexion / groupe

Rôles de connexion / groupe (46)

- puis créer un rôle en activant dans l'onglet droits '**connexion autorisé**'
- Lui donner un mot de passe dans l'onglet *Définition* (ex : *garyXX*) (La saisie d'un mot de passe est obligatoire pour définir un rôle de connexion).
- Laisser les autres paramètres par défaut dans l'onglet *Droits* (hérite des droits des rôles parents activé).



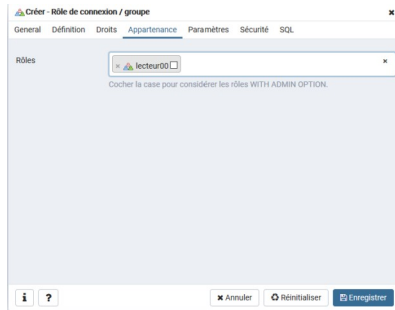
- Indiquez dans l'onglet appartenance du rôle qu'il est membre de *ecrivainsXX* et de *lecteurXX*

La syntaxe SQL dans l'onglet SQL doit être :

```
CREATE ROLE gary00 WITH
LOGIN
NOSUPERUSER
```

```
NOCREATEDB
NOCREATEROLE
INHERIT
NOREPLICATION
CONNECTION LIMIT -1
PASSWORD 'xxxxxxx';
GRANT ecrivains00, lecteur00 TO gary00;
COMMENT ON ROLE gary00 IS 'gary Scherman est le fondateur du projet QGIS
!';
```

La gestion des droits



De même créer le rôle de connexion *michaelXX* (Michael Stonebraker est le fondateur du projet PostgreSQL) et indiquer qu'il est membre du groupe *lecteurXX*.

3. Utilisation avec DBManager sous QGIS

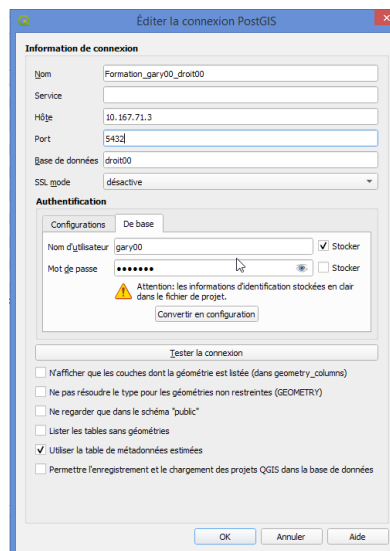


Méthode : Nous allons maintenant utiliser QGIS pour vérifier les droits avec les deux rôles.

Comme indiqué précédemment dans le paragraphe sur la gestion des connexions dans QGIS, créer la connexion suivante ci-contre.

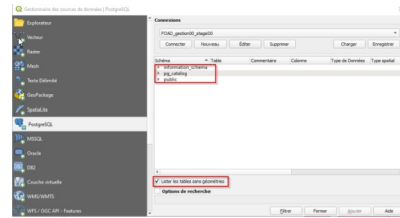
On souhaite que cette connexion soit toujours réalisée avec l'utilisateur *garyXX*, on utilisera donc l'option **Enregistrer le nom d'utilisateur**.

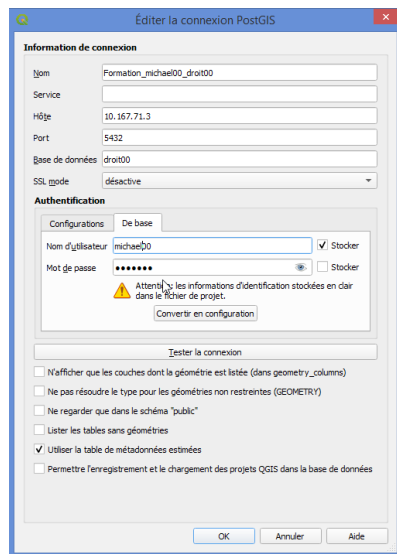
Le recours à la table de métadonnées estimées (option) permet d'optimiser la rapidité d'ouverture des couches.



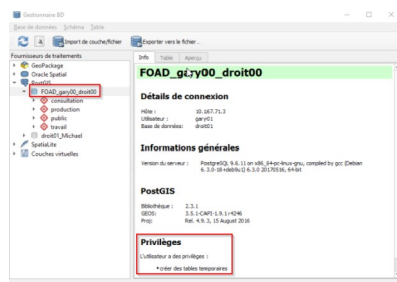
La gestion des droits

Constater qu'il n'y a aucune table avec géométrie accessible pour ce rôle (le rôle n'accède pas, pour l'instant, aux schémas *consultation* et *production*).



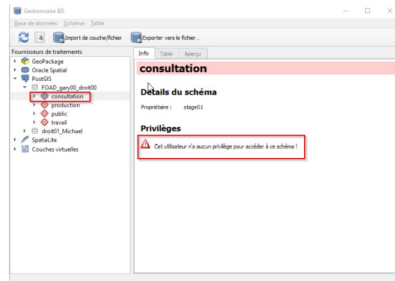


Faire **Fermer**, puis créer cette connexion.

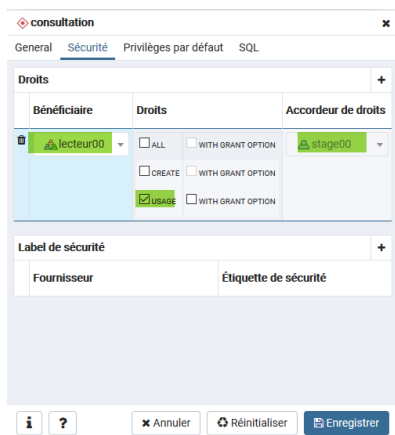


Lancer *DBManager* (Menu *Base de données* -> *Gestionnaire BD*) et vérifier les droits pour les deux connexions (le cas échéant faire clic droit reconnecter si nécessaire).

Pour l'instant le seul privilège de *garyXX* est de créer des tables temporaires.



On peut vérifier qu'il n'a aucun droit sur les schémas en se positionnant dessus.



Indiquons maintenant sous *pgAdmin* les droits sur le schéma *consultation* pour le rôle de groupe *lecteurXX*.

- Se positionner sur le schéma *consultation*, faire un clic droit puis **Propriétés**
- Onglet *sécurité* (permet d'indiquer les droits sur le schéma) :

Cliquer sur la croix à droite de Droits pour ajouter une nouvelle ligne de droits.

Sélectionner le rôle *lecteurXX*

dans le menu déroulant, puis dans la colonne Droits activer **USAGE**.

Ceci nous permet d'accorder le droit USAGE (on ne souhaite pas que les membres du groupe lecteur puisse eux-mêmes créer de nouveaux objets dans le schéma, on ne leur donne pas le privilège CREATE).

Notez que celui qui accorde les droits est le rôle en cours (ici stage00)

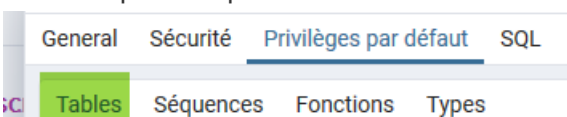
Valider.

On souhaite maintenant préciser les droits (privilèges) par défaut sur les **futurs** objets d'un schéma **créés par un utilisateur ou par des rôles dont il est membre**.

voir la documentation de PostgreSQL sur *ALTER DEFAULT PRIVILEGES*²⁶

Nous souhaitons préciser les droits par défaut (dans notre cas le droit en lecture) pour le rôle de groupe lecteurXX sur les éventuels futures objets créés par **garyXX** (on admet pour les besoins de la démonstration que garyXX aura des droits de création dans le schéma consultation).

Relancer la boîte de dialogue des propriétés du schéma consultation. Nous allons utiliser l'onglet **Privilèges par défaut** pour disposer du canevas de l'ordre SQL...

- Dans l'onglet *Tables*,  ajouter une ligne en cliquant sur la croix à droite.
- Choisir le bénéficiaire *lecteurXX*, (remplacer XX par votre numéro)

26 - <https://docs.postgresql.fr/9.5/sql-alterdefaultprivileges.html>

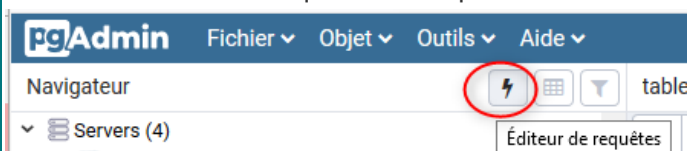
- Dans la colonne 'droits' activer **SELECT**

Dans pgadmin, il n'est pas possible de choisir l'accordeur de droit qui est par défaut le propriétaire du schéma, hors nous voulons absolument que l'accordeur de droit soit garyXX pour que les droits par défaut s'appliquant aux futurs objets créés par lui.

Aller dans l'onglet SQL et copier l'ordre SQL (sélectionner les lignes et appuyez que CTRL C) :

```
ALTER DEFAULT PRIVILEGES IN SCHEMA consultation
GRANT SELECT ON TABLES TO lecteur00;
```

Activer l'éditeur de requête en cliquant sur



Copier (CTRL V) le SQL dans l'éditeur de requête.

Important : Pour indiquer que ce sont les droits par défaut pour les éventuels futurs objets de garyXX, nous allons modifier l'ordre SQL dans l'éditeur, pour prendre provisoirement le rôle de garyXX avec une instruction SQL SET ROLE juste avant le ALTER DEFAULT PRIVILEGES

Ceci est possible car vous devez être connecté avec le rôle de connexion **stageXX** qui est **SUPERUSER** et qui a donc le droit d'utiliser SET ROLE.

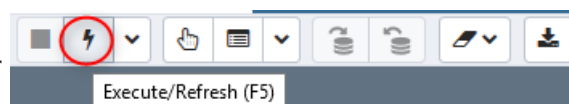
(pour utiliser SET ROLE nomrole, Il est obligatoire que l'utilisateur de la session courante soit membre du rôle nomrole - si l'utilisateur de la session est **superutilisateur**, tous les rôles sont utilisables.)

Ajouter SET ROLE garyXX ; avant ALTER DEFAULT... et RESET ROLE (pour réinitialiser le rôle à la valeur de la session) en dernière ligne.

La commande complète dans l'éditeur SQL doit donc être :

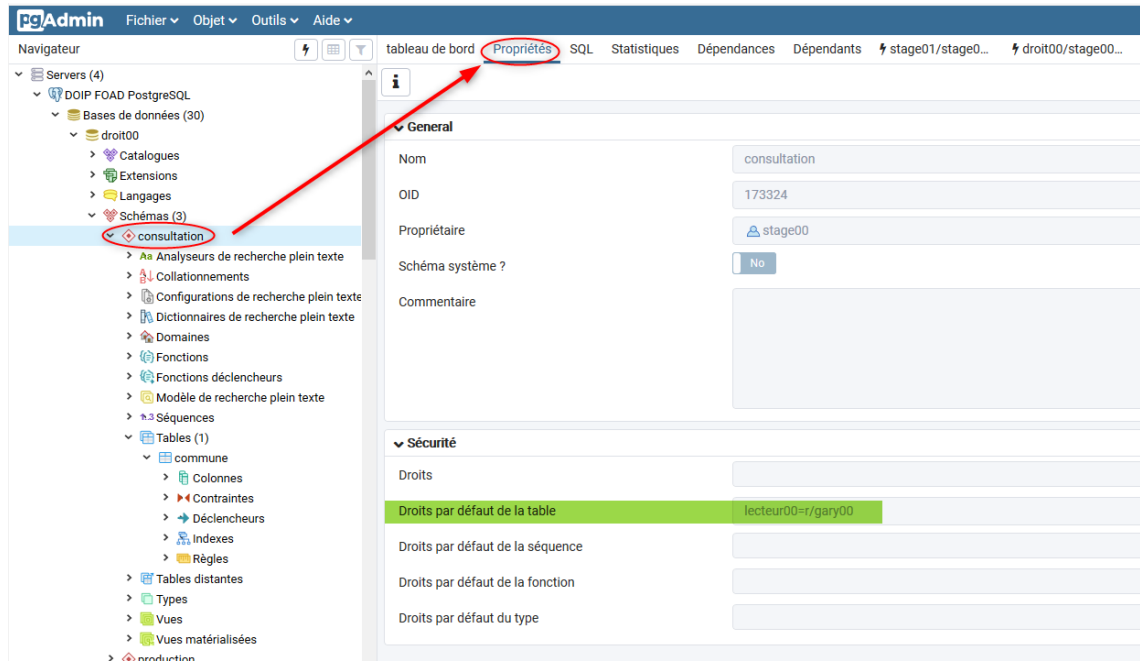
```
SET ROLE garyXX ;
ALTER DEFAULT PRIVILEGES IN SCHEMA consultation
GRANT SELECT ON TABLES TO lecteurXX;
RESET ROLE ;
```

Exécuter la commande SQL en cliquant sur



Vous devez avoir un message 'Requête exécutée avec succès en 67 msec

En étant positionné sur le schéma *consultation* dans le navigateur, vérifier dans l'onglet propriété du schéma que vous avez bien : droits par défaut de la table '*lecteur00=r/gary00*'



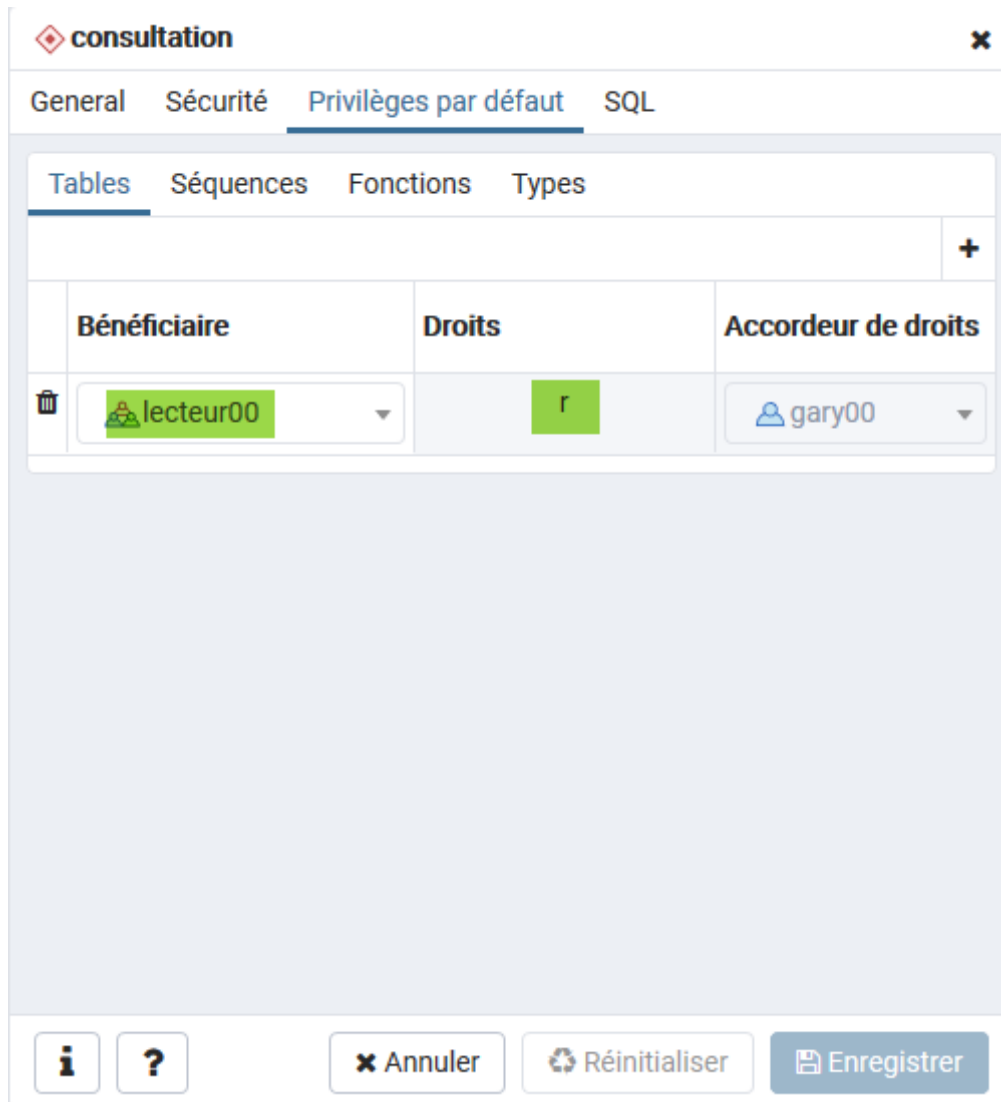
(Remplacer le 00 par votre numéro de stagiaire)

Ceci indique que gary00 a octroyé les privilèges r (SELECT) au rôle lecteur00

Vérifier maintenant les droits dans DBManager en actualisant avec le bouton

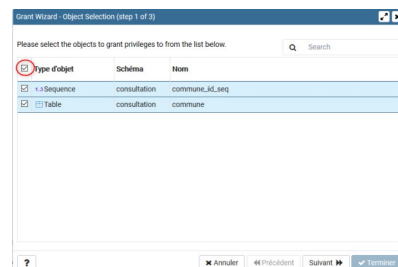


- Constater que, sur le schéma, l'utilisateur garyXX a maintenant le privilège *Accéder aux objets*
- Se positionner sur la table *commune* dans le schéma *consultation* : constater que l'utilisateur n'a aucun privilège.
- En effet nous avons indiqué les droits par défaut qui sont les droits qui seront appliqués par défaut aux futurs objets, mais qui ne s'appliquent pas aux objets existants.



Sous PgAdmin, faire un clic droit sur le schéma *consultation*, puis *Assistant de gestion des droits*.

Cliquer sur la case à cocher en haut à gauche pour tout cocher



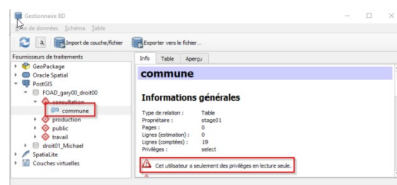
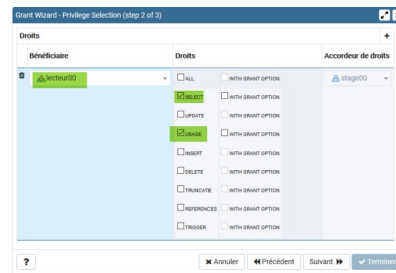
La gestion des droits

Appuyer sur le bouton 'suivant' et ajouter les privilèges *SELECT* et *USAGE* pour le rôle *lecteurXX*.

Appuyer à nouveau sur 'suivant' et Vérifier l'ordre SQL :

```
GRANT SELECT, USAGE ON SEQUENCE
consultation.commune_id_seq TO lecteurXX;
GRANT SELECT ON TABLE consultation.commune TO lecteurXX;
```

Terminer.



Vérifier les droits sous QGIS :

- vérifier que la table *commune* est maintenant chargeable sous QGIS par un glisser-lâcher dans QGIS, ou par un clic droit 'ajouter au canevas'.
- Vérifier que cette table n'est pas modifiable sous QGIS.



Complément : droit par défaut sur les schéma

Il faut bien comprendre que les droits par défaut sur les schémas ne s'appliquent que pour les futurs objets créés par un utilisateur ou un rôle dont il est membre.

Sous pgadmin on ne voit dans l'onglet *propriétés* -> *sécurité* qu'une seule ligne décrivant les droits par défaut pour les tables, mais on peut très bien avoir plusieurs rôles ayant attribués des droits par défaut.

Pour avoir une vision exhaustive on peut utiliser la requête suivante :

```
SELECT
nspname, -- schema name
defaclobjtype, -- object type
defaclacl -- default access privileges
FROM pg_default_acl a JOIN pg_namespace b ON a.defaclnamespace=b.oid;
qui renvoi par exemple si gary16b est un deuxième rôle de connexion membre de
écrivains16 ayant attribués des droits par défaut en lecture sur ses futures objets :
```

Panneau sortie			
Sortie de données			
	nspname name	defaclobjtype "char"	defaclacl aclitem[]
1	consultation	r	{lecteur16=r/gary16b}
2	consultation	r	{lecteur16=r/gary16}

On a donc ici deux lignes décrivant les droits par défaut de gary16 et gary16b sur le

schéma consultation.

Il est possible d'attribuer des droits par défaut directement dans une suite de commande SQL, par exemple :

```

droit00/stage00@DOIP FOAD PostgreSQL
Éditeur de requêtes Historique
1 SET ROLE gary16 ;
2 ALTER DEFAULT PRIVILEGES IN SCHEMA consultation
3 GRANT SELECT ON TABLES TO lecteur16;
4 RESET ROLE ;
5
6 SET ROLE gary16b ;
7 ALTER DEFAULT PRIVILEGES IN SCHEMA consultation
8 REVOKE ALL ON TABLES FROM lecteur16;
9 RESET ROLE ;
10

```

E. 01 (tutoré) - gestion des droits

Après le cours et l'exercice guidé, voici un exercice à effectuer seul sur la gestion des droits : se positionner sous PgAdmin sur le schéma *production*.


Question

Mettre en pratique les acquis pour faire en sorte que *garyXX* :

- puisse créer de nouvelles tables.
- dispose des droits sur les tables existantes (ajouter, modifier et supprimer des enregistrements) du schéma *production*
- Dispose des droits d'écriture sur les objets créé par *stageXX* dans le schéma *production*

Pour vérifier cette dernière possibilité , charger dans QGIS la table *commune_densite.shp* (fournie dans les données du stage).

Créer une connexion sur la base droitXX avec le rôle stageXX, puis en utilisant cette connexion sous DBManager (Base de données -> gestionnaire de BD)

utiliser le bouton  Import de couche/fichier... pour charger cette couche dans le schéma *production* (on appellera la table *commune_densite* et on n'utilisera pas d'option de création que nous verrons plus loin dans le cours).

Changer de connexion pour vous connectez avec le rôle *garyXX* et vérifier les privilèges.

Faire également en sorte que *michaelXX* puisse accéder en lecture seule au schéma *production* et aux objets qu'il contient.


Consigne : Envoyer un message aux tuteurs pour indiquer que vous avez réalisé

l'exercice en rappelant votre numéro de stagiaire.

Si vous avez mis des mots de passe autres que le nom de login pour *garyXX* et *michaelXX*, merci de les communiquer aux tuteurs.

F. ASGARD

Présentation

ASGARD²⁷  **ASGARD PostgreSQL** est une extension PostgreSQL créée par le Service du Numérique du Ministère de l'Ecologie pour favoriser l'usage de PostgreSQL dans les services.

ASGARD met en œuvre ;

- un modèle de gestion des droits simplifié, qui alloue par défaut des droits homogènes pour tous les objets d'un même schéma ;
- une classification des schémas en blocs fonctionnels avec des règles de nommage ;
- une nomenclature nationale des schémas pour le bloc fonctionnel « consultation » reprenant les deux niveaux de l'arborescence de la Géobase ;

L'idée centrale d'ASGARD pour la gestion des droits est de considérer que le mécanisme par défaut de PostgreSQL n'est généralement pas conforme à un comportement simplifié souhaité par les services. En effet, il implique des actions pour les créateurs d'objets s'ils souhaitent que d'autres utilisateurs aient accès à leurs objets. Lorsque cette attribution explicite de droits est réalisée à la main, elle peut rapidement devenir fastidieuse et source d'erreurs.

Avec ASGARD des droits standards sont appliqués **automatiquement** au moment de la création des objets dans les schémas et **le propriétaire est ré-affecté** de manière à rester cohérent avec le propriétaire du schéma.

Règle : Avec ASGARD Le propriétaire d'un schéma est le propriétaire de tous les objets qu'il contient.

Les profils de droits

L'idée est de définir trois profils de droits – producteur, éditeur et lecteur.

- Les LECTEURS ont accès en lecture seule aux données du schéma. En pratique, ils ont les droits USAGE sur les schémas et SELECT sur les tables et séquences.
- Les ÉDITEURS ont tous les droits des lecteurs, auxquels s'ajoutent les droits qui permettent la modification des données : INSERT, UPDATE et DELETE sur les tables ou assimilés et USAGE sur les séquences.
- Les PRODUCTEURS sont propriétaires des schémas et reçoivent donc automatiquement la propriété de tous les objets qui y sont créés. À ce titre (cf. I.3.A), ils ont tous les droits sur les schémas et les objets qu'ils contiennent. En plus de la visualisation et l'édition des données, les membres du groupe producteur peuvent créer des objets dans le schéma, les supprimer, modifier leur définition ou encore attribuer ou révoquer des droits sur ces

27 - <https://snum.scenari-community.org/Asgard/Documentation/>

objets pour les autres rôles de groupe.

Avec ASGAR, tout schéma a un rôle PRODUCTEUR (propriétaire) et au plus un rôle ÉDITEUR et un rôle LECTEUR.

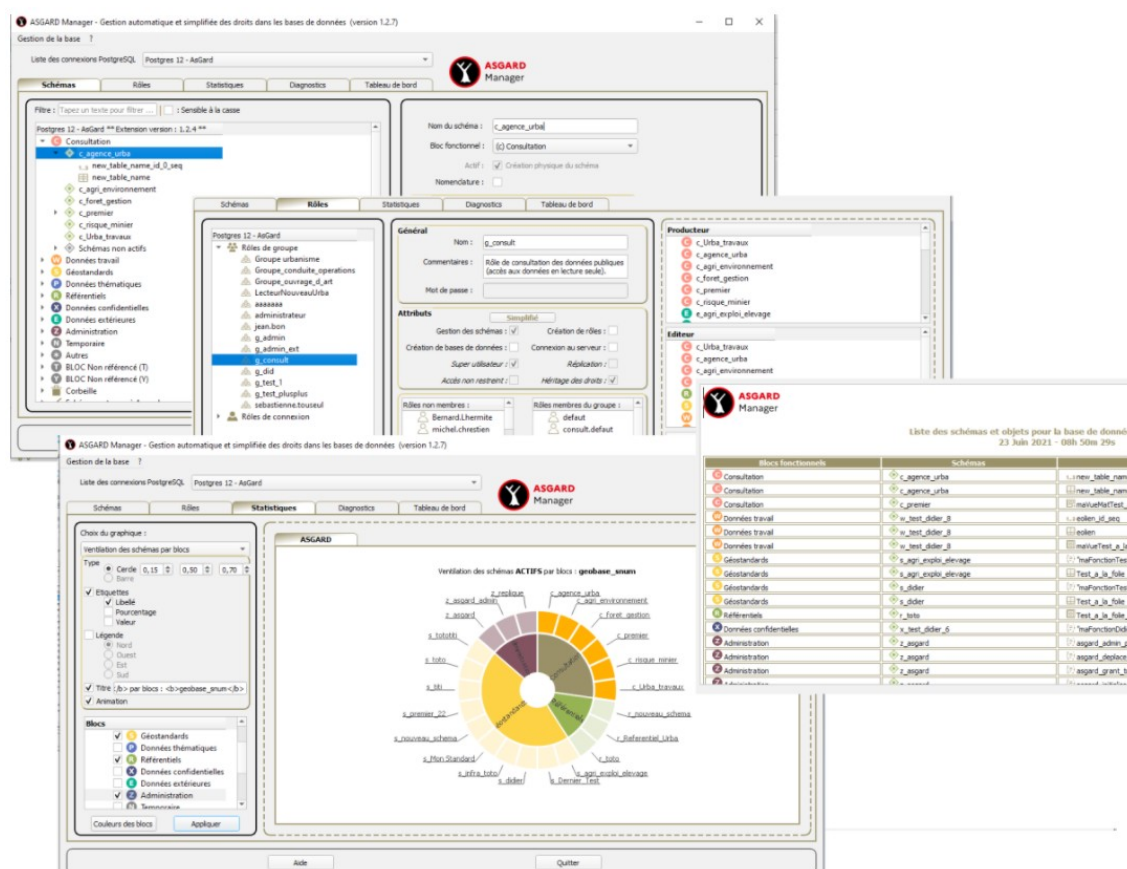
Rôles d'ASGAR

ASGAR créé automatiquement quatre rôles : trois rôles de groupe et un rôle de connexion : g_consult, g_admin, g_admin_ext et consult.default.

- g_consult a pour fonction d'offrir à ses membres un accès en lecture seule à l'ensemble du patrimoine de données du service, à l'exclusion des données confidentielles.
- g_admin est le groupe des administrateurs. Il peut créer, modifier et supprimer des bases (CREATEDB), créer, gérer et supprimer des rôles (CREATEROLE), Il a de plus le privilège CREATE sur la base, qui lui permet de créer de nouveaux schémas.
- consult.default est un rôle de connexion générique qui permet de se connecter à la base avec le mot de passe « consult.default »
- g_admin_ext est un rôle de groupe technique.

AsgardManager

Le plugin AsgardManager permet d'administrer les bases directement sous QGIS



Accompagnement

Une formation spécifique à l'utilisation d'ASGAR sera proposée courant 2022.

Sauvegarde et restauration

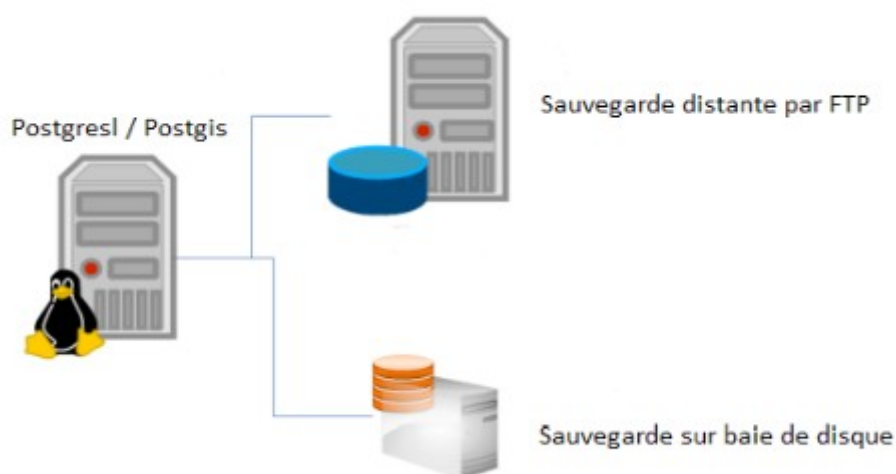
III

Sauvegarde logique (dump)	54
Sauvegarder le serveur (pg_dumpall)	60
Restauration logique	60
02 - sauvegarde et restauration	64
Utilisation d'une console PSQL	64
Sauvegarde au niveau système de fichiers	65
Archivage continu	66

Une stratégie de sauvegarde doit être définie. Elle dépend du niveau de sécurité requis. Elle doit s'inscrire dans une démarche plus globale d'analyse de la sécurité des systèmes d'information et dépend des systèmes physiques mis en place (disque RAID, baie SAN,...).

exemple de stratégie :

- sauvegarde hebdomadaire pour les bases peu évolutive, sauvegarde journalière pour les autres.
- Duplication de sauvegarde sur une baie de disque et duplication de la sauvegarde sur un serveur distant par FTP
- conservation des fichiers de sauvegarde sur un mois avec effacement automatique au-delà
- effacement manuel en cas de saturation (transmission d'un rapport de sauvegarde par mel)



Il y a trois approches fondamentalement différentes pour sauvegarder les données de PostgreSQL :

- la sauvegarde logique ;
- la sauvegarde au niveau du système de fichiers ;
- l'archivage continu.

Chacune a ses avantages et ses inconvénients.

A. Sauvegarde logique (dump)

Sauvegarder (pg_dump) avec PgAdmin

Le principe est de produire un fichier texte de commandes SQL (appelé « fichier dump »), qui, si on le renvoie au serveur, recrée une base de données identique à celle sauvegardée. PostgreSQL propose pour cela le programme utilitaire `pg_dump`. L'usage basique en ligne de commande est :

```
pg_dump base_de_donnees > fichier_de_sortie
```

la sauvegarde peut être effectuée depuis n'importe quel ordinateur ayant accès à la base. `Pg_dump` doit avoir un accès en lecture à toutes les tables que vous voulez sauvegarder, donc pour sauvegarder une base complète, vous devez pratiquement toujours utiliser un superutilisateur.

Un des gros avantages de `pg_dump` sur les autres méthodes de sauvegarde est que la sortie de `pg_dump` peut être généralement re-chargée dans des versions plus récentes de PostgreSQL, alors que les sauvegardes au niveau fichier et l'archivage continu sont tous les deux très spécifiques à la version du serveur. `Pg_dump` est aussi la seule méthode qui fonctionnera lors du transfert d'une base de données vers une machine d'une architecture différente (par exemple d'un serveur 32 bits à un serveur 64 bits).

Les options de la `Pg_dump` en ligne de commande sont décrites *ici*²⁸

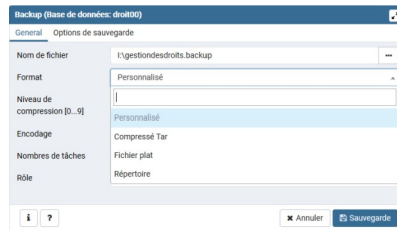
28 - <https://docs.postgresql.fr/current/app-pgdump.html>

Nous allons les examiner au travers de l'interface de pgadmin :



Méthode

Sous pgadmin Clic droit sur une base → sauvegarder (ou menu Outils → sauvegarder).



Les différents formats sont proposés :

- **Personnalisé** : archive personnalisée utilisable par `pg_restore`. Avec le format de sortie répertoire, c'est le format le plus souple, car il permet la sélection manuelle et le

réordonnancement des objets archivés au moment de la restauration. Ce format est aussi compressé par défaut.

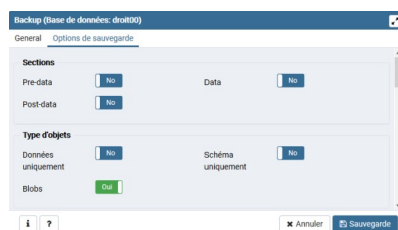
- **Tar** : archive tar utilisable par `pg_restore`. Le format tar est compatible avec le format répertoire; l'extraction d'une archive au format tar produit une archive au format répertoire valide. Toutefois, le format tar ne supporte pas la compression et a une limite de 8Go sur la taille des tables individuelles. Par ailleurs, l'ordre de restauration des données des tables ne peut pas être changé au moment de la restauration.
- **Fichier plat** : fichier de scripts SQL en texte simple. Ce format offre une grande souplesse à la restauration des données, lors d'un changement de version de PostgreSQL ou pour utiliser les données dans d'autres SGBD.
- **Répertoire** : Produire une archive au format répertoire utilisable en entrée de `pg_restore`. Cela créera un répertoire avec un fichier pour chaque table et blob exporté, ainsi qu'un fichier appelé Table of Contents (Table des matières) décrivant les objets exportés dans un format machine que `pg_restore` peut lire. Une archive au format répertoire peut être manipulée avec des outils Unix standard; par exemple, les fichiers d'une archive non-compressée peuvent être compressés avec l'outil `gzip`. Ce format est compressé par défaut et supporte les sauvegardes parallélisées.

Les autres options sont :

- **Encodage** : Par défaut, la sauvegarde utilise celui de la base de données
- **Nombre de tâches** : Exécute une sauvegarde parallélisée. Cette option réduit la durée de la sauvegarde mais, elle augmente aussi la charge sur le serveur de base de données. Vous ne pouvez utiliser cette option qu'avec le format de sortie répertoire car c'est le seul format où plusieurs processus peuvent écrire leurs données en même temps.
- **Rôle** : Spécifie un rôle à utiliser pour créer la sauvegarde. Avec cette option, `pg_dump` utilise une commande `SET ROLE nomrole` après s'être connecté à la base. C'est utile quand l'utilisateur authentifié n'a pas les droits dont `pg_dump` a besoin, mais peut basculer vers un rôle qui les a. Certaines installations ont une politique qui est contre se connecter directement en tant que superutilisateur, et l'utilisation de cette option permet que les extractions soient faites sans violer cette politique.



Méthode : Onglet "Options de sauvegarde"



Sections : Sauvegarde seulement les sections indiquées (pre-données, données ou post-données). (Par défaut toutes les sections).

La section 'données' (ou Data) contient toutes les données des tables ainsi que la définition des Large Objects et les

valeurs des séquences.

Exemple en SQL : `COPY commune (id, geom, ...)`

Les éléments 'post-données' (Post-data) incluent la définition des index, triggers, règles et contraintes (autres que les contraintes de vérification).

Exemple en SQL :

```
ALTER TABLE ONLY "TRONCON_ROUTE"
ADD CONSTRAINT "TRONCON_ROUTE_pkey" PRIMARY KEY (id);
```

Les éléments 'pré-données' (Pre-data) incluent tous les autres éléments de définition.

Exemple en SQL : `CREATE TABLE commune (id integer NOT NULL,...)`

Type des objets :

- **Données uniquement** : (--data-only). Seules les données sont sauvegardées, pas le schéma (définition des données). Les données des tables, les Large Objects, et les valeurs des séquences sont sauvegardées. Cette option est similaire à cocher 'données' dans 'sections' mais, pour des raisons historiques, elle n'est pas identique.
- **Schéma uniquement** : Seule la définition des objets (le schéma) est sauvegardée, pas les données. Cette option est l'inverse de 'données uniquement'. Elle est similaire, mais pas identique (pour des raisons historiques), à cocher Pré-données et post-données dans 'sections'.
- **Blobs** : Inclut les objets larges (blob) dans la sauvegarde. C'est le comportement par défaut, sauf si une des options suivantes est ajoutée : --schema, --table ou --schema-only. L'option -b n'est utile que pour ajouter les objets larges aux sauvegardes sélectives.

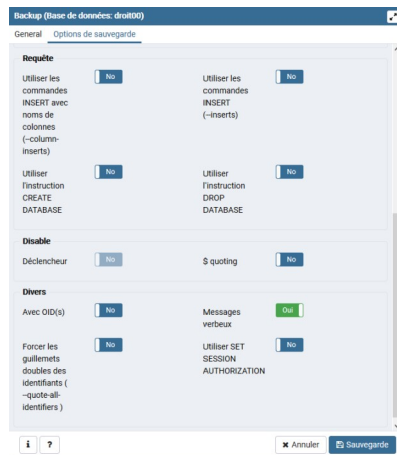
Ne pas sauvegarder :

- **propriétaire** (--no-owner) : Les commandes d'initialisation des possessions des objets au regard de la base de données originale ne sont pas produites. L'option -O est utilisée pour créer un script qui puisse être restauré par n'importe quel utilisateur. En revanche, c'est cet utilisateur qui devient propriétaire de tous les objets à la restauration.
- **Droits** (--no-privileges) : Les droits d'accès (commandes grant/revoke) ne sont pas sauvegardés.
- **Tablespace** (--no-tablespaces) : Ne pas générer de commandes pour créer des tablespace, ni sélectionner de tablespace pour les objets. Avec cette option, tous les objets seront créés dans le tablespace par défaut durant la restauration.
- **Données de la table non enregistrées dans les journaux** (--no-unlogged-table-data) : Ne pas exporter le contenu des tables non journalisées (unlogged)
- **Commentaires** (--no-comments) : Ne pas sauvegarder les commentaires



Méthode : Options de sauvegarde (suite)





Requêtes :

- Utiliser les INSERT avec noms de colonnes** (`--column-inserts`) : Extraire les données en tant que commandes INSERT avec des noms de colonnes explicites (`INSERT INTO table (colonne, ...) VALUES ...`). Ceci rendra la restauration très lente ; c'est surtout utile pour créer des extractions qui puissent être chargées dans des bases de données autres que PostgreSQL.
- Utiliser des commandes INSERT** (`--inserts`) : Extraire les données en tant que commandes INSERT (plutôt que COPY). Ceci rendra la restauration très lente ; c'est surtout utile pour créer des extractions qui puissent être chargées dans des bases de données autres que PostgreSQL, Notez que la restauration peut échouer complètement si vous avez changé l'ordre des colonnes. L'option `--column-inserts` est plus sûre, mais encore plus lente.
- Utiliser l'instruction CREATE DATABASE** (`--create`) : La sortie de sauvegarde débute par une commande de création de la base de données et de connexion à cette base. Peu importe, dans ce cas, la base de données de connexion à la restauration. De plus, si `--clean` est aussi spécifié (`DROP DATABASE`), le script supprime puis crée de nouveau la base de données cible avant de s'y connecter
- Utiliser l'instruction DROP DATABASE** (`--clean`) : Les commandes de nettoyage (suppression) des objets de la base sont écrites avant les commandes de création.
- Charger Via la partition racine** : (`--load-via-partition-root`) : Lors de l'export de données d'une partition, faire que les instructions COPY ou INSERT ciblent la racine du partitionnement qui contient cette partition, plutôt que la partition elle-même.

Désactiver :

- trigger (déclencheurs)** (`--disable-triggers`) : Cette option ne s'applique que dans le cas d'une extraction de données seules. Ceci demande à `pg_dump` d'inclure des commandes pour désactiver temporairement les triggers sur les tables cibles pendant que les données sont rechargées. Utilisez ceci si, sur les tables, vous avez des contraintes d'intégrité ou des triggers que vous ne voulez pas invoquer pendant le rechargement. Cette option n'a de sens que pour le format texte simple (SQL). Pour les formats d'archive, vous pouvez spécifier cette option quand vous appelez `pg_restore`. À l'heure actuelle, les commandes émises pour `--disable-triggers` doivent être exécutées en tant que superutilisateur.
- Guillemet dollar** (`--disable-dollar-quoting`) : Cette option désactive l'utilisation du caractère dollar comme délimiteur de corps de fonctions, et force leur délimitation en tant que chaîne SQL standard.

Divers :

- Avec OID** (`--oids`) : Les identifiants d'objets (OID) sont sauvegardés comme données des tables. Cette option est utilisée dans le cas d'applications utilisant

des références aux colonnes OID (dans une contrainte de clé étrangère, par exemple). Elle ne devrait pas être utilisée dans les autres cas.

- **Message en verbeux (--verbose)** : Mode verbeux. pg_dump affiche des commentaires détaillés sur les objets et les heures de début et de fin dans le fichier de sauvegarde. Des messages de progression sont également affichés sur la sortie d'erreur standard.
- **Forcer les guillemets doubles des identifiants (--quote-all-identifiers)** : Force la mise entre guillemets de tous les identifiants. Ceci peut être utile si vous exportez votre base en vue d'une migration dans une nouvelle version qui aurait introduit de nouveaux mots clés.
- **Utilisez SET SESSION AUTHORIZATION (--use-set-session-authorization)** : Émettre des commandes SQL standard SET SESSION AUTHORIZATION à la place de commandes ALTER OWNER pour déterminer l'appartenance d'objet. Ceci rend l'extraction davantage compatible avec les standards, mais, suivant l'historique des objets de l'extraction, peut ne pas se restaurer correctement. Par ailleurs, une extraction utilisant SET SESSION AUTHORIZATION nécessitera certainement des droits superutilisateur pour se restaurer correctement, alors que ALTER OWNER nécessite des droits moins élevés

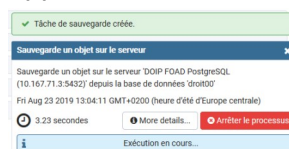


Méthode : rapport de sauvegarde

```

Processus Watcher - Sauvegarde un objet sur le serveur
-----
Sauvegarde un objet sur le serveur 'DOIP FOAD PostgreSQL (10.167.71.3:5432)' depuis la base de données 'drole00'...
Commande exécutée:
C:\Program Files (x86)\pgAdmin 4\bin\pg_dump.exe -h 10.167.71.3 -p 5432 --username 'drole00' --no-password --verbose --format=c --table 'drole00'
-----
Heure de début: Fri Aug 23 2019 13:04:11 GMT+0200 (heure d'été d'Europe centrale)
-----
pg_dump: creating publication membership
pg_dump: reading subscription
pg_dump: reading trigger objects
pg_dump: reading dependency data
pg_dump: saving encoding 'UTF8'
pg_dump: saving triggers, constraints, triggers + on
pg_dump: saving search paths
pg_dump: saving database definition
pg_dump: dumping contents of table 'consultation_contenu'
pg_dump: dumping contents of table 'production_contenu_detail'
pg_dump: dumping contents of table 'production_contenu_xy'
pg_dump: dumping contents of table 'production_contenu_spectacle'
pg_dump: dumping contents of table 'public_contenu_detail'
pg_dump: dumping contents of table 'public_contenu_xy'
-----
Terminé avec succès. Temps d'exécution: 70.2 secondes
  
```

Pendant la sauvegarde un message apparaît :



Il est possible de demander plus de détails sur la sauvegarde avec le bouton 'More détails...'



Complément

En ligne de commande, il est possible de 'dumper' une base directement dans une autre sans passer par un fichier intermédiaire.

La commande sera du type :

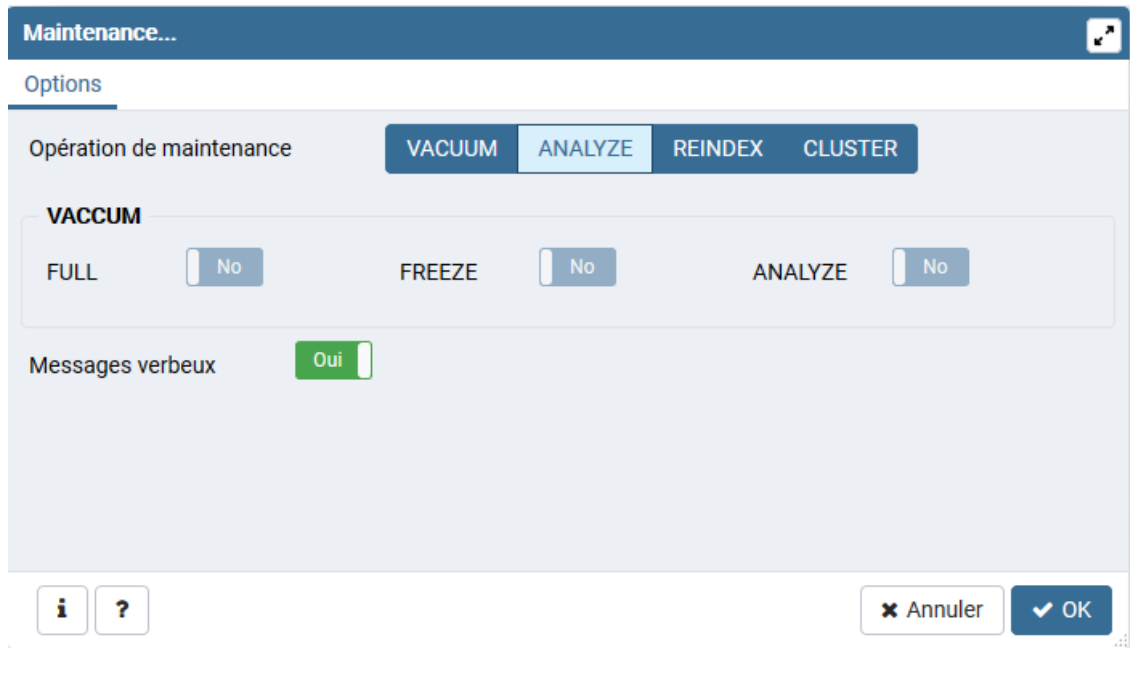
```
pg_dump -h serveur1 base_de_donnees | psql -h serveur2 base_de_donnees
```



Conseil

Après la restauration d'une sauvegarde, il est conseillé d'exécuter ANALYZE sur chaque base de données pour que l'optimiseur de requêtes dispose de statistiques utiles.

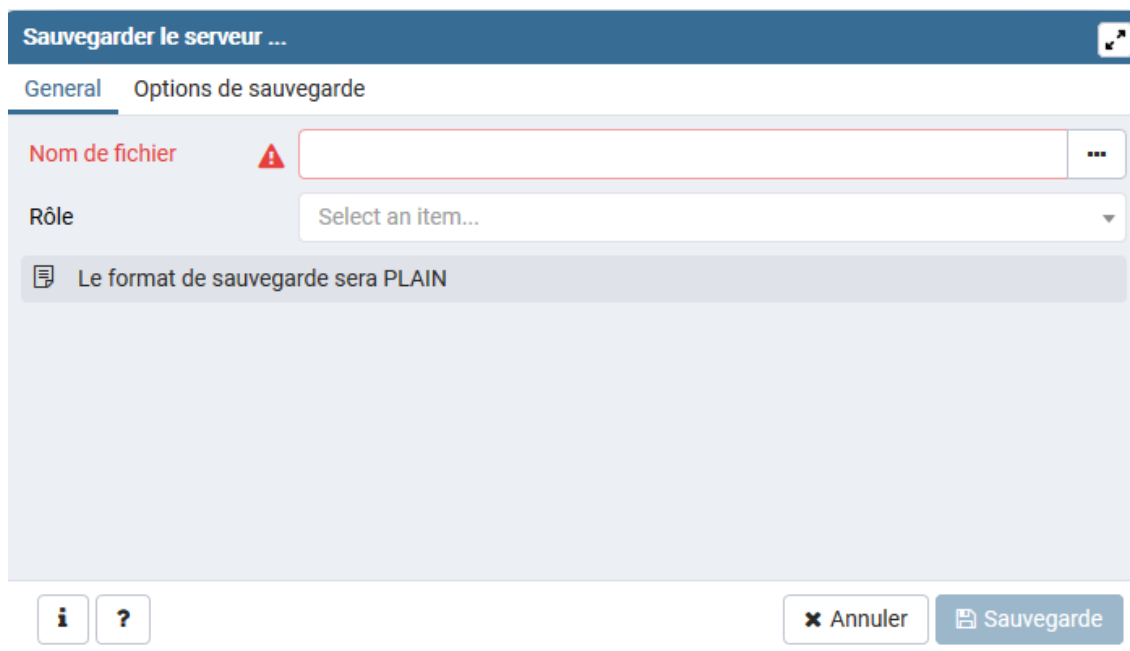
On utilise pour cela le menu Outils-> maintenance puis ANALYZE.



B. Sauvegarder le serveur (pg_dumpall)

pg_dumpall permet de sauvegarder toutes les bases de données d'une instance dans le format PLAIN (format texte SQL). pg_dumpall sauvegarde aussi les objets globaux, communs à toutes les bases de données (pg_dump ne sauvegarde pas ces objets.) Cela inclut aussi les informations concernant les utilisateurs et groupes des bases de données, ainsi que les tablespaces et les propriétés telles que les droits d'accès s'y appliquant.

Avec PgAdmin, sur clic droit sur un serveur → sauvegarder le serveur, on obtient la boîte de dialogue :



C. Restauration logique

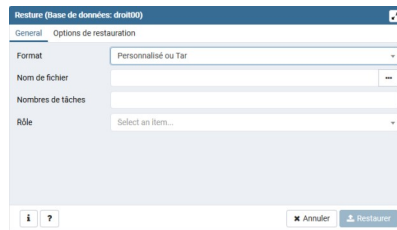
pg_restore sous PgAdmin

La restauration de données peut s'effectuer avec l'outil `pg_restore` pour les formats binaires et avec `psql` pour les sauvegardes en format texte SQL.



Méthode

Avec pgadmin, se positionner dans une base de données, puis clic droit → restaurer :



Constater que seul les formats *personnalisé* ou *tar* et *répertoire* sont disponibles.

Nombre de tâches (`--jobs`) : Exécute les parties les plus consommatrices en temps de `pg_restore` -- celles des chargements de données, créations

d'index et créations de contraintes -- en utilisant plusieurs jobs concurrents. Cette option peut réduire de beaucoup le temps pour restaurer une grosse base de données pour un serveur fonctionnant sur une machine multi-processeurs. La valeur optimale pour cette option dépend de la configuration matérielle du serveur, du client et du réseau. Les facteurs incluent le nombre de cœurs CPU et la configuration disque. Un bon moyen pour commencer est le nombre de cœurs CPU du serveur, mais une valeur plus grande que ça peut amener des temps de restauration encore meilleurs dans de nombreux cas. Bien sûr, les valeurs trop hautes apporteront des performances en baisse.

Seuls les formats d'archivage personnalisé et répertoire sont supportés avec cette option

Rôle (`--role`) : Indique un nom de rôle utilisé pour la restauration. Cette option fait que `pg_restore` exécute un `SET ROLE nom_rôle` après connexion à la base de données. C'est utile quand l'utilisateur authentifié (indiqué par l'option `-U`) n'a pas les droits demandés par `pg_restore`, mais peut devenir le rôle qui a les droits requis. Certains installations ont une politique contre la connexion en super-utilisateur directement, et utilisent cette option pour permettre aux restaurations de se faire sans violer cette règle.



Méthode : Onglet "Options de restauration"



Sections : Restaure seulement la section nommée (Par défaut toutes les sections).

Type des objets :

- **Données uniquement** (`--data-only`) : Restaure seulement les données, pas les schémas (définitions des données). Les

données des tables, les Large Objects, et les valeurs des séquences sont restaurées si elles sont présentes dans l'archive.

Cette option est similaire à `--section=data` mais, pour des raisons historiques, elle n'est pas identique.

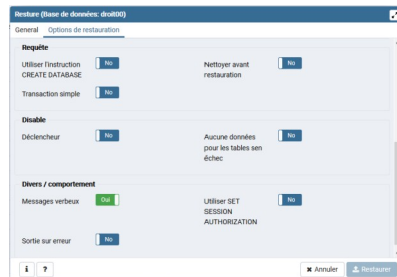
- **Schéma uniquement** (`--schema-only`) : Restaure seulement le schéma (autrement dit, la définition des données), mais pas les données, à condition que cette définition est présente dans l'archive.

Ne pas enregistrer :

- **propriétaire** (`--no-owner`) : Ne pas donner les commandes initialisant les propriétaires des objets pour correspondre à la base de données originale. Par défaut, `pg_restore` lance des instructions `ALTER OWNER` ou `SET SESSION AUTHORIZATION` pour configurer le propriétaire des éléments du schéma créé. Ces instructions échouent sauf si la connexion initiale à la base de données est réalisée par un superutilisateur (ou le même utilisateur que le propriétaire des objets du script)
- **Droits** (`--no-privileges`) : Empêche la restauration des droits d'accès (commandes `grant/revoke`).
- **Tablespace** (`--no-tablespaces`) : Ne sélectionne pas les tablespaces. Avec cette option, tous les objets seront créés dans le tablespace par défaut lors de la restauration.
- **Commentaires** : ne pas restaurer les commentaires



Méthode : Option de restauration (suite)



Requête :

- **Inclure l'instruction CREATE DATABASE (--create)** : Crée la base de données avant de la restaurer. Si l'option --clean est aussi indiquée, supprime puis crée de nouveau la base de données cible avant de s'y connecter.

- **Nettoyer avant restauration (--clean)** : Nettoie (supprime) les objets de la base de données avant de les créer.
- **Transaction simple (--single-transaction)** : Exécute la restauration en une seule transaction (autrement dit, toutes les commandes de restauration sont placées entre un BEGIN et un COMMIT). Ceci assure l'utilisateur que soit toutes les commandes réussissent, soit aucun changement n'est appliqué.

Désactiver :

- **trigger (--disable-triggers)** : Cette option n'est pertinente que lors d'une restauration des données seules. Elle demande à pg_restore d'exécuter des commandes pour désactiver temporairement les déclencheurs sur les tables cibles pendant que les données sont rechargées. Utilisez ceci si vous avez des vérifications d'intégrité référentielle sur les tables que vous ne voulez pas appeler lors du rechargement des données. Actuellement, les commandes émises pour --disable-triggers doivent être exécutées par un superutilisateur
- **Aucune données pour les tables en échec (--no-data-for-failed-tables)** : Par défaut, les données de la table sont restaurées même si la commande de création de cette table a échoué (par exemple parce qu'elle existe déjà). Avec cette option, les données de cette table seront ignorées. Ce comportement est utile si la base cible contient déjà des données pour cette table. Par exemple, les tables supplémentaires des extensions de PostgreSQL comme PostGIS pourraient avoir déjà été créées et remplies sur la base cible ; indiquer cette option empêche l'ajout de données dupliquées ou obsolètes.

Divers / comportement :

- **Message verbeux (--verbose)** : Mode verbeux. On décochera cette option lors de la restauration de gros fichiers de backup.
Utilisez SET SESSION AUTHORIZATION (--use-set-session-authorization) : Affiche les commandes SET SESSION AUTHORIZATION du standard SQL à la place des commandes ALTER OWNER pour déterminer le propriétaire de l'objet. Ceci rend la sauvegarde plus compatible avec les standards mais, suivant l'historique des objets dans la sauvegarde, pourrait restaurer correctement.
- **Sortie sur erreur (--exit-on-error)** : Quitte si une erreur est rencontrée lors de l'envoi des commandes SQL à la base de données. La valeur par défaut est de continuer et d'afficher le nombre d'erreurs à la fin de la restauration.



Complément : Manuels de formation de Dalibo

La société Dalibo met à disposition ses manuels de formation. Concernant les

|sauvegardes ont pourra lire avec intérêt le chapitre dédié dans le manuel *Formation DBA1*²⁹, également disponible *en ligne*³⁰.

D. 02 - sauvegarde et restauration

Mise en pratique d'une sauvegarde et restauration pour dupliquer deux schémas d'une base

Question

[Solution n°1 p 75]

Nous souhaitons dupliquer les schémas *consultation* et *production* dans une nouvelle base de données.

Avec pgadmin, il n'est pas possible de choisir les schémas à sauvegarder dans les options de sauvegarde au niveau base de données. Il faut donc procéder à la sauvegarde de chaque schéma et les restaurer successivement.

Réaliser des sauvegardes (dump) de chaque schéma *consultation* et *production* de la base *droitXX*. On utilisera le format 'personnalisé'.

Créer une nouvelle base *newdroitXX* (propriétaire *stageXX*).

Restaurer dans cette nouvelle base les deux schémas.

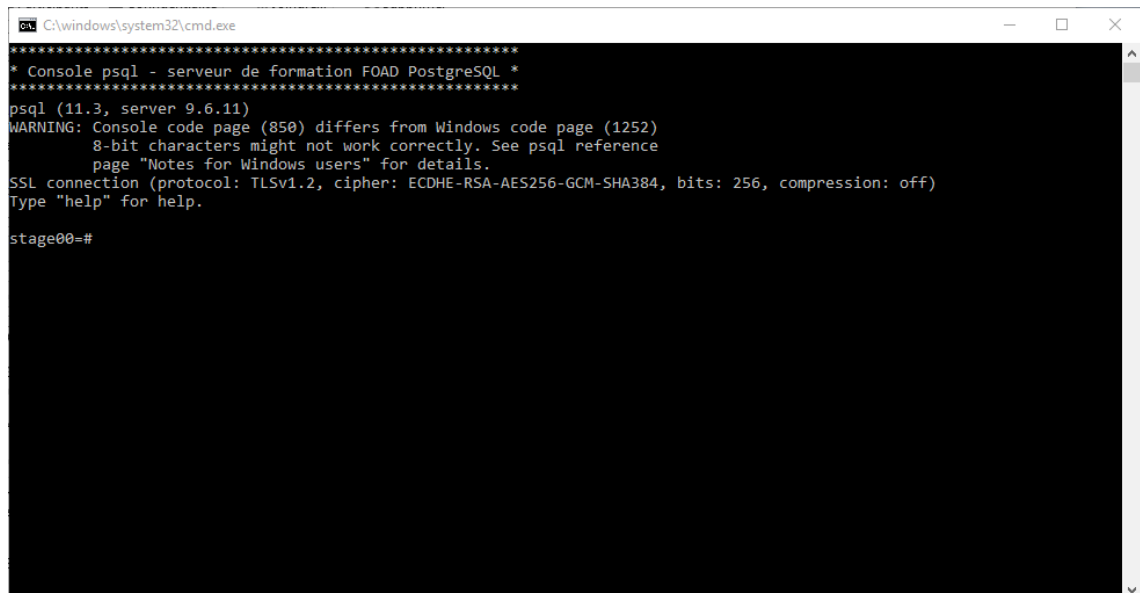
E. Utilisation d'une console PSQL

Certains référentiels (comme les fichiers fonciers) sont livrés sous forme de dump SQL.

A titre d'exemple nous allons utiliser une procédure permettant de restaurer un dump SQL dans la base *stageXX* en utilisant une console PSQL

29 - <https://public.dalibo.com/exports/formation/manuels/formations/dba1/dba1.handout.pdf>

30 - <https://public.dalibo.com/exports/formation/manuels/modules/i0/i0.handout.html>



```
C:\windows\system32\cmd.exe
*****
* Console psql - serveur de formation FOAD PostgreSQL *
*****
psql (11.3, server 9.6.11)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

stage00=#
```

Nous pouvons utiliser la commande `\i` pour exécuter le script de commande SQL `ff_annexes_metropole_2013.sql` qui vous est fourni dans le jeu de données.

(il s'agit d'un jeu de fichier fictif qui ne contient qu'une seule ligne)

taper la commande :

```
\i i :/ff_annexes_metropole_2013.sql
```

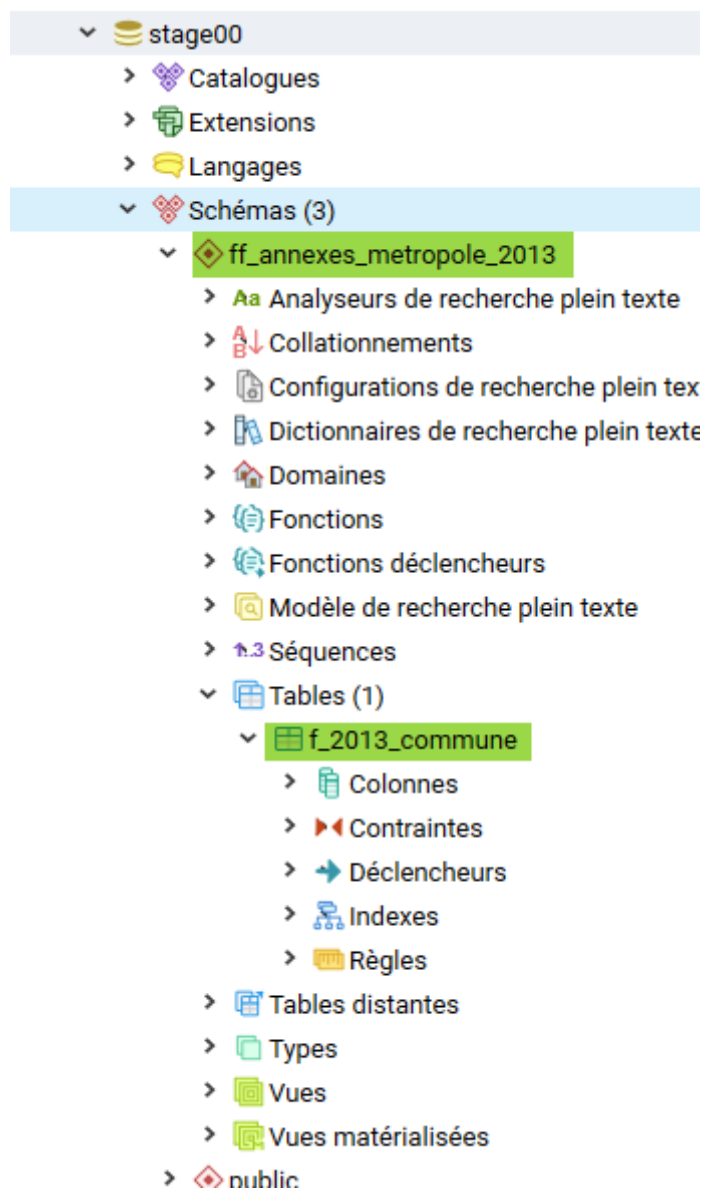
remplacer le `i :/` par le chemin d'accès au jeu de données.

(si le chemin comporte un blanc il faut le mettre entre ' ' simple)

Vous devez voir défiler la liste des commandes SQL qui se termine par une `ALTER TABLE`

fermer la console par `\q`

Vous pouvez vérifier que le schéma `ff_annexes_metropole_2013` a bien été créé dans la base `stage00` et qu'il contient une table `f_2013_commune`



F. Sauvegarde au niveau système de fichiers

Une autre stratégie de sauvegarde (pour les administrateurs systèmes) consiste à copier les fichiers utilisés par PostgreSQL pour le stockage des données. Par défaut, les fichiers de base de données sont stockés dans le répertoire DATA de POSTGRE.

N'importe quelle méthode de sauvegarde peut être utilisée, par exemple :

```
tar -cf sauvegarde.tar /usr/local/pgsql/data
```

Cependant, deux restrictions rendent cette méthode peu pratique ou en tout cas inférieure à la méthode `pg_dump`.

- Le serveur de base de données doit être arrêté pour obtenir une sauvegarde

utilisable

- Les sauvegardes du système de fichiers fonctionnent uniquement pour les restaurations complètes d'un cluster de bases de données

Nous ne rentrerons donc pas plus dans le cadre de cette formation sur ce type de sauvegarde.

Pour *en savoir plus*³¹...

G. Archivage continu

PostgreSQL maintient en permanence des journaux WAL (write ahead log). Ces journaux décrivent chaque modification effectuée sur les fichiers de données des bases. Ils existent principalement pour se prémunir des suites d'un arrêt brutal : si le système s'arrête brutalement, la base de données peut être restaurée dans un état cohérent en « rejouant » les entrées des journaux enregistrées depuis le dernier point de vérification.

Cette approche est plus complexe que les autres.

Pour *en savoir plus*³²...

31 - <http://docs.postgresql.fr/current/backup-file.html>

32 - <http://docs.postgresql.fr/9.6/continuous-archiving.html>



Gestion de l'espace disque	67
Opérations de maintenance sur les bases	69

A. Gestion de l'espace disque

Chaque table est composée d'un fichier disque principal. Si la table contient des colonnes pouvant recevoir des valeurs étendues, il pourrait aussi y avoir un fichier TOAST associé à la table. Des index peuvent également être associés.

Chaque table ou index est stocké dans un fichier distinct ou plusieurs si la taille du fichier dépasse 1 Go.

L'espace disque peut être surveillé de plusieurs façons différentes : en utilisant les fonctions SQL listées dans «*Fonctions de taille des objets*»³³, en utilisant le module *oid2name*³⁴(1) ou en inspectant manuellement les catalogues système. Les fonctions SQL sont les plus simples à utiliser et sont généralement recommandées.

33 - <https://docs.postgresql.fr/current/functions-admin.html#FUNCTIONS-ADMIN-DBOBJECT>

34 - <https://docs.postgresql.fr/current/oid2name.html>



Méthode : Connaître l'espace disque consommé par une table

Pour connaître l'espace disque consommé par une table, avec pgadmin : se positionner dans l'onglet statistique sur un table :

Statistiques	valeur
Parcours séquentiel	9
Lecture séquentielle de lignes	114
Parcours d'index	0
Lignes d'index récupérées	0
Lignes insérées	19
Lignes mises à jour	0
Lignes supprimées	0
Lignes mises à jour avec HOT	0
Lignes vivantes	19
Lignes mortes	0
Lecture de blocs d'en-tête	30
Accès aux blocs d'en-tête	51
Blocs d'index lus	2
Blocs d'index atteints	24
Lecture des blocs TOAST	4
Accès aux blocs TOAST	17
Lecture des blocs d'index TOAST	8
Accès aux blocs d'index TOAST	24
Dernier VACUUM	
Dernier AUTOVACUUM	
Dernière ANALYSE	
Dernière auto-ANALYSE	
Compteur de VACUUM	0
Compteur d'AUTOVACUUM	0
Compteur d'ANALYSE	0
Compteur d'auto-ANALYSE	0
Taille de la table	56 kB
Taille de la table TOAST	24576
Taille de l'index	16 kB

Dans cet exemple la taille de *commune_densite* est de 56 Ko, la table d'index fait 16 Ko.

La *table TOAST*³⁵ (The Oversized-Attribute Storage Technique) permet de stocker les attributs trop grands (typiquement les données géométriques). Ceci est transparent pour l'utilisateur.

On peut également utiliser des requêtes comme :

```
SELECT
relname as "Table",
pg_size_pretty(pg_total_relation_size(relid)) As "Size",
```

```

pg_size_pretty(pg_total_relation_size(relid) - pg_relation_size(relid))
as "External Size"
FROM      pg_catalog.pg_statio_user_tables      ORDER      BY
pg_total_relation_size(relid) DESC;

```

	Données	EXPLAIN	Messages	Notifications
	Table name	Size text	External Size text	
1	FR_communes	20 MB	2624 kB	
2	spatial_ref_sys	4552 kB	224 kB	

qui indique :

le nom de la table

la taille de la table

les tailles des objets liés (comme les index)

La tâche la plus importante d'un administrateur de base de données, en ce qui concerne la surveillance des disques, est de s'assurer que *les disques n'arrivent pas à saturation*³⁶.

B. Opérations de maintenance sur les bases

Sous PgAdmin il est possible d'utiliser le menu Outils > Maintenance... celui-ci permet de lancer différentes opération de maintenance.

Maintenance...

Options

Opération de maintenance: VACUUM ANALYZE REINDEX CLUSTER

VACCUM

FULL: No FREEZE: No ANALYZE: No

Messages verbeux: Oui

Annuler OK



Méthode : VACUUM

VACUUM récupère l'espace inutilisé et, optionnellement, analyse une base.

VACUUM récupère l'espace de stockage occupé par des lignes mortes. Lors des opérations normales de PostgreSQL, les lignes supprimées ou rendues obsolètes par une mise à jour ne sont pas physiquement supprimées de leur table. Elles restent présentes jusqu'à ce qu'un VACUUM soit lancé. C'est pourquoi, il est nécessaire de faire un VACUUM régulièrement, spécialement sur les tables fréquemment mises à jour.

Sans paramètre, VACUUM traite toutes les tables de la base de données courante pour lequel l'utilisateur connecté dispose du droit d'exécution du VACUUM. Avec un paramètre, VACUUM ne traite que cette table.

Les options :

- **FULL** récupère plus d'espace (compactage des tables), mais est beaucoup plus long et prend un verrou exclusif sur la table (inaccessible à d'autres requêtes pendant le VACCUM). Cette méthode requiert aussi un espace disque supplémentaire, car elle écrit une nouvelle copie de la table et ne supprime l'ancienne copie qu'à la fin de l'opération. Habituellement, cela doit seulement être utilisé quand une quantité importante d'espace doit être récupérée de la table. .
- **FREEZE** : les versions des lignes sont gelées si elles sont suffisamment vieilles pour être visibles de toutes les transactions en cours. En particulier, sur une base en lecture seulement, VACUUM FREEZE aura pour résultat de geler toutes les lignes de la base. Donc, tant que la base n'est pas modifiée, aucun nettoyage supplémentaire n'est nécessaire.
- **ANALYZE** : Met à jour les statistiques utilisées par l'optimiseur pour déterminer la méthode la plus efficace pour exécuter une requête.

Pour exécuter un VACUUM sur une table, vous devez habituellement être le propriétaire de la table ou un superutilisateur. Néanmoins, les propriétaires de la base de données sont autorisés à exécuter VACUUM sur toutes les tables de leurs bases de

données, sauf sur les catalogues partagés. Cette restriction signifie qu'un vrai VACUUM sur une base complète ne peut se faire que par un superutilisateur.)

Il est recommandé que les bases de données actives de production soient traitées par VACUUM fréquemment (au moins toutes les nuits), pour supprimer les lignes mortes. Après avoir ajouté ou supprimé un grand nombre de lignes, il peut être utile de faire un VACUUM ANALYZE sur la table affectée. Cela met les catalogues système à jour de tous les changements récents et permet à l'optimiseur de requêtes de PostgreSQL™ de faire de meilleurs choix lors de l'optimisation des requêtes

PostgreSQL inclut un « *autovacuum*³⁷ » qui peut automatiser la maintenance par VACUUM. Certains administrateurs de bases de données voudront suppléer ou remplacer les activités du démon avec une gestion manuelle des commandes VACUUM, qui seront typiquement exécutées suivant un planning par des scripts cron ou par le planificateur de tâches.

A noter que si on est positionné dans le navigateur au niveau d'un objet (table,...) la maintenance ne s'exécutera qu'au niveau de l'objet sélectionné.

À noter que le VACCCUM est également disponible pour l'administrateur système sous forme de la commande système vacuumdb.



Méthode : ANALYZE

ANALYZE³⁸ collecte des statistiques sur le contenu des tables de la base de données et stocke les résultats dans le catalogue système *pg_statistic*³⁹.

L'optimiseur de requêtes les utilise pour déterminer les plans d'exécution les plus efficaces.

ANALYZE peut être exécuté comme une option de VACUUM.



Méthode : REINDEX

REINDEX reconstruit un index en utilisant les données stockées dans la table, remplaçant l'ancienne copie de l'index. Il y a plusieurs raisons pour utiliser REINDEX :

- Un index a été corrompu et ne contient plus de données valides
- L'index en question a « explosé », c'est-à-dire qu'il contient beaucoup de pages d'index mortes ou presque mortes
- Vous avez modifié un paramètre de stockage (par exemple, fillfactor) pour un index et vous souhaitez vous assurer que la modification a été prise en compte.
- La construction d'un index avec l'option `CONCURRENTLY` a échoué, laissant un index « invalide »

Dans les vieilles versions de PostgreSQL, le gain avec REINDEX peut-être énorme.

Nota bene : disponible pour l'administrateur système avec la commande *reindexdb*⁴⁰.



Méthode : CLUSTER

CLUSTER commande permet de réécrire les données d'une table dans un ordre donné (suivant un index). Cela peut être utile pour optimiser des requêtes SQL, selon la façon dont les données sont accédées.

Pour des tables spatiales, elle permet d'ordonner physiquement la table selon l'index

37 - <http://docs.postgresql.fr/current/maintenance.html#autovacuum>

38 - <http://docs.postgresql.fr/current/sql-analyze.html>

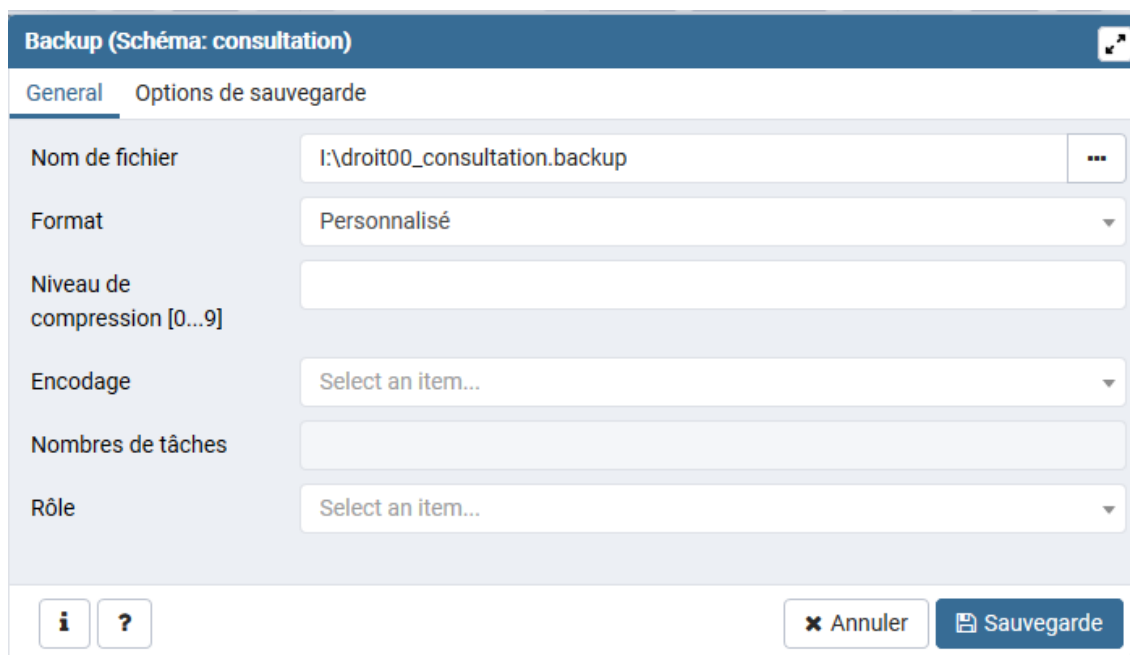
39 - <http://docs.postgresql.fr/current/catalog-pg-statistic.html>

40 - <http://docs.postgresql.fr/current/app-reindexdb.html>

Solution des exercices

> Solution n°1 (exercice p. 64)

sous PgAdmin se connecter avec un rôle superuser.
Sur chacun des schéma *consultation* puis *production* de la base *droitXX* Faire **clic droit** → **sauvegarder**



The screenshot shows the 'Backup (Schéma: consultation)' dialog box in PgAdmin. The 'General' tab is selected. The fields are as follows:

Field	Value
Nom de fichier	I:\droit00_consultation.backup
Format	Personnalisé
Niveau de compression [0...9]	
Encodage	Select an item...
Nombres de tâches	
Rôle	Select an item...

At the bottom, there are buttons for 'i', '?', 'Annuler', and 'Sauvegarde'.

Choisir de ne pas enregistrer les '*tablespaces*'
Puis faire de même pour le schéma *production*.
Créer la base *newdroitXX* avec comme propriétaire *stageXX* avec comme template (modèle) *template_SIG* (remplacer XX par votre numéro de stagiaire)
Se positionner sur la base *newdroitsXX*
faire **clic droit** → **restaurer**
ne pas oublier de demander de restaurer les pré-données, Données et post-données

Resture (Base de données: newdroit00)

General Options de restauration

Format: Personnalisé ou Tar

Nom de fichier: I:\droit00_consultation.backup

Nombres de tâches:

Rôle: Select an item...

Annuler Restaurer

On peut vérifier par exemple que les droits du schéma 'production' ont bien été restaurés. (le cas échéant rafraîchir l'affichage pour faire apparaître les schémas restaurés).

production
✕

General
Sécurité
Privilèges par défaut
SQL

Droits			+
	Bénéficiaire	Droits	Accordeur de droits
✕	ecrivains00	CU	stage00
✕	lecteur00	U	stage00
✕	stage00	CU	stage00

Label de sécurité		+
Fournisseur	Étiquette de sécurité	

i
?

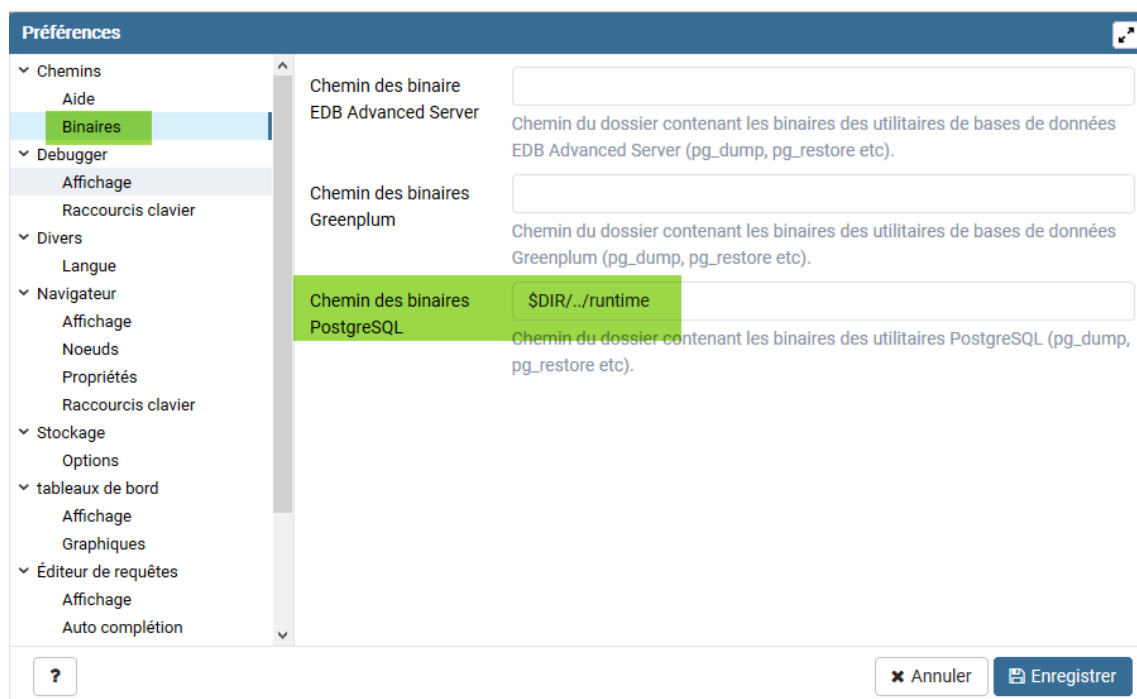
✕ Annuler
🔄 Réinitialiser
📄 Enregistrer



Attention : pg_dump erreur : annulation à cause de la différence entre les versions

Si vous rencontrez des messages d'erreur concernant les versions de pg_dump, il faut préciser à pgadmin d'aller chercher les bons exécutables binaires

(par exemple : C:\Program Files\PostgreSQL\13\bin si PostgreSQL est installé localement sur votre poste de travail)



Conseil : pgadmin server

Si vous utilisez pgAdmin en version server, pgadmin enregistre le fichier dans un espace de stockage sur le serveur. Pour le télécharger sur votre poste de travail, il faut passer par le gestionnaire de stockage (Menu Outil -> Gestionnaire de stockage

et utiliser le bouton  près avoir sélectionné le fichier.

Télécharger le fichier

Il convient, pour ne pas saturer le serveur, de faire le ménage sur la partition utilisée par l'espace de stockage et donc de ne pas conserver les sauvegardes sur cet espace qui doit être considéré comme un espace de dépôt temporaire à libérer à l'issue de l'utilisation par suppression du fichier.