
PostgreSQL

1-Introduction

Base de données avec un moteur transactionnel, qui dispose d'un langage PL/PgSql et de nombreux outils autour :

- Client cli : psql (ligne de commande)
- Client GUI : pgAdmin (interface graphique)
- Postgis : volet géographique
- Nombreux système de réplications
- Système d'extensions

Nous avons un outil pour la bonne gestion des transactions le MVCC, client/serveur avec une gestion de connexion.
Il fait appel au moteur relationnel ACID : atomic, sonsitent, isolated, durable.
Atomic : une transaction se fait en entier ou pas du tout
Consistent : en cas d'erreur lors d'une transaction l'état redevient celui d'avant celle-ci
Isolated : une transaction en cours n'interfère pas avec les autres.
Durable : les données sont dispos en cas de redémarrage (système de recovery)

Définitions
+- Cluster Machine > Cluster PG > Database > Rôle / Schémas > Tables > Champs
Nous avons le cluster au sens machine (différentes machines qui communiquent les uns avec les autres [cloud, VM, conteneur …) et le cluster au sens postgre c'est une instance de postgre c'est une première enveloppe contenant un ensemble de database avec un moteur et une certaine version.

+- Cluster Machine : plusieurs machines communiquant entre elles avec de la réplication.
+- Cluster PG : instances avec allocation de ressources spécifiques et définition de configuration.
+- Database : ensemble de données structurée (une application par exemple)
+-Rôles/Users : utilisateurs avec login et mot de passe.
+-Schéma : Namespace ou espace de nom dans une base de données (un group d'éléments comme attribuer les droits sur certaines tables et pas toutes du namespace)
+-Tables : espace de stockage logique dans une base de données.
+-Champs : une colonne de tables
+-Lignes : une table est structurée en colonne/ligne

2-Installation

Sur Debian
apt-get install postgresql
apt-cache search postgresql

service postgresql status
service postgresql stop
service postgresql start
service postgresql reload (évite les coupure de service pour faire des modification de configuration)

lancer une instance de postgre : ???

Supprimer postgresql
apt purge postgresql-15 postgresql-client-15

Les principaux répertoires

+- configuration : /etc/postgresql/<version>/<clusterpg (par defaut main)>

```
root@debian:/etc/postgresql/15/main# ls
conf.d       pg_ctl.conf  pg_ident.conf    start.conf
environment  pg_hba.conf  postgresql.conf
root@debian:/etc/postgresql/15/main# ls -l
total 60
drwxr-xr-x 2 postgres postgres  4096 Nov 10 11:22 conf.d
-rw-r--r-- 1 postgres postgres   315 Nov 10 11:22 environment
-rw-r--r-- 1 postgres postgres   143 Nov 10 11:22 pg_ctl.conf
-rw-r----- 1 postgres postgres  5002 Nov 10 11:22 pg_hba.conf
-rw-r----- 1 postgres postgres  1636 Nov 10 11:22 pg_ident.conf
-rw-r--r-- 1 postgres postgres 29721 Nov 10 11:22 postgresql.conf
-rw-r--r-- 1 postgres postgres   317 Nov 10 11:22 start.conf
```

+- data : /var/lib/postgresql/<version>/<clusterpg (par defaut main)>

Eléments permettant le bon fonctionnement du moteur.

```
root@debian:/var/lib/postgresql/15/main# ls -l
total 84
drwx------ 5 postgres postgres 4096 Nov 10 11:22 base
drwx------ 2 postgres postgres 4096 Nov 10 11:25 global
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_commit_ts
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_dynshmem
drwx------ 4 postgres postgres 4096 Nov 10 11:27 pg_logical
drwx------ 4 postgres postgres 4096 Nov 10 11:22 pg_multixact
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_notify
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_replslot
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_serial
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_snapshots
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_stat
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_stat_tmp
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_subtrans
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_tblspc
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_twophase
-rw------- 1 postgres postgres    3 Nov 10 11:22 PG_VERSION
drwx------ 3 postgres postgres 4096 Nov 10 11:22 pg_wal
drwx------ 2 postgres postgres 4096 Nov 10 11:22 pg_xact
-rw------- 1 postgres postgres   88 Nov 10 11:22 postgresql.auto.conf
-rw------- 1 postgres postgres  130 Nov 10 11:22 postmaster.opts
-rw------- 1 postgres postgres  108 Nov 10 11:22 postmaster.pid
```

+- binaire principal : /usr/lib/postgresql/<version>/bin/postgresql

Nous avons les binaires comme le pg_ctl pour lancer postgresql au tout début.

```
root@debian:/usr/lib/postgresql/15/bin# ls -l
total 12624
-rwxr-xr-x 1 root root   80632 May  9  2023 clusterdb
-rwxr-xr-x 1 root root   88984 May  9  2023 createdb
-rwxr-xr-x 1 root root   80984 May  9  2023 createuser
-rwxr-xr-x 1 root root   72376 May  9  2023 dropdb
-rwxr-xr-x 1 root root   72312 May  9  2023 dropuser
-rwxr-xr-x 1 root root  151088 May  9  2023 initdb
-rwxr-xr-x 1 root root   47872 May  9  2023 oid2name
-rwxr-xr-x 1 root root  105976 May  9  2023 pg_amcheck
-rwxr-xr-x 1 root root   47344 May  9  2023 pg_archivecleanup
-rwxr-xr-x 1 root root  151000 May  9  2023 pg_basebackup
-rwxr-xr-x 1 root root  200408 May  9  2023 pgbench
-rwxr-xr-x 1 root root   68232 May  9  2023 pg_checksums
-rwxr-xr-x 1 root root   51520 May  9  2023 pg_config
-rwxr-xr-x 1 root root   63832 May  9  2023 pg_controldata
-rwxr-xr-x 1 root root   72376 May  9  2023 pg_ctl
-rwxr-xr-x 1 root root  405816 May  9  2023 pg_dump
-rwxr-xr-x 1 root root  114496 May  9  2023 pg_dumpall
-rwxr-xr-x 1 root root   72248 May  9  2023 pg_isready
-rwxr-xr-x 1 root root   97208 May  9  2023 pg_receivewal
-rwxr-xr-x 1 root root   97344 May  9  2023 pg_recvlogical
-rwxr-xr-x 1 root root   68280 May  9  2023 pg_resetwal
-rwxr-xr-x 1 root root  188120 May  9  2023 pg_restore
-rwxr-xr-x 1 root root  142128 May  9  2023 pg_rewind
-rwxr-xr-x 1 root root   51560 May  9  2023 pg_test_fsync
-rwxr-xr-x 1 root root   39232 May  9  2023 pg_test_timing
-rwxr-xr-x 1 root root  162752 May  9  2023 pg_upgrade
-rwxr-xr-x 1 root root  105048 May  9  2023 pg_verifybackup
-rwxr-xr-x 1 root root  113560 May  9  2023 pg_waldump
-rwxr-xr-x 1 root root 8863224 May  9  2023 postgres
lrwxrwxrwx 1 root root       8 May  9  2023 postmaster -> postgres
-rwxr-xr-x 1 root root  785944 May  9  2023 psql
-rwxr-xr-x 1 root root   89016 May  9  2023 reindexdb
-rwxr-xr-x 1 root root   89304 May  9  2023 vacuumdb
-rwxr-xr-x 1 root root   47712 May  9  2023 vacuumlo
```

3-binaires

Les binaires à connaître et indispensable.

+- pg_ctl :
- +- gestion de l'instance/cluster
- +-start/stop/kill
- +-init : création autre espace de datas
- +-promote : promotion du standby

+- psql : (un autre interface graphique avec pgadmin)
- +- client de connexion à un cluster
- +-précision utilisateur et/ou db
- +-passage de sql en cli ou script sql
- Permet d'avoir toutes l'aide et le faire de suite.

Binaires spécifiques à debian

+- pg_createcluster :

+- création d'un cluster (une instance PG)

+- créatin des répetoires (/etc /var/lib/)

Permet de revenir vers une instance postgre (crée l'instance et les répertoires)

+- pg_dropcluster :

+- suppression d'un cluster

+- cluster arrêté

Permet de supprimer un cluster ou l'arrêter

+- pg_lscluster :

+- liste les clusters

Permet de lister les cluster présent sur la .

+- pg_ctlcluster :

+- équivalent du pg_ctl

+- contrôle du cluster (stop/start d'un cluster…)

Gérer le cluster spécifique au cluster et interagir individuellement

Spécifiques Sauvegarde

+- pg_dump:

+- sauvegarde d'une instance

+-différent formats : plain text, binaire ….

+-différents niveaux d'objets (cluster/db/table/schéma)

Permet de sauvegarder avec finesse une instance ou certaine table avec des formats de sortie. (binaire utilisable que pour postgre) si c'est en plain-text on pourra le restaurer directement pas psql

+- pg_dumpall: (intégrale du cluster)

+- sauvegarde intégrale en format binaire

+- pg_restore:

+- restauration à partir d'une sauvegarde à partir de (pg_dumpall)

Wrappers

+-équivalent de commandes sql

+-createdb :

+- création d'une base de données

Evite de se connecter en psql et lancer la commande create …

+-dropdb :

+- suppression d'une base de données

+-createuser :

+- création d'un utilisateur

(script shells)

+-dropuser :

+- supprime un utilisateur

Maintenance

+-reindexdb :
  +- ré-indexation des index avec des paramètres pour restreindre le périmètre car il peut avoir des ralentissements.

+-vacuumdb :
  +- tâche de maintenance (ménage)

+-vacuumlo :
  +- suppression de large objects

Spécifiques systèmes avancés

+-pg_controldata:
  +- vérifie l'état du serveur et des infos critiques (control files vitaux pour le bon état de postgre à checked)

+-pg_resetwal:
  +- en cas de crash avec pb de WAL (Write Ahead Logging) = mode d'archivage permet une autre sauvegarde et manière de restaurer mais en cas de crash mais des pertes de data dernier recours)
  +- attention : datas inconsistentes (dernier recours)

+-pg_receive_wal:
  +- récupération des WAL d'une autre DB
Dans la réplication et permet de jouer les fichier d'archivages et récupérer à distance ou local .

+-pg_controldata:
  +- vérifie l'état du serveur et des infos critiques (control files vitaux pour le bon état de postgre à checked)
+-pg_basebackup:
  +- récupération de datas par une connexion à une autre BD (ex : init réplication)
   Pour mettre une réplication en place.

4-Procédures arrêt et démarrage

> service postgresql status
>ps aux | grep postgres

```
root@debian:~# service postgresql start
root@debian:~# ps aux | grep postgres
postgres    6509  0.0  0.2 219680 30416 ?        Ss   15:07   0:00 /usr/lib/postgresql/
15/bin/postgres -D /var/lib/postgresql/15/main -c config_file=/etc/postgresql/15/main/p
ostgresql.conf
postgres    6510  0.0  0.0 219812  6236 ?        Ss   15:07   0:00 postgres: 15/main: c
heckpointer
postgres    6511  0.0  0.0 219828  6212 ?        Ss   15:07   0:00 postgres: 15/main: b
ackground writer
postgres    6513  0.0  0.0 219680 10476 ?        Ss   15:07   0:00 postgres: 15/main: w
alwriter
postgres    6514  0.0  0.0 221272  9100 ?        Ss   15:07   0:00 postgres: 15/main: a
utovacuum launcher
postgres    6515  0.0  0.0 221252  7192 ?        Ss   15:07   0:00 postgres: 15/main: l
ogical replication launcher
root        6536  0.0  0.0   6332  2092 pts/5    S+   15:08   0:00 grep postgres
root@debian:~#
```

Nous avons le binaire et la localisation des datas et éléments de configuration -c ….

>less /lib/systemd/system/postgresql.service

```
[Unit]
Description=PostgreSQL RDBMS

[Service]
Type=oneshot
ExecStart=/bin/true
ExecReload=/bin/true
RemainAfterExit=on

[Install]
WantedBy=multi-user.target
/lib/systemd/system/postgresql.service (END)
```

Nous avons un bin/true mais pour voir les commandes de lancement ! bin bash
>less /etc/init.d/postgresql

Postgresql : démarrage et arrêt (au même moment)
+-important pour un moteur de BDD surtout en relationnel
+- consistence des données
+- capacité à récupérer les données manquantes

Démarrage

1. Ressources ? : Affectation de la mémoire partagée
   +- shared_buffer (défaut 128 MB)
   +- paramètre dans /etc/…/postgresql.conf
   +- paramétré par clusterPG
2. Où on en est ? : Lecture des fichiers de contrôle (control files)
   +- pg_control dans $PGDATA/global (/var/lib/postgresql/15/main/global/pg_control) où se situe les control files état arrêt/démarre en cas d'arrêt brutal.
   +- état de la base en fonction de l'arrêt
3. Si récupération nécessaire ? : checkpoint
   +- où sont disponibles les transactions manquantes ?
   +- sur data file (disque) ou non ?
4. Si transactions disponibles sur WAL : (pour compléter les datas en jouant les transactions)
   +- relancement des transactions
5. Si pas de de récupération possible ? erreur
   On ne peut pas démarrer
6. Si tout est ok écoute 5432 et autorise les connexions

Arrêt

1. OS : Signal d'arrêt
   +- Si Type SIGINT (intermédiaire)
2. Nouvelles connexions coupées
3. Stop les autres connexions
4. Rollback des transactions en cours (ne laisse pas terminer même en cours et revient en arrière)
   +- Si Type SIGTERM (soft)
5. Coupe les nouvelles connexions
6. Fini les transactions en cours

Dans tous les cas :

Ecriture d'un checkpoint
Ecriture de la mémoire sur disque (data files)
Mise à jour des control files (pg_control)
Rq : prend du temps (datas) et attente si un checkpoint est déjà en cours.
Si SIGQUIT (arrêt brutal)
+- pas de checkpoint
+- pas d'écriture des datas sur disque
+-pas de maj des control files
+- recovery
+- pertes de transaction (datas)

5-Fonctionnement : les processus

+- processus père : postgres -D <data_dir> -c <conf_dir>
+- principe de fork
+- 2 type de fils
+- les processus du moteur
+- les processus affectés aux connexions
Les processus ??
+- le checkpointeur : permet de poser des jalons mémoire/disque
+- écriture des dirty pages (mémoire) sur les data files (disque)
+- paramétrage de cette fréquence d'écriture
+- soit en durée et/ou en nombre de WAL switch.
(Write Ahead Logging) stock toutes les transactions pour rejouer en cas de mauvais arrêt très utile.
+- configuration des postgressql.conf
+- param : checkpoint_segments et checkpoint timeout (position du jalon)
+- important pour le recovery notamment (coût en perf)
Pause des jalons pour savoir où on est arrivé en écriture disque et ensuite on continu et ainsi de suite.

+- background writter : écriture des dirty pages (mémoire) sur data files (disque)

+- wal writer : écriture de wal buffer sur disque
Wal buffer tous les logs transactions (insert …) sont dans un buffer mais pas sur disque et qui est charger sur disque avec des tailles définie (16Mo)

+- autovaccum launcher : en charge de l'autovaccum
Faire de la défragmentation (cleaner)

+- stats collector : calcule les statistiques
+- hyper important : calcule les statistiques
| (traitement efficace des requêtes)
+- écriture dans pg stats_temp
Faire les requêtes plus rapidement comme un select contrôle le volume des données en statisqtique pour exécuter plus rapidement.

+- logical replication launcher :

+- en cas de réplication logique
+- lance un process pour les standbys qui se connectent

+postgres (idle) : pour chaque connexion
+- allocation de la ressource +- work memory

```
root@debian:/home/debian# ps auxf | grep postgres
postgres     602  0.0  0.2 219680 30236 ?        Ss   09:17   0:00 /usr/lib/postgresql/
15/bin/postgres -D /var/lib/postgresql/15/main -c config_file=/etc/postgresql/15/main/p
ostgresql.conf
postgres     645  0.0  0.0 219812  8616 ?        Ss   09:17   0:00  \_ postgres: 15/mai
n: checkpointer
postgres     646  0.0  0.0 219828  6084 ?        Ss   09:17   0:00  \_ postgres: 15/mai
n: background writer
postgres     720  0.0  0.0 219680 10452 ?        Ss   09:17   0:00  \_ postgres: 15/mai
n: walwriter
postgres     721  0.0  0.0 221272  8752 ?        Ss   09:17   0:00  \_ postgres: 15/mai
n: autovacuum launcher
postgres     722  0.0  0.0 221252  7072 ?        Ss   09:17   0:00  \_ postgres: 15/mai
n: logical replication launcher
root        2503  0.0  0.0   6332  2100 pts/1    S+   12:14   0:00  |
     \_ grep postgres
```

La première instance de postgres est main mais si on veut plusieurs instance et isolation c'est par cluster

Nous avons les fichiers de configuration de la data

```
root@debian:/home/debian# ls /var/lib/postgresql/15/main/
base           pg_multixact  pg_stat      PG_VERSION            postmaster.pid
global         pg_notify     pg_stat_tmp  pg_wal
pg_commit_ts   pg_replslot   pg_subtrans  pg_xact
pg_dynshmem    pg_serial     pg_tblspc    postgresql.auto.conf
pg_logical     pg_snapshots  pg_twophase  postmaster.opts
```

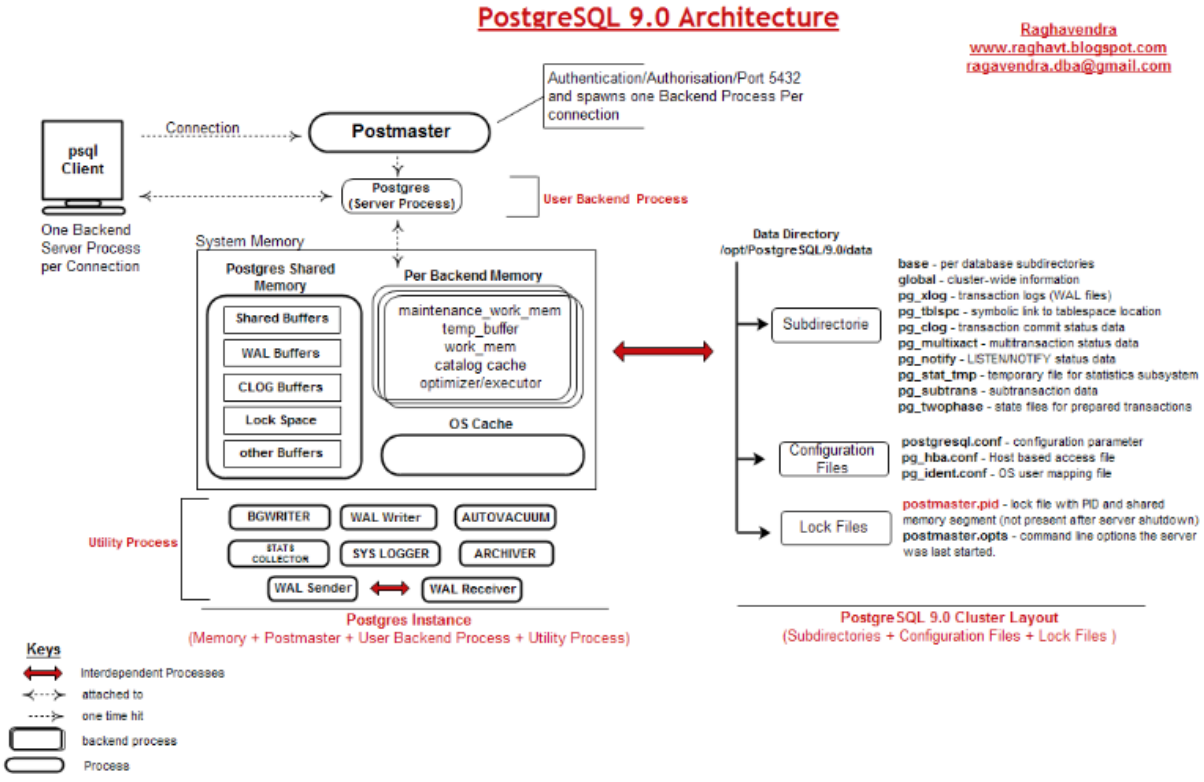
Nous avons les fichiers d'instances ici une c'est le main

```
root@debian:/home/debian# ls /etc/postgresql/15/main/
conf.d       pg_ctl.conf  pg_ident.conf    start.conf
environment  pg_hba.conf  postgresql.conf
```

6-Fonctionnement : la mémoire

+- important pour un moteur de BDD
+- accessibilité rapide à la donnée
+- la mémoire n'est pas pérenne
    +-cas d'arrêt : prévu > système d'écriture
    +- incident : imprévu > perte de la mémoire / memory (rejoue à partir des wal)

+- postgres scinde en 4 la mémoire disponible :
    +- shared buffer : mémoire partagée : (shared bufferwal buffer, clog buffer)
    +- wal buffers :
    +- work mem
    +- maintenance work mem
+- on appelle shared memory = shared buffer + val buffer + c-log-buffer
+- aide pour configure : https://pgtune.leopard.in.va/



Shared buffer
+- par défaut 128 Mo
+- au mini 128ko + 16ko par max connexion
+-si plus 25% de celle du système
+- cette partie de la mémoire est ensuite écrite sur datafile
+- data plus rapide que sur datafile
+- background writer : process chargé de la copie

WAL buffer
+- en charge des archives des transactions
+- cette partie est ensuite écrite dans les WAL files puis archivé
+-wal write : en charge de la copie (wal buffer vers sur le disque)

+- archiver + process en charge de l'archivage

Work mem
+- mémoire allouée pour chaque utilisateur
+-permet les tri, classement, filtre, jointures… des requêtes clientes
+- affectation pour chaque connexion
+- cf postgresql.conf > reloead nécessaire

Maintenance work mem
+-mémoire dédiée aux opérations de maintenance
+-vaccum (défragmentation en cas de table supprimée), create index, reindex …
+-valeur qui peut jouer sur la vitesse de restauration

7-Fonctionnement : les répertoires

+- 2 essentiels
/var/lib/postgresql/15/main/ (instance et base sql pour restauration ou graphique)
/etc/postgresql/15/main/ (configuration)

```
root@debian:/home/debian# ls /var/lib/postgresql/15/main/
base           pg_multixact  pg_stat      PG_VERSION            postmaster.pid
global         pg_notify     pg_stat_tmp  pg_wal
pg_commit_ts   pg_replslot   pg_subtrans  pg_xact
pg_dynshmem    pg_serial     pg_tblspc    postgresql.auto.conf
pg_logical     pg_snapshots  pg_twophase  postmaster.opts
```

+-base
- +-localisation des bases
- +-template 0 et template 1 : appelée lors de la création de BD (base modèle pur créer des databases)
- +-postgresql
- +-classement par oid (tables de correspondance entre oid et objects (oid id spécifique à Postgres. /main/base/ls (requetes sur oid corrrespond une table ou autre équivalence en objet et oid.
- +-SELECT relname , oid, refilemode FROM pg_class wher relname='paul';
- +-ou SELECT pg_relation_filepath('paul')
- +-free Space Map (FSM) et visibility Map (VM) : espace dispo, lien avec Vacuum

+-global
- +- control files : pg_control appelé au démarrage
- +- connaitre l'état du moteur

Dans le main/global/
/pg_control
Gère les control file et permet à postgres de savoir où il est et couplé au checkpoint.
Pour reprendre si nécessaire
/pg_wal
Contient les fichiers d'archivage les écritures de log de transaction contiennent à chaque fois que postgres reçoit une requête.
/pg_stat
Statistique permet l'élaboration des plans de requêtes (select retranscrit sur plusieurs requêtes pour optimiser la requête et pour gagner du temps)

+- pg_tblspc :
    +- pointeur vers les tablespaces
+- pg_repslot :
    +- informations sur la réplication par slot

```
root@debian:/home/debian# ls /etc/postgresql/15/main/
conf.d        pg_ctl.conf  pg_ident.conf    start.conf
environment   pg_hba.conf  postgresql.conf
```

Le conf.d principale fichier de postgres personnalise des configurations ou faire des includes.
Le pg_hba permet la connexion à la base de données (certaines ip ou range_ip pour toutes les bases ou certaines ou sur des users)
Pg_ctl.conf passer des options supplémentaires au lancement du service postgresql
Le start.conf mode auto pour la machine qui redémarre ou manuel.
Le pg_ident.conf fait du mapping user système et user postgres (user system pour psql)

Entrer dans psql
>su – postgres
>psql
Postgres# SELECT relname, oid, relfilenode FROM pg_class;

relname	oid	relfilenode
pg_statistic	2619	2619
pg_type	1247	0
pg_toast_1255	2836	0
pg_toast_1255_index	2837	0
pg_toast_1247	4171	0
pg_toast_1247_index	4172	0
pg_toast_2604	2830	2830
pg_toast_2604_index	2831	2831
pg_toast_2606	2832	2832
pg_toast_2606_index	2833	2833
pg_toast_2612	4157	4157
pg_toast_2612_index	4158	4158
pg_toast_2600	4159	4159
pg_toast_2600_index	4160	4160
pg_toast_2619	2840	2840
pg_toast_2619_index	2841	2841
pg_toast_3381	3439	3439
pg_toast_3381_index	3440	3440
pg_toast_3429	3430	3430
pg_toast_3429_index	3431	3431
pg_toast_2618	2838	2838

8-Autorisation d’accès: pg_hba

/etc/postgresql/15/main
Netstat -ntaup
Le postgres sql écoute 5432 et écoute le 127.0.0.1
+- ip d’écoute : listen adresse
    +- ou ip
    +- plusieurs séparés par virgule
    +- changement restart
/etc/postgres/15/main/postgresql.conf
Listen_addresses = ‘*’

service postgresql restart
si on a 0.0.0.0 :5432 on écoute sur toutes les ip

+- pg_hba.conf (host base authentification)
	+-Host Based Authntification
	+- /etc/postgresql/…
	+- lu de haut en bas (pour le déroulement de la configuration)
	+- reload pour prise en compte

+- filtre multifacteurs :
	+- types
	+- database
	+-user
	+-ip/net
	+-méthode d'authentification md5, ldap)
Les facteurs
+-type : local ou host (distant)
+-database : spécifique ou global (all)
+-user : spécifique ou global (all)
+-adresse :
	+-une ip, un masque de sous réseau (CIDR)
	+-ipv4 ou ipv6
+-méthode :
	+trust : sans login
	+peer : user de l'OS (exemple postgres)
	+password : user/password simple
	+md5 : avec chiffrement
	+reject : refus (utile en cas de maintenance)
	+ldap…

Exemple
Entrer dans psql
>su – postgres
>psql
>\q
postgres@postgres: -$ logout

on va créer un usr avec psql
>create user paul encrypted password 'password'
Puis dans pg_hba: host all all 127.0.0.1/32 md5
psql -h 127.0.01 -U paul postgres
Password for user paul : …
Connexion à la base

Modif pg_hba: host all all 127.0.0.1/32 reject
Nous n'avons plus accès
Si nous avons des problèmes il faut avoir le reflexe :
less /var/log/postgresql/postgresql-11-main.log

/---création de database---/
>su – postgres
>psql
>create database pilpaul
>\q

pg_hba> host pipaul   paul   127.0.0.1/32 md5 #paul peut se connecter à la table pipaul
pg_hba> host postgres criss 127.0.0.1/32 md5 #le user criss peut uniquement se connecter à la base postgres

connexions : memory
 +- max_connexion : 400 bytes par connexion
Rq : si elles ne sont pas établies
+- connexions établies : paramètres work_mem par connexion établie (128mo)

9- psql : le client indispensable

PSQL : les débuts
2 modes de connexions en local ou passe par la socket
Remarque :
+- cluster : main
+- port : 5432
+- serveur : localhost
+-user : postgres

Créer un utilisateur postgresql
Depuis root >su postgres   (l'utilisateur par défaut n'a pas de mot de passe et il est accessible que en local)
postgres@debian: createuser - -interactive criss   (option -P pour un mot de passe sinon le faire via psql)
shall the new role be a superuser? (y/n) y
connexion psql: postgres@debian: psql
postgres=# \password criss
Enter new password for user criss: criss
Enter it again: criss

+- accès à psql avec des options et en passant par la socket (pas local)
        psql -h 127.0.0.1 -p 5432 -U criss -d postgres

        Rq : --cluster dans le cas de plusieurs cluster ou nom différent

Il faut aussi le déclarer dans /etc/postgres/15/main/pg_hba.conf
IPV4 local connections
host   all   all   127.0.0.1/32   scram-sha-256
host   postgres   criss   127.0.0.1/32   scram-sha-256

**********UTILISATION DECOULE**********

postgresql:commande psql :
\l = liste des bases
\d = liste des tables
\du = liste des utlisateurs
\dn+ = lister les schémas et les droits
\q = quitter
\h = aide
USE labase = pour se connecter sur la base
\c labase = pour se connecter sur la base
SELECT version() ; = version PostgreSQL
SELECT current date ; = date actuelle
\ ? aide mémoire psql
\c nom_base nom_utilisateur #connecte à la base nom_base avec e rôle nom_utilisateur

Créer un utilisateur postgresql non sécurisé avec les commandes SQL

```
postgres=# CREATE ROLE <nom_utilisateur> LOGIN;
postgres=# ALTER ROLE <nom_utilisateur> CREATEDB;
postgres=# CREATE DATABASE <nom_base_de_donnee> OWNER <nom_utilisateur>;
```

```
postgres=# ALTER ROLE <nom_utilisateur> WITH ENCRYPTED PASSWORD
'mon_mot_de_passe';
```

----------Linux----------

>cat /etc/passwd | cut -d : -f 1 |sort

```
polkitd
postgres
proxy
root
rtkit
```

Créer un utilisateur :
root>adduser criss
Ajouter l'utilisateur au groupe wheel :
root>usermod -a -G wheel criss
root>groups criss
supprime l'utilisateur :
root>userdel -r criss
l'utilisateur perds la commande sudo su :
 User is not in the sudoers file
l'utilisateur debian n'a pas accès à sudo donc
> su root
  password:???
root@debian:/home/debian# nano /etc/sudoers
#User privilege specification
root ALL=(ALL:ALL) ALL
debian ALL=(ALL:ALL) ALL   #ajouter debian pour utiliser sudo
---

```
debian@debian:~$ psql -h 127.0.0.1 -p 5432 criss -d postgres
Password for user criss:
psql (15.3 (Debian 15.3-0+deb12u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=#
```

```
debian@debian:~$ psql -h 127.0.0.1 -p 5432 -U  criss -d postgres
Password for user criss:
psql (15.3 (Debian 15.3-0+deb12u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=#
```

Pour se connecter à une autre instance
psql –cluster 15/instance1 -p 5432 -h 127.0.0.1 -U criss -d maBase

Nous pouvons passer des commandes psql en ligne de commande directement sans être dans psql :
postgres@debian: psql -c "\l+" / psql -c "\dt" #liste toute les tables

pour la doc : postgres@debian: psql - -help

Nous pouvons faire directement du sql :
postgres@debian: psql -c "SELECT * FROM table"

postgres@debian: psql -f monficsql.sql

10- Databases : template, création

Databases:
Serveur > Cluster > Databases > Tables

+-database = un contenant partageant ses ressources (au sein du cluster)
+- l'objet élémentaire est la table
+- les tables sont composée de colonnes et de lignes
+-les colonnes sont différentes types : int, varchar, date ….
/etc/postgresql/15/main/pg_hba.conf

# TYPE	DATABASE	USER	ADDRESS	METHOD
host	all	criss	127.0.0.1/32	md5
Local	all	all	127.0.0.1/32	peer

Postgres=# \l
 Tables : postgres
           template0 #ne pas toucher c'est un template de référence
           template1 #templace créer depuis template0 donc mieux utiliser template1

Création d'une database
CREATE DATABASE name
[[WITH] [OWNER] [=] user_name]
[TEMPLATE [=] template]
[ENCODING [=] encoding] >> encodeage (utf8 ….)
[LC_COLLATE [=] lc_collate] >> tri des strings
[LC_CTYPE [=] lc_type] >> type de caractère (maj, min …
[TABLESPACE [=] tablespace] >>endroit où sont stockées la database autre
[CONNECTION LIMIT [=] connlimit] >>limite la connexion

Postgres=# CREATE DATABASE crissbas ;
CREATE DATABASE
Postgres=#\l #visualise les databases

```
crissbase | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc
```

Postgres=#drop database crissbase ; #supprime la base

+-créer une base : CREATE DATABASE mydb WITH ENCODING 'UTF-8' TEMPLATE TEMPLATE1;
Templare1 > template0 > creation DB
+- avec le wrapper
created -T template1 mydn
Rqm: on peut utiliser n'importe quelle base à l'arrêt comme template (modèle)

Postgres=#alter database mydb
ALLOW LIMIT OWNER TO RENAME TO SET
CONNECTION LIMIT OWNER TO
Postgres=#alter database owner to criss; #renomme le propriétaire de la database

Postgres=#create database mytemplate1,
CREATE DATABASE
Postgres=#create table matable (id int, champs1 varchar(255));

Postgres=#\c #postgres template0 template1 mytemplate1
Postgres=#\c mytemplate1 #you are now connected to database mytable as user postgres
Postgres=#\dt #did not find any relations
Ici nous avons créé la table dans la base de données postgres donc non !!!!
Postgres=#drop table matable ; #supprime matable sur la base de données postgres on la veux dans mytemplate.
Postgres=#\c mytemplate1#you are now connected to database mytempplate1 as user postgres
mytemplate1=# create table matable (id int, champs1 varchar(255));
mytemplate1=#\dt
mytemplate1=#\q
Postgres=#create database mydatabase #sans template
mydatabase =#\c mydatabase
mydatabase =#\dt #did not find any relations on efface la base de donnée on va créer avec un template
Postgres=#create database mydatabase template mytemplate;
Postgres=#\c mydatabase
Postgres=#\dt et nous avons les autres tables qui ont été créée sur l'autre template.

Postgres=#

Postgres=#
Postgres=#

Postgres=#

11- POSTGRESQL: Tables

Serveur > Cluster > Databases > Tables

+- database = un contenant partageant ses ressources (au sein du cluster)
+- l'objet élémentaire est la table
+- les tables sont composées de colonnes et de lignes
+- les colonnes de différents types int, char, varchar, date, float, double, timstamp …
+- différents types de tables :
    +- classiques (create table standard)
    +-temporaires : seulement en mémoire (perdue en cas de crash ou arrêt mais très rapide) pour job
    +-avec héritage : récupère les colonnes de la table mère
    +-partitionnement : découpage en plusieurs (sous table) par élément d'une colonne
    +-non loguée : sans écriture des logs (traces des transactions) > plus rapide mais perte datas si crash.

Création d'une table
CREATE TABLE matable (id int, champs1 varchar(255))
Supprimer une table
DROP matable ;
Vider la table :
TRUNCATE TABLE matable ;
Modifier
ALTER TABLE (ALTER COLUMN ou ADD COLUMN) …

```
postgres=# CREATE DATABASE criss;
CREATE DATABASE
postgres=# \c criss
You are now connected to database "criss" as user "postgres".
criss=# CREATE TABLE matable (id int, champs1 varchar(255));
CREATE TABLE
criss=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | matable | table | postgres
(1 row)

criss=#
```

Inserer 2 millions de valeurs dans la table toto

```
criss=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | matable | table | postgres
(1 row)

criss=# CREATE TABLE toto (champs1 int);
CREATE TABLE
criss=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | matable | table | postgres
 public | toto    | table | postgres
(2 rows)

criss=# \dt+
criss=# \timing
Timing is on.
criss=# INSERT INTO toto SELECT * FROM GENERATE_SERIES(1,2000000);
INSERT 0 2000000
Time: 1798.232 ms (00:01.798)
criss=#
```

\dt+

```
public | matable | table | postgres | permanent   | heap          | 0 bytes |
public | toto    | table | postgres | permanent   | heap          | 69 MB   |
```

Compte les lignes :

```
criss=# select count(*) from toto
  count
---------
 2000000
(1 row)

Time: 64.106 ms
```

Nous passons en mode non logué (sans les logs nous avons divisé par deux le temps d'insertion mais on ne peut rejouer la transaction))

```
criss=# alter table toto
ADD                       DROP                      RENAME
ALTER                     ENABLE                    REPLICA IDENTITY
ATTACH PARTITION          FORCE ROW LEVEL SECURITY  RESET
CLUSTER ON                INHERIT                   SET
DETACH PARTITION          NO                        VALIDATE CONSTRAINT
DISABLE                   OWNER TO
criss=# alter table toto set
(              LOGGED         TABLESPACE     WITH
ACCESS METHOD  SCHEMA         UNLOGGED       WITHOUT
criss=# alter table toto set unlogged;
ALTER TABLE
Time: 576.543 ms
criss=# INSERT INTO toto SELECT * FROM GENERATE_SERIES(1,2000000);
INSERT 0 2000000
Time: 634.972 ms
criss=#
```

Temporaires (en mémoire)
CREATE TEMPORARY TABLE (champs1 int);

Héritage
CREATE TABLE titi (champs2 char(2)) INHERITES (toto);

```
criss=# \dS toto
            Unlogged table "public.toto"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 champs1 | integer |           |          |

criss=# create table titi (champs2 char(2)) inherits (toto);
CREATE TABLE
Time: 5.589 ms
criss=# \dS titi
                  Table "public.titi"
 Column  |     Type     | Collation | Nullable | Default
---------+--------------+-----------+----------+---------
 champs1 | integer      |           |          |
 champs2 | character(2) |           |          |
Inherits: toto
```

Partitionnement
CREATE TABLE temperature (measure_timestamp TIMESTAMPZ,
Sensor_id INTEGER,
Measure_value DOUBLE PRECISION,
Measure_unite custom_enum_unit)
PARTITION BY RANGE (measure_timestamp);

CREATE TABLE temperature_201709
PARTITION OF temperature
FOR VALUES FROM (‘2017-09-01’) TO (‘2017-10-01’);

CREATE TABLE temperature_201710
PARTITION OF temperature
FOR VALUES FROM (‘2017-10-01’) TO (‘2017-11-01’);

INSERT INTO temperature SELECT
Tval, sensorid, (100.0*random())-50.0, ‘°C’
FROM generate_series(‘2017-10-01 00:00:00’, CURRENT_TIMSTAMP, ‘1 minute’::interval) tval
CROSS JOIN generate_series(1,1000,1) sensorid

https://www.loxodata.com/post/cantwaitpg11_part/

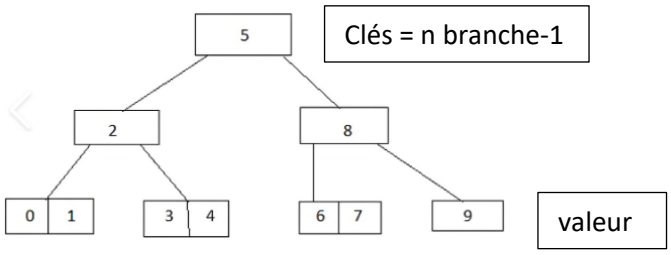
partitioner une table pour travailler avec des tables plus rapidement sans trop de données

12- POSTGRESQL: Index : B-Tree

+-Objectif : augmente la vitesse des requêtes
+- index = table structurée pour un besoin
+-index = pointeur
+-attention : demande de la ressource et demande de la performance
+- index = entretien (réindex)
+-index = baisse performance en écriture
+-postgresql dispose de plusieurs types d'index :
+- b-tree (r-tree spatial postgis)
+- hash
+- Gist *
+-Gin *
+- Brin *
+-création d'un index :
CREATE INDEX idx_criss ON crisstable USING BTREE (t_champs1);
+-B-Tree : cas d'utilisateion : < <= >= > (Ne s'applique pas : recherche text ou plainText recherche de mot)
+-table longue
+-champs de jointures
+-peu de valeurs rapatriées
+-éviter sur les tables mise à jour régulièrement

+-complément : https://fr.wikipedia.org/wiki/Arbre_B

B-Tree
+- Balancing tree : arbre équilibré (bonne répartition des données arbre inverse)
+- créé par Rudolf (boeing)
+- Principe :: ex : 0 1 2 3 4 5 6 7 8 9



Recherche 6 = 6>5 => 8>6 = 6
Création d'une table B-tree

```
criss=# CREATE TABLE Btree1 (id int);
CREATE TABLE
Time: 4.669 ms
criss=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | btree1  | table | postgres
 public | matable | table | postgres
 public | titi    | table | postgres
 public | toto    | table | postgres
(4 rows)

criss=# \dS btree1
              Table "public.btree1"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 id     | integer |           |          |

criss=# INSERT INTO btree1 SELECT * FROM GENERATE_SERIES(1,2000000);
INSERT 0 2000000
Time: 998.701 ms
```

Recherche d'une valeur :

```
criss=# SELECT * FROM btree1 WHERE id=1555555;
   id
---------
 1555555
(1 row)

Time: 46.527 ms
criss=# EXPLAIN ANALYZE select * from btree1 WHERE id=1555555;
                                                      QUERY PLAN

------------------------------------------------------------------------------------------
------------------------------
 Gather  (cost=1000.00..20266.77 rows=1 width=4) (actual time=47.649..51.130 rows=1 loo
ps=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Seq Scan on btree1  (cost=0.00..19266.67 rows=1 width=4) (actual time=2
8.473..31.368 rows=0 loops=3)
         Filter: (id = 1555555)
         Rows Removed by Filter: 666666
 Planning Time: 0.061 ms
 Execution Time: 51.152 ms
(8 rows)
```

<u>Nous allons créer un index</u>

```
criss=# CREATE INDEX idx_btree1 ON btree1 (id);
CREATE INDEX
Time: 813.521 ms
criss=# \dS btree1
              Table "public.btree1"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 id     | integer |           |          |
Indexes:
    "idx_btree1" btree (id)
```

```
criss=# SELECT * FROM btree1 WHERE id=1555555;
   id
---------
 1555555
(1 row)

Time: 1.269 ms
```
Nous avons réduit le temps de la requête

13- POSTGRESQL: Index : hash

+-Objectif : augmente la vitesse des requêtes
+- index = table structurée pour un besoin
+-index = pointeur
+-attention : demande de la ressource et demande de la performance
    +- index = entretien (réindex)
    +-index = baisse performance en écriture
+-postgresql dispose de plusieurs types d'index :
    +- b-tree (r-tree spatial postgis)
    +- hash
    +- Gist *
    +-Gin *
    +- Brin *
+-création d'un index :
CREATE INDEX idx_criss ON crisstable USING HASH (t_champs1);
+-B-Tree : cas d'utilisateion : = ou <> (performant s'il trouve cette valeur ou qui n'est pas cette valeur)
    +-table longue
    +-champs de jointures
    +-peu de valeurs rapatriées
    +-éviter sur les tables mise à jour régulièrement

+-complément : https://fr.wikipedia.org/wiki/Arbre_B

Hash
+- une fonction de hashage => hash la clé (clé/valeur)
+- le résultat est un entier qui détermine le positionnement du pointeur
+- le pointeur pointe vers la ligne correspondante

Exemple :

1. La table

Nom	Données
Paul	Rouge
Jean	Vert
René	bleu

2. Le hachage
   hash(Paul) = 2 (prend Paul et hash Paul a une valeur entière 2)
   hash(Jean) = 3
   hash(René) = 1

3. L'index

La ligne généré par le hash comme hash(Paul) = 2 correspond à la ligne du pointeur ligne 2 et on lui affecte un pointeur (B)

Index	Pointeur
René	A
Paul	B
Jean	C

4. Recherche

   1- je cherche Paul

   2- hash(Paul)=2 #ligne 2 et on trouve le pointeur B et je suis renvoyer à la ligne Paul Rouge

   3- recherche du pointeur dans l'index 2

   4- correspondance du pointeur dans la table

   5-rouge

Select Time : 0.518 ms au lieu de 43.784 ms : avec le hash

```
criss=# CREATE TABLE hash1 (id int);
CREATE TABLE
Time: 6.896 ms
criss=# INSERT INTO hash1 SELECT * FROM GENERATE_SERIES(1,2000000);
INSERT 0 2000000
Time: 1125.797 ms (00:01.126)
criss=# SELECT * FROM hash1 WHERE id=1555555;
   id
---------
 1555555
(1 row)

Time: 43.784 ms
criss=# CREATE INDEX idx_hash1 ON HASH  (id);
ERROR:  relation "hash" does not exist
Time: 0.567 ms
criss=# CREATE INDEX idx_hash1 ON hash1 USING  HASH  (id);
CREATE INDEX
Time: 2218.823 ms (00:02.219)
criss=# SELECT * FROM hash1 WHERE id=1555555;
   id
---------
 1555555
(1 row)
```

14- POSTGRESQL: Vues classique et matérialisée

+-vue classique : requête utilisable sous forme de table

Permet aussi de fournir des droits sur une vue et non sur la table comme un ex tract de données.

Création de la table et insertions

```
criss=# CREATE TABLE crissvue1 (id INT, champs1 VARCHAR);
CREATE TABLE
Time: 12.745 ms
criss=# INSERT INTO crissvue1 (id, champs1) VALUES(1, 'pierre');
INSERT 0 1
Time: 4.447 ms
criss=# INSERT INTO crissvue1 (id, champs1) VALUES(2, 'paul');
INSERT 0 1
Time: 3.406 ms
criss=# INSERT INTO crissvue1 (id, champs1) VALUES(3, 'jacques');
INSERT 0 1
Time: 3.160 ms
```

+- création de la vue

```
criss=# select count(*) from crissvue1;
 count
-------
     3
(1 row)

Time: 0.524 ms
criss=# CREATE VIEW criss_count AS SELECT COUNT(*) FROM crissvue1;
CREATE VIEW
Time: 4.332 ms
criss=# \dv
           List of relations
 Schema |    Name     | Type |  Owner
--------+-------------+------+----------
 public | criss_count | view | postgres
(1 row)

criss=# \dS

criss=# CREATE VIEW criss_champs1 AS SELECT champs1 FROM crissvue1;
CREATE VIEW
Time: 4.180 ms
criss=# \dv
            List of relations
 Schema |     Name      | Type |  Owner
--------+---------------+------+----------
 public | criss_champs1 | view | postgres
 public | criss_count   | view | postgres
(2 rows)
```

```
criss=# select * from criss_champs1;
 champs1
---------
 pierre
 paul
 jacques
(3 rows)
```

Permet de restituée la vue

Si nous supprimons une ligne de la table d'origine la conséquence sur la vue et prend automatiquement le changement Mais attention si on fait évoluer la structure de la table il faut recréer la vue !!!

```
criss=# select * from crissvue1;
 id | champs1
----+---------
  1 | pierre
  2 | paul
  3 | jacques
(3 rows)

Time: 1.188 ms
criss=# DELETE FROM crissvue1 WHERE id=1;
DELETE 1
Time: 2.885 ms
criss=# select * from criss_champs1;
 champs1
---------
 paul
 jacques
(2 rows)

Time: 0.793 ms
criss=#
```

+- Nous faisons évoluer note table ajouter un champs2

```
criss=# ALTER TABLE crissvue1 ADD COLUMN champs2 VARCHAR NOT NULL DEFAULT NOW
()::date;
ALTER TABLE
Time: 3.554 ms
criss=# \dS crissvue1
                    Table "public.crissvue1"
 Column  |       Type        | Collation | Nullable |   Default
---------+-------------------+-----------+----------+-------------
 id      | integer           |           |          |
 champs1 | character varying |           |          |
 champs2 | character varying |           | not null | now()::date
```

Nous recréons une vue constituer avec un select * c'est important quand on va refaire évoluer ajouter un champs3

```
criss=# select * from crissvue1;
 id | champs1 |  champs2
----+---------+------------
  2 | paul    | 2023-11-17
  3 | jacques | 2023-11-17
(2 rows)

Time: 0.228 ms
criss=# CREATE VIEW v_crissvue1_all AS SELECT * FROM crissvue1;
CREATE VIEW
Time: 4.786 ms
criss=# select * from v_crissvue1_all;
 id | champs1 |  champs2
----+---------+------------
  2 | paul    | 2023-11-17
  3 | jacques | 2023-11-17
(2 rows)
```

Ajouter un champs3 et utiliser la dernière vue et ici elle n'est pas à jour.

```
criss=# ALTER TABLE crissvue1 ADD COLUMN champs3 VARCHAR NOT NULL DEFAULT NOW
():::date;
ALTER TABLE
Time: 4.942 ms
criss=# select * from crissvue1;
 id | champs1 |  champs2   |  champs3
----+---------+------------+------------
  2 | paul    | 2023-11-17 | 2023-11-17
  3 | jacques | 2023-11-17 | 2023-11-17
(2 rows)

Time: 1.171 ms
criss=# select * from v_crissvue1_all;
 id | champs1 |  champs2
----+---------+------------
  2 | paul    | 2023-11-17
  3 | jacques | 2023-11-17
(2 rows)

Time: 1.186 ms
```

Donc obliger de supprimer la vue et la recréer

```
criss=# DROP VIEW v_crissvue1_all;
DROP VIEW
Time: 4.372 ms
criss=# CREATE VIEW v_crissvue1_all AS SELECT * FROM crissvue1;
CREATE VIEW
Time: 5.144 ms
criss=# select * from v_crissvue1_all;
 id | champs1 |  champs2   |  champs3
----+---------+------------+------------
  2 | paul    | 2023-11-17 | 2023-11-17
  3 | jacques | 2023-11-17 | 2023-11-17
(2 rows)

Time: 1.064 ms
```

+- La vue matérialisée : (cela permet de faire des backups à un instant)

+- vue matérialisée = copie de la table (data) à un instant donné

+- création de la vue matérialisée

C'est une copie d'une table en parlant des datas et structure à un instant donné.
Donc si on supprime une donnée la vue matérialisée ne pourra pas se mettre à jour.

```
criss=# CREATE MATERIALIZED VIEW vm_criss AS SELECT * FROM crissvue1;
SELECT 2
Time: 11.745 ms
criss=# select * from cm_criss;
criss=# select * from vm_criss;
 id | champs1 |  champs2   |  champs3
----+---------+------------+------------
  2 | paul    | 2023-11-17 | 2023-11-17
  3 | jacques | 2023-11-17 | 2023-11-17
(2 rows)

Time: 0.568 ms


criss=# DELETE FROM crissvue1 WHERE id=2;
DELETE 1
Time: 2.994 ms
criss=# select * from crissvue1;
 id | champs1 |  champs2   |  champs3
----+---------+------------+------------
  3 | jacques | 2023-11-17 | 2023-11-17
(1 row)

Time: 0.406 ms
criss=# select * from vm_criss;
 id | champs1 |  champs2   |  champs3
----+---------+------------+------------
  2 | paul    | 2023-11-17 | 2023-11-17
  3 | jacques | 2023-11-17 | 2023-11-17
(2 rows)

Time: 0.252 ms
```

\dmv ou \dm

```
criss=# \dvm
                    List of relations
 Schema |      Name       |       Type        |  Owner
--------+-----------------+-------------------+----------
 public | criss_champs1   | view              | postgres
 public | criss_count     | view              | postgres
 public | v_crissvue1_all | view              | postgres
 public | vm_criss        | materialized view | postgres
(4 rows)
```

15- POSTGRESQL: Droits : Rôles et Users

+- gestion des droits : databases, schémas, tables, colonnes …
Il se gère principalement par le pg_hba ou au sein de l'instance …
+- en postgresql : users = rôles (pas avec oracle) users Paul , rôle développeur
+- un user/role existe pour un cluster PG (instance)
+- lien avec le pg_hba pour les autorisation de connexions à la socket

```
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             criss           127.0.0.1/32            md5
```

+- possibilité d'héritage = principe de roles (similaire à Oracle)
    +- user = login + mdp + droits

Listes les users/roles : \du
Créer un ustilisateur : create user
Options :
    +- superutilisateur : fait ce qu'il veut
    +- createdb : autorise la création de database (nocreatedb)
    +- createrole : autorise la création d'autres rôles/users (nocreaterol) …
    +- inherit role : héritage d'un autre rôle en mentionnant son nom
    +- login : autorise la connexion à l'instance
    +- replication : spécifique
    +- connection limit : nombre maximum de connexions concurrentes
    +- password / encrypted password : définition du mot de passe entre simples quotes
    +- valid until : définir une date de péremption/validité
    +- admin : niveau admin (DBA oracle)

<u>Exemple d'utilisateurs de niveaux différents</u>

DROP USER toto ; #supprime l'utilisateur toto
ALTER USER tutu CREATEDB ; #modifie les accès

CREATE USER toto NOLOGIN NOCREATEDB;
CREATE USER tutu WITH ENCRYPTED PASSWORD 'password' LOGIN NOCREATEDB;
CREATE ROLE titi LOGIN NOCREATEDB;
CREATE USER tata LOGIN CREATDB;
\du

```
 criss     | Superuser, Create role, Create DB
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
 tata      | Create DB
 titi      |
 toto      | Cannot login
 tutu      |
```

Connexion par port: il faut compléter le pg_hba.config
postgres@debian:/home/debian$ psql -h 127.0.0.1 -U tutu -d postgres

```
postgres=# \c  postgres toto
connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FA
TAL:  Peer authentication failed for user "toto"
Previous connection kept
```

16- POSTGRESQL: Droits : grants et databases

La gestion des droits spécifiques des databases.

+- important pour du mutualisé : sécurité et isolation
+- instance peu de limite en nb de DB
https://www.endpoint.com/blog/2008/11/10/10000-databases-on-postgresql-cluster
Rq : limite du filesystem
+- gestion des droits via la clause GRANT

Crée une database madb avec le propriétaire criss puis nous allons changer le propriétaire par celui que l'on a créée

```
postgres=# CREATE DATABASE madb OWNER criss;
CREATE DATABASE
postgres=# \l
postgres=# CREATE USER pipi;
CREATE ROLE
postgres=# ALTER DATABASE ladb IWNER TO pipi
```

Affectation des droits GRANT (sur database, table , séquence, …)

```
Command:     GRANT
Description: define access privileges
Syntax:
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER
 }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]
    [ GRANTED BY role_specification ]

GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )
    [, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
    [ GRANTED BY role_specification ]

GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE sequence_name [, ...]
         | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ]
}
    ON DATABASE database_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
    [ GRANTED BY role_specification ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON DOMAIN domain_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
    [ GRANTED BY role_specification ]
```

Le CREATE permet uniquement de créer des schémas dans database

On se sert que sur le GRANT DATABASE

```
criss     | Superuser, Create role, Create DB
pipi      |
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
tata      | Create DB
titi      |
toto      | Cannot login
tutu      | Create DB
```

Crée la database

```
postgres=# CREATE DATABASE pipibase;
CREATE DATABASE
postgres=# \l
 pipibase  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |           | l
ibc            |
```

Modifie les droits de propriétés sur cette database

```
postgres=# ALTER DATABASE pipibase OWNER TO pipi;
ALTER DATABASE
 pipibase  | pipi     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |           | l
ibc            |
```

+-révocation des droits pour tous les users sauf le owner (droit juste pour utilisateur et postgres c'est tout)

```
postgres=# REVOKE ALL ON DATABASE pipibase FROM PUBLIC;
REVOKE
```

+- GRANT ouvrir les droits de connexion à la base à toto

```
postgres=# GRANT CONNECT ON DATABASE pipibase TO pipi;
GRANT
postgres=# \c pipbase pipi
Password for user pipi:
connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: fe
_sendauth: no password supplied
Previous connection kept
```

+-création utilisateur

Create user didi login createdb;

Create user dodo login created;

+- creation d'une database en tant que user postgres

Create database mondb

- Peer : signifie qu'il fera confiance à l'identité (authenticité) de l'utilisateur UNIX. Donc ne demande pas de mot de passe.
- md5 : signifie qu'il demandera toujours un mot de passe et le validera après le hachage avec MD5.
- Trust : signifie qu'il ne demandera jamais de mot de passe et fera toujours confiance à n'importe quelle connexion.

Bon nous avons un problème entre peer et md5

RESOLUTION CREER DES UTILISATEUR AVEC PASSWORD :

CREATE USER tutu WITH ENCRYPTED PASSWORD ‘tutu’ LOGIN CREATEDB;

+- ne peut pas créer de schéma
CREATE SCHEMA myschema ;

+-autorisation de création des schémas :
GRANT CREATE ON DATABASE maDB TO toto;
Attention si un schema existe il faut gérer les droits sur le schéma (REVOKE ….)
On crée ce qu’on veut en public puis on va cloisonner on va créer un schéma et créer des zone sécuriser et ensuite on crée des tables à l’intérieur.

17- POSTGRESQL: les Schémas

+- schémas = vues des objects de la base de données. (les droits à travers les Schémas)
+- objectif : faciliter la gestion des droits au sein d’une base de données.
+- schéma par défaut = public
C’est une sorte de conteneur qui va faciliter les droits au niveau au sein d’une base de données
Par défaut nous avons le Schéma : Public
Création d’un schéma

```
psql (15.3 (Debian 15.3-0+deb12u1))
Type "help" for help.

postgres=# \dn
      List of schemas
  Name  |       Owner
--------+-------------------
 public | pg_database_owner
(1 row)

postgres=# CREATE SCHEMA monshema;
CREATE SCHEMA
postgres=# \dn
       List of schemas
   Name   |       Owner
----------+-------------------
 monshema | postgres
 public   | pg_database_owner
(2 rows)
```

Autoriser la création de schéma pour un utilisateur :
GRANT CREATE DATABASE maBase TO titi;
Supprimer les droits de création de Schéma d’un utilisateur :
REVOKE CREATE DATABASE maBase TO titi;

Tables et Schéma:

+-Comment créer une table dans un schéma ?

+- deux méthodes :

- +- search_path
- +- préfix <nom_schéma>.<nom_table>

1. > CREATE TABLE monschema.tbl1 (id int); #création d'une table

\dt pour visualiser la table ne fonctionne pas il faut faire un search_path = monshema

Mais pour revenir il faut se mette dans son schéma public set search_path = public ;

```
postgres=# CREATE TABLE monshema.tbl1 (id int);
CREATE TABLE
postgres=# \dt
Did not find any relations.
postgres=# select * from monshema.tbl1 ;
 id
----
(0 rows)

postgres=# set search_path = monshema;
SET
postgres=# \dt
         List of relations
  Schema  | Name | Type  |  Owner
----------+------+-------+----------
 monshema | tbl1 | table | postgres
(1 row)
```

Pour créer une table mais sans préfixage on se remet dans le schéma public

```
postgres=# set search_path = public;
SET
postgres=# CREATE TABLE tbl2 (id int);
CREATE TABLE
postgres=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | tbl2 | table | postgres
(1 row)

postgres=# set search_path = monshema;
SET
postgres=# \dt
         List of relations
  Schema  | Name | Type  |  Owner
----------+------+-------+----------
 monshema | tbl1 | table | postgres
(1 row)

postgres=# CREATE TABLE tbl2 (id int);
CREATE TABLE
postgres=# \dt
         List of relations
  Schema  | Name | Type  |  Owner
----------+------+-------+----------
 monshema | tbl1 | table | postgres
 monshema | tbl2 | table | postgres
(2 rows)
```

Nous pouvons remarquer le cloisonnement des tables avec les schémas.
Savoir où l'on est > SHOW SEARCH_PATH;

Comment avec un \dt savoir les tables même si l'on n'est pas dans le bon schéma :
ALTER USER criss SET SEARCH_PATH = monshema;
À la connexion par défaut on sera connecté à monshema

+-2 permissions
    +- USAGE : lecture dans un schéma
    +- CREATE : création de tables dans un schéma

+- grant

REVOKE ALL ON SCHEMA monshema FROM PUBLIC;
\c mabase toto
CREATE TABLE monshema.matable (id int);
GRANT CREATE ON SCHEMA monshema TO toto;
CREATE TABLE monshema.matable (id int);

Ce qu'on souhaite c'est que le user toto ne puisse pas se connecter à monshema ?
Pour tout le monde:
REVOKE ALL ON SCHEMA monshema FROM PUBLIC;
\c mabase toto #on se connecte à mabase avec toto
Set search_path = monshema ;
SET
Show search_path ; #on est bien connecté à monshema
Select * from monshema.tbl1;
 Permission refused
GRANT USAGE SCHEMA monshema TO toto; #permet la lecture des éléments dans shema.
Set search_path = monshema ;
SET
\dt #liste les relations
GRANT CREATE SCHEMA monshema TO toto;
CREATE TABLE monshema.tbl3 (id int);

****ON PEUT CREER DES DROITS ASSEZ FIN AVEC LES SCHEMA SANS GERER LES DROITS UN A UN****

18- POSTGRESQL: Droits : Tables et Colonnes

+-Limitation pour les tables :
    +- SELECT : sélection
    +-INSERT : insertion de lignes
    +-UPDATE : mise à jour de lignes
    +-DELETE : suppression de lignes
    +-TRUNCATE : vider la table
    +-REFERENCES : utilisation de la table comme clef (clé étrangère …)
    +-TRIGGER : mise en place de déclencheurs
    +-ALL

+- Syntax
GRANT ALL ON TABLE tbl1 TO <role_ou_public>
REVOKE ALL ON TABLE tbl1 FROM <role_ou_public>

CREATE TABLE tbl1 (id int, champs1 varchar);
INSERT INTO tbl1 VALUES (1, 'hello');
INSERT INTO tbl1 VALUES (2, 'world');
REVOKE ALL ON TABLE tbl1 FROM toto;
\c user

>\du #utilisateur
>\l #database
>\c criss mabase
CREATE TABLE tbl1 (id int, champs1 varchar);
\dt #on voit la table créée
INSERT INTO tbl1 VALUES (1, 'hello');
INSERT INTO tbl1 VALUES (2, 'world');
REVOKE ALL ON TABLE tbl1 FROM toto;
\c mabase toto #
GRANT INSERT ON TABLE tbl1 TO toto ;
!!Mais pas le droit d'un select

+-Limitation pour les colonnes/champs de tables
    +- SELECT
    +-INSERT
    +-UPDATE
    +-REFERENCES

\c mabase toto
SELECT * FROM tbl1 ;
\c criss mabase
REVOKE ALL ON TABLE tbl1 FROM toto;
GRANT SELECT(champs1) ON TABLE tbl1 to toto;
\c mabase toto
SELECT * FROM tbl1; #la requete veux retourber l'intégralité des champs

SELECT champs1 FROM tbl1;#plus pécis

Remq: partir d'un revoke all et donner les droits petit à petit et de créer les table depuis l'utilisateur et non avec postgres *

19- POSTGRESQL:Logs : DDL, formats

+-Ne pas confondre avec les logfiles (logs de transaction – WAL)
+-répertoire : $PG_DATA/pg_log/ (VAR/LOG/PG_SQL
+- postgresql = decouplage entre la collecte du log et son écriture
    +- processus = logging collector

+-paramétrage dans le /etc/postgresql/../../postgresql.conf
+-paramètres log_destination : type de logs
    +- stderr
    +- csvlog
    +- syslog
    +- eventlog
+-log_directory = localisation
+- log_filename = format des noms des fichiers (standard strftime)
+- attention : log_truncate écrase les fichiers si ON

Less /var/log/postgresql/postgresql-15-main.log
Nano /etc/postgresql/15/maim/postgresql.conf

Log_destination = csvlog (logging collector to be on
Logging_collector = pn
Lod_directory = /tmp
Log_filename = criss
Log_file_mode = 0660

…
Quesqu'on va loger
+-information contenue dans les logs = logs_statement
    +- attention aux performances
    +- none : erreur uniquement
    +- ddl : non + erreur
    +- mod : ddl + changements de datas (insert, update, delete)
    +- all : mod + select
    +- log_line_prefix = format des lignes de logs #mise en page

Log_statement= all ;

Psql> select * from mabase
Et voici les logs de la requête
Ls -larth /tmp
Less /tmp/dernier_fichier.csv

+- logs concernant d'autres éléments : log des connexions
    +- log_checkpoints = on
    +- log_connectionx = on
    +- log_disconnections = on
    +- log_duration = on

Logs – Debug
+- logguer peu pour la performance
+- ponctuelle pour debugger augmenter les logs
+-ajout en fonction du besoin
+-pour une database :
    ALTER DATABASE madb SET log_statement = 'none' #madb plus de log sur une database.
Modification pour debugger mettre à all
    ALTER DATABASE madb SET log_statement = 'all'
+-pour un user:
ALTER USER madb IN DATANASE criss SET log_statement = 'all
+- logger requêtes lentes:
Log_min_duration = -1
Rq: -1 = rien sinon en millisecondes / indépendant de log_statement
Ce qui permet de par exemple 1000ms donc 1sc toutes les requêtes qui dépassent seront logguer.

20- POSTGRESQL: COPY : export et import personnalisés

COPY – réaliser des exports et se passe de INSERT INTO
+- /h COPY #aide
+- création d'une table
CREATE TABLE tbl1 (id int, champsl varchar);
INSERT INTO tbl1 SELECT *, 'l' FROM generate_series(1, 200000);

+- export vers un fichier texte
 COPY tbl1 TO '/tmp/export_tbl.txt'

+- export en csv
COPY tbl1 TO ' /tmp/export_tbl.txt' CSV HEADER;

Requête personnalisé
+-exporter une requête précise
COPY (SELECT COUNT(*) FROM tbl1) TO '/tmp/export_tbl1.txt' ;
txt avec le nombre de ligne.
+- export vers une command shell
COPY tbl1 TO PROGRAM 'wc -l > /tmp/shell_count.txt';
+- import de données – ex : curl 'interroger une api pour collecter des datas'
COPY tbl1 FROM PROGRAM 'curl http://exemple.com'

21- POSTGRESQL: Sauvegarde : pg_dump

+- important mais cela reste une image (on ne peut pas revenir dans le temps)
+- complémentaire à l'archive log
+- pad de méthode unique => à adapter à sa situation
+- ne copie pas les users et autres spécifiques à l'instance
Ne copie pas les users donc juste les tables champs séquences.

+- deux outils (complémentaire) :
    +- pg_dump : personnalisation au maximum
    +- pg_dumpall : backup intégral de l'instance (y compris users …)

Beaucoup d'options (une partie identique à psql) :
+- -h : hostname (eh oui c'est pas l'aide)
+- -d : database
+- -p : port
+- -U : user
+- -f : fichier de sortie
+- -F : précision du format de sortie
    +- c : custom (binaire)
    +- d : directory (permet de parallélisation)
    +- t : tar
    +- p : plain text
+- -j : parallélisation

Liens

https://xavki.blog/postgresql-tutoriaux-francais/
support postgre (créations d'outils comme psql)
https://www.dalibo.com/
https://www.google.com/imgres?imgurl=https://blog.anayrat.info/img/2014/PGArchOverview.png&imgrefurl=https://blog.anayrat.info/2014/12/22/introduction-%25C3%25A0-postgres-part1/&h=756&w=1100&tbnid=34cln6ja2t-JHM&tbnh=186&tbnw=271&usg=AI4_-kTUaXiRhyoIDpqit1vRSaJEE_STFA&vet=1&docid=kkfKAnx7yPHtzM

https://www.postgresql.org/docs/current/reference.html

https://www.postgresql.org/docs/15/app-dropuser.html