

# Espace disque et BDD Postgres

La manipulation des Backups (.dump) de PostgreSQL peut entrainer une saturation d'espace disque et il convient lors de la duplication (ou de la restauration) de bases d'attacher une vigilance particulière sur la volumétrie du disque disponible sur le serveur.

Ce document a pour but de vous « éclairer » sur cet aspect.

- [Prérequis](#)
- [Manipulation des bases de données](#)
  - [Présentation](#)
  - [Restauration d'une base de test](#)
  - [Impact d'une montée de version \(sql-update\)](#)
  - [Déterminer la volumétrie d'une base de données](#)
  - [Echec de restauration de Base de données](#)
- [Pour aller plus loin](#)
- [Procédures](#)
  - [Espace disque et BDD Postgres](#)
  - [Mise en place de filtre Database](#)

# Prérequis

Les principales commandes Ubuntu utilisées dans ce document seront :

- `df` qui nous donne le taux d'occupation disque,
- `du` qui nous donne la taille d'un répertoire,
- `grep` qui nous permet d'effectuer une recherche sur un fichier « txt »,
- `psql`, `pg_dump` & `pg_restore` sont propres à PostgreSQL.

Pour avoir une information complémentaire sur l'usage de ces commandes, taper en ligne de commande `man <cmd>` pour avoir un complément d'information sur leur usage respectif sur votre serveur. Par exemple :

```
openprod@ubuntu18-jmp:~ $ man df
```

La plupart des commandes fournies dans cette documentation le sont à titre indicatif : elles fonctionnent parfaitement sur un serveur PostgreSQL 10 (Ubuntu 18) mais peuvent être à adapter en fonction de votre système d'exploitation ou votre SGBD.

# Manipulation des bases de données

# Présentation

Sous PostgreSQL, la sauvegarde et la restauration de bases s'effectuent par deux commandes distinctes : **pg\_dump** et **pg\_restore**. Chacune de ces deux commandes exploitent des fichiers « dump », l'équivalent sous SQL Server du « .bak ».

Si on regarde un peu plus en détail le contenu d'un fichier .dump, on constate que le .dump est une succession de commande SQL, que le moteur va « rejouer », créant ainsi la structure de la base de données (ses extensions, ses fonctions, ses tables, etc..) et les enregistrements contenus dans chacun des fichiers au moment où la sauvegarde a été produite.



```
-- PostgreSQL database dump
--
-- Dumped from database version 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1)
-- Dumped by pg_dump version 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- Name: ppython3u; Type: PROCEDURAL LANGUAGE; Schema: -; Owner: -
--
CREATE OR REPLACE PROCEDURAL LANGUAGE ppython3u;

--
-- Name: unaccent; Type: EXTENSION; Schema: -; Owner: -
--
CREATE EXTENSION IF NOT EXISTS unaccent WITH SCHEMA public;

--
-- Name: EXTENSION unaccent; Type: COMMENT; Schema: -; Owner: -
--
COMMENT ON EXTENSION unaccent IS 'text search dictionary that removes accents';

--
-- Name: get_field(text, integer, text, text, text, text, integer, text); Type: FUNCTION; Schema: public; Owner: -
--
CREATE FUNCTION public.get_field(server text, port integer, login text, password text, database text, model text, obj_id integer, field text) RETURNS text
    LANGUAGE ppython3u
    AS $$
    if obj_id is None:
        return '0'
    else :
        from xmlrpc.client import ServerProxy
        res = {}
```

Sur Open-Prod, il existe deux type des sauvegardes : celle en .dump, et celle avec le filestore inclus.

Leur contenu est le suivant :

1. Si on produit une sauvegarde avec **pg\_dump**, on obtiendra un fichier directement exploitable dans une version compatible pour tout moteur PostgreSQL de la même version :

 <b>BDD_2023-02-28T16h46.dump</b>	<b>28/02/2023 17:53</b>	<b>Fichier DUMP</b>	<b>1 186 896 Ko</b>
--	-------------------------	---------------------	---------------------

2. Si on produit une sauvegarde avec le filestore inclus, le fichier sera produit au format zip et contiendra les fichiers suivants :

13-08-31.zip

Trier Trier Afficher Afficher Extraire tout

Ce PC > DATA (D:) > Telec > REFERENCE\_2023-02-22\_13-08-31.zip

Nom	Type	Taille compressée	Protégé pa...	Taille	Ratio	Modifié le
documents	Dossier de fichiers					10/03/2022 13:06
filestore	Dossier de fichiers					08/02/2023 14:24
dump.sql	Fichier source SQL	7 506 Ko	Non	62 413 Ko	88 %	22/02/2023 13:08
manifest.json	Fichier source JSON	1 Ko	Non	4 Ko	75 %	22/02/2023 13:08

Le zip sera porteur du nom de la base sur le serveur d’origine et le .dump sera intégré dans ce dernier.

**Attention ! la volumétrie du .dump n’a rien a voir avec l’espace disque qu’utilisera la base de données une fois restaurée. En effet, le .dump est un fichier « txt » qui ne contient que les commandes permettant de réalimenter la base. Avec des champs « vides » par exemple occuperont nécessairement un espace dans un SGBD mais pas dans le dump.**

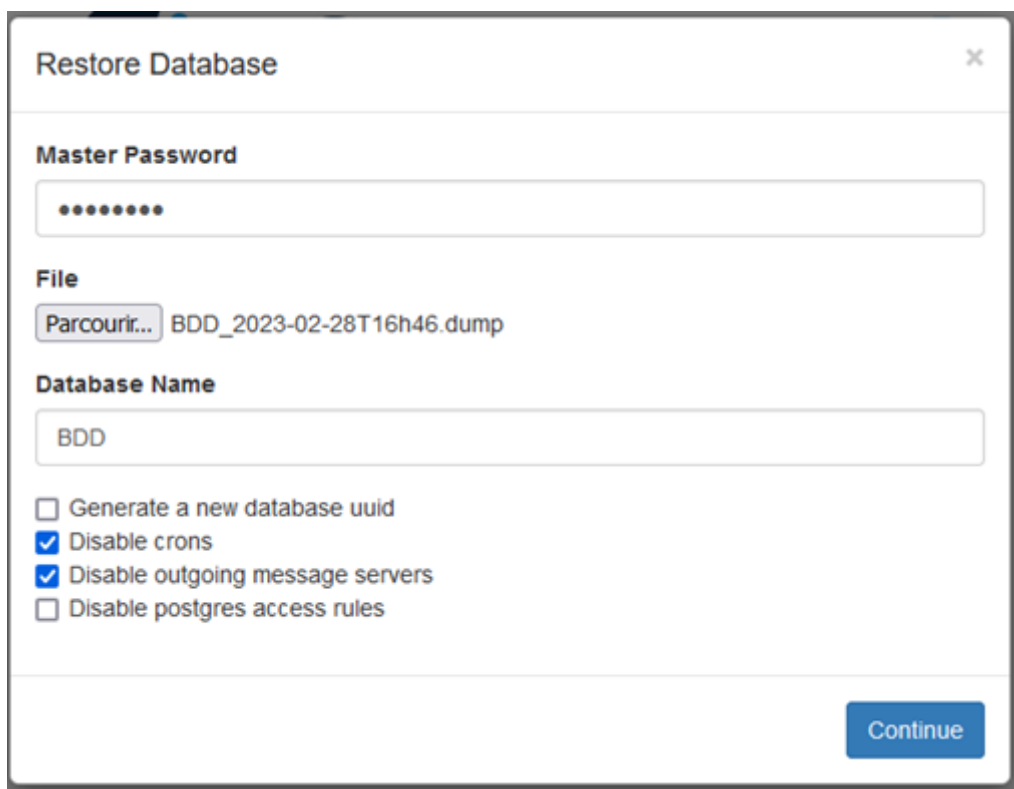
# Restauration d'une base de test

La première étape est de vérifier l'espace disque sur le serveur :

```
openprod@ubuntu18-jmp:~$ df -H
Filesystem      Size  Used Avail Use% Mounted on
udev            4,1G   0    4,1G   0% /dev
tmpfs           817M  938k  816M   1% /run
/dev/mapper/ubun-67G  13G   51G  21% /
tmpfs           4,1G   17k   4,1G   1% /dev/shm
tmpfs           5,3M    0   5,3M   0% /run/lock
tmpfs           4,1G    0   4,1G   0% /sys/fs/cgroup
/dev/sda2       1,1G  160M  791M  17% /boot
tmpfs           817M    0  817M   0% /run/user/1000
openprod@ubuntu18-jmp:~$
```

Nous constatons ici que le disque dur a une taille de 67 Go utiles et que nous avons une disponibilité de 51 Go.

Nous allons donc restaurer notre fichier dump cité plus haut sur ce disque dur au travers de l'interface d'Open-Prod.



**Restore Database**

**Master Password**

.....

**File**

Parcourir... BDD\_2023-02-28T16h46.dump

**Database Name**

BDD

☐ Generate a new database uuid

☒ Disable crons

☒ Disable outgoing message servers

☐ Disable postgres access rules

**Continue**

Avec la commande `tail -f /var/log/openprod/openprod-server.log`, il est possible de "suivre" les traitements réalisés par Open-Prod sur le moment. Nous trouvons l'information suivante :

Le serveur exécute la commande `pg_restore` sur le fichier « /tmp/tmp5SX1I3 », préalablement téléchargé via le navigateur. Une fois restauré, on contrôle à nouveau l'espace disque :

```
openprod@ubuntu18-jmp:~$ df -H
Filesystem                Size      Used Avail Use% Mounted on
udev                     4,1G         0   4,1G   0% /dev
tmpfs                    817M    2,0M    815M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 67G    25G    39G  40% /
tmpfs                    4,1G     17k    4,1G   1% /dev/shm
tmpfs                    5,3M         0   5,3M   0% /run/lock
tmpfs                    4,1G         0   4,1G   0% /sys/fs/cgroup
/dev/sda2                 1,1G   160M    791M  17% /boot
tmpfs                    817M         0   817M   0% /run/user/1000
openprod@ubuntu18-jmp:~$
```

**Nous constatons que le fichier dump de 1.13 Go, une fois restauré occupe +/- 12 Go d'espace disque.**

# Impact d'une montée de version (sql-update)

De nouveau, une montée de version d'Open-Prod peut fortement impacter l'espace disque de l'environnement. Par exemple, si de nouveaux champs fonctionnels sont mis en place par l'éditeur (ou vos modules) dans un grand nombre d'enregistrements, l'occupation définitive du disque peut arriver à saturation.

Naturellement, PostgreSQL va créer des fichiers d'échange temporaire sur disque et il convient que de l'espace disque soit disponible en quantité suffisante lors de l'opération.

Si nous poursuivons sur l'exemple précédent, voici l'espace disque suite à un `sql-update` sur notre base de donnée :

```
openprod@ubuntu18-jmp:~$ df -H
Filesystem                Size      Used Avail Use% Mounted on
udev                     4,1G         0   4,1G   0% /dev
tmpfs                     817M       1,9M   815M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 67G       56G    7,5G  89% /
tmpfs                     4,1G       17k    4,1G   1% /dev/shm
tmpfs                     5,3M         0   5,3M   0% /run/lock
tmpfs                     4,1G         0   4,1G   0% /sys/fs/cgroup
/dev/sda2                 1,1G     160M    791M  17% /boot
tmpfs                     817M         0   817M   0% /run/user/1000
```

**Après la montée de version de notre base dans un format ancien, notre espace disque disponible est tombé à 7.5 Go, la taille de la base a donc augmenté de 32 Go ! Prenez soin d'avoir toujours le maximum d'espace disque disponible lors d'une montée de version et de tester la taille définitive de la future base sur un serveur de test.**



# Déterminer la volumétrie d'une base de données

Il existe de nombreux moyen pour déterminer la taille d'une base de données spécifique.

Via la commande `sudo -u postgres psql` puis `\l+` :

```
openprod@ubuntu18-jmp:~$ sudo -u postgres psql
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \l+

```

Name	Owner	Encoding	Collate	Ctype	Access privileges	Size	Tablespace	Description
BDD	openprod	UTF8	fr_FR.UTF-8	fr_FR.UTF-8		41 GB	pg_default	
openprod	openprod	UTF8	fr_FR.UTF-8	fr_FR.UTF-8		32 MB	pg_default	
postgres	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8		7615 kB	pg_default	default administrative connection database
template0	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	=c/postgres postgres=CTc/postgres	7481 kB	pg_default	unmodifiable empty database
template1	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	=c/postgres postgres=CTc/postgres	7615 kB	pg_default	default template for new databases

```
(5 rows)

postgres=#
```

Via la commande `du` si on connaît le lieu de stockage de nos bases. Ici la base « BDD » correspond au répertoire 94118, soit 42477984 Mo :

```
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base# du -H
7636      ./1
7492      ./13017
4         ./pgsql_tmp
42477984  ./94118
33068     ./16385
7636      ./13018
42533824  .
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base#
```

Depuis l'interface de pgadmin4 et `\` **SELECT pg\_size\_pretty( pg\_database\_size('BDD') )**



# Echec de restauration de Base de données

Saturer l'espace disque d'un serveur de production va fatalement interrompre/impacter un grand nombre de services. Veiller toujours à avoir de l'espace en adéquation avec la manipulation à réaliser. Par exemple, si on veut dupliquer une base par la production d'un fichier zip, le serveur va occuper l'espace de la base de données initiale plus la base de données compressée (dans le .zip uploadé) et la taille de la base de données de destination.

Voici un exemple d'erreur qui peut être rencontrée :

```
Database restore error: Postgres subprocess ('/usr/bin/pg_restore', u'--dbname=BDD_TEST', '--no-privileges', '--role=openprod', '--no-owner', '/tmp/tmpfz2jX9') error 1
```

Et le

log joint :

```
2023-03-08 17:05:51,461 1187 ERROR BDD_TEST openerp.addons.web.controllers.main: restore Database restore error: Postgres subprocess ('/usr/bin/pg_restore', u'--dbname=BDD_TEST', '--no-privileges', '--role=openprod', '--no-owner', '/tmp/tmpfz2jX9') error 1
2023-03-08 17:05:51,540 1187 INFO BDD_TEST openerp.addons.web.controllers.main: restore time:197.283s mem: 1448172k -> 1480572k (diff: 38400k)
2023-03-08 17:05:51,547 1187 INFO BDD_TEST werkzeug: 192.168.10.56 - - [08/Mar/2023 17:05:51] "POST /web/database/restore HTTP/1.1" 200 -
2023-03-08 17:06:20,605 1187 WARNING BDD_TEST openerp.addons.base.ir.ir_cron: Skipping database BDD_TEST as its base version is not 9.0.1.9.
2023-03-08 17:06:28,534 1187 WARNING BDD_TEST openerp.addons.base.ir.ir_cron: Skipping database BDD_TEST as its base version is not 9.0.1.9.
```

**Si vous obtenez ce type de message lors d'une restauration, contrôler rapidement l'espace à disposition sur le serveur et effectuer le correctif nécessaire.**

# Pour aller plus loin

Les éléments précédemment présentés vous donnent les principaux aspects à connaître et à maîtriser afin d'appréhender convenablement une mise à jour ou une restauration de base de données. Il est possible d'aller encore plus loin dans l'analyse afin d'identifier plus précisément les composants les plus volumineux d'un environnement.

## 1. Répertoire de stockage des bases de données

PostgreSQL stocke ses bases de données sur un répertoire que l'on peut explorer/quantifier. Cette information peut être retrouvée sur la clé « `data_directory` » présente dans le fichier `postgresql.conf` et peut être trouvé via un simple `grep` (ou via SQL, voir plus bas). Exécuter la commande : `grep data_directory /etc/postgresql/10/main/postgresql.conf`

```
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base# grep data_directory /etc/postgresql/10/main/postgresql.conf
data_directory = '/var/lib/postgresql/10/main'          # use data in another directory
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base#
```

En effet, dans le répertoire `/var/lib/postgresql/10/main/base` ("10" représentant ici la version de l'instance PostgreSQL et "main" son nom), l'utilisateur pourra trouver ses bases de données.

**Cette documentation étant réalisée avec une version 18 d'Ubuntu, il faut l'adapter pour les autres versions (PostgreSQL 12 pour Ubuntu 20, PostgreSQL 14 pour Ubuntu 22, etc...) ou pour les autres noms d'instance (main) !**

Considérer que dans la plupart des cas, vos fichiers de configuration PostgreSQL seront sous :  
**`/etc/postgresql/<version_instance>/<nom_instance>`**

Et vos fichiers data sous :

**`/var/lib/postgresql/<version_instance>/<nom_instance>`**

Une commande SQL vous donne la localisation des fichiers de votre instance :

**`SHOW data_directory;`**

Une commande SQL vous permet de déterminer le répertoire de stockage de votre base :

**`SELECT oid from pg_database where datname = 'BDD';`**

Cet espace est protégé et seul l'utilisateur « root » peut l'explorer...

Il faut donc s'authentifier comme l'utilisateur root comme ci-dessous :

```

openprod@ubuntu18-jmp:/var/lib/postgresql/10 $ sudo su root
root@ubuntu18-jmp:/var/lib/postgresql/10# cd main
root@ubuntu18-jmp:/var/lib/postgresql/10/main# cd base
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base# ls -l
total 68
drwx----- 2 postgres postgres 12288 mars  8 15:32 1
drwx----- 2 postgres postgres 4096 mars  2 10:20 13017
drwx----- 2 postgres postgres 12288 mars  8 15:11 13018
drwx----- 2 postgres postgres 36864 mars  8 15:12 16385
drwx----- 2 postgres postgres 4096 mars  2 12:25 pgsql_tmp
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base# sudo -u postgres psql
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.1))
Type "help" for help.

```

```

postgres=# \l

```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
openprod	openprod	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	
postgres	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	
template0	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	=c/postgres + postgres=CTc/postgres

```

(4 rows)

postgres=#

```

Détail des commandes lancées :

- \$ `sudo su root` --> permet de passer sur l'utilisateur root.
- # `cd main` --> change de répertoire et va dans le répertoire « main ».
- # `cd base` --> Change de répertoire et va dans le répertoire « base ».
- # `ls -l` --> Affiche le contenu du répertoire en mode liste.

Contrairement à SQL Server, la base de données sous PostgreSQL n'est pas composée d'un seul fichier (.dat) mais d'un ensemble de fichiers, chaque fichier constituant une table.

```

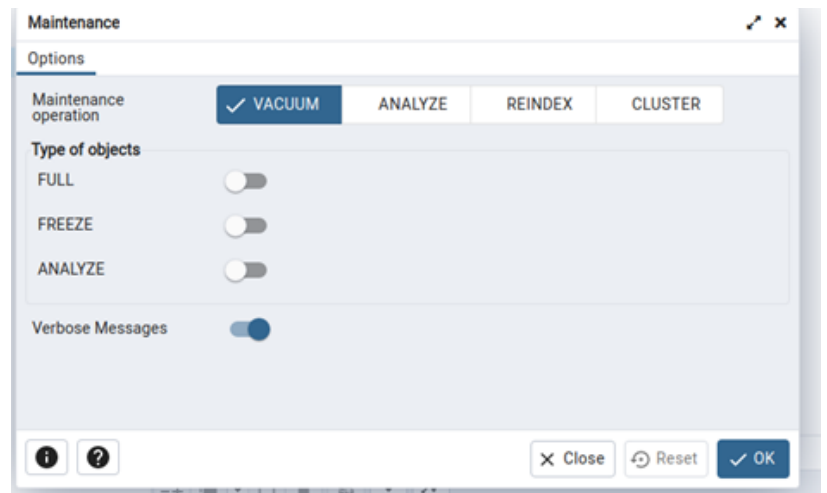
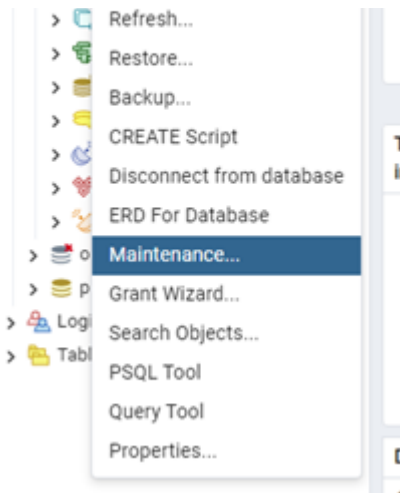
root@ubuntu18-jmp:/var/lib/postgresql/10/main/base/94118# ls -l
total 42477908
-rw----- 1 postgres postgres      0 mars  8 16:24 100002
-rw----- 1 postgres postgres    8192 mars  8 16:24 100004
-rw----- 1 postgres postgres    8192 mars  8 16:30 100005
-rw----- 1 postgres postgres      0 mars  8 16:25 100007
-rw----- 1 postgres postgres      0 mars  8 16:25 100010
-rw----- 1 postgres postgres    8192 mars  8 16:25 100012
-rw----- 1 postgres postgres    8192 mars  8 16:30 100013
-rw----- 1 postgres postgres      0 mars  8 16:25 100015
-rw----- 1 postgres postgres      0 mars  8 16:25 100018
-rw----- 1 postgres postgres    8192 mars  8 16:25 100020
-rw----- 1 postgres postgres      0 mars  8 16:25 100021
-rw----- 1 postgres postgres      0 mars  8 16:25 100024
-rw----- 1 postgres postgres    8192 mars  8 16:25 100026
-rw----- 1 postgres postgres    8192 mars  8 16:30 100027
-rw----- 1 postgres postgres    8192 mars  8 16:30 100029
-rw----- 1 postgres postgres      0 mars  8 16:25 100031
-rw----- 1 postgres postgres      0 mars  8 16:25 100034
-rw----- 1 postgres postgres      0 mars  8 16:25 100037
-rw----- 1 postgres postgres      0 mars  8 16:25 100040
-rw----- 1 postgres postgres      0 mars  8 16:25 100043
-rw----- 1 postgres postgres      0 mars  8 16:25 100046
-rw----- 1 postgres postgres    8192 mars  8 16:25 100048
-rw----- 1 postgres postgres    8192 mars  8 16:30 100049
-rw----- 1 postgres postgres      0 mars  8 16:25 100051
-rw----- 1 postgres postgres      0 mars  8 16:25 100054
-rw----- 1 postgres postgres    8192 mars  8 16:25 100056
-rw----- 1 postgres postgres    8192 mars  8 16:30 100057
-rw----- 1 postgres postgres      0 mars  8 16:25 100059
-rw----- 1 postgres postgres      0 mars  8 16:25 100062
-rw----- 1 postgres postgres      0 mars  8 16:25 100065
-rw----- 1 postgres postgres      0 mars  8 16:25 100068
-rw----- 1 postgres postgres    8192 mars  8 16:25 100070
-rw----- 1 postgres postgres    8192 mars  8 16:30 100071
-rw----- 1 postgres postgres      0 mars  8 16:25 100073
-rw----- 1 postgres postgres      0 mars  8 16:25 100076
-rw----- 1 postgres postgres    8192 mars  8 16:25 100078
-rw----- 1 postgres postgres    8192 mars  8 16:30 100079
-rw----- 1 postgres postgres      0 mars  8 16:25 100081
-rw----- 1 postgres postgres    8192 mars  8 16:30 100084
-rw----- 1 postgres postgres    8192 mars  8 16:34 100086
-rw----- 1 postgres postgres      0 mars  8 16:25 100089
-rw----- 1 postgres postgres    8192 mars  8 16:25 100091
-rw----- 1 postgres postgres    8192 mars  8 17:30 100092
-rw----- 1 postgres postgres      0 mars  8 16:25 100095
-rw----- 1 postgres postgres    8192 mars  8 16:25 100097
-rw----- 1 postgres postgres    8192 mars  8 16:30 100098
-rw----- 1 postgres postgres    8192 mars  8 16:34 100100
-rw----- 1 postgres postgres      0 mars  8 16:25 100103
-rw----- 1 postgres postgres    8192 mars  8 16:25 100105
-rw----- 1 postgres postgres    8192 mars  8 16:30 100106
-rw----- 1 postgres postgres    8192 mars  8 16:34 100108
-rw----- 1 postgres postgres      0 mars  8 16:25 100111
-rw----- 1 postgres postgres    8192 mars  8 16:25 100113

```

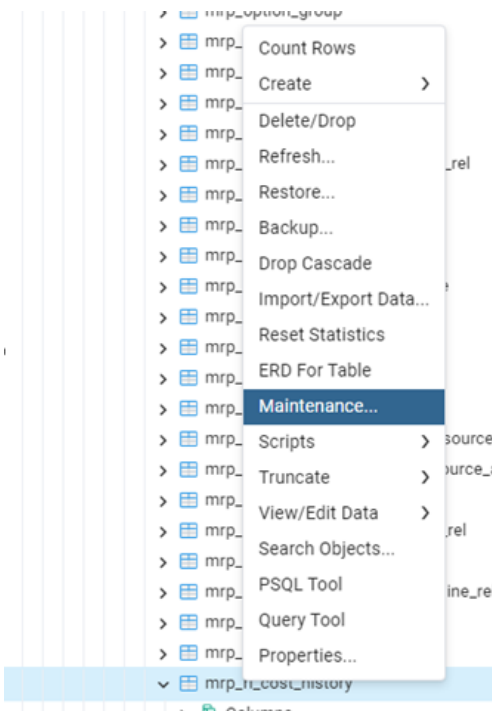
Comme nous l'avons vu plus haut, le fait de regarder la taille de ce répertoire nous indique la taille de la base de données. Bien que le moteur de base de données exécute ses propres « plans de maintenance » automatiquement, ces derniers ne seront réalisés que si certains seuils sont atteints par les tables. Il peut être souhaitable de déclencher manuellement certaines opérations de maintenance.

**Ce sont des opérations d'administration importantes et une sauvegarde est impérative avant le déclenchement de ces traitements.**

Via pgadmin, sélectionner la base (clique-droit) puis « Maintenance ».



Mais aussi sur les fichiers eux-mêmes :



## 2. Analyse de la volumétrie individuelle des fichiers

Autre point, il est possible de déterminer l'espace qu'occupe chaque fichier individuellement. Pour ce faire, il faut se connecter sur la base souhaitée et exécuter la requête suivante :

```
SELECT table_name,  
pg_relation_size(table_schema || '.' || table_name) As Taille_donnees,  
pg_total_relation_size(table_schema || '.' || table_name) As Taille_totale  
FROM information_schema.tables  
ORDER BY Taille_totale DESC;
```

Vous obtiendrez ainsi la taille des fichiers les plus volumineux de votre base de données :



BDD/openprod@192.168.10.59

Query Query History

```

1 select * from pg_database;
2 SELECT table_name,
3 pg_relation_size(table_schema || '.' || table_name) As Taille_donnees,
4 pg_total_relation_size(table_schema || '.' || table_name) As Taille_totale
5 FROM information_schema.tables
6 ORDER BY Taille_totale DESC;
```

Data Output Messages Notifications

	table_name character varying	taille_donnees bigint	taille_totale bigint
1	mrp_rl_cost_history	20924104704	29111926784
2	mrp_component_cost_history	3714105344	4795809792
3	mrp_routing_cost_history	3198500864	4386177024
4	mrp_bom_cost_history	3234144256	3677003776
5	excel_import_processing	172032	592797696
6	mrp_bom	89219072	173752320
7	stock_move	25886720	77463552
8	quotation	90112	71901184
9	product_product	42221568	58753024
10	ir_cron_log	49496064	55566336
11	ir_translation	13680640	47407104
12	mrp_workorder	15982592	27836416
13	excel_import_log	24592384	27222016
14	ir_attachment	2088960	21168128
15	account_invoice_line	8232960	16220160
16	purchase_order_line	5865472	14049280
17	ir_model_data	5898240	12894208
18	account_move_line	3203072	12656640
19	stock_valuation	5595136	12460032
20	real_time_valuation	4308992	12230656
21	pg_depend	4292608	12156928
22	pg_attribute	8224768	11313152

```
SELECT count(id) from mrp_rl_cost_history;
```



# Procédures

Procédures

# Espace disque et BDD Postgres

[Espace disque et BDD Postgres](#)

Procédures

# Mise en place de filtre Database

[Mise en place de filtre Database](#)