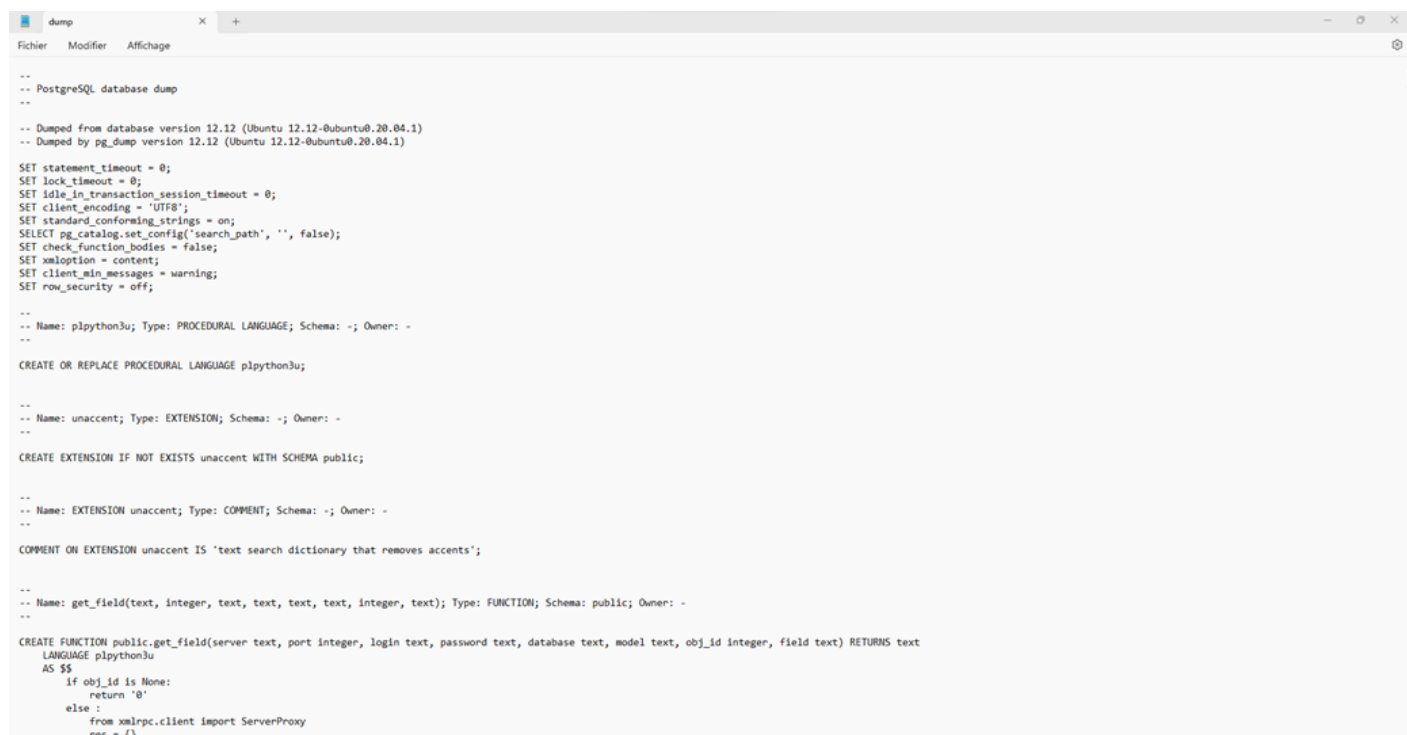


Présentation

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: plpython3u; Type: PROCEDURAL LANGUAGE; Schema: -; Owner: -
--

CREATE OR REPLACE PROCEDURAL LANGUAGE plpython3u;

--
-- 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, 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 plpython3u
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 :

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

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

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	7506 Ko	Non	62413 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.

Revision #1

Created 15 March 2023 08:22:27 by Alexis CHAPEL

Updated 15 March 2023 08:32:07 by Alexis CHAPEL