01_Actualisations des bases de données Geometries dans PostGIS (OSM, IRIS)

1/ actualisation OSM - exercice du 260402

provence-alpes-cote 100%[===================>] 363.10M 34.6MB/s   in 11s

```
# 1. Création de l'arborescence (le -p crée les parents si besoin)
mkdir -p /home/debian/data/Regions/93/OSM_93

# 2. On se déplace dans le dossier cible
cd /home/debian/data/Regions/93/OSM_93

# 3. Téléchargement du fichier de la Région PACA
# Note : C'est le fichier qui contient La Ciotat
wget https://download.geofabrik.de/europe/france/provence-alpes-cote-d-azur-latest.osm.pbf
```

Pour éviter de retomber sur l'erreur `column "highway" does not exist`, nous allons utiliser une commande `osm2pgsql` qui force l'extraction des colonnes standards (style par défaut) et active le **hstore** (pour stocker tous les autres tags dans un champ flexible).

Nécessité d'installer osm2pgsql directement dans le VPS car outil indisponible sur Docker

https://osm2pgsql.org/ (version 1.8.0)

2/ SCRIPT DE TRANSFERT OSM PACA VERS PUBLIC DANS POSTGIS

compter 6 minutes.

```
osm2pgsql -d alteris_geo -U alteris_admin -H localhost -P 5432 -W \
  --slim \
  --hstore \
  --multi-geometry \
  --number-processes 2 \
  /home/debian/data/Regions/93/OSM_93/provence-alpes-cote-d-azur-latest.osm.pbf
```

1. La Source (Le fichier .pbf)

Tu as bien fait de télécharger sur **Geofabrik**. C'est le dépôt de référence qui découpe la "Planet OSM" (le monde entier) en morceaux digestes (comme la Région PACA). Ton fichier `/home/debian/data/Regions/93/OSM_93/provence-alpes-cote-d-azur-latest.osm.pbf` est parfait.

2. L'Outil (Le traducteur)

L'image Docker `osm2pgsql/osm2pgsql` va lire ce fichier et "traduire" les données OpenStreetMap (points, lignes, polygones) en tables SQL exploitables par Alteris.

3. La Destination (Les tables dans PostgreSQL)

Par convention historique, cet outil nomme toujours les tables créées avec le préfixe `planet_osm_`. C'est là que tu retrouveras :

- `planet_osm_line` (qui deviendra ton réservoir pour les **filaires**).
- `planet_osm_point` (tes futurs équipements/écoles).
- `planet_osm_polygon` (tes bâtis et zones végétalisées).

Résumé du flux de travail Alteris

1. **Extraction** : Tu as récupéré le PBF chez Geofabrik.
2. **Injection** : Tu lances la commande Docker `osm2pgsql`.
3. **Rangement** : On renomme `planet_osm_line` en `osm_raw.roads` (pour que ton script de filaires fonctionne).

Résultat du transfert pour comparaison

```
debian@vps-02c1f983-vps-ovh-net:~/data/Regions/93/OSM_93$ sudo docker exec -it
a
                lteris_postgis psql -U alteris_admin -d alteris_geo -c "CREATE EXTENSION IF
NOT
```

```
                  EXISTS hstore;"
CREATE EXTENSION
debian@vps-02c1f983-vps-ovh-net:~/data/Regions/93/OSM_93$ osm2pgsql -d
alteris_g
                        eo -U alteris_admin -H localhost -P 5432 -W \
  --slim \
  --hstore \
  --multi-geometry \
  --number-processes 2 \
  /home/debian/data/Regions/93/OSM_93/provence-alpes-cote-d-azur-latest.osm.pbf
2026-04-02 07:57:46  osm2pgsql version 1.8.0
Password:
2026-04-02 07:57:54  Database version: 15.4 (Debian 15.4-1.pgdg110+1)
2026-04-02 07:57:54  PostGIS version: 3.3
2026-04-02 07:57:54  Setting up table 'planet_osm_point'
2026-04-02 07:57:54  Setting up table 'planet_osm_line'
2026-04-02 07:57:54  Setting up table 'planet_osm_polygon'
2026-04-02 07:57:54  Setting up table 'planet_osm_roads'



 2026-04-02 08:01:35  Reading input files done in 221s (3m 41s).
2026-04-02 08:01:35    Processed 39612462 nodes in 75s (1m 15s) - 528k/s
2026-04-02 08:01:35    Processed 5322338 ways in 108s (1m 48s) - 49k/s
2026-04-02 08:01:35    Processed 52147 relations in 38s - 1k/s
2026-04-02 08:01:35  Clustering table 'planet_osm_line' by geometry...
2026-04-02 08:01:35  Clustering table 'planet_osm_point' by geometry...
2026-04-02 08:01:42  Creating geometry index on table 'planet_osm_point'...
2026-04-02 08:01:44  Creating osm_id index on table 'planet_osm_point'...
2026-04-02 08:01:45  Analyzing table 'planet_osm_point'...
2026-04-02 08:01:45  Clustering table 'planet_osm_polygon' by geometry...
2026-04-02 08:01:48  Creating geometry index on table 'planet_osm_line'...
2026-04-02 08:01:51  Creating osm_id index on table 'planet_osm_line'...
2026-04-02 08:01:53  Analyzing table 'planet_osm_line'...
2026-04-02 08:01:53  Clustering table 'planet_osm_roads' by geometry...
2026-04-02 08:01:55  Creating geometry index on table 'planet_osm_roads'...
2026-04-02 08:01:55  Creating osm_id index on table 'planet_osm_roads'...
2026-04-02 08:01:55  Analyzing table 'planet_osm_roads'...
2026-04-02 08:01:56  Done postprocessing on table 'planet_osm_nodes' in 0s
2026-04-02 08:01:56  Building index on table 'planet_osm_ways'
2026-04-02 08:02:11  Creating geometry index on table 'planet_osm_polygon'...
```

```
2026-04-02 08:02:21  Creating osm_id index on table 'planet_osm_polygon'...
2026-04-02 08:02:23  Analyzing table 'planet_osm_polygon'...
2026-04-02 08:02:23  Building index on table 'planet_osm_rels'
2026-04-02 08:03:14  Done postprocessing on table 'planet_osm_ways' in 78s (1m 18s)
2026-04-02 08:03:14  Done postprocessing on table 'planet_osm_rels' in 2s
2026-04-02 08:03:14  All postprocessing on table 'planet_osm_point' done in 10s.
2026-04-02 08:03:14  All postprocessing on table 'planet_osm_line' done in 17s.
2026-04-02 08:03:14  All postprocessing on table 'planet_osm_polygon' done in 37s.
2026-04-02 08:03:14  All postprocessing on table 'planet_osm_roads' done in 2s.
2026-04-02 08:03:14  osm2pgsql took 319s (5m 19s) overall.
debian@vps-02c1f983-vps-ovh-net:~/data/Regions/93/OSM_93$
```

3/ BASCULEMENT OSM PACA VERS OSM_RAW DANS POSTGIS

```
sudo docker exec -it alteris_postgis psql -U alteris_admin -d alteris_geo -c "
-- 1. On s'assure que le schéma cible existe
CREATE SCHEMA IF NOT EXISTS osm_raw;

-- 2. On supprime les vieux restes pour éviter les conflits
DROP TABLE IF EXISTS osm_raw.roads CASCADE;
DROP TABLE IF EXISTS osm_raw.points CASCADE;
DROP TABLE IF EXISTS osm_raw.buildings CASCADE;
DROP TABLE IF EXISTS osm_raw.transit_major CASCADE;

-- 3. On bascule les nouvelles tables vers osm_raw
ALTER TABLE public.planet_osm_line SET SCHEMA osm_raw;
ALTER TABLE public.planet_osm_point SET SCHEMA osm_raw;
ALTER TABLE public.planet_osm_polygon SET SCHEMA osm_raw;
ALTER TABLE public.planet_osm_roads SET SCHEMA osm_raw;

-- 4. Renommage pour une nomenclature propre
ALTER TABLE osm_raw.planet_osm_line RENAME TO roads;
ALTER TABLE osm_raw.planet_osm_point RENAME TO points;
ALTER TABLE osm_raw.planet_osm_polygon RENAME TO buildings;
ALTER TABLE osm_raw.planet_osm_roads RENAME TO transit_major;
"
```

4/ FAIRE L'INVENTAIRE DE OSM_RAW

```
sudo docker exec -it alteris_postgis psql -U alteris_admin -d alteris_geo -c "
SELECT
    relname AS table_name,
    pg_size_pretty(pg_total_relation_size(c.oid)) AS taille
FROM pg_class c
JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE n.nspname = 'osm_raw'
  AND c.relkind = 'r'
ORDER BY pg_total_relation_size(c.oid) DESC;
"
```

Rang	Table	Usage Alteris
1	`buildings`	Emprises bâties et limites administratives (La Ciotat est dedans).
2	`roads`	Réseau routier complet (La source de tes filaires).
3	`points`	Équipements, services, commerces.
4	`landuse`	Occupation du sol (Zones industrielles, parcs, forêts).
5	`transit_major`	Grands axes structurants (pour cartographie grand angle).
6	`pois`	Points d'intérêt spécifiques (doublon partiel de `points`).
7	`waterways`	Rivières, canaux, fossés.
8	`water`	Plans d'eau, bassins, mer.
9	`railways`	Réseau ferré (Barrière physique majeure pour I-T6-01).
10	`places`	Noms de quartiers, lieux-dits et communes.

```
 table_name   | taille
--------------+---------
 buildings    | 1374 MB
 roads        | 625 MB
 points       | 232 MB
 landuse      | 131 MB
 transit_major | 64 MB
 pois         | 64 MB
 waterways    | 29 MB
 water        | 14 MB
 railways     | 3680 kB
 places       | 2920 kB
(10 rows)
```

Catégorie	Tables incluses	Poids estimé
Gros volumes	`buildings` + `roads`	~2 000 MB
Moyens volumes	`points` + `landuse`	~363 MB
Volumes légers	`transit_major`, `pois`, `waterways`, `water`	~171 MB
Petits volumes	`railways`, `places`	~6 MB
TOTAL	**Miroir OSM PACA**	**~2,54 GB**

```
debian@vps-02c1f983-vps-ovh-net:~$
debian@vps-02c1f983-vps-ovh-net:~$ sudo docker exec -it alteris_postgis psql -U alteris_admin
-d alteris_geo -c "
SELECT pg_size_pretty(SUM(pg_total_relation_size(c.oid))) AS poids_total_osm_raw
FROM pg_class c
JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE n.nspname = 'osm_raw'
  AND c.relkind = 'r';
"
```

```
------------------------
 poids_total_osm_raw
---------------------
 2539 MB
(1 row)

debian@vps-02c1f983-vps-ovh-net:~$
```

Transfert des contours IRIS dans insee_raw

Pour ne pas réinventer la roue et éviter les galères de mot de passe ou de drivers manquants, voici la **méthode "Golden"** que nous avons validée.

L'astuce consiste à utiliser **le binaire du VPS (ton host)** pour lire le fichier et **injecter le résultat dans le Docker** via un "pipe" (flux). C'est la méthode la plus robuste car elle ne dépend pas de la configuration interne du conteneur.

□□ La Méthode "Standard" (à copier-coller)

Voici la commande universelle pour recharger tes couches SIG :

Bash

```
# 1. On génère le SQL depuis le fichier local
# 2. On l'injecte dans le psql du Docker
ogr2ogr -f PGDump /vsistdout/ "/home/debian/data/contours-iris-pe.gpkg" \
  -nln insee_raw.contours_iris_2025 \
  -lco OVERWRITE=YES \
  -lco GEOMETRY_NAME=geom \
  -lco DROP_TABLE=YES \
  -nlt PROMOTE_TO_MULTI | docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo
```

□□ Pourquoi cette méthode gagne à tous les coups ?

1. **Indépendance des Drivers** : Tu utilises le `ogr2ogr` de ton Debian (GDAL 3.6.2), qui est beaucoup plus complet et à jour que celui souvent pré-installé dans les images Docker.
2. **Bypass du Mot de Passe** : En envoyant le flux vers `docker exec -i ... psql`, tu profites de l'authentification "peer" ou de l'absence de mot de passe interne du conteneur. Plus besoin de `PGPASSWORD`.
3. **Propreté (OVERWRITE/DROP)** : L'option `-lco DROP_TABLE=YES` et `-lco OVERWRITE=YES` garantit que tu ne crées pas de doublons et que tu nettoies les restes d'une couche corrompue.
4. **Standardisation (PROMOTE_TO_MULTI)** : Elle force toutes les géométries en `MultiPolygon`. C'est vital car QGIS ou PostGIS détestent avoir un mélange de `Polygon` et `MultiPolygon` dans une même table.

□□ Le mémo de secours (si ça bloque encore)

Si un jour tu as une erreur de connexion, vérifie ces 3 points :

- **Le chemin** : Fais toujours un `ls /chemin/vers/ton/fichier` pour être sûr qu'il n'a pas bougé.
- **Le schéma** : Assure-toi que le schéma cible (ici `insee_raw`) existe déjà dans ta base :
  - `CREATE SCHEMA IF NOT EXISTS insee_raw;`
- **Les droits** : Si tu as une erreur de permission, utilise `sudo` devant la commande `ogr2ogr`.

Tu as maintenant la "recette secrète" pour tes imports. Est-ce qu'on s'en sert pour une autre couche (DVF, Sirene...) ou on finit la synthèse de ton indicateur A1 ?

Revision #1
Created 2026-04-10 09:15:15 UTC by Julien
Updated 2026-04-10 09:15:15 UTC by Julien