# Mise à jour de GeoNature Atlas

# Requêtes liés aux zones géographiques de l'Atlas

L'Atlas nécessite de définir un territoire correspondant à l'emprise géographique des observations stockées dans le module Synthèse de GeoNature.

Dans le cadre du SINP PACA, nous avons gérées la définition du territoire et la suppression des zones géographiques inutiles via un script de gestion des zones géographiques liées à un territoire.

Dans le cadre du SINP AURA, un script basé sur le même principe est aussi disponible mais nous avons préféré utiliser le référentiel géographique construit par Flavia dans leur GeoNature. En effet, les contours des départements et de la région ont été construits à l'aide des contours des communes. Cela offre l'avantage d'avoir une parfaite superposition de ces différentes entités géographiques. De plus, l'intégration de différentes entités géographiques fournit par l'INPN a été réalisé et l'ensemble des entités géographiques hors région AURA supprimées.

Vous pouvez consulter les requêtes initialement exécutées pour créer le territoire et supprimer des zones géographiques dans le cadre du SINP PACA.

## Adapatation des vues

#### **Pour AURA**

- Ressources : Script SQL dans le dépôt Github SINP AURA Data
- Se connecter au serveur "db-srv" en tant que geonat : ssh geonat@db-<region>-sinp
- Se placer dans le dossier ~/data : cd ~/data
- Mettez à jour la VM vm\_observations avec : psql -h localhost -U geonatadmin -d gnatlas -f ./db-atlas/data/sql/01\_update\_vm\_observations.sql
- Corriger la base *gnatlas* en tant que super utilisateur : sudo -u postgres -s psql -d gnatlas -f ./db-atlas/data/sql/03\_fix\_as\_superuser.sql
- Corriger la base *gnatlas* en tant que propriétaire : psql -h localhost -U geonatadmin d gnatlas -f ./db-atlas/data/sql/04\_fix\_as\_user.sql

#### **Pour PACA**

 Augmenter le nombre ligne récupérable en une seule fois par Postgresql au niveau des tables étrangères (foreign table), par défaut 100 le passé à 100 000 :

```
ALTER SERVER geonaturedbserver OPTIONS (SET fetch size '100000');
```

Si l'option n'a jamais été ajouté, le faire avec :

```
ALTER SERVER geonaturedbserver OPTIONS (ADD fetch size '100000');
```

• Remplir les la colonne centroid de la table ref geo.l areas car elle est utilisé par la vue

synthese.syntheseff de l'Atlas. Or par défaut, cette colonne est vide :

```
UPDATE ref_geo.l_areas
SET centroid = ST Centroid(geom)
WHERE centroid IS NULL;
```

• TRAVAIL EN COURS : Modifier la vue synthese.syntheseff car la requête par défaut ne fonctionne pas avec 46 millions de lignes dans la table cor area synthese. C'est la requêtes associant les id synthese avec les différents types de zone géo qui posent problème. L'affichage dans l'Atlas se faisant sous forme de mailles, nous allons supprimé cette partie le temps de trouver une solution :

```
CREATE OR REPLACE VIEW synthese.syntheseff AS
SELECT
    s.id_synthese,
    s.id dataset,
    s.cd nom,
    s.date min AS dateobs,
    s observers AS observateurs,
    (s.altitude min + s.altitude_max) / 2 AS altitude_retenue,
    st transform(s.the geom point, 4326) AS the geom point,
    s.count min AS effectif total,
    c.insee,
    dl.cd nomenclature::INTEGER AS diffusion level
FROM synthese synthese s
    JOIN atlas.l communes AS c
        ON ( st intersects(s.the geom point, c.the geom) )
    LEFT JOIN synthese.t nomenclatures AS dl
        ON (s.id nomenclature diffusion level = dl.id nomenclature)
    LEFT JOIN synthese t nomenclatures AS st
        ON (s.id nomenclature observation status = st.id nomenclature)
WHERE ( NOT dl.cd nomenclature::text = '4'::text OR
s.id nomenclature diffusion level IS NULL )
    AND st.cd nomenclature::text = 'Pr'::TEXT;
```

• À partir de la v1.5 de l'Atlas, la VM atlas.vm communes se comporte normalement. Elle met toujours un peu de temps à se générer (15mn en local sur disque SSD Nyme). Avant la version 1.5 de l'Atlas, la VM atlas.vm communes ne contient pas toutes les communes du territoire. Il semblerait que les communes en bord de mer ne soient pas prise en compte même avec l'utilisation de la fonction Postgis st buffer () ... Le problème vient peut être des îles des communes côtières. De plus, la génération de son contenu prend énormément de temps, nous pouvons donc utiliser la subdivision du territoire à la place. Au final, nous pouvons utiliser à la place:

```
DROP MATERIALIZED VIEW IF EXISTS atlas.vm_stats ;
DROP MATERIALIZED VIEW IF EXISTS atlas.vm communes;
CREATE MATERIALIZED VIEW atlas.vm communes
TABLESPACE pg default
AS
```

```
SELECT DISTINCT
        c.insee,
        c.commune maj,
        c.the geom,
        c.commune geojson
    FROM atlas.l communes c
        JOIN atlas t_subdivided_territory t
            ON (st_intersects(t.geom, c.the_geom))
WITH DATA;
-- View indexes:
CREATE UNIQUE INDEX vm communes insee idx ON atlas.vm communes USING
btree(insee) ;
CREATE INDEX vm_communes_commune_maj_idx ON atlas.vm_communes USING
btree(commune maj) ;
CREATE INDEX index gist vm communes the geom ON atlas.vm communes USING
gist(the geom) ;
CREATE MATERIALIZED VIEW atlas.vm stats
TABLESPACE pg_default
AS SELECT 'observations'::text AS label,
    COUNT(*) AS RESULT
   FROM atlas.vm_observations
UNION
 SELECT 'municipalities'::text AS label,
   COUNT(*) AS RESULT
   FROM atlas.vm communes
UNION
SELECT 'taxons'::text AS label,
    COUNT(DISTINCT vm taxons.cd ref) AS RESULT
   FROM atlas.vm taxons
UNION
 SELECT 'pictures'::text AS label,
    COUNT(DISTINCT m.id media) AS RESULT
   FROM atlas.vm medias m
     JOIN atlas.vm taxons t ON t.cd ref = m.cd ref
 WHERE m.id type = ANY (ARRAY[1, 2])
WITH DATA:
-- View indexes:
CREATE UNIQUE INDEX vm stats label idx ON atlas.vm stats USING
btree(label) ;
-- Restore permissions
GRANT SELECT ON TABLE atlas.vm communes TO geonatatlas;
GRANT SELECT ON TABLE atlas.vm stats TO geonatatlas;
```

Ajout d'un index sur la VM atlas.vm\_observations\_mailles avec :

```
CREATE UNIQUE INDEX vm_observations_mailles_id_obs_geojson_idx ON atlas.vm_observations_mailles USING btree (id_observation,
```

```
geojson maille);
```

• Solution alternative à la vue syntheseff qui utilise une vue matérialisée :

```
CREATE MATERIALIZED VIEW atlas.cor_synthese_area
TABLESPACE pg default
AS
    SELECT DISTINCT ON (sa.id synthese, t.type code)
        sa.id synthese,
        sa.id area,
        st transform(a.centroid, 4326) AS centroid 4326,
        t.type code,
        a.area code
    FROM synthese.cor_area_synthese AS sa
         JOIN ref geo.l areas AS a
             ON (sa.id_area = a.id_area)
         JOIN ref geo.bib areas types AS t
             ON (a.id type = t.id type)
    WHERE t.type code IN ('M10', 'COM', 'DEP')
WITH DATA;
-- View indexes:
CREATE UNIQUE INDEX cas pk idx ON atlas.cor synthese area USING btree
(id synthese, id area);
CREATE INDEX cas type code idx ON atlas.cor synthese area USING btree
(type code);
CREATE OR REPLACE FUNCTION
atlas.get blurring centroid geom by code(code CHARACTER VARYING,
idSynthese INTEGER)
RETURNS geometry
LANGUAGE plpgsql
IMMUTABLE
AS $function$
    -- Function which return the centroid for a sensitivity or
diffusion level code and an id synthese
    DECLARE centroid geometry;
    BEGIN
        SELECT INTO centroid csa.centroid 4326
        FROM atlas.cor synthese area AS csa
        WHERE csa.id synthese = idSynthese
            AND csa.type code = (CASE WHEN code = '1' THEN 'COM' WHEN
code = '2' THEN 'M10' WHEN code = '3' THEN 'DEP' END)
        LIMIT 1:
        RETURN centroid;
    END;
$function$
```

```
;
CREATE OR REPLACE VIEW synthese.syntheseff AS
SELECT
    s.id synthese,
    s.id dataset,
    s.cd nom,
    s.date_min AS dateobs,
    s.observers AS observateurs,
    (s.altitude_min + s.altitude_max) / 2 AS altitude_retenue,
    CASE
        WHEN (sens.cd nomenclature::INT >= 1 AND
sens.cd nomenclature::INT <= 3 AND dl.cd nomenclature::INT >= 1 AND
dl.cd nomenclature::INT <= 3) THEN</pre>
            CASE
                WHEN (sens.cd nomenclature::INT >=
dl.cd nomenclature::INT) THEN (
atlas.get blurring centroid geom by code(sens.cd nomenclature,
s.id synthese)
                WHEN (sens.cd nomenclature::INT <</pre>
dl.cd nomenclature::INT) THEN (
atlas.get_blurring_centroid_geom_by_code(dl.cd_nomenclature,
s.id_synthese)
            END
        WHEN (sens.cd nomenclature::INT >= 1 AND
sens.cd nomenclature::INT <= 3) AND (dl.cd nomenclature::INT < 1 OR</pre>
dl.cd nomenclature::INT > 3) THEN (
atlas.get blurring centroid geom by code(sens.cd nomenclature,
s.id_synthese)
           WHEN (dl.cd nomenclature::INT >= 1 AND
dl.cd nomenclature::INT <= 3) AND (sens.cd nomenclature::INT < 1 OR</pre>
sens.cd nomenclature::INT > 3) THEN (
atlas.get blurring centroid geom by code(dl.cd nomenclature,
s.id synthese)
        ELSE st transform(s.the geom point, 4326)
    END AS the geom point,
    s.count min AS effectif total,
    c.insee,
    sens.cd nomenclature AS sensitivity,
    dl.cd_nomenclature AS diffusion_level
FROM synthese synthese s
    JOIN atlas.l communes AS c
        ON ( st_intersects(s.the_geom_point, c.the_geom) )
    LEFT JOIN synthese t nomenclatures AS sens
        ON (s.id nomenclature sensitivity = sens.id nomenclature)
    LEFT JOIN synthese.t nomenclatures AS dl
        ON (s.id nomenclature diffusion level = dl.id nomenclature)
```

```
LEFT JOIN synthese.t_nomenclatures AS st
        ON (s.id_nomenclature_observation_status = st.id_nomenclature)
WHERE ( NOT dl.cd_nomenclature = '4' OR
s.id_nomenclature_diffusion_level IS NULL )
    AND ( NOT sens.cd_nomenclature = '4' OR
s.id_nomenclature_sensitivity IS NULL )
    AND st.cd_nomenclature = 'Pr';
```

### Mise à jour de l'application

### En utilisant une release de l'Altas

#### • Ressources :

https://github.com/PnX-SI/GeoNature-atlas/blob/master/docs/installation.rst#mise-%C3%A0-jour-de-lapplication

- Sur "web-srv" :
  - o Se connecter au serveur : ssh geonat@<sinp-web>
  - Basculer l'Atlas en mode maintenance :
    - sudo nginx\_dissite atlas.conf ; sudo nginx\_ensite atlas maintenance.conf ; sudo service nginx reload
    - Arrêter le service de Supervisor avant de faire la mise à jour : sudo supervisorctl stop atlas
  - Se placer dans le dossier dwl de l'utilisateur geonat : cd ~/dwl/
  - Exporter la dernière version de GeoNature-Atals dans une variable d'env locale à la session : export GNAV=\$(curl -s https://api.github.com/repos/PnX-SI/GeoNature-atlas/releases/latest | grep tag\_name | cut -d\" -f4)
    - Vérifier la version en question : echo "\${GNAV}"
  - Télécharger l'archive : wget
    - https://github.com/PnX-SI/GeoNature-atlas/archive/\${GNAV}.zip -0
      atlas\_v\${GNAV}.zip
  - Décompresser l'archive dans le dossier www de l'utilisateur geonat : unzip atlas\_v\${GNAV}.zip -d ~/www/
  - ∘ Se rendre dans le dossier www : cd ~/www/
  - Renommer le dossier au format GeoNature (⇒ uniformité): mv GeoNature-atlas-\${GNAV} atlas\_v\${GNAV}
  - o Pour faciliter la migration créer le lien symbolique suivant :
    - cd ~/www; rm -f atlas\_old; ln -s atlas\_v<ancienne-version>
       atlas old
  - Copier les fichiers de config de l'ancien dossier vers le nouveau :
    - le fichier de paramètres pour les scripts d'installation et Gunicorn : cp
      - ~/www/atlas old/atlas/configuration/settings.ini
      - ~/www/atlas v\${GNAV}/atlas/configuration/settings.ini
    - le fichier de config du backend Python : cp
      - ~/www/atlas old/atlas/configuration/config.py
      - ~/www/atlas\_v\${GNAV}/atlas/configuration/config.py
  - Vérifier l'existence de nouveaux paramètres de config :

- Dans *settings.ini* : diff
  - ~/www/atlas old/atlas/configuration/settings.ini.sample
  - ~/www/atlas v\${GNAV}/atlas/configuration/settings.ini.sample
- Dans config.py : diff
  - ~/www/atlas\_old/atlas/configuration/config.py.sample
  - ~/www/atlas\_v\${GNAV}/atlas/configuration/config.py.sample
- Copier le contenu du dossier "static" : cp -aR ~/www/atlas\_old/static/custom/ ~/www/atlas\_v\${GNAV}/static
- Modifier le lien symbolique pour pointer vers la nouvelle version : cd ~/www; rm -f atlas; ln -s "atlas v\${GNAV}" atlas
- Consulter les éventuelles notes de version spécifiques décrites au niveau de chaque version : https://github.com/PnX-SI/GeoNature-atlas/releases
  - **ATTENTION** : Avant de lancer un script, l'éditer et corriger les chemins vers l'installation de GeoNature
  - Sur l'instance "web-srv" exécuter tous les scripts en lien avec l'interface.
- Surtout, si les notes de version indiquent qu'il faut exécuter un script de migration de base de données synchroniser "db-srv" :
  - Synchroniser l'ensemble de "www" de "web-srv" vers "db-srv" : rsync -av -e "ssh -p <port-ssh-db>" /home/geonat/www/ geonat@db-<region>-sinp:/home/geonat/www/
- Sur "db-srv" :
  - o Se connecter au serveur : ssh geonat@<sinp-db>
  - Exécuter les scripts de mise à jour de la base de données. Vous pouvez utiliser une commande du type : psql -h localhost -U geonatadmin -d gnatlas -f ~/www/atlas/data/<mon-fichier-de-mise-a-jour>.sql
    - **Note** : la requête doit être exécuter avec l'utilisateur *geonatadmin* et non *geonatatlas* car les vues matérialisées de la base *gnatlas* appartiennent à *geonatadmin*. Il aurait peut être été mieux d'installer la base de l'atlas avec *geonatatlas*...
- Sur "web-srv" :
  - Relancez l'installation automatique de l'application : cd ~/www/atlas/;
     ./install app.sh
  - Consulter les logs de l'installation dans ~/www/atlas/log/errors\_atlas.log avec :
     tail -f ~/www/atlas/log/errors\_atlas.log
  - Réactiver le service de *Supervisor* : supervisorctl start atlas
  - Désactiver le mode maintenance de l'Atlas: sudo nginx\_dissite atlas\_maintenance.conf; sudo nginx\_ensite atlas.conf; sudo service nginx reload

### À partir d'une branche

Toutes étapes suivantes se déroulent sur "web-srv" à l'exception des mises à jour de la base à exécuter via psql sur "db-srv" :

- Exporter le nom de la branche dans une variable d'env locale à la session. C'est le nom utilisé pour les fichiers il faut donc y remplacer les éventuels "/" par "-" : export GNAB="<br/>branche>"
- Indiquer la version de l'Atlas sur laquelle se base les développements
  - Pour voir la version, en local dans le dossier du code de l'Atlas contenant les développements : cat VERSION

- Récupérer aussi les 7 derniers caractères du SHA1 du dernier commit de la branche à récupérer
- Exporter la version de l'Atlas obtenu avec cat en la concaténant avec les 7 premiers caractères du commit séparé par "-" . Ex. : export GNAV="1.5.2-dev0-04dbbcb"
- Vérifier les versions en question : echo "Branche: \${GNAB} ; TaxHub : \${GNAV}"
- Se placer dans le dossier dwl de l'utilisateur geonat : cd ~/dwl/
- Télécharger l'archive : wget

```
https://github.com/PnX-SI/GeoNature-atlas/archive/<branche>.zip -0
"atlas ${GNAB} v${GNAV}.zip"
```

- Il est aussi possible d'indiquer un SHA1 de commit pour l'URL : https://github.com/PnX-SI/GeoNature-atlas/archive/<shal>.zip
- Décompresser l'archive dans le dossier dwl de l'utilisateur geonat : unzip atlas \${GNAB} v\${GNAV}.zip -d ~/dwl/
- Renommer le dossier au format GeoNature (⇒ uniformité), dans le nom de la branche les "/" sont remplacés par des "-": mv GeoNature-atlas-\${GNAB} atlas \${GNAB} v\${GNAV}
- Déplacer le dossier au format GeoNature dans www/: mv atlas \${GNAB} v\${GNAV} ~/www/
- Se rendre dans le dossier www : cd ~/www/
- Copier les fichiers de config de l'ancien dossier vers le nouveau :
  - le fichier de paramètres pour les scripts d'installation : cp
    - ~/www/atlas/atlas/configuration/settings.ini
    - ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/settings.ini
  - le fichier de config du backend Python : cp
    - ~/www/atlas/atlas/configuration/config.py
    - ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/config.py
- Vérifier l'existence de nouveaux paramètres de config :
  - Dans config.py: diff ~/www/atlas/atlas/configuration/config.py.sample ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/config.py.sample
  - Dans settings.ini :
    - diff~/www/atlas/atlas/configuration/settings.ini.sample ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/settings.ini.sampl
- Compléter/modifier les paramètres de config en fonction des besoins de la branches dans les fichiers:
  - ∘ config.py: vi
    - ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/config.py
  - ∘ settings.ini : vi
    - ~/www/atlas \${GNAB} v\${GNAV}/atlas/configuration/settings.ini
- Copier le contenu du dossier "static" : cp -aR ~/www/atlas/static/custom/ ~/www/atlas \${GNAB} v\${GNAV}/static
- Modifier les liens symboliques :
  - ∘ pour la nouvelle version : cd ~/www; rm -f atlas; ln -s atlas \${GNAB} v\${GNAV} atlas
  - o pour l'ancienne version :
    - cd ~/www; rm -f atlas old; ln -s atlas v\${GNAV} atlas old
    - ou si l'ancienne version était déjà sur une branche de test : cd ~/www; rm -f atlas old; In -s atlas \${GNAB} v<ancienne-version> atlas old
- La mise à jour de la base de données passe par la commande geonature depuis la version 2.9.0, voir cet étape dans la section consacré à GeoNature.
- Consulter les éventuelles notes de version spécifiques décrites au niveau de chaque version :

### https://github.com/PnX-SI/GeoNature-atlas/releases

• Sur l'instance "web-srv" exécuter tous les scripts en lien avec l'interface.

### Import des images de l'INPN

Si vous souhaitez utiliser dans l'Atlas les images issues de l'INPN vous pouvez suivre la procédure suivante :

- Se connecter sur le serveur : ssh geonat@sinp-<region>-web
- Lancer une session avec Screen car l'exécution du script peut prendre du temps : screen -S "images-inpn-import"
  - Voir la documentation générale concernant les commandes à utiliser avec Screen pour quitter puis se reconnecter à une session.
- Se placer dans le dossier du script dans TaxHub : cd
  - ~/www/taxhub/data/scripts/import\_inpn\_media
- Créer un environnement virtuel : python3 -m venv venv
- Activer l'environnement virtuel : source venv/bin/activate
- Redonner les droits d'exécution à GCC pour tout le monde si l'on veut pouvoir installer correctement les paquets Python dans le venv : sudo chmod o+x /usr/bin/gcc
- Installer les paquets suivant : pip install psycopg2 requests
- Retirer les droits d'exécution à GCC pour tout le monde : sudo chmod o-x /usr/bin/gcc
- Créer le fichier de configuration : cp config.py.sample config.py
  - Modifier les paramètres :
    - SQLALCHEMY\_DATABASE\_URI = "postgresql://geonatadmin:<mot-depasse>@10.0.1.20:5432/geonature2db"
    - Pour l'intégration intiale :

```
QUERY_SELECT_CDREF = """SELECT DISTINCT cd_ref FROM
taxonomie.bib_noms ORDER BY cd_ref LIMIT 100"""
```

- Supprimer le LIMIT 100 une fois un premier test effectué
- Pour les mise à jour :

```
QUERY_SELECT_CDREF = """
SELECT DISTINCT cd_ref
FROM taxonomie.bib_noms AS bn
WHERE NOT EXISTS (
    SELECT 'X'
    FROM taxonomie.t_medias tm
    WHERE tm.cd_ref = bn.cd_ref
)
ORDER BY cd_ref
"""
```

Si la table taxonomie.bib\_noms est vide. Cela peut arrivé si aucun module de saisie n'est actif (cas du SINP). Il est possible de la remplir à partir des données de gn\_synthese.synthese avec la requête suivante :

```
INSERT INTO taxonomie.bib_noms (cd_nom, cd_ref)
SELECT DISTINCT s.cd_nom, t.cd_ref
```

```
FROM gn_synthese.synthese AS s
  JOIN taxonomie.taxref AS t
    ON s.cd_nom = t.cd_nom
WHERE NOT s.cd_nom IN (SELECT DISTINCT cd_nom FROM taxonomie.bib_noms);
```

- Utiliser Psql avec la requête suivante si bib\_noms est vide (Temps d'exécution 20s pour 9,3 millions de lignes dans synthese): psql -h localhost -U geonatadmin -d geonature2db -c "INSERT INTO taxonomie.bib\_noms (cd\_nom, cd\_ref) SELECT DISTINCT s.cd\_nom, t.cd\_ref FROM gn\_synthese.synthese AS s JOIN taxonomie.taxref AS t ON s.cd nom = t.cd nom;"
- Lancer le script : python import\_inpn\_media.py
- Vérifier la présence des médias dans la table taxonomie.t medias.
- Si tout c'est bien passé, désactiver l'environnement virtuel : deactivate
- Le script ajoute les photos en tant que "secondaire" pour désigner celle avec le plus petit id media (=~ au hasard) comme "principale", utiliser la requête :
  - o lors de l'intégration initiale :

```
WITH first_media AS (
    SELECT MIN(id_media) AS first_id_media_founded, cd_ref
    FROM taxonomie.t_medias
    GROUP BY cd_ref
)
UPDATE taxonomie.t_medias AS tm
    SET id_type = 1
    FROM first_media AS fm
    WHERE tm.id_media = fm.first_id_media_founded
    AND tm.cd_ref = fm.cd_ref;
```

lors des mises à jours :

```
WITH first_media AS (
    SELECT MIN(id_media) AS first_id_media_founded, cd_ref
    FROM taxonomie.t_medias
    WHERE cd_ref NOT IN (
    SELECT cd_ref
    FROM taxonomie.t_medias stm
    WHERE id_type = 1
    )
    GROUP BY cd_ref
)
UPDATE taxonomie.t_medias AS tm
    SET id_type = 1
    FROM first_media AS fm
    WHERE tm.id_media = fm.first_id_media_founded
    AND tm.cd_ref = fm.cd_ref;
```

 Soit:psql -h localhost -U geonatadmin -d geonature2db -c "WITH first\_media AS (SELECT MIN(id\_media) AS first\_id\_media\_founded,

```
cd_ref FROM taxonomie.t_medias GROUP BY cd_ref) UPDATE
taxonomie.t_medias AS tm SET id_type = 1 FROM first_media AS fm
WHERE tm.id_media = fm.first_id_media_founded AND tm.cd_ref =
fm.cd ref;"
```

• Pour afficher les images sur l'Atlas, il est nécessaire de rafraichir les données des vues matérialisées atlas.vm\_medias et atlas.vm\_taxons\_plus\_observes :

```
REFRESH MATERIALIZED VIEW atlas.vm_medias WITH DATA;
REFRESH MATERIALIZED VIEW atlas.vm_taxons_plus_observes WITH DATA;
REFRESH MATERIALIZED VIEW atlas.vm_stats WITH DATA;
TRUNCATE atlas.t_cache;
```

### Mise à jour des données

• Ressources :

https://github.com/PnX-SI/GeoNature-atlas/blob/master/docs/vues materialisees maj.rst

- Se connecter au serveur : ssh geonat@<sinp-db>
- Lancer une nouvelle session avec Screen : screen -S "update-atlas"
  - Voir la documentation générale concernant les commandes à utiliser avec Screen pour quitter puis se reconnecter à une session.
- NOTES: si le refresh est réalisé CONCURRENTLY, il est possible de lancer le rafraîchissement des VMs sur l'installation de production. L'Atlas restera fonctionnel même pendant le rafraîchissement.
- Les actions disponibles :
  - Pour mettre à jour uniquement les **observations** (table *synthese*) utiliser la requête :
     psql -h localhost -U geonatadmin -d gnatlas -c "SELECT atlas.refresh\_materialized\_view\_data();"
  - Pour mettre à jour uniquement les zones géographiques (à faire uniquement si le territoire a été modifié): psql -h localhost -U geonatadmin -d gnatlas -c "SELECT atlas.refresh\_materialized\_view\_ref\_geo();"
  - Pour tout mettre à jour sans distinction (la mise à jour des données géographiques peut être longue): psql -h localhost -U geonatadmin -d gnatlas -c "SELECT RefreshAllMaterializedViews('atlas');"
- Il est possible d'automatiser cette tâche via un *cron* ⇒ Dans notre cas, ce n'est pas utile...

From

https://sinp-wiki.cbn-alpin.fr/ - CBNA SINP

Permanent link:

https://sinp-wiki.cbn-alpin.fr/serveurs/installation/web-srv/geonature-atlas-mise-a-jour?rev=1648240168

Last update: 2022/03/25 20:29

