Import des données via GN2PG

Notes

- C'est le serveur *db-srv* qui se charge d'héberger le script Gn2Pg dans le dossier `/home/geonat/data/gn2pg`
  - Le serveur *bkp-srv* peut également héberger Gn2Pg pour l'espace de préprod.
- Un script Bash `'/home/geonat/data/gn2pg/bin/gn2pg_update.sh` encapsule l'appel de Gn2Pg
- Le script Bash a une dépendance à *Pipenv* qui se charge de télécharger et installer le paquet Python *gn2pg-client* pour nous
- Il est nécessaire d'installer *Pipenv* sur le système et de préparer le dossier `/home/geonat/data/gn2pg/gn2pg`, pour cela voir le fichier /home/geonat/data/gn2pg/README.md
- Un cron (`/etc/crond.d/gn2pg`) se charge de lancer Gn2Pg aux heures déterminées pour chaque config définie.
- Il est possible de vérifier si Gn2Pg est en cours de fonctionnement en filtrant les processus sur le mot-clé "gn2pg_update" : `ps -aux|grep gn2pg_update`
- Les fichiers de log de Gn2Pg sont consultables ici : `/home/geonat/data/gn2pg/log/`

Relancer un import GN2PG interrompu

- Connectez vous à la base de donnée
- Afficher la table : `gn2pg_lpo.data_json`
- Trier les données par ordre décroissant du champ `update_ts`. La date la plus récente devrait s'afficher en premier
- Le champ "id_date" contient l'id_synthese à copier.
- Modifier ensuite le fichier `~/data/gn2pg/config/lpo_config.toml` sur le serveur hébergeant *Gn2Pg*.
  - Ajouter/Modifier le paramètre `filter_n_up_id_synthese` et lui associer l'id synthese précédemment copié comme valeur.
  - **ATTENTION** : penser une fois le téléchargement via Gn2PG terminé à commenter cette valeur.

Principes pour les données transmises

- Nous gardons l'ID synthèse du GeoNature de chaque fournisseur de données (pôle en AURA) en guise de `entity_source_pk_value`.
- Nous gardons la contrainte d'unicité entre `id_source` et `entité_source_pk_value`.
  - Avec la fourniture du « vrai » identifiant produit par le producteur au niveau du champs additionnel sous la clé « `source_id_data` ».
- Pas d'utilisation de l'UUID à ce stade car effets de bords qui pourront s'avérer problématiques selon les usages.
- Les champs `identifier` et `email` des données sur les utilisateurs devront avoir la valeur `NULL`. C'est important pour éviter tout conflit avec les utilisateurs créant des comptes

directement depuis les interfaces. Pour éviter aussi des bugs au niveau de l'inscription et du renouvellement des mots de passe, il y a un index unique sur ces 2 champs. Enfin, même si cela crée des doublons, nous distinguerons les entrées dans la table `t_roles` pour les utilisateurs s'inscrivant au SINP vis à vis des entrées générées par GN2PG ou les scripts d'intégrations des données au format CSV. La fonction se chargeant d'insérer les utilisateurs dans la table `t_roles` de GeoNature a été modifiée pour insérer `NULL` dans le champ `email` et l'email dans le champ `additional_data` sous l'attribut `gn2pg_data.email`.

- Le champ `meta_validation_date` de la synthèse du fournisseur utilisera l'alias `validation_date` soit : `s.meta_validation_date AS validation_date`
- Le champ `additional_data` de la synthèse du fournisseur utilisera l'alias `donnees_additionnelles` soit : `s.additional_data::text AS donnees_additionnelles`
- Les données à intégrer au champs additionnel en tant qu'attributs de l'objet principal sont :
  - « `source_nom` »
  - « `source_id_data` »

Mise à jour de l'installation de Gn2Pg

- Sur le dépôt Github *sinp-<region>-data* :
  - modifier la version du client *gn2pg-client* dans le fichier `gn2pg/Pipfile`.
    - Installer la nouvelle version du client avec : `pipenv install`
    - Si besoin, mettre à jour le fichier Pipfile.lock avec la commande : `pipenv lock`
    - Créer un nouveau commit avec ces modifications
  - corriger les fichier `gn2pg/data/sql/…to_synthese.sql` en :
    - comparant les modifications effectuées sur le fichier to_gnsynthese.sql du dépôt GN2PG depuis la précédente version utilisée.
  - Mettre à jour le fichier `README.md` si nécessaire
  - Corriger les fichiers `config/…_config.toml` si les paramètres de config ont évolués (voir check_conf.py, Changelog et realeases)
    - En local, les tester avec : `pipenv run gn2pg_cli --custom-script ./data/sql/<...>_to_synthese.sql <...>_config.toml`
  - Tester le script en local depuis le dossier `gn2pg/bin/` avec : `./gn2pg_update.sh -v -c <…>_config.toml`
- Sur le serveur *db-srv* ou *bkp-srv* (pré-production) :
  - Se placer dans le dossier ~/data/ avec : `cd ~/data/`
  - Récupérer les changements : `git pull ; git submodule update`
  - Mettre à jour Pipenv :
    - afficher la version actuellement installée de Pipenv via Pipx : `pipx list`
    - mise à jour : `pipx upgrade pipenv`
    - vérifier la version de Pipenv : `pipenv –version`
  - Mettre à jour les dépendances des paquets Python : `pipenv sync`
  - Se placer dans le dossier de gn2pg : `cd ~/data/gn2pg/`
  - Vérifier l'installation de Gn2Pg :
    - s'assurer que `~/.gn2pg/` est bien un lien vers `~/data/gn2pg/config/` avec : `ls -al ~/.gn2pg`
      - si nécessaire, supprimer le dossier et recréer le lien : `rm -fr ~/.gn2pg/; ln -s ~/data/gn2pg/config ~/.gn2pg`
    - vérifier la version de Gn2pg : `pipenv run gn2pg_cli --version`
  - Vérifier/Ajuster les paramètres de config : `vi <…>_config.toml`
  - Si nécessaire, réinstaller les fichiers `…_to_synthese.sql` avec : `pipenv run`

```
gn2pg_cli --custom-script ./data/sql/<...>_to_synthese.sql
<...>_config.toml
```

Installer le Dashboard Gn2Pg

- Ressources : https://github.com/lpoaura/GN2PG/blob/main/docs/dashboard.rst