Exemple d'import de données via un fichier SQL

Processus d'importation

Pour importer les données (initialement ou en mise à jour), nous utiliserons des fichiers CSV associé à la commande *COPY*. Ces fichiers CSV devront :

- être encodée en **UTF-8**
- utiliser une **tabulation** comme caractère de séparation des champs
- posséder une **première ligne d'entête** indiquant les noms des champs
- utiliser les caractères **\N** pour indiquer une valeur nulle (*NULL*) pour un champ
- si nécessaire utiliser le caractère **guillemet** (") pour préfixer et suffixer une valeur de champ
- si nécessaire utiliser **deux guillemets** successifs ("") pour échapper le caractère guillemet dans une valeur de champ préfixé et suffixé par des guillemets.

TODO :

- voir si on utilise plutôt un chaine vide sans guillemet pour indiquer une valeur NULL
- A priori, l'utilisation d'une tabulation (qui ne se rencontre pratiquement jamais dans les valeurs des champs) doit permettre d'éviter l'utilisation des guillements pour encapsuler une valeur de champ même si celui-ci en contient.

Import initial

- L'import initial concerne plusieurs millions de données. Elles seront transmises sous forme de fichiers CSV.
    - Nous utiliserons plusieurs fichiers CSV respectant tous le même format (voir ci-dessus)
    - Le détail des différents fichiers (organism, acquistion_framework, dataset, source, synthese) est présentés ci-dessous
    - Un seul fichier CSV est obligatoire *synthese* mais les autres sont nécessaire pour éviter une saisie manuelle sous GeoNature des métadonnées.
    - Dans ces fichiers CSV, nous utiliserons les codes alphanumériques des valeurs pour les champs devant contenir des **identifiant de nomenclatures**. Un script Python se connectant à la base de GeoNature permettra de récupérer l'identifiant spécifique à une instance de base de données.
    - Pour les lien avec les **organismes**, **cadres d'acquisition**, **jeux de données** et **sources**, nous utiliserons le même principe que pour les nomenclatures. Les liens se feront sur les codes ou noms et le script Python permettra de récupérer les identifiants spécifiques à chaque instance de base de données.
- L'import des fichiers CSV se fera à l'aide d'un script Bash exécutant des scripts SQL et Python.
    - Un script Python sera chargé de vérifier les données CSV et de remplacer les différents codes standard par leur identifiant numérique spécifique à la base de données courante.
    - Des scripts SQL se chargeront de désactiver triggers, contraintes… puis de les réactiver et exécuter globalement après la commande de *COPY* d'insertion du fichier CSV lancée.
- De cette façon, nous pouvons espérer intégrer jusqu'à 3 millions d'observations en moins d'une

heure dans la table *synthese* de GeoNature.

Mise à jour

- Pour chaque mise à jour, nous devrons importer seulement un différentiel :
  - le différentiel sera réalisé en local sur une machine disposant d'assez de mémoire et d'un disque dur de type SSD NVMe de façon à réduire au maximum les temps de traitement. Idéalement, le différentiel doit pouvoir être extrait des bases de données d'origines.
  - le nombre moins important de données nous permettra de réaliser rapidement la mise à jour de la table *synthese* de GeoNature sans gêner son utilisation via les interface web.
  - le différentiel sera fournie au format CSV (voir ci-dessus)
  - le fichier CSV sera importé dans une première table d'import à partir de laquelle nous réaliserons les requêtes d'import dans la table *synthese*
- Nous procéderons pour l'import du différentiel dans la table "gn_synthese.synthese" à partit de la table d'import en 3 étapes :
  - Ajout des nouvelles observations
  - Modification des observations existantes qui ont été modifiée depuis le dernier import
  - Suppression des observations qui ne sont plus présente dans l'import courant. Dans le cas du différentiel, les lignes supprimées doivent être présentent dans l'import.
- Pour réaliser le différentiel, nous pouvons utiliser :
  - La clé primaire des données d'origine (champ "*entity_source_pk_value*") pour vérifier si la données existe déjà ou pas dans la table "*gn_synthese.synthese*". Idéalement, il faudrait aussi y ajouter le champ "code_source" et/ou "code_dataset".
  - Le champ "*last_action*" permettra d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").
  - Dans le cas des observations modifiées, le champ "*meta_update_date*" permettra de déterminer la version de la modification. Une date plus récente dans la table d'import indiquera la nécessité de mettre à jour l'enregistrement.
- Pour les champs contenant des **identifiant de nomenclatures**, une fonction Postgresql présente dans la base de GeoNature permettra de récupérer l'identifiant numérique spécifique à chaque instance de base de données et correspondant au code fournie dans le CSV. Les codes de nomenclature à utiliser sont disponibles dans le champ "*cd_nomenclature*" de la table "*ref_nomenclatures.t_nomenclatures*".
- Pour les lien avec les **organismes**, **cadres d'acquisition**, **jeux de données** et **sources**, nous utiliserons le même principe que pour les nomenclatures. Les liens se feront sur les codes ou noms et des fonctions permettront de récupérer les identifiants spécifiques à chaque instance de base de données.
- L'ordre d'import des tables devrait être le suivant : organism, acquistion_framework, dataset, source, synthese.

Format SYNTHESE d'import

Permet de fournir les informations sur les observations. Correspond à la table "*gn_synthese.synthese*".

- unique_id_sinp [UUID] : UUID SINP s'il existe déjà dans les données sources.
- unique_id_sinp_grp [UUID] : UUID SINP du relevé s'il existe déjà dans les données sources.
- code_source [VARCHAR(255)] (=*id_source*) : code alphanumérique permettant d'identifier la

source des données (voir SOURCE).
- entity_source_pk_value [VARCHAR(25)] : code alphanumérique correspondant à l'équivalant d'une clé primaire pour les données sources.
- code_dataset [VARCHAR(25)] (=*id_dataset*) : code alphanumérique permettant d'identifier le jeu de donnée de l'observation (voir DATASET)
- code_nomenclature_geo_object_nature [VARCHAR(25)] (=*id_nomenclature_geo_object_nature*) : code alphanumérique de la valeur du type de nomenclature *NAT_OBJ_GEO*.
- code_nomenclature_grp_typ [VARCHAR(25)] (=*id_nomenclature_grp_typ*) : code alphanumérique de la valeur du type de nomenclature *TYP_GRP*.
- code_nomenclature_obs_meth [VARCHAR(25)] (=*id_nomenclature_obs_meth*) : code alphanumérique de la valeur du type de nomenclature *METH_OBS*.
- code_nomenclature_obs_technique [VARCHAR(25)] (=*id_nomenclature_obs_technique*) : code alphanumérique de la valeur du type de nomenclature *TECHNIQUE_OBS*.
- code_nomenclature_bio_status [VARCHAR(25)] (=*id_nomenclature_bio_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_BIO*.
- code_nomenclature_bio_condition [VARCHAR(25)] (=*id_nomenclature_bio_condition*) : code alphanumérique de la valeur du type de nomenclature *ETA_BIO*.
- code_nomenclature_naturalness [VARCHAR(25)] (=*id_nomenclature_naturalness*) : code alphanumérique de la valeur du type de nomenclature *NATURALITE*.
- code_nomenclature_exist_proof [VARCHAR(25)] (=*id_nomenclature_exist_proof*) : code alphanumérique de la valeur du type de nomenclature *PREUVE_EXIST*.
- code_nomenclature_valid_status [VARCHAR(25)] (=*id_nomenclature_valid_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_VALID*.
- code_nomenclature_diffusion_level [VARCHAR(25)] (=*id_nomenclature_diffusion_level*) : code alphanumérique de la valeur du type de nomenclature *NIV_PRECIS*.
- code_nomenclature_life_stage [VARCHAR(25)] (=*id_nomenclature_life_stage*) : code alphanumérique de la valeur du type de nomenclature *STADE_VIE*.
- code_nomenclature_sex [VARCHAR(25)] (=*id_nomenclature_sex*) : code alphanumérique de la valeur du type de nomenclature *SEXE*.
- code_nomenclature_obj_count [VARCHAR(25)] (=*id_nomenclature_obj_count*) : code alphanumérique de la valeur du type de nomenclature *OBJ_DENBR*.
- code_nomenclature_type_count [VARCHAR(25)] (=*id_nomenclature_type_count*) : code alphanumérique de la valeur du type de nomenclature *TYP_DENBR*.
- code_nomenclature_sensitivity [VARCHAR(25)] (=*id_nomenclature_sensitivity*) : code alphanumérique de la valeur du type de nomenclature *SENSIBILITE*.
- code_nomenclature_observation_status [VARCHAR(25)] (=*id_nomenclature_observation_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_OBS*.
- code_nomenclature_blurring [VARCHAR(25)] (=*id_nomenclature_blurring*) : code alphanumérique de la valeur du type de nomenclature *DEE_FLOU*.
- code_nomenclature_source_status [VARCHAR(25)] (=*id_nomenclature_source_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_SOURCE*.
- code_nomenclature_info_geo_type [VARCHAR(25)] (=*id_nomenclature_info_geo_type*) : code alphanumérique de la valeur du type de nomenclature *TYP_INF_GEO*.
- reference_biblio [VARCHAR(255)] :
- count_min [INT(4)] :
- count_max [INT(4)] :
- cd_nom [INT(4)] :
- **nom_cite** [VARCHAR(1000)] :
- meta_v_taxref [VARCHAR(50)] :
- sample_number_proof [TEXT] :
- digital_proof [TEXT] :

- non_digital_proof [TEXT] :
- altitude_min [INT(4)] :
- altitude_max [INT(4)] :
- geom_4326 [geometry(Geometry,4326)] :
- geom_point [geometry(Point,4326)] :
- geom_local [geometry(Geometry,2154)] :
- **date_min** [DATE(YYYY-MM-DD HH:MM:SS)] :
- **date_max** [DATE(YYYY-MM-DD HH:MM:SS)] :
- validator [VARCHAR(1000)] :
- validation_comment [TEXT] :
- observers [VARCHAR(1000)] :
- determiner [VARCHAR(1000)] :
- id_digitiser [INT(4)]: TODO voir si on garde ou pas ce champ.
- code_nomenclature_determination_method [VARCHAR(25)] (=*id_nomenclature_determination_method*) : code alphanumérique de la valeur du type de nomenclature *METH_DETERMIN*.
- comment_context [TEXT] :
- comment_description [TEXT] :
- meta_validation_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de validation de l'observation.
- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement de l'observation.
- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement de l'observation.
- **meta_last_action** [CHAR(1)] (=*last_action*) : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

Format SOURCE d'import

Permet de décrire la source des données.
Correspond à la table "*gn_synthese.t_sources*".

- **name** [VARCHAR(255)] : correspond au champ "*name_source*". Doit correspondre à la valeur du champ "*code_source*" de la table synthese d'import.
- desc [TEXT] : description de la source des données. Correspond au champ "*desc_source*"
- entity_source_pk_field [VARCHAR(255)] : nom du champ dans les données sources servant de clé primaire et dont la valeur est présente dans le champ "*entity_source_pk_value*" de la table SYNTHESE d'import.
- url [VARCHAR(255)] : adresse web décrivant la source des données ou permettant d'accéder aux données sources. Correspond au champ "*url_source*".
- **meta_create_date** [DATE YYYY-MM-DD HH:MM:SS] : date et heure de création de l'enregistrement de la source.
- meta_update_date [DATE YYYY-MM-DD HH:MM:SS] : date et heure de mise à jour de l'enregistrement de la source.
- **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

Format DATASET d'import

Permet de fournir les informations sur les jeux de données.
Correspond à la table "*gn_meta.t_datasets*".
NOTES : pour éviter trop de complexité, nous ne tenons pas compte des "protocoles" liées au jeu de données, ni des acteurs de type "personne". Si cela s'avérait nécessaire, il est possible de les ajouter manuellement une fois l'import effectué.

- unique_id [UUID] (=*unique_dataset_id*) : UUID du jeu de données si pré-existant.
- **code_acquisition_framework** [VARCHAR(255)] (=*id_acquisition_framework*) : code/nom du cadre d'acquisition auquel le jeu de données appartient. Permet d'effectuer le lien avec le champ "*name*" de la table d'import ACQUISITION_FRAMEWORK.
- **name** [VARCHAR(255)] (=*dataset_name*)
- **shortname** [VACHAR(255)] (=*dataset_shortname*) : permet de faire le lien avec la table SYNTHESE et le champ "*code_dataset*".
- **desc** [TEXT] (=*dataset_desc*)
- code_nomenclature_data_type [VARCHAR(25)] (=*id_nomenclature_data_type*)
- keywords [TEXT]
- marine_domain [BOOL]
- terrestrial_domain [BOOL]
- code_nomenclature_dataset_objectif [VARCHAR(25)] (=*id_nomenclature_dataset_objectif*)
- bbox_west [FLOAT]
- bbox_est [FLOAT]
- bbox_south [FLOAT]
- bbox_north [FLOAT]
- code_nomenclature_collecting_method [VARCHAR(25)] (=*id_nomenclature_collecting_method //) * code_nomenclature_data_origin [VARCHAR(25)] (*=id_nomenclature_data_origin *) * code_nomenclature_source_status [VARCHAR(25)] (*=id_nomenclature_source_status *) * code_nomenclature_resource_type [VARCHAR(25)] (*=id_nomenclature_resource_type *) * cor_territory [TEXT(ARRAY-ARRAY)] : champ de type tableau de tableau de textes. Les tableaux de textes du plus bas niveau contiendront comme première valeur le code de nomenclature correspondant au champ* "id_nomenclature_territory" *de la table* "gn_meta.cor_dataset_territory" *et comme seconde valeur une description du territoire (champ* "territory_desc*) ou une chaine vide. Exemple : <code>* '*reu_myt*' *</code> * cor_actors_organism [TEXT(ARRAY-ARRAY)] : champ de type tableau de tableau de textes. Les tableaux de textes du plus bas niveau contiendront comme première valeur le nom de l'organisme et comme seconde valeur le code de nomenclature correspondant au champ* "id_nomenclature_actor_role" *de la table* "gn_meta.cor_dataset_actor"*. Exemple : <code>* '*cbn-alpin_1_pne_5*' *</code> * **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement du jeu de données. * meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement du jeu de données. * **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import (*"I"*), modifiées (*"U"*) ou supprimées (*"D//").

Format de la table ACQUISITION_FRAMEWORK d'import

Permet de fournir les informations sur les cadres d'acquisition des jeux de données.
Correspond à la table "*gn_meta.t_acquisition_framework*".

NOTES : pour éviter trop de complexité, nous ne tenons pas compte des "publications" liées au cadre d'acquisition, ni des acteurs de type "personne". Si cela s'avérait nécessaire, il est possible de les ajouter manuellement une fois l'import effectué.

- unique_id [UUID] (=*unique_acquisition_framework_id*) UUID du cadre d'acquisition si pré-existant.
- **name** [VARCHAR(255)] (=*acquisition_framework_name*)
- **desc** [TEXT] (=*acquisition_framework_desc*)
- code_nomenclature_territorial_level [VARCHAR(25)] (=*id_nomenclature_territorial_level*)
- territory_desc [TEXT]
- keywords [TEXT]
- code_nomenclature_financing_type [VARCHAR(25)] (=*id_nomenclature_financing_type*)
- target_description [TEXT]
- ecologic_or_geologic_target [TEXT]
- parent_code [VARCHAR(255)](=*acquisition_framework_parent_id*)
- is_parent [BOOL]
- **start_date** [DATE(YYYY-MM-DD)] (=*acquisition_framework_start_date*)
- end_date [DATE(YYYY-MM-DD)] (=*acquisition_framework_end_date*)
- cor_objectifs [TEXT(ARRAY)] : champ de type tableau de textes. Le tableau contiendra une succession de codes de nomenclature correspondant au champ "*id_nomenclature_objectif*" (="Objectif du cadre d'acquisition") de la table "*gn_meta.cor_acquisition_framework_objectif*". Exemple :

```
'{'4','5','6'}'
```

- cor_voletsinp [TEXT(ARRAY)] : champ de type tableau de textes. Le tableau contiendra une succession de codes de nomenclature correspondant au champ "*id_nomenclature_voletsinp*" (="Volet SINP") de la table "*gn_meta.cor_acquisition_framework_voletsinp*". Exemple :

```
'{'1'}'
```

- cor_actors_organism [TEXT(ARRAY-ARRAY)] : champ de type tableau de tableau de textes. Les tableaux de textes du plus bas niveau contiendront comme première valeur le nom de l'organisme et comme seconde valeur le code de nomenclature correspondant au champ "*id_nomenclature_actor_role*" de la table "*gn_meta.cor_acquisition_framework_actor*". Exemple :

```
'{{'CBN-Alpin', '1'},{'PNE','5'}}'
```

- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement du jeu de données.
- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement du jeu de données.
- **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*")

Format ORGANISM d'import

Permet de fournir les informations sur les organismes liées aux jeux de données et cadres d'acquisition.

Correspond à la table table "*utilisateurs.bib_organismes*".

- unique_id [UUID] : UUID de l'organisme si pré-existant dans les données sources. Correspond au champ "uuid_organisme".
- **name** [VARCHAR(100)] : correspond au champ "nom_organisme". Sert de lien avec les tables DATASET et ACQUISITION_FRAMEWORK.
- address [VARCHAR(128)] : correspond au champ "adresse_organisme".
- postal_code [VARCHAR(5)] : correspond au champ "cp_organisme".
- city [VARCHAR(100)] : correspond au champ "ville_organisme".
- phone [VARCHAR(14)] : correspond au champ "tel_organisme".
- fax [VARCHAR(14)] : correspond au champ "fax_organisme".
- email [VARCHAR(100)] : correspond au champ "email_organisme".
- url [VARCHAR(255)] : correspond au champ "url_organisme".
- logo_url [VARCHAR(255)] : correspond au champ "url_logo".
- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement de l'organisme.
- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement de l'organisme.
- **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

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

Permanent link:
https://sinp-wiki.cbn-alpin.fr/database/exemple-import-synthese?rev=1596705527

Last update: **2020/08/06 09:18**