Formats d'import/export - Format v1.0



Accéder à la page de discussions sur le format d'import/export en cliquant sur l'icône de la barre d'outil à droite.

Objectif : proposer un format d'échange de données **simple** (autant que possible), bien **documenté**, **pratique**, **facilitant l'intégration des données sous GeoNature** et respectant le **standard OccTax** du SINP.

Processus d'importation



Ce document est en cours de travail. Les informations qu'il contient peuvent donc être amenées à changer à tout moment !

Transmission des fichiers

Les fichiers seront transmis dans un fichier d'archive au format ZIP. Le nom du fichier devra être en minuscule et contenir plusieurs parties séparées par des underscores ("_"). Les parties du fichier seront les suivantes :

1. date au format ISO 8601 : 2020-08-26
2. sujet : sinp
3. abréviation de la région concernée : paca / aura
4. abréviation de l'organisme fournisseur : cbna / cbnmed / cbnmc / cenpaca ...
5. extension : .zip

Exemple : *2020-08-26_sinp_paca_cbna.zip*

L'archive devra contenir au minimum les fichiers suivants :

- ***meta_archive.ini*** : fichier contenant les métadonnées sur le fournisseur de l'archive et la version du format d'échange utilisée.
- ***synthese.csv*** : les données d'observation à intégrer au module Synthèse. Peux être remplacer par un fichier *occtax.csv*.

Les fichiers suivants pourront aussi être fournis sans être obligatoires :

- *source.csv* : permet de décrire la source des données. Correspond à la table "*gn_synthese.t_sources*".
- *dataset.csv* : permet de fournir les informations sur les jeux de données. Correspond à la table "*gn_meta.t_datasets*".
- *acquisition_framework.csv* : permet de fournir les informations sur les cadres d'acquisition des

jeux de données. Correspond à la table "*gn_meta.t_acquisition_framework*".
- *organism.csv* : permet de fournir les informations sur les organismes liées aux utilisateurs, jeux de données et cadres d'acquisition. Correspond à la table "*utilisateurs.bib_organismes*".
- *user.csv* : permet de fournir des informations sur les utilisateurs à l'origine des données. Correspond à la table "*utilisateurs.t_roles*".
- *occtax.csv* : les données d'observation à intégrer au module OccTax. Les données seront automatiquement répercutées dans le module Synthèse via des *triggers* présents dans la base de données.
- *validation.csv* : permet de fournir les statuts de validations actuel et passés des observations. Généralement, utilisé en complément du fichier *occtax.csv*.
- *meta_additional_data.csv* : contient la description des champs additionnels présents dans les différents fichiers à importer. uniquement pour la documentation. Non utilisé.

Principe de la procédure d'import initial

- L'import initial peut concerner plusieurs millions de données. Elles seront transmises sous forme de fichiers textes.
    - Nous utiliserons plusieurs fichiers textes respectant tous le même format CSV (voir ci-dessous)
    - Le détail des différents fichiers (user, organism, acquistion_framework, dataset, source, synthese) est présentés ci-dessous
    - Un seul fichier est obligatoire *synthese* (ou *occtax*) mais les autres sont nécessaire pour éviter une saisie manuelle sur GeoNature des métadonnées et sur UsersHub des utilisateurs et organismes.
    - Dans ces fichiers, 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 numérique spécifique à une instance de base de données.
    - Pour les lien avec les **organismes**, **cadres d'acquisition** et **jeux de données**, nous utiliserons un mécanisme différent des nomenclatures. Les liens se feront sur les UUID et un script SQL permettra de récupérer les identifiants spécifiques à chaque instance de base de données. Les champs sur lesquels les liens sont basés peuvent être amené à évoluer dans la base d'origine. Lors d'une mise à jour nous nous appuierons aussi sur les champs "*unique_id*" (UUID) de ces entités (s'ils sont fournis) pour retrouver la correspondance avec un enregistrement en base de données. Il est donc important que les bases de données sources stockent l'UUID des ces entités...
    - Pour les liens avec les **sources**, nous sommes obligés d'utilisé son nom (champ *name*) car elle ne possède pas d'UUID. Cela peut donc poser problème si le nom change. Par défaut, la source correspond à la base de données d'un administrateur de données. Leurs noms devraient peut changer.
- L'import des fichiers se fera à l'aide d'un script Bash exécutant des scripts SQL et Python.
    - Un script Python sera chargé de vérifier les données et de remplacer les différents codes standards par leur identifiant numérique spécifique à la base de données de destination.
    - Des scripts SQL se chargeront de désactiver triggers, contraintes... puis de les réactiver et exécuter globalement après le lancement de la commande de *COPY* d'insertion du fichier CSV.
- De cette façon, nous pouvons espérer intégrer jusqu'à 3 millions d'observations en moins d'une heure dans la table *synthese* de GeoNature.

Principe de la procédure de 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-dessous)
  - le fichier 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 partir de la table d'import en 3 étapes :
  - Ajout des nouvelles observations
  - Modification des observations existantes qui ont subit un changement depuis le dernier import
  - Suppression des observations qui ne sont plus présentes dans l'import courant. Dans le cas du différentiel, les lignes supprimées doivent être présentes dans l'import.
- Pour réaliser le différentiel, nous pouvons utiliser :
  - La clé primaire des données d'origine (champ "*entity_source_pk_value*") ou mieux l'UUID de l'observation (champ "*unique_id_sinp*") 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. Les microsecondes étant accepté pour ce champ, il est possible d'indiquer très précisément la version de l'enregistrement d'une observation.
- Pour les champs contenant des **identifiant de nomenclatures**, comme pour l'importation initiale, une fonction Postgresql présente dans la base de GeoNature ou le parseur Python (développé pour traiter ce point précis) 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. Si l'utilisation des codes ou noms posait problème, il est possible d'utiliser comme alternative les UUID correspondant présent dans les champs "*unique_id_sinp*" ou "*unique_id*" (sauf pour le lien avec la *source*).
- Afin d'éviter tout problème relationnel entre les tables mises à jour dans la base de données, il est important de réaliser :
  - les actions suivantes *parser* le fichier CSV, *copier* le CSV préparé dans la base, *insérer* les lignes dont *meta_last_action* vaut "I" et *mettre à jour* les lignes dont *meta_last_action* vaut "U" sur les entités dans l'ordre suivant : `source`, `organism`, `user`, `acquistion_framework`, `dataset`, `synthese`.
  - enfin, il est possible de *supprimer* les lignes dont *meta_last_action* vaut "D" sur les entités dans l'ordre suivant : `synthese`, `dataset`, `acquistion_framework`, `user`, `organism`,

source.

Gestion du différentiel dans la base des locale d'un administrateur de données

Il faut gérer un différentiel par rapport à la date du dernier export vers le SINP.

Seules les données ajoutées, modifiées ou supprimées depuis le dernier export doivent être transmises.

Pour les données ajoutées :

- `meta_last_acton` doit prendre la valeur "`I`"
- `meta_create_date` doit indiquer la date et heure de création de l'enregistrement.

Pour les données modifiées (1 ou plusieurs fois) :

- `meta_last_action` doit prendre la valeur "U"
- `meta_update_date` doit indiquer la date et heure de **la dernière** modification depuis l'export précédent

Pour les données supprimées :

- `meta_last_action` doit prendre la valeur "`D`".

Cas particuliers :

- une donnée ajoutée (et éventuellement modifiée), puis supprimée depuis le dernier export n'est pas à transmettre ;
- une donnée ajoutée puis modifiée doit être transmise avec `meta_last_action` = "`I`" (ajout) et non "U" (mise à jour).

Format du fichier des métadonnées de l'archive "meta_archive.ini"

Ce fichier au format INI a pour objectif de fournir les informations sur l'origine des autres fichiers fournis dans l'archive.
Il concerne le créateur de l'archive.

Ce fichier devra:

- être encodé en **UTF-8**
- être nommé (en minuscule) : ***meta_archive.ini***

Les règles à respecter pour ce format INI sont le suivantes :

- une ligne peut contenir soit un commentaire débutant par le caractère **#** ou une entrée ***clé / valeur***
- la clé doit être séparé de sa valeur par un **=**
- les clés doivent être en minuscule et utilisé l'underscore (_) comme séparateur de mots
- des espaces peuvent encadrer les clés et valeurs (ils seront supprimés).
- Si la valeur contient plusieurs lignes, encadrée là par des guillemets doubles (") et indenter les ligne supplémentaires.

Format (en gras les champs obligatoires) :

- **format_version** [VARCHAR(8)] : version du format d'échange utilisé pour les fichiers à importer.
- export_date [DATE(YYYY-MM-DD HH:MM)] : date et heure de l'export des observations de la synthèse hors de la base d'origine.
- taxref_version [VARCHAR(8)] : version de TaxRef utilisée lors de la génération de l'archive.
- habref_version [VARCHAR(8)] : version de HabRef utilisée lors de la génération de l'archive.
- **editor** [VARCHAR(100)] : nom de l'organisme créateur de l'archive.
- **contact** [VARCHAR(100)] : infos sur la personne ayant créé l'archive. Format : `NOM Prénom <email>`.
- notes [TEXT] : remarques divers sur les fichiers de l'archive.

Exemple :

```
format_version = 1.0
export_date = 2020-08-27 10:15
taxref_version = 13
editor = Conservatoire Botanique National Alpin
contact = jp.milcent@cbn-alpin.fr
notes = "Données de test.
    À utiliser seulement lors de la phase de conception."
```

Format des fichiers d'import

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**
- avoir un nom au singulier, en minuscules et avec des underscores comme séparateur de mots.
- avoir l'extension *.csv*
- avoir un des noms suivant : *synthese.csv*, *source.csv*, *dataset.csv*, *acquisition_framework.csv*, *organism.csv*, *user.csv*, *occtax.csv*, *validation.csv* ou *meta_additional_data.csv*.

Le format CSV (en réalité plutôt TSV) qu'ils contiendront devra respecter les règles suivantes :

- 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.
- respecter l'ordre des champs décrits dans cette documentation
- si possible fournir les fichiers avec l'ensemble des champs décrits même s'ils doivent contenir des valeurs nulles

Il faut vous assurer d'avoir supprimé, remplacé ou protégé les caractères suivant dans les valeurs des champs :

- les caractères **anti-slash** (\) doivent être supprimé
- les caractères **tabulation** (Tab, ASCII 9) doivent être absolument supprimé du contenu des

champs ou remplacé par `\t`
- les caractères **fin de ligne** (LF, Newline, ASCII 10) sont à supprimer ou à remplacer par `\n`
- les caractères **retour chariot** (CR, Carriage return, ASCII 13) sont à supprimer ou à remplacer par `\r`
- les caractères **tabulation verticale** (Vertical tab, ASCII 11) sont à supprimer ou à remplacer par `\v`



Ce mécanisme de remplacement des caractères `\n`, `\r`... dans la base n'est pas fonctionnel pour l'instant. L'utilisation du mode CSV désactive ce mécanisme comme indiqué ici dans la doc de Postgresql.

Notes diverses

- Les champs contenant des dates au format `[DATE(YYYY-MM-DD HH:MM:SS)]` peuvent éventuellement contenir aussi les microsecondes séparées des secondes par un point. Ex. : `2019-03-07 12:09:14.451907`. C'est le cas pour les champs `meta_create_date` et `meta_update_date`. L'ajout de cette information peut servir à indiquer la nécessité de mise à jour d'un enregistrement en cas de modification très rapprochées réalisées par un outil automatique par exemple. La plupart du temps, cette information est inutile et alourdit le fichier CSV.
- Les champs de type UUID, INT, JSON ou DATE ne devrait pas contenir de valeur vide mais la chaine indiquant l'utilisation de la valeur NULL (`\N`). Si les fichiers transmis contiennent des valeurs vides pour les champs de ces types, elle sera transformé en NULL par le parser (script Python) lors de l'intégration. Pour les champs `meta_create_date` et `meta_update_date`, ça prendra la valeur par défaut `now()` (date d'import en base GN).

Champs additionnels

Le standard prévoit la transmission de champs additionnels.

Nous prévoyons la possibilité de fournir des champs additionnels pour toutes les types de ressources de 2 façons :

- si les champs ne sont pas valables pour toutes les lignes, 1 seule possibilité :
  - utiliser le champ *additional_data* dont les valeurs doivent être au format JSON et être valide. Le JSON en question devra correspondre à un objet d'un seul niveau avec attribut et valeur. Ex. :

    ```
    {"attribut1": "valeur1", "attribut2": "valeur2"}
    ```

- si les champs sont valables pour toutes les lignes du fichier d'import, ~~2~~ 1 possibilités~~s~~ :
  - utiliser le champ *additional_data* (voir ci-dessus)
  - ~~ajouter des colonnes supplémentaires au fichier~~ : non implémenté !

Afin de respecter le standard, il est nécessaire de fournir les méta-données de ces champs additionnel en fournissant le fichier *meta_additional_data.csv* (voir META_ADDITIONAL_DATA). Ce fichier contient la descriptions des champs additionnels.

Pour permettre l'intégration des données, l'utilisation du champ *additional_data* est nécessaire car la possibilité d'ajouter des colonnes supplémentaires aux fichiers CSV n'est pas implémentée.
Les champs additionnels ne seront pas forcément traités pour toutes les ressources. Seuls les champs de la ressource SYNTHESE seront transmis au niveau national.

À faire / Améliorations

- ☐ Ne pas insérer dans les tables les valeurs des champs meta_create_date ou meta_update_date : les valeurs de ces champs issues de la base d'origine peuvent éventuellement être stockés dans le champ `additional_data` de la base GN de destination. Les champs `meta_create_date` et `meta_update_date` des tables de la base GN de destination doivent contenir les timestamps de création/modification de l'enregistrement dans cette base. Cela permettra ainsi de connaitre les dates de création/modification d'un enregistrement dans la base SINP elle-même.
- ☐ Ajouter le champ ''parent_code'' au fichier ORGANISM
- ☐ Tester les outils Frictionless Data et le format de données Data Package : voir Frictionless Data, Data Package, Check List Bank Data formats.
- ☐ Dans le format OccTax, renommer les champs UUID en unique_id_couting, unique_id_occurence et unique_id_survey pour que cela soit plus clair !
- ☐ Dans le format OccTax, prevoir l'ajout des champ code_nomenclature_tech_collect_campanule et sample_number_proof
- ☐ Trouver une solution au problème du remplacement des caractères \n, \r... qui ne fonctionne pas lors de l'import dans la base
- ☐ Modifier le nom du champ //determiner// de la ressource //SYNTHESE// en //determiner**s**//
- ☐ Intégrer si nécessaire les nouveaux champs de GeoNature v2.6.0 aux formats (-- @jpmilcent 2021-02-21) :
  - ACQUISITION_FRAMEWORK : opened, id_digitizer, initial_closing_date
  - DATASET : id_nomenclature_resource_type, active, validable, id_digitizer
  - SYNTHESE : sample_number_proof,
- ☐ Revoir le format des champs "cor_actors_organism" et "cor_actors_user" du format ACQUISITION_FRAMEWORK. La récupération des valeurs de tableaux imbriqués nécessite de passer par du JSON (idem pour les champs de type ARRAY du DATASET). Il serait donc plus efficace d'utiliser directement le format JSON pour ces champs.-- @jpmilcent 2021-02-21.
- ☐ [✓ jpmilcent, 2024-10-23]~~Dans le format DATASET, indiquer les valeurs par défaut des nomenclatures à utiliser. Les champs correspondant dans la table GeoNature étant obligatoire ! Les nomenclatures ne sont pas obligatoires pour les cadres d'acquisition et la synthese.~~
- ☐ [✓ jpmilcent, 2024-10-23]~~Pour le format USER, trouver un solution pour éviter les doublons d'email et d'identifiant => se baser sur l'email => trouver une solution pour la mise à jour de l'email => ajouter un champ "new_email" + champ "email" (= ancien email) + "meta_last_action" = "U".~~ ⇒ ajout d'index unique à la base de données.
- ☐ [✓ jpmilcent, 2024-10-23]~~Pour le format ORGANISM, trouver une solution pour éviter les doublon inter-fournisseur => se baser sur l'UUID du [[http://standards-sinp.mnhn.fr/referentiel-des-organismes/|référentiel national SINP des organismes]].~~
- ☐ [✓ choarau, 2026-01-21]~~Pour les formats ACQUISITION_FRAMEWORK et DATASET, trouver une solution pour éviter les doublon inter-fournisseur => se baser sur le site "[[https://inpn.mnhn.fr/docs-web/docs/download/263010|Référentiel National SINP - Métadonnées]]".~~

Évolutions

- 2026-06-17 :
  - Ajout des statuts des différents fichiers.
  - Ajout de complément sur le format des fichiers CSV attendus.
  - Mise à jour des textes introductifs.
- 2026-04-02 :
  - Utilisation de `VARCHAR(50)` pour tous les champs `code_nomenclature_…` de SYNTHESE.
- 2026-03-13 :
  - Ajout du format VALIDATION. Nécessaire pour intégrer un historique des validations à la Synthese ou dans le cadre de l'utilisation du format OCCTAX.
- 2025-04-30 :
  - Ajout du format OCCTAX supporté pour les ajouts uniquement. Non nécessaire pour l'instant dans le cadre des SINP.
- 2024-10-23 :
  - Nous privilégions maintenant l'utilisation des UUID comme identifiant de lien entre les ressources à la place des codes ou noms.
  - Le champ *unique_id* devient obligatoire pour les ressources ACQUISITION FRAMEWORK et DATASET.
  - Le champ *parent_code* de la ressource ACQUISITION FRAMEWORK doit maintenant indiquer l'UUID du cadre d'acquisition parent.
  - Le champ *code_acquisition_framework* de la ressource DATASET doit maintenant indiquer l'UUID du cadre d'acquisition.
  - Le champ *cor_actors_organism* des ressources ACQUISITION FRAMEWORK et DATASET doit contenir l'UUID de l'organisme et non plus son nom.
  - Le champ *code_dataset* de la ressource SYNTHESE doit contenir l'UUID du jeu de données correspondant.
  - Le champ *code_organism* de la ressource USER doit contenir l'UUID de l'organisme correspondant.
- 2023-07-03 :
  - Ajout de l'UUID au format à plat d'un "utilisateur" (=*user*) permettant de stocker les infos sur les observateurs (*observers*) et les déterminateurs (*determiner*).
  - Utilisation de l'UUID (champ *unique_id* de la ressource *USER*) à la place de l'*identifier* pour le champ *code_digitiser* de la ressource *SYNTHESE*.

Format SYNTHESE d'import

- But : Permet de fournir les informations sur les observations.
- Table GeoNature : "*gn_synthese.synthese*".
- Standard : OccTax v2
- Correspondance GeoNature/Standard : gn_synthese.synthese et OccTax
- Statut : **Production (utilisable et fonctionnel)**

NOTES : nous n'utilisons par le champ *meta_v_taxref* car la transmission de l'info via le fichier *meta_archive.ini* suffit.

Description du format SYNTHESE

Pour chaque ligne : `nom_du_champ [format du champ] (=*nom_champ_table_geonature*) : description du champ.`. Les champs **en gras** sont obligatoires. Pour les nomenclatures, le nom de la mnémonique du type dans GeoNature est indiqué en italique en fin de description. La correspondance avec le nom de cette nomenclature dans le standard OccTax est indiqué entre parenthèses et un lien pointe vers ses valeurs sur le site Standards d'échanges du SINP. Pour les nomenclatures, la valeur à transmettre est celle présente dans la colonne "**Code**" du standard qui est équivalente au champ `cd_nomenclature` de la table `ref_nomenclatures.t_nomenclatures` de GeoNature.

- 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.
- source_id [VARCHAR(25)] (=*entity_source_pk_value*) : code alphanumérique correspondant à l'équivalant d'une clé primaire pour les occurrences des données sources.
- source_id_grp [VARCHAR(25)] (Pas dans synthese) : code alphanumérique correspondant à l'équivalant d'une clé primaire pour les relevés des données sources.
- **code_source** [VARCHAR(255)] (=*id_source*) : code alphanumérique permettant d'identifier la source des données. Sert de lien avec la ressource SOURCE et son champ "*name*".
- **code_dataset** [VARCHAR(255)] (=*id_dataset*) : code alphanumérique permettant d'identifier le jeu de donnée de l'observation. Sert de lien avec la ressource DATASET et son champ "*unique_id*".
- code_nomenclature_geo_object_nature [VARCHAR(50)] (=*id_nomenclature_geo_object_nature*) : code alphanumérique de la valeur du type de nomenclature *NAT_OBJ_GEO* (NatureObjetGeoValue|3|page64).
- code_nomenclature_grp_typ [VARCHAR(50)] (=*id_nomenclature_grp_typ*) : code alphanumérique de la valeur du type de nomenclature *TYP_GRP* (TypeRegroupementValue|24|page90).
- grp_method [VARCHAR(255)] : description de la méthode ayant présidé au regroupement, de façon aussi succincte que possible : champ libre.
- code_nomenclature_obs_technique [VARCHAR(50)] (=*id_nomenclature_obs_technique*) : code alphanumérique de la valeur du type de nomenclature *METH_OBS* (ObservationTechniqueValue|14|page69).
- code_nomenclature_bio_status [VARCHAR(50)] (=*id_nomenclature_bio_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_BIO* (OccurrenceStatutBiologiqueValue|13|page76).
- code_nomenclature_bio_condition [VARCHAR(50)] (=*id_nomenclature_bio_condition*) : code alphanumérique de la valeur du type de nomenclature *ETA_BIO* (OccurrenceEtatBiologiqueValue|7|page75).
- code_nomenclature_naturalness [VARCHAR(50)] (=*id_nomenclature_naturalness*) : code alphanumérique de la valeur du type de nomenclature *NATURALITE* (OccurrenceNaturaliteValue|8|page77).
- code_nomenclature_exist_proof [VARCHAR(50)] (=*id_nomenclature_exist_proof*) : code alphanumérique de la valeur du type de nomenclature *PREUVE_EXIST* (PreuveExistanteValue|15|page83).
- code_nomenclature_valid_status [VARCHAR(50)] (=*id_nomenclature_valid_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_VALID* (NiveauValidationValue|80|page67).
- code_nomenclature_diffusion_level [VARCHAR(50)] (=*id_nomenclature_diffusion_level*) : code alphanumérique de la valeur du type de nomenclature *NIV_PRECIS*

(NiveauPrecisionValue|5|page64).
- code_nomenclature_life_stage [VARCHAR(50)] (=*id_nomenclature_life_stage*) : code alphanumérique de la valeur du type de nomenclature *STADE_VIE* (OccurrenceStadeDeVieValue|10|page78).
- code_nomenclature_sex [VARCHAR(50)] (=*id_nomenclature_sex*) : code alphanumérique de la valeur du type de nomenclature *SEXE* (OccurrenceSexeValue|9|page78).
- code_nomenclature_obj_count [VARCHAR(50)] (=*id_nomenclature_obj_count*) : code alphanumérique de la valeur du type de nomenclature *OBJ_DENBR* (ObjetDenombrementValue|6|page68).
- code_nomenclature_type_count [VARCHAR(50)] (=*id_nomenclature_type_count*) : code alphanumérique de la valeur du type de nomenclature *TYP_DENBR* (TypeDenombrementValue|21|page86).
- code_nomenclature_sensitivity [VARCHAR(50)] (=*id_nomenclature_sensitivity*) : code alphanumérique de la valeur du type de nomenclature *SENSIBILITE* (SensibiliteValue|16|page84).
- code_nomenclature_observation_status [VARCHAR(50)] (=*id_nomenclature_observation_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_OBS* (StatutObservationValue|18|page84).
- code_nomenclature_blurring [VARCHAR(50)] (=*id_nomenclature_blurring*) : code alphanumérique de la valeur du type de nomenclature *DEE_FLOU* (dEEFloutageValue|4|page62).
- code_nomenclature_source_status [VARCHAR(50)] (=*id_nomenclature_source_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_SOURCE* (StatutSourceValue|19|page85).
- code_nomenclature_info_geo_type [VARCHAR(50)] (=*id_nomenclature_info_geo_type*) : code alphanumérique de la valeur du type de nomenclature *TYP_INF_GEO* (TypeInfoGeoValue|23|page89).
- code_nomenclature_behaviour [VARCHAR(50)] (=*id_nomenclature_behaviour*) : code alphanumérique de la valeur du type de nomenclature *OCC_COMPORTEMENT* (OccurrenceComportementValue|110|page72).
- code_nomenclature_biogeo_status [VARCHAR(50)] (=*id_nomenclature_biogeo_status*) : code alphanumérique de la valeur du type de nomenclature *STAT_BIOGEO* (StatutBiogeographiqueValue|11|page81).
- reference_biblio [VARCHAR(255)] : référence de la source de l'observation lorsque celle-ci est de type « Littérature », au format ISO690. La référence bibliographique doit concerner l'observation même et non uniquement le taxon ou le protocole.
- count_min [INT(4)] : nombre minimum d'individus du taxon composant l'observation.
- count_max [INT(4)] : nombre maximum d'individus du taxon composant l'observation. Mettre la même valeur que *count_min* si 1 seule valeur de dénombrement.
- cd_nom [INT(4)] : code du taxon en vigueur « cd_nom » de TaxRef référençant au niveau national le taxon. Il peut être trouvé dans la colonne "CD_NOM" de TaxRef.
- cd_hab [INT(4)] : code HABREF de l'habitat où le taxon de l'observation a été identifié. Il peut être trouvé dans la colonne "CD_HAB" d'HabRef.
- **nom_cite** [VARCHAR(1000)] : nom du taxon cité à l'origine par l'observateur. Ne pas mettre de valeur NULL dans ce champ mais une valeur vide ("").
- digital_proof [TEXT] : si possible une URL (idéalement un permalien) vers la preuve en ligne.
- non_digital_proof [TEXT] : adresse ou nom de la personne ou de l'organisme qui permettrait de retrouver la preuve non numérique de l'observation.
- altitude_min [INT(4)] : altitude minimum de l'observation en mètres.
- altitude_max [INT(4)] : altitude maximum de l'observation en mètres.
- depth_min [INT(4)] : profondeur Minimum de l'observation en mètres selon le référentiel des profondeurs indiqué dans les métadonnées (système de référence spatiale verticale).

- depth_max [INT(4)] : profondeur Maximale de l'observation en mètres selon le référentiel des profondeurs indiqué dans les métadonnées (système de référence spatiale verticale).
- place_name [VARCHAR(500)] : Nom du lieu ou de la station où a été effectuée l'observation. **ATTENTION** : cet attribut ne pourra pas être flouté !
- geom [geometry(Geometry,2154)] (=*the_geom_local* ⇒ *the_geom_4326*, *the_geom_point*) : géométrie détaillée de l'observation transféré au format WKT. Utiliser la syntaxe Postgis pour indiquer le SRID utilisé au début du WKT (Ex. `SRID=2154;POLYGON((786000 6515000,786500 6515000,786500 6514500,786000 6514500,786000 6515000))` ou `SRID=2154;POINT(959204.517 6405928.812)`). Utiliser préférentiellement le SRID 2154 (à défaut le SRID 4326).
- precision [INT(4)] : estimation en mètres d'une zone tampon autour de l'objet géographique. Cette précision peut inclure la précision du moyen technique d'acquisition des coordonnées (GPS,...) et/ou du protocole naturaliste.
- code_area_attachment [VARCHAR(25)] : (ex : COM.05170). Permet d'indiquer dans le cas de données imprécises une zone géographique déjà présente dans la base. Cela évite l'utilisation d'une géométrie volumineuse car nous devibs seulement stocker le centroïde dans le champ `geom`. Le format de ce champ correspond au type de zone géographique, tel qu'il est définit dans le champ `type_code` de la table `l.areas`, agrégé par un point au code INSEE de cette zone (ex : DEP.05) soit le champ `area_code` de la table `l.areas`. Les types de zone géographique disponibles : COM (communes) et DEP (Département). Il est envisageable d'utiliser d'autres zones en fonction des possibilités offertes par la base GeoNature de destination. Penser à remplir en correspondant le champ `code_nomenclature_info_geo_type`.
- **date_min** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure du jour, dans le système local de l'observation dans le calendrier grégorien. En cas d'imprécision, cet attribut représente la date la plus ancienne de la période d'imprécision. La date doit être écrite suivant la norme ISO8601. L'heure est dans le fuseau horaire de la zone d'observation. Ce champ est obligatoire et ne peut pas contenir de valeur nulle.
- **date_max** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure du jour, dans le système local de l'observation dans le système grégorien. Lorsqu'une observation est faite sur un jour, les dates et heures de début et de fin sont les mêmes (cas le plus courant). La date doit être écrite suivant la norme ISO8601. L'heure est dans le fuseau horaire de la zone d'observation. Ce champ est obligatoire et ne peut pas contenir de valeur nulle. Ce champ doit contenir une date supérieur ou égale à celle du champ "date_min".
- validator [VARCHAR(1000)] : personne ayant procédé à la validation (et organisme). Voir le détail du format à plat des infos sur une personne.
- validation_comment [TEXT] : commentaire sur la validation.
- validation_date [DATE(YYYY-MM-DD HH:MM:SS)] (=*meta_validation_date*) : date et heure de validation de l'observation.
- observers [VARCHAR(1000)] : Nom, prénom, email, organisme et UUID de la ou des personnes ayant réalisé l'observation. Voir le détail du format à plat des infos sur une personne.
- determiner [VARCHAR(1000)] : Prénom, nom et organisme de la ou les personnes ayant réalisé la détermination taxonomique de l'observation. Ex : Jérémie VAN ES (CBNA).
- determination_date [DATE(YYYY-MM-DD HH:MM:SS)] (Pas dans synthese) : date/heure de la dernière détermination du taxon de l'observation dans le calendrier grégorien.
- code_digitiser [VARCHAR(50)]: UUID de l'utilisateur à l'origine de la saisie de l'observation. Sert de lien avec la ressource USER et son champ "*unique_id*".
- code_nomenclature_determination_method [VARCHAR(50)] (=*id_nomenclature_determination_method*) : code alphanumérique de la valeur du type de nomenclature *METH_DETERMIN* (champ libre dans OccTax) (Voir les codes disponibles).
- comment_context [TEXT] : description libre du contexte de l'observation, aussi succincte et

précise que possible. Informations sur le lieu (=*où*), le moment (=*quand*) , la méthode (=*comment*) et la personne ayant réalisé l'observation (=*qui*).
- comment_description [TEXT] : description libre de l'observation, aussi succincte et précise que possible. Informations sur le(s) individu(s) observé(s) (=*quoi*).
- additional_data [JSON] : permet d'associer des champs complémentaires à la synthèse si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **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*").

Précisions sur le champ additionnal_data

Correspondance avec la commune et le département

Dans le champ "*additional_data*", il est intéressant de faire figurer des informations concernant la commune et/ou le département d'appartenance de l'observation. Cela permet de "forcer" le rapprochment d'une observation à sa commune et/ou département quand celle-ci est très proche de la limite de sa géométrie. Le JSON peut alors contenir un objet avec les attributs suivant :

```
{
     "communeInseeCode": "<code-insee-de-la-commune-de-l'obs>",
     "departementInseeCode": "<code-insee-du-departement>"
}
```

Mettre dans l'attribut "*communeInseeCode*" le code INSEE de la commune où l'observation a été réalisée si l'info est disponible.

Mettre dans le champ "*departementInseeCode*" le code INSEE du département uniquement si le code INSEE de la commune où l'observation a été réalisée est inconnu. Cela devrait être le cas uniquement pour les observations imprécises...

Du coup, le format JSON le plus courant devrait être :

```
{
     "communeInseeCode": "<code-insee-de-la-commune-de-l'obs>"
}
```

Notion de confidentialité

Vous pouvez ajouter une notion de confidentialité dans le champ `additional_data` avec l'attribut "*confidential*". Cet attribut prend en valeur un bouléen (`true` ou `false`). Son abscence équivault à `false`, vous pouvez donc l'ajouter que pour les observations confidentielles.

Donnée confidentielle : donnée privée dont niveau diffusion

(`code_nomenclature_diffusion_level`) est différent de 5 ou donnée sensible dont valeur de sensibilité (`code_nomenclature_sensitivity`) est différente de 0.

Exemple du rendu JSON :

```
{
     "confidential": true
}
```

Champ précision et libellé

Dans le cadre du SINP PACA, le CEN-PACA utilise une notion de précision sous forme de libellés : "précis", "lieu-dit", "commune", "indéterminé".

Le champ `précision` est rempli en conséquence ainsi :

- "précis" : 25 (mètres)
- "lieu-dit" : 250 (mètres)
- "commune" : calcul du rayon moyen en mètre de la commune de l'observation en se basant sur la solution (n°2) de la requête. La solution (n°2) calcule la distance moyenne du centroïde du polygone de la commune a chaque point constituant son périmètre :

  ```
  round(AVG(ST_Distance(st_centroid(la.geom), perimeters.geom)))
  ```

- "indéterminé" : `NULL`

Du coup, il serait intéressant de faire figurer dans le champ `additional_data` les libellés qui pourront être associé à l'attribut "*precisionLabel*" d'un objet JSON.
Pour ce faire, attribuer un des 4 valeurs ci-dessous :

- `précis` : si l'observation possède un valeur dans le champ précision inférieur ou égale à 25m.
- `lieu-dit` : si l'observation possède un valeur dans le champ précision supérieur à 25m et inférieure ou égale à 250m.
- `commune` : si l'observation possède un valeur dans le champ précision supérieur à 250m.
- `indéterminé` : si la précision de l'observation n'est pas connue.

Exemple du rendu JSON :

```
{
     "precisionLabel": "lieu-dit"
}
```

Format SOURCE d'import

- But : Permet de décrire la source des données.
- Table GeoNature : "*gn_synthese.t_sources*".
- Statut : **Production (utilisable et fonctionnel)**

Description du format SOURCE

Pour chaque ligne : `nom_du_champ [format du champ] (=nom_champ_table_geonature) : description du champ.`. Les champs **en gras** sont obligatoires.

- **name** [VARCHAR(255)] (=*name_source*) : correspond au champ "*name_source*". Sert de lien avec la ressource SYNTHESE et son champ "*code_source*".
- desc [TEXT] (=*desc_source*) : 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)] (=*url_source*) : adresse web décrivant la source des données ou permettant d'accéder aux données sources. Correspond au champ "*url_source*".
- additional_data [JSON] (=*champs_addi*) : permet d'associer des champs complémentaires à la source si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **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

- But : permet de fournir les informations sur les jeux de données.
- Table GeoNature : "*gn_meta.t_datasets*" et aux tables liées *cor_dataset_actor*, *cor_dataset_territory*.
- Standard : Métadonnées v1.3.10
- Correspondance GeoNature/Standard : gn_meta.t_datasets et Métadonnées

NOTES :

- pour éviter trop de complexité, nous ne tenons pas compte des "protocoles" liées au jeu de données. Si cela s'avérait nécessaire, il est possible de les ajouter manuellement une fois l'import effectué.
- Dans la table GeoNature correspondante à ce fichier, les nomenclatures ne peuvent pas être nulle. Il faut donc indiquer la valeur par défaut ou ne pas inclure la colonne dans le fichier CSV.

Description du format DATASET

Pour chaque ligne : `nom_du_champ [format du champ] (=nom_champ_table_geonature) : description du champ.`. Les champs **en gras** sont obligatoires.

- **unique_id_sinp** [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. Sert de le lien avec la ressource ACQUISITION_FRAMEWORK et son champ "*unique_id*".
- **name** [VARCHAR(150)] (=*dataset_name*) : nom du jeu de donnée (**150 caractères max**).
- **shortname** [VACHAR(30)] (=*dataset_shortname*) : libellé court du jeu de données (**30 caractères max**). Permet de faire le lien avec la ressource SYNTHESE et son champ "*code_dataset*".
- **desc** [TEXT] (=*dataset_desc*) : description du jeu de données.
- *code_nomenclature_data_type* [VARCHAR(25)] (=*id_nomenclature_data_type*) : code de la valeur du type de nomenclature *DATA_TYP* (TypeDonneesValue). Si null ou non fourni, le code "1" (Occurrences de taxons) sera utilisé.
- keywords [TEXT] : mot(s)-clé(s) représentatifs du jeu de données.
- **marine_domain** [BOOL] : indique si le jeu de données concerne le domaine marin. Ne peut pas être NULL.
- **terrestrial_domain** [BOOL] : indique si le jeu de données concerne le domaine terrestre. Ne peut pas être NULL.
- *code_nomenclature_dataset_objectif* [VARCHAR(25)] (=*id_nomenclature_dataset_objectif*) : code de la valeur du type de nomenclature *JDD_OBJECTIFS* (ObjectifJeuDonneesValue|92|page43). Si null ou non fourni, le code "1.1" (Observations naturalistes opportunistes) sera utilisé.
- bbox_west [FLOAT] : extrême ouest de l'emprise géographique du jeu de données (rectangle permettant d'englober le jeu de données et défini par 4 extrêmes : nord, sud, est, et ouest). La valeur doit correspondre à la longitude ouest dans le SRID 4326.
- bbox_east [FLOAT] : extrême est de l'emprise géographique du jeu de données (rectangle permettant d'englober le jeu de données et défini par 4 extrêmes : nord, sud, est, et ouest). La valeur doit correspondre à la longitude est dans le SRID 4326.
- bbox_south [FLOAT] : extrême sud de l'emprise géographique du jeu de données (rectangle permettant d'englober le jeu de données et défini par 4 extrêmes : nord, sud, est, et ouest). La valeur doit correspondre à la latitude sud dans le SRID 4326.
- bbox_north [FLOAT] : extrême nord de l'emprise géographique du jeu de données (rectangle permettant d'englober le jeu de données et défini par 4 extrêmes : nord, sud, est, et ouest). La valeur doit correspondre à la latitude nord dans le SRID 4326.
- *code_nomenclature_collecting_method* [VARCHAR(25)] (=*id_nomenclature_collecting_method //) : code de la valeur du type de nomenclature //METHO_RECUEIL* (MethodeRecueilValue|91|page42). Si null ou non fourni, le code "1" (Observation directe) sera utilisé.
- *code_nomenclature_data_origin* [VARCHAR(25)] (=*id_nomenclature_data_origin //) : code de la valeur du type de nomenclature //DS_PUBLIQUE* (DSPubliqueValue [Standard OccTax]|2|page63). Si null ou non fourni, le code "NSP" (Ne Sait Pas) sera utilisé.
- *code_nomenclature_source_status* [VARCHAR(25)] (=*id_nomenclature_source_status //) : code de la valeur du type de nomenclature //STATUT_SOURCE* (StatutSourceValue|19|page85). Si null ou non fourni, le code "NSP" (Ne Sait Pas) sera utilisé.
- *code_nomenclature_resource_type* [VARCHAR(25)] (=*id_nomenclature_resource_type //) : code de la valeur du type de nomenclature //RESOURCE_TYP* (?).Si null ou non fourni, le code "1" (Jeu de données) sera utilisé.
- 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*" (= *TERRITOIRE* TerritoireValue|87|page50) de la table "*gn_meta.cor_dataset_territory*" et comme seconde valeur une description du territoire (champ "*territory_desc*) ou une chaine vide. Exemple :

```
"{{""REU "","""""},{""MYT"","""""}}"
```

- 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 l'UUID de l'organisme et comme seconde valeur le code de nomenclature correspondant au champ "*id_nomenclature_actor_role*" (="ROLE_ACTEUR" / RoleActeurValue|86|page49) de la table "*gn_meta.cor_dataset_actor*". Exemple :

```
 "{{""5a433bd0-1fc0-25d9-e053-2614a8c026f8"",
""1""},{""5a433bd0-1ffc-25d9-e053-2614a8c026f8"",""5""}}"
```

- cor_actors_user [TEXT(ARRAY-ARRAY)] : champ de type tableau de tableau de textes. Les tableaux de textes du plus bas niveau contiendront comme première valeur l'identifiant de l'utilisateur/personne et comme seconde valeur le code de nomenclature correspondant au champ "*id_nomenclature_actor_role*" (="ROLE_ACTEUR" / RoleActeurValue|86|page49) de la table "*gn_meta.cor_dataset_actor*". Exemple :

```
 "{{""jpmilcent"",""1""},{""pmartin"",""5""}}"
```

- additional_data [JSON] : permet d'associer des champs complémentaires au jeu de données si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **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

- But : permet de fournir les informations sur les cadres d'acquisition des jeux de données.
- Tables GeoNature : "*gn_meta.t_acquisition_framework*" et aux tables liées *cor_acquisition_framework_actor*, *cor_acquisition_framework_objectif*, *cor_acquisition_framework_publication*, *cor_acquisition_framework_voletsinp* et *sinp_datatype_publications*.
- Standard : Métadonnées v1.3.10
- Correspondance GeoNature/Standard : gn_meta.t_acquisition_framework et Métadonnées
- Statut : **Production (utilisable et fonctionnel)**

NOTES :

- Au moins un des deux champs *cor_actor_organism* ou *cor_actor_user* doit être renseigné par cadre d'acquisition.

Description du format ACQUISITION_FRAMEWORK

Pour chaque ligne : `nom_du_champ [format du champ] (=nom_champ_table_geonature) : description du champ.`. Les champs **en gras** sont obligatoires.

- **unique_id** [UUID] (=*unique_acquisition_framework_id*) : UUID du cadre d'acquisition si pré-

existant.

- **name** [VARCHAR(255)] (=*acquisition_framework_name*) : nom complet du cadre d'acquisition. Sert de le lien avec la ressource DATASET et son champ "*code_acquisition_framework*" mais aussi avec le champ *parent_code*.
- **desc** [TEXT] (=*acquisition_framework_desc*) : description du cadre d'acquisition.
- code_nomenclature_territorial_level [VARCHAR(25)] (=*id_nomenclature_territorial_level*) : code alphanumérique de la valeur du type de nomenclature *NIVEAU_TERRITORIAL* (NiveauTerritorialValue|84|page23).
- territory_desc [TEXT] : description du territoire concerné.
- keywords [TEXT] : mot(s)-clé(s) représentatifs du cadre d'acquisition, séparés par des virgules.
- code_nomenclature_financing_type [VARCHAR(25)] (=*id_nomenclature_financing_type*) : code du type de financement pour le cadre d'acquisition, tel que défini dans la nomenclature *TYPE_FINANCEMENT* (TypeFinancementValue|88|page28).
- target_description [TEXT] : description de la cible taxonomique ou géologique pour le cadre d'acquisition (Ex. : pteridophyta).
- ecologic_or_geologic_target [TEXT] : champ composé de *cd_nom* de TaxRef, séparés par des points virgules, s'il s'agit de taxons, ou de *cd_hab* de HabRef, séparés par des points virgules, s'il s'agit d'habitats.
- parent_code [VARCHAR(255)](=*acquisition_framework_parent_id*) : si ce cadre d'acquisition est contenu dans un autre, le code/nom du cadre acquisition parent sera indiqué. Sert de le lien avec la ressource ACQUISITION_FRAMEWORK parente et son champ "unique_id".
- is_parent [BOOL] : indique si ce dispositif est un métacadre, et donc s'il contient d'autres cadres d'acquisition.
- **start_date** [DATE(YYYY-MM-DD)] (=*acquisition_framework_start_date*) : date de lancement du cadre d'acquisition. **Ne peut pas être NULL**.
- end_date [DATE(YYYY-MM-DD)] (=*acquisition_framework_end_date*) : cate de clôture du cadre d'acquisition. Si elle n'est pas remplie, on considère que le cadre est toujours en activité.
- 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*" (="CA_OBJECTIFS" / ObjectifsValue|page24) 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 " / VoletSINPValue|90|page29) 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 l'UUID de l'organisme et comme seconde valeur le code de nomenclature correspondant au champ "*id_nomenclature_actor_role*" (="ROLE_ACTEUR" / RoleActeurValue|86|page25) de la table "*gn_meta.cor_acquisition_framework_actor*". Exemple :

```
"{{""5a433bd0-1fc0-25d9-e053-2614a8c026f8"",
""1""},{""5a433bd0-1ffc-25d9-e053-2614a8c026f8"",""5""}}"
```

Note : Au moins un organisme (ce champ) ou un utilisateur (le champ *cor_actors_user*) doit être renseigné par cadre d'acquisition.

- cor_actors_user [TEXT(ARRAY-ARRAY)] : champ de type tableau de tableau de textes. Les tableaux de textes du plus bas niveau contiendront comme première valeur l'identifiant de l'utilisateur/personne et comme seconde valeur le code de nomenclature correspondant au champ "*id_nomenclature_actor_role*" (="ROLE_ACTEUR" / RoleActeurValue|86|page25) de la table "*gn_meta.cor_acquisition_framework_actor*". Exemple :

```
"{{""jpmilcent"",""1""},{""pmartin"",""5""}}"
```

  Note : Au moins un utilisateur (ce champ) ou un organisme (le champ *cor_actors_organism*) doit être renseigné par cadre d'acquisition.

- cor_publications [JSON] : champ contenant un tableau d'objets au format JSON valide. Ce champ correspond à la table *cor_acquisition_framework_publication* de GeoNature. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL. Chaque objet du JSON représentera une publication avec les attributs suivant :
  - ***reference*** : attribut obligatoire contenant la référence complète de la publication au format ISO 690.
  - *url* : permalien permettant de la publication si disponible sur le web.
  - *uuid* : UUID de cette publication.
- additional_data [JSON] : permet d'associer des champs complémentaires au cadre d'acquisition si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **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

- But : permet de fournir les informations sur les organismes liées aux jeux de données et cadres d'acquisition.
- Table GeoNature : "*utilisateurs.bib_organismes*".
- Statut : **Production (utilisable et fonctionnel)**

Description du format ORGANISM

Pour chaque ligne : `nom_du_champ [format du champ] (=nom_champ_table_geonature) : description du champ.`. Les champs **en gras** sont obligatoires.

- **unique_id** [UUID] (=*uuid_organisme*) : UUID de l'organisme si pré-existant dans les données sources. Correspond au champ "uuid_organisme". Privilégier les UUID issus de la base du référentiel des organismes de l'INPN. Rechercher un organisme avec le web service : `https://odata-sinp.mnhn.fr/organizations?shortNameFragmentOrLongNameFragment=<nom-organisme>` . Ex. : https://odata-sinp.mnhn.fr/organizations?shortNameFragmentOrLongNameFragment=Alpin . Alternative : utiliser une sauvegarde au format tableur du référentiel.

- **name** [VARCHAR(100)] (=*nom_organisme*) : correspond au champ "nom_organisme". Sert de lien avec les tables DATASET (*cor_actors_organism*) et ACQUISITION_FRAMEWORK (*cor_actors_organism*).
- address [VARCHAR(128)] (=*adresse_organisme*) : correspond au champ "adresse_organisme".
- postal_code [VARCHAR(5)] (=*cp_organisme*) : correspond au champ "cp_organisme".
- city [VARCHAR(100)] (=*ville_organisme*) : correspond au champ "ville_organisme".
- phone [VARCHAR(14)] (=*tel_organisme*) : correspond au champ "tel_organisme".
- fax [VARCHAR(14)] (=*fax_organisme*) : correspond au champ "fax_organisme".
- email [VARCHAR(100)] (=*email_organisme*) : correspond au champ "email_organisme".
- url [VARCHAR(255)] (=*url_organisme*) : correspond au champ "url_organisme".
- logo_url [VARCHAR(255)] (=*url_logo*) : correspond au champ "url_logo".
- additional_data [JSON] : permet d'associer des champs complémentaires à l'organisme si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide.
- **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*").

Format USER d'import

- But : permet de fournir les informations sur les utilisateurs, c'est à dire les personnes désireuses d'accéder aux informations demandant un compte. Normalement, ce fichier ne devrait être utilisé que pour l'import initial mais des champs "meta_*" sont fournis s'il s'avérait nécessaire de l'utiliser aussi en mise à jour.
- Table GeoNature : "*utilisateurs.t_roles*"
- Statut : **Production (utilisable et fonctionnel)**

NOTES : nous ne gérons pas dans cette version la notion de groupe de GeoNature.

Description du format USER

Pour chaque ligne : `nom_du_champ [format du champ] (=nom_champ_table_geonature) : description du champ.`. Les champs **en gras** sont obligatoires.

- **unique_id** [UUID] (=*uuid_role*) : UUID associé à l'utilisateur.
- **identifier** [VARCHAR(100)] (=*identifiant)* : identifiant/login de l'utilisateur. Peut éventuellement correspondre à la première partie de l'email (avant le @). Sert de lien avec la ressource SYNTHESE (*code_digitiser*).
- firstname [VARCHAR(50)] (=*prenom_role*) : prénom.
- name [VARCHAR(50)] (=*nom_role*) : nom de famille.
- email [VARCHAR(250)] : courriel.
- code_organism [VARCHAR(100)] (=*id_organisme*) : UUID permettant d'identifier l'organisme de l'utilisateur (voir ORGANISM) ou éventuellement son code alphanumérique. Sert de le lien avec la ressource ORGANISM et son "unique_id" ou éventuellement "name".
- comment [TEXT] (=*remarques*) : contient des éventuelles informations internes (date d'importation, par exemple).

- enable [BOOL] (=*active*) : indique si l'utilisateur est actif (=true) et donc peut se connecter ou pas (=false). Par défaut, "True".
- additional_data [JSON] (=*champs_addi*) : permet d'associer des champs complémentaires à l'utilisateur si toutes les entrées ne partagent pas les même champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement de l'utilisateur.
- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement de l'utilisateur.
- **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

Format OCCTAX d'import

- But : Permet de fournir les informations sur les observations à intégrer aux tables du module OccTax de GeoNature.
- Table GeoNature : "*pr_occtax.**".
- Standard : OccTax v2
- Statut : ** Bêta (fonctionnel mais des corrections peuvent subvenir) **

Description du format OCCTAX

Pour chaque ligne : `nom_du_champ [format du champ] (=table_geonature.nom_champ) : description du champ.`. Les champs **en gras** sont obligatoires. Pour les nomenclatures, le nom de la mnémonique du type dans GeoNature est indiqué en italique en fin de description. La correspondance avec le nom de cette nomenclature dans le standard OccTax est indiqué entre parenthèses et un lien pointe vers le document PDF Standards d'échanges du SINP. Pour les nomenclatures, la valeur à transmettre est celle présente dans la colonne "**Code**" du standard qui est équivalente au champ `cd_nomenclature` de la table `ref_nomenclatures.t_nomenclatures` de GeoNature.

- **unique_id_sinp_grp** [UUID] (*t_releve_occtax.unique_id_sinp_grp*) : UUID SINP du relevé s'il existe déjà dans les données sources.
- **code_dataset** [VARCHAR(255)] (=*T_releves_occtax.id_dataset*) : code alphanumérique permettant d'identifier le jeu de donnée de l'observation. Sert de lien avec la ressource DATASET et son champ "shortname".
- code_digitiser [VARCHAR(50)] (*t_releves_occtax.id_digitiser*): UUID de l'utilisateur à l'origine de la saisie de l'observation. Sert de lien avec la ressource USER et son champ "unique_id".
- observers [VARCHAR(1000)] (*t_releves_occtax.observers_txt*] : Nom, prénom, email, organisme et UUID de la ou des personnes ayant réalisé l'observation. Voir le détail du format à plat des infos sur une personne.
- **code_nomenclature_grp_typ** [VARCHAR(25)] (=*t_releves_occtax.id_nomenclature_grp_typ*) : code alphanumérique de la valeur du type de nomenclature TYP_GRP (TypeRegroupementValue|24|page90).
- grp_method [VARCHAR(255)] (=t_releves_occtax.grp_method) : description de la méthode ayant présidé au regroupement, de façon aussi succincte que possible : champ libre.
- **date_min** [DATE(YYYY-MM-DD HH:MM:SS)] (=*t_releves_occtax.date_min*) : date et heure du jour, dans le système local de l'observation dans le calendrier grégorien. En cas d'imprécision,

cet attribut représente la date la plus ancienne de la période d'imprécision. La date doit être écrite suivant la norme ISO8601. L'heure est dans le fuseau horaire de la zone d'observation et correspond à l'heure stocké dans le champ `hour_min` de la table `t_releve_occtax`. Ce champ est obligatoire et ne peut pas contenir de valeur nulle.

- **date_max** [DATE(YYYY-MM-DD HH:MM:SS)] (=*t_releves_occtax.date_max*) : date et heure du jour, dans le système local de l'observation dans le système grégorien. Lorsqu'une observation est faite sur un jour, les dates et heures de début et de fin sont les mêmes (cas le plus courant). La date doit être écrite suivant la norme ISO8601. L'heure est dans le fuseau horaire de la zone d'observation et correspond à l'heure stocké dans le champ `hour_max` de la table `t_releve_occtax`. Ce champ est obligatoire et ne peut pas contenir de valeur nulle. Ce champ doit contenir une date supérieur ou égale à celle du champ "date_min".
- cd_hab [INT(4)] (=*t_releves_occtax.cd_hab*) : code HABREF de l'habitat où le taxon de l'observation a été identifié. Il peut être trouvé dans la colonne "CD_HAB" d'HabRef.
- altitude_min [INT(4)] (=*t_releves_occtax.altitude_min*) : altitude minimum de l'observation en mètres.
- altitude_max [INT(4)] (=*t_releves_occtax.altitude_max*) : altitude maximum de l'observation en mètres.
- depth_min [INT(4)] (=*t_releves_occtax.depth_min*) : profondeur Minimum de l'observation en mètres selon le référentiel des profondeurs indiqué dans les métadonnées (système de référence spatiale verticale).
- depth_max [INT(4)] (=*t_releves_occtax.depth_max*) : profondeur Maximale de l'observation en mètres selon le référentiel des profondeurs indiqué dans les métadonnées (système de référence spatiale verticale).
- place_name [VARCHAR(500)] (=*t_releves_occtax.place_name*) : Nom du lieu ou de la station où a été effectuée l'observation. ATTENTION : cet attribut ne pourra pas être flouté !
- meta_device_entry : mettre NULL.
- comment_context [TEXT] (=*t_releves_occtax.comment*): description libre du contexte de l'observation, aussi succincte et précise que possible. Informations sur le lieu (=où), le moment (=quand) , la méthode (=comment) et la personne ayant réalisé l'observation (=qui).
- geom [geometry(Geometry,2154)] (=*t_releves_occtax.geom_local et .geom_4326* ⇒ the_geom_4326, the_geom_point) : géométrie détaillée de l'observation transféré au format WKT. Utiliser la syntaxe Postgis pour indiquer le SRID utilisé au début du WKT (Ex. `SRID=2154;POLYGON((786000 6515000,786500 6515000,786500 6514500,786000 6514500,786000 6515000))` ou `SRID=2154;POINT(959204.517 6405928.812)`). Utiliser préférentiellement le SRID 2154 (à défaut le SRID 4326).
- code_nomenclature_geo_object_nature [VARCHAR(25)] (=*t_releves_occtax.id_nomenclature_geo_object_nature*) : code alphanumérique de la valeur du type de nomenclature NAT_OBJ_GEO (NatureObjetGeoValue|3|page64).
- precision [INT(4)] (=*t_releves_occtax.precision*) : estimation en mètres d'une zone tampon autour de l'objet géographique. Cette précision peut inclure la précision du moyen technique d'acquisition des coordonnées (GPS,...) et/ou du protocole naturaliste.
- additional_fields_releve [JSON] : permet d'associer des champs complémentaires et/ou si toutes les entrées ne partagent pas les même champs pour le relevé. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **unique_id_occurrence** [UUID] : UUID de l'occurrence s'il existe déjà dans les données sources.
- **code_nomenclature_tech_collect_campanule** [VARCHAR(25)] (=*t_releves_occtax.id_nomenclature_tech_collect_camp*) : code alphanumérique de la valeur du type de nomenclature METH_OBS (ObservationTechniqueValue|14|page69). Attention : MethodeObservation devient techniqueObservation, renommé "Technique de collecte (Campanule)(2018).

- **code_nomenclature_bio_condition** [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_bio_condition*) : code alphanumérique de la valeur du type de nomenclature ETA_BIO (OccurrenceEtatBiologiqueValue|7|page75).
- code_nomenclature_bio_status [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_bio_status*) : code alphanumérique de la valeur du type de nomenclature STATUT_BIO (OccurrenceStatutBiologiqueValue|13|page76).
- code_nomenclature_naturalness [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_naturalness*) : code alphanumérique de la valeur du type de nomenclature NATURALITE (OccurrenceNaturaliteValue|8|page77).
- code_nomenclature_exist_proof [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_exist_proof*) : code alphanumérique de la valeur du type de nomenclature PREUVE_EXIST (PreuveExistanteValue|15|page83).
- code_nomenclature_diffusion_level [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_diffusion_level*) : code alphanumérique de la valeur du type de nomenclature NIV_PRECIS (NiveauPrecisionValue|5|page64).
- code_nomenclature_observation_status [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_observation_status*) : code alphanumérique de la valeur du type de nomenclature STATUT_OBS (StatutObservationValue|18|page84).
- code_nomenclature_blurring [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_blurring*) : code alphanumérique de la valeur du type de nomenclature DEE_FLOU (dEEFloutageValue|4|page62).
- code_nomenclature_source_status [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_source_status*) : code alphanumérique de la valeur du type de nomenclature STATUT_SOURCE (StatutSourceValue|19|page85).
- code_nomenclature_behaviour [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_behaviour*) : code alphanumérique de la valeur du type de nomenclature OCC_COMPORTEMENT (OccurrenceComportementValue|110|page72).
- determiner [VARCHAR(1000)] (=*t_occurrences_occtax.determiner*) : Prénom, nom et organisme de la ou les personnes ayant réalisé la détermination taxonomique de l'observation. Ex : Jean MARTIN (CBNA).
- code_nomenclature_determination_method [VARCHAR(25)] (=*t_occurrences_occtax.id_nomenclature_determination_method*) : code alphanumérique de la valeur du type de nomenclature METH_DETERMIN (champ libre dans OccTax) (Voir les codes disponibles).
- cd_nom [INT(4)] (=*t_occurrences_occtax.cd_nom*) : code du taxon en vigueur « cd_nom » de TaxRef référençant au niveau national le taxon. Il peut être trouvé dans la colonne "CD_NOM" de TaxRef.
- **nom_cite** [VARCHAR(1000)] (=*t_occurrences_occtax.nom_cite*) : nom du taxon cité à l'origine par l'observateur. Ne pas mettre de valeur NULL dans ce champ mais une valeur vide ("").
- meta_v_taxref [varchar(50)] (=*t_occurrences_occtax.meta_v_taxref*) : numéro de version de Taxref. Ex : "Taxref V16.0".
- digital_proof [TEXT] (=*t_occurrences_occtax.digital_proof*) : si possible une URL (idéalement un permalien) vers la preuve en ligne.
- non_digital_proof [TEXT] (=*t_occurrences_occtax.non_digital_proof*) : adresse ou nom de la personne ou de l'organisme qui permettrait de retrouver la preuve non numérique de l'observation.
- comment_description [TEXT] (=*t_occurrences_occtax.comment*) : description libre de l'observation, aussi succincte et précise que possible. Informations sur le(s) individu(s) observé(s) (=quoi).
- additional_fields_occurence [JSON] : permet d'associer des champs complémentaires et/ou si toutes les entrées ne partagent pas les même champs pour l'occurence de taxon. Les valeurs

de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- unique_id_counting [UUID] : UUID du dénombrement s'il existe déjà dans les données sources.
- **code_nomenclature_life_stage** [VARCHAR(25)] (=*cor_counting_occtax.id_nomenclature_life_stage*) : code alphanumérique de la valeur du type de nomenclature STADE_VIE (OccurrenceStadeDeVieValue|10|page78).
- **code_nomenclature_sex** [VARCHAR(25)] (=*cor_counting_occtax.id_nomenclature_sex*) : code alphanumérique de la valeur du type de nomenclature SEXE (OccurrenceSexeValue|9|page78).
- **code_nomenclature_obj_count** [VARCHAR(25)] (=*cor_counting_occtax.id_nomenclature_obj_count*) : code alphanumérique de la valeur du type de nomenclature OBJ_DENBR (ObjetDenombrementValue|6|page68).
- code_nomenclature_type_count [VARCHAR(25)] (=*cor_counting_occtax.id_nomenclature_type_count*) : code alphanumérique de la valeur du type de nomenclature TYP_DENBR (TypeDenombrementValue|21|page86).
- count_min [INT(4)] (=*cor_counting_occtax.count_min*) : nombre minimum d'individus du taxon composant l'observation.
- count_max [INT(4)] (=*cor_counting_occtax.count_max*) : nombre maximum d'individus du taxon composant l'observation. Mettre la même valeur que count_min si 1 seule valeur de dénombrement.
- additional_fields_counting [JSON] : permet d'associer des champs complémentaires et/ou si toutes les entrées ne partagent pas les même champs pour le dénombrement. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL.
- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement.
- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement.
- **meta_last_action** [CHAR(1)] (=*last_action*) : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

À faire

- ☐ Faire le point sur l'ordre des champs : voir si nous ordonnons les champs suivant l'ordre hiérarchiques des tables d'OccTax.
- ☐ Faire le point sur les champs `unique_id_…`
  - ☐ Renommer ''unique_id_sinp_grp'' en ''unique_id_survey''
    - Le champ `unique_id_sinp_grp` est utilisé par le parser pour la Synthèse également, donc si on modifie ce champ pour OccTax, il faudrait le faire pour la Synthèse également pour une modification commune dans le parser.
  - ☐ [✓ choarau, 2026-01-21]~~Renommer~~ `unique_id_occurence_occtax` en `unique_id_occurrence` si c'est bien la même chose
  - ☐ Renommer `unique_id_sinp_occtax` en `unique_id_counting`
- ☐ [✓ choarau, 2026-01-21]~~Voir si on garde ou pas le champ~~ `meta_device_entry` (le garder si présent dans le standard ou important pour GN)
  - Le champ est présent dans `t_releves_occtax`. Il permet de savoir si l'info a été saisie via l'application mobile ou web.
- ☐ Le champ additional_data est utilisé pou le champ additional_fields de t_releve_occtax. Il faudrait voir si cela convient ou s'il faut en gérer d'autres pour les autres champs `additional_fields` des autres tables d'OccTax.
- ☐ Suite aux modifications précédentes tenir à jour les script SQL du dépôt sinp-shared-data

Format VALIDATION d'import

- But : Permet de fournir les informations sur les validations des observations à intégrer à la table des validations de GeoNature.
- Table GeoNature : *gn_commons.t_validations*.
- Standard : OccTax v2
- Statut : ** Bêta (fonctionnel mais des corrections peuvent subvenir) **

Description du format VALIDATION

Pour chaque ligne : `nom_du_champ [format du champ] (=table_geonature.nom_champ) : description du champ.`. Les champs **en gras** sont obligatoires. Pour les nomenclatures, le nom de la mnémonique du type dans GeoNature est indiqué en italique en fin de description. La correspondance avec le nom de cette nomenclature dans le standard est indiqué entre parenthèses et un lien pointe vers le document PDF Standards d'échanges du SINP. Pour les nomenclatures, la valeur à transmettre est celle présente dans la colonne "**Code**" du standard qui est équivalente au champ `cd_nomenclature` de la table `ref_nomenclatures.t_nomenclatures` de GeoNature.

- **unique_id_sinp** [UUID] (=*gn_commons.t_validations.uuid_attached_row*) : UUID SINP de l'observation correspondante à cette validation. Si un historique des validations doit être importé, cet UUID pourra être répété plusieurs fois mais le champ `creation_date` devra être distinct.
- **code_nomenclature_valid_status** [VARCHAR(25)] (=*gn_commons.t_validations.id_nomenclature_valid_status*) : code alphanumérique de la valeur du type de nomenclature *STATUT_VALID* (NiveauValidationValue|80|page67). Pour la validation la plus récente d'une observation, ce champ devrait contenir la même valeur que le champ `code_nomenclature_valid_status` du fichier `synthese.csv`.
- validator [VARCHAR(1000)] (=*gn_commons.t_validations.id_validator*) : personne ayant procédé à la validation (et organisme). Voir le détail du format à plat des infos sur une personne. Pour la validation la plus récente d'une observation, ce champ devrait contenir la même valeur que le champ `validator` du fichier `synthese.csv`.
- comment [TEXT] (=*gn_commons.t_validations.validation_comment*) : commentaire sur la validation. Pour la validation la plus récente d'une observation, ce champ devrait contenir la même valeur que le champ `validation_comment` du fichier `synthese.csv`.
- automatic [BOOL] (=*gn_commons.t_validations.validation_auto*) : indique si la validation a été obtenue par un processus automatique (TRUE) ou manuel (FALSE). Par défaut, la valeur TRUE sera utilisée.
- creation_date [DATE(YYYY-MM-DD HH:MM:SS)] (=*gn_commons.t_validations.validation_date*) : date et heure de validation de l'observation. Pour la validation la plus récente d'une observation, ce champ devrait contenir la même valeur que le champ `validation_date` du fichier `synthese.csv`.
- additional_data [JSON] : permet d'associer des champs complémentaires et/ou si toutes les entrées ne partagent pas les mêmes champs. Les valeurs de ce champ doivent être au format JSON et être valide. Ne pas mettre de valeur vide dans ce champ mais une valeur NULL. Il n'y a pas de champs équivalant dans la table de la base GeoNature, il n'est donc pas utilisé pour l'instant.
- **meta_create_date** [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de création de l'enregistrement. Il n'y a pas de champs équivalant dans la table de la base GeoNature, il n'est donc pas utilisé pour l'instant. Mais il peut servir au débogage.

- meta_update_date [DATE(YYYY-MM-DD HH:MM:SS)] : date et heure de mise à jour de l'enregistrement. Il n'y a pas de champs équivalant dans la table de la base GeoNature, il n'est donc pas utilisé pour l'instant. Mais il peut servir au débogage.
- **meta_last_action** [CHAR(1)] : permet d'identifier les lignes ajoutées depuis le dernier import ("*I*"), modifiées ("*U*") ou supprimées ("*D*").

Format META_ADDITIONAL_DATA d'import

- But : permet de fournir la description des champs additionnels présent dans les fichiers de données à importer.
- Standard : OccTax v2 extension *AttributAdditionel*.
- Statut : **Alpha (non utilisé ; présent uniquement pour fournir une documentation)**

Description du format META_ADDITIONAL_DATA

Actuellement, ce fichier peut être fourni pour fournir une documentation des informations stockées dans les champs `additional_data` des différents fichiers CSV. Il ne permet pas d'ajouter des colonnes aux fichiers CSV car l'ordre et le nom des colonnes doit être respectés dans les différents fichiers CSV.
Pour chaque ligne : `nom_du_champ [format du champ] (=`*`nom_attribut_standard`*`) : description du champ.`. Les champs **en gras** sont obligatoires.

- **resource** [VARCHAR(50)] : nom de la ressource (=fichier d'import) pour laquelle le champ additionnel est utilisé. Valeurs : *synthese*, *source*, *dataset*, *acquisition_framework*, *organism*, *user*.
- **name** [VARCHAR(50)] (=*nomAttribut*) : nom du champ additionnel.
- **desc** [TEXT] (=*definitionAttribut*) : définition/description du contenu du champ additionnel.
- **keyword** [VARCHAR(50)] (=*thematiqueAttribut*) : un mot-clé indiquant la catégorie/thématique du champ.
- **type** [VARCHAR(25)] (=*typeAttribut*) : indique si l'attribut additionnel est de type quantitatif (=*QTA*), qualitatif (=*QUAL*) ou inconnu (=*NSP*).
- unity [VARCHAR(50)] (=*uniteAttribut*) : unité du système international du champ additionnel (Ex. : °C, m, kg). **Obligatoire** si *type* = *QUAL*.

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

Permanent link:
https://wiki-sinp.cbn-alpin.fr/database/import-formats?rev=1787296138

Last update: **2026/08/21 07:08**