2025/12/16 04:35 1/11 exemple-import-synthese

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 fichier CSV (voir ci-dessus).
- L'import des fichiers CSV se fera à l'aide d'un script Bash exécutant des scripts SQL et Python.
 - Le 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.
 - Les scripts SQL se chargeront de désactiver triggers, contraintes... puis de les réactiver et exécuter globalement après la commande de COPY 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.
- Nous procéderons pour l'import du différentiel dans la table "gn_synthese.synthese" en 3 étapes :
 - Ajout des nouvelles observations

- o 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.
- L'ordre d'import des tables devrait être le suivant : organism, acquistion_framework, dataset, source, synthese.

Format de la table SYNTHESE d'import

Permet de fournir les informations sur les observations.

- **id_synthese** [INT(4)] : id auto-incrémenté servant de clé primaire. TODO: voir s'il est vraiment nécessaire de garder ce champ.
- 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 de la table SOURCE d'import

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

- id_source [INT(4)] : id auto-incrémenté servant de clé primaire. TODO: voir s'il est vraiment nécessaire de garder ce champ.
- **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 de la table 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.

- id_dataset [INT(4)] : id auto-incrémenté servant de clé primaire. TODO: voir s'il est vraiment nécessaire de garder ce champ.
- 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)

2025/12/16 04:35 5/11 exemple-import-synthese

- 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

"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 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 :

```
'{{'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 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'acquistion, ni des acteurs de type "personne". Si cela s'avérait nécessaire, il est possible de les ajouter manuellement une fois l'import effectué.

- id_acquisition_framework [INT(4)] : id auto-incrémenté servant de clé primaire. TODO: voir s'il est vraiment nécessaire de garder ce champ.
- unique_id [UUID] (=unique_acquisition_framework_id) UUID du cadre d'acquisition si préexistant.
- **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 de la table 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".

- id_organism [INT(4)] : id auto-incrémenté servant de clé primaire. TODO: voir s'il est vraiment nécessaire de garder ce champ.
- 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".

2025/12/16 04:35 7/11 exemple-import-synthese

- 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").

Exemple de code SQL d'import

- Ceci est un exmple pouvant servir de base au format d'import
- Le fichier SQL doit être présent en local sur le serveur où se trouve la base de données
- Nous importons les données dans un schema "**imports**" et une table "**synthese_faune**" à l'aide de la commande PSQL \copy
- La ligne de commande à utiliser :

```
psql -h localhost -d "<nom-base-de-donnees>" -U "<nom-utilisateur>" -v
ON_ERROR_STOP=1 -f ./fichier_import_utf8.sql
```

- Nous supprimons toutes les clés étrangères et les liens avec des tables extérieures pour simplifier l'import...
- [jpmilcent] Trouver un solution pour lier les nomenclatures et jeu de données : utiliser les fonctions et utiliser les valeurs de nomenclature à la place des identifiants numériques dans la table ?
- [jpmilcent] Voir comment nommer les contraintes et index pour éviter le message d'erreur indiquant qu'elles existent déjà : ajout d'un suffixe ?
- [jpmilcent] Voir si on importe avec tous les champs où seulement ceux utilisés ? : exporter avec seulement les champs nécessaire réduit la taille et rend la table plus lisible mais implique des différences dans chaque import...

```
-- PostgreSQL - Example GeoNature synthese import
-- BEGIN;

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = ON;
SET check_function_bodies = FALSE;
SET client_min_messages = warning;

CREATE SCHEMA IF NOT EXISTS imports;

SET search_path = imports, pg_catalog;
SET default_tablespace = '';
SET default_with_oids = FALSE;

DROP SEQUENCE IF EXISTS synthese_id_synthese_seq CASCADE ;
```

```
CREATE SEQUENCE synthese id synthese seq
   INCREMENT BY 1
   MINVALUE 1
   MAXVALUE 9223372036854775807
   START 1
   CACHE 1
   NO CYCLE;
DROP TABLE IF EXISTS synthese faune CASCADE;
CREATE TABLE synthese faune (
    id synthese INTEGER DEFAULT
NEXTVAL('synthese id synthese seq'::regclass) NOT NULL,
   unique id sinp uuid,
   unique_id_sinp_grp uuid,
   id source INTEGER,
    id module INTEGER,
   entity source pk value CHARACTER VARYING,
   id dataset INTEGER,
   id nomenclature geo object nature INTEGER NULL,
   id_nomenclature_grp_typ INTEGER NULL,
    id nomenclature obs meth INTEGER NULL,
    id nomenclature obs technique INTEGER NULL,
   id nomenclature bio status INTEGER NULL,
    id nomenclature bio condition INTEGER NULL,
   id_nomenclature_naturalness INTEGER NULL,
    id nomenclature exist proof INTEGER NULL,
   id nomenclature valid status INTEGER NULL,
    id nomenclature diffusion level INTEGER NULL,
   id nomenclature life stage INTEGER NULL,
    id nomenclature sex INTEGER NULL,
    id nomenclature obj count INTEGER NULL,
    id nomenclature type count INTEGER NULL,
   id nomenclature sensitivity INTEGER NULL,
    id nomenclature observation status INTEGER NULL,
    id_nomenclature_blurring INTEGER NULL,
    id nomenclature source status INTEGER NULL,
    id nomenclature info geo type INTEGER NULL,
    count min INTEGER,
    count max INTEGER,
    cd nom INTEGER,
   nom cite CHARACTER VARYING(1000) NOT NULL,
   meta v taxref CHARACTER VARYING(50) NULL,
    sample_number_proof text,
   digital proof text,
   non digital proof text,
   altitude min INTEGER,
   altitude max INTEGER,
    the geom 4326 public geometry (Geometry, 4326),
    the geom point public geometry (Point, 4326),
```

2025/12/16 04:35 9/11 exemple-import-synthese

```
the geom local public geometry (Geometry, 2154),
   date min TIMESTAMP WITHOUT TIME zone NOT NULL,
   date max TIMESTAMP WITHOUT TIME zone NOT NULL,
   validator CHARACTER VARYING(1000),
   validation comment text,
    observers CHARACTER VARYING(1000),
   determiner CHARACTER VARYING(1000),
   id digitiser INTEGER,
    id nomenclature determination method INTEGER NULL,
    comment context text,
    comment description text,
   meta validation date TIMESTAMP WITHOUT TIME zone,
   meta create date TIMESTAMP WITHOUT TIME zone DEFAULT now(),
   meta_update_date TIMESTAMP WITHOUT TIME zone DEFAULT now(),
    last action CHARACTER(1),
    CONSTRAINT check synthese altitude max CHECK ((altitude max >=
altitude min)),
   CONSTRAINT check synthese count max CHECK ((count max >= count min)),
   CONSTRAINT check synthese date max CHECK ((date max >= date min)),
    CONSTRAINT enforce dims the geom 4326 CHECK
((public.st ndims(the geom 4326) = 2)),
   CONSTRAINT enforce dims the geom local CHECK
((public.st_ndims(the_geom_local) = 2)),
    CONSTRAINT enforce dims the geom point CHECK
((public.st ndims(the geom point) = 2)),
    CONSTRAINT enforce geotype the geom point CHECK
(((public geometrytype(the geom point) = 'POINT'::text) OR (the geom point
IS NULL))),
   CONSTRAINT enforce_srid_the_geom_4326 CHECK
((public.st srid(the geom 4326) = 4326)),
    CONSTRAINT enforce srid the geom local CHECK
((public.st_srid(the_geom_local) = 2154)),
    CONSTRAINT enforce srid the geom point CHECK
((public.st srid(the geom point) = 4326))
);
\copy synthese_faune (id_synthese, unique_id_sinp, unique_id_sinp_grp,
id source, id module, entity source pk value, id dataset,
id nomenclature geo object nature, id nomenclature grp typ,
id nomenclature obs meth, id nomenclature obs technique,
id nomenclature bio status, id nomenclature bio condition,
id nomenclature naturalness, id nomenclature exist proof,
id_nomenclature_valid_status, id_nomenclature_diffusion_level,
id nomenclature life stage, id nomenclature sex, id nomenclature obj count,
id nomenclature type count, id nomenclature sensitivity,
id nomenclature_observation_status, id_nomenclature_blurring,
id nomenclature source status, id nomenclature info geo type, count min,
count max, cd nom, nom cite, meta v taxref, sample number proof,
digital_proof, non_digital_proof, altitude_min, altitude_max, the_geom_4326,
the geom point, the geom local, date min, date max, validator,
```

```
validation comment, observers, determiner, id digitiser,
id nomenclature determination method, comment context, comment description,
meta validation date, meta create_date, meta_update_date, last_action) FROM
stdin WITH NULL '\N';
24977967
           \N
                 \N
                             \N
                                   96495756
                                                3
                                                     \N
                                                           \N
                                                                  \N
                        1
                                                                        \ N
\N
      \N
            \ N
                  \N
                         \N
                               \N
                                     \N
                                            \N
                                                  \N
                                                        \N
                                                               \ N
                                                                     \N
                                                                           \N
\N
      \N
                       3297
                               LARFUS
                                          \N
                                                \N
                                                      \N
                                                            \N
                                                                   0
                                                                        0
0101000020E61000004777103B53881240ADFA5C6DC5D64540
0101000020E61000004777103B53881240ADFA5C6DC5D64540
01010000206A080000B81E85EBCB61294148E17AA487FC5741
                                                        1994-01-09 00:00:00
1994-01-09 00:00:00
                                 KAYSER Yves
                                                      \N
                                                            \N
                                                                   \N
                                                                         ad ;
          2016-10-18 00:00:00
                                  2020-02-12 14:04:42.735559
fuscus
                                                                  \ N
                                                                        Ι
24977968
                             \N
                                   96595425
                                                3
           \N
                 \N
                        1
                                                     \N
                                                           \ N
                                                                  \N
                                                                        \ N
\ N
      \ N
            \N
                  \ N
                         \N
                               \N
                                     \ N
                                            \N
                                                  \N
                                                              \ N
                                                                     \ N
                                                                           \N
                                                        \N
\N
      \ N
                       3297
                               LARFUS
                                          \N
                                                      \ N
                                                            \N
0101000020E61000004777103B53881240ADFA5C6DC5D64540
0101000020E61000004777103B53881240ADFA5C6DC5D64540
01010000206A080000B81E85EBCB61294148E17AA487FC5741
                                                        1994-06-24 00:00:00
1994-06-24 00:00:00
                                 KAYSER Yves
                                                      \N
                                                            \N
                                                                   \N
                                                                         juv
2016-03-01 00:00:00
                        2020-02-12 14:04:42.735559
                                                       \ N
                                                             Ι
١.
ALTER TABLE ONLY synthese faune
    ADD CONSTRAINT pk synthese PRIMARY KEY (id synthese);
ALTER TABLE ONLY synthese faune
    ADD CONSTRAINT unique id sinp unique UNIQUE (unique id sinp);
CREATE INDEX i synthese altitude max ON synthese faune USING btree
(altitude max);
CREATE INDEX i synthese altitude min ON synthese faune USING btree
(altitude min);
CREATE INDEX i synthese cd nom ON synthese faune USING btree (cd nom);
CREATE INDEX i synthese date max ON synthese faune USING btree (date max
DESC);
CREATE INDEX i synthese date min ON synthese faune USING btree (date min
DESC);
CREATE INDEX i synthese id dataset ON synthese faune USING btree
(id dataset);
CREATE INDEX i synthese t sources ON synthese faune USING btree (id source);
CREATE INDEX i synthese the geom 4326 ON synthese faune USING gist
(the geom 4326);
```

2025/12/16 04:35 11/11 exemple-import-synthese

```
CREATE INDEX i_synthese_the_geom_local ON synthese_faune USING gist
  (the_geom_local);

CREATE INDEX i_synthese_the_geom_point ON synthese_faune USING gist
  (the_geom_point);

COMMIT;
```

From:

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

Permanent link:

https://sinp-wiki.cbn-alpin.fr/database/exemple-import-synthese?rev=1596704328



