Accéder aux données agrégées à partir d'un dump du hub national

L'accès aux données partagées sur le serveur de la FCBN étant parfois limité/limitant pour des raisons de qualité de la connexion, il est possible de faire à la FCBN une demande de dump de la base nationale pour en disposer en local (dans le cadre du réseau des CBN uniquement, à l'exclusion des données précises).

Cela a été fait pour le CBN de Bailleul en mars 2017.

Procédure d'installation

Création d'une bdd si_flore_national en local

```
CREATE DATABASE si_flore_national
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       CONNECTION LIMIT = -1;

ALTER DATABASE si_flore_national
  SET search_path = "$user", public, topology;
GRANT CONNECT, TEMPORARY ON DATABASE si_flore_national TO public;
GRANT ALL ON DATABASE si_flore_national TO postgres;


CREATE SCHEMA ref;
CREATE SCHEMA agregation;
CREATE EXTENSION postgis;
```

Restauration à partir de backup totaux

- Restauration du schéma ref (siflore_ref.backup);
- Restauration du schéma agregation (siflore_agregation.backup);

On peut le faire directement sous pgadmin avec un clic droit sur la base de données (“restaurer”).

Restauration à partir de fichiers plats sql

```
--SOUS WINDOWS (en ligne de commande cmd)
cd c:\Program Files\PostgreSQL\9.2\bin
psql -U postgres  -p 5432 -d si_flore_national -c "CREATE EXTENSION
postgis;"
```

```
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_agregation_table_observation.sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_agregation_table_releve.sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_agregation_table_releve_territoire.
sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_agregation_table_releve_acteur.sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_agregation_table_entite_metadonnee.
sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_ref_table_taxref_et_autre.sql
psql -U postgres -p 5432 si_flore_national
<"F:\dump_agregation_alexis_desse\schema_ref_table_geo.sql
```

Exploitation des données (au choix 1 ou 1bis)

0-CREATION DES FONCTIONS POUR LES LISTES DE TAXONS

```
--Ou alors charger l'intégrité des fonctions contenues sur cette page
https://github.com/fedecbn/hub.sql/blob/master/hub.sql


--- Initiations de tables indispensables pour le fonctionement du hub
CREATE TABLE IF NOT EXISTS public.zz_log (lib_schema character
varying,lib_table character varying,lib_champ character varying,typ_log
character varying,lib_log character varying,nb_occurence character
varying,date_log timestamp,user_log varchar);
CREATE TABLE IF NOT EXISTS public.bilan (uid integer NOT NULL,lib_cbn
character varying,data_nb_releve integer,data_nb_observation
integer,data_nb_taxon integer,taxa_nb_taxon integer,temp_data_nb_releve
integer,temp_data_nb_observation integer,temp_data_nb_taxon
integer,temp_taxa_nb_taxon integer,derniere_action character varying,
date_derniere_action date,CONSTRAINT bilan_pkey PRIMARY KEY (uid));
DROP TABLE IF EXISTS twocol CASCADE;    CREATE TABLE public.twocol (col1
varchar, col2 varchar);
DROP TABLE IF EXISTS threecol CASCADE; CREATE TABLE public.threecol (col1
varchar, col2 varchar, col3 varchar);

------------------------------------------------------------------------
----------------------------
------------------------------------------------------------------------
----------------------------
--- Nom : hub_import_taxon
--- Description : Importer une liste de taxon dans un hub
------------------------------------------------------------------------
```

```
-----------------------------
-------------------------------------------------------------------------------
-----------------------------
CREATE OR REPLACE FUNCTION hub_import_taxon(libSchema varchar, path varchar,
files varchar) RETURNS setof zz_log AS
$BODY$
DECLARE out zz_log%rowtype;
BEGIN
--- Commande
CASE WHEN files <> '' THEN
    EXECUTE 'TRUNCATE TABLE "'||libSchema||'".zz_log_liste_taxon; TRUNCATE
TABLE "'||libSchema||'".zz_log_liste_taxon_et_infra;';
    EXECUTE 'COPY "'||libSchema||'".zz_log_liste_taxon FROM
'''||path||files||''' HEADER CSV DELIMITER '';'' ENCODING ''UTF8'';';
    out.lib_log := files||' importé depuis '||path;
ELSE out.lib_log := 'Paramètre "files" incorrect'; END CASE;

--- Output&Log
out.lib_schema := libSchema;out.lib_champ := '-';out.lib_table :=
'zz_log_liste_taxon';out.typ_log := 'hub_import_taxon';out.nb_occurence :=
1; SELECT CURRENT_TIMESTAMP INTO out.date_log;out.user_log :=
current_user;PERFORM hub_log (libSchema, out);RETURN next out;
END; $BODY$  LANGUAGE plpgsql;

-------------------------------------------------------------------------------
-----------------------------
-------------------------------------------------------------------------------
-----------------------------
--- Nom : hub_txinfra
--- Description : Générer une table avec les taxon infra depuis la table
zz_log_liste_taxon
-------------------------------------------------------------------------------
-----------------------------
-------------------------------------------------------------------------------
-----------------------------
CREATE OR REPLACE FUNCTION hub_txinfra(libSchema varchar, version_taxref
integer = 7) RETURNS setof zz_log AS
$BODY$
DECLARE out zz_log%rowtype;
DECLARE i varchar;
BEGIN
--- Commande
FOR i in EXECUTE 'select cd_ref from "'||libSchema||'".zz_log_liste_taxon'
    LOOP
    EXECUTE
        'INSERT INTO "'||libSchema||'".zz_log_liste_taxon_et_infra
(cd_ref_demande, nom_valide_demande, cd_ref_cite,
nom_complet_cite,cd_taxsup_cite,rang_cite)
        select '''||i||''' as cd_ref_demande, '''' as nom_valide_demande,
foo.* from
        (WITH RECURSIVE hierarchie(cd_nom,nom_complet, cd_taxsup, rang) AS (
```

```
        SELECT cd_nom, nom_complet, cd_taxsup, rang
        FROM ref.taxref_v'||version_taxref||' t1
        WHERE t1.cd_nom = '''||i||''' AND t1.cd_nom = t1.cd_ref
        UNION
        SELECT t2.cd_nom, t2.nom_complet, t2.cd_taxsup, t2.rang
        FROM ref.taxref_v'||version_taxref||' t2
        JOIN hierarchie h ON t2.cd_taxsup = h.cd_nom
        WHERE t2.cd_nom = t2.cd_ref
        ) SELECT * FROM hierarchie) as foo';
    end loop;
EXECUTE 'update  "'||libSchema||'".zz_log_liste_taxon_et_infra set
nom_valide_demande = nom_valide from "'||libSchema||'".zz_log_liste_taxon
where zz_log_liste_taxon_et_infra.cd_ref_demande= zz_log_liste_taxon.cd_ref
' ;
out.lib_log := 'Liste de sous taxons générée';

--- Output&Log
out.lib_schema := libSchema;out.lib_champ := '-';out.lib_table :=
'zz_log_liste_taxon_et_infra';out.typ_log := 'hub_txinfra';out.nb_occurence
:= 1; SELECT CURRENT_TIMESTAMP INTO out.date_log;out.user_log :=
current_user;PERFORM hub_log (libSchema, out);RETURN next out;
END; $BODY$  LANGUAGE plpgsql;

------------------------------------------------------------------------------
----------------------------
------------------------------------------------------------------------------
----------------------------
--- Nom : hub_log
--- Description : ecrit les output dans le Log du schema et le log global
------------------------------------------------------------------------------
----------------------------
------------------------------------------------------------------------------
----------------------------
CREATE OR REPLACE FUNCTION hub_log (libSchema varchar, outp zz_log, action
varchar = 'write') RETURNS void AS
$BODY$
DECLARE exist integer;
BEGIN

/*ajout du user_log dans le zzlog*/
EXECUTE 'SELECT 1 FROM information_schema.columns WHERE table_schema =
'''||libSchema||''' AND table_name = ''zz_log'' AND column_name =
''user_log'';' INTO exist;
CASE WHEN exist IS NULL THEN
    EXECUTE 'ALTER TABLE '||libSchema||'.zz_log add column user_log
varchar;';
ELSE END CASE;

CASE WHEN action = 'write' THEN
    EXECUTE 'INSERT INTO "'||libSchema||'".zz_log
(lib_schema,lib_table,lib_champ,typ_log,lib_log,nb_occurence,date_log,user_l
```

```
og) VALUES
(''||outp.lib_schema||''','''||outp.lib_table||''','''||outp.lib_champ||'''
,'''||outp.typ_log||''','''||outp.lib_log||''','''||outp.nb_occurence||''','
''||outp.date_log||''','''||outp.user_log||''');';
    CASE WHEN libSchema <> 'public' THEN EXECUTE 'INSERT INTO
"public".zz_log
(lib_schema,lib_table,lib_champ,typ_log,lib_log,nb_occurence,date_log,user_l
og) VALUES
(''||outp.lib_schema||''','''||outp.lib_table||''','''||outp.lib_champ||'''
,'''||outp.typ_log||''','''||outp.lib_log||''','''||outp.nb_occurence||''','
''||outp.date_log||''','''||outp.user_log||''');';
    ELSE PERFORM 1; END CASE;
WHEN action = 'clear' THEN
    EXECUTE 'TRUNCATE "'||libSchema||'".zz_log';
ELSE SELECT 1;
END CASE;
PERFORM hub_mail(outp.lib_schema,outp.typ_log,outp.date_log,outp.user_log);
END;$BODY$ LANGUAGE plpgsql;




------------------------------------------------------------------------------
------------------------------
------------------------------------------------------------------------------
------------------------------
--- Nom : hub_mail
--- Description : Renseigne la table emailing_queue - infos transmises par
mail à l'admin et aux cbn
------------------------------------------------------------------------------
------------------------------
------------------------------------------------------------------------------
------------------------------
CREATE OR REPLACE FUNCTION hub_mail (libSchema varchar,action
varchar,date_log timestamp with time zone,user_log name) RETURNS void AS
$BODY$
BEGIN
CASE WHEN action = 'hub_export' OR action = 'hub_push' OR action =
'hub_connect' OR action = 'hub_publicate' OR action = 'hub_import' OR action
= 'siflore_data_refresh' THEN
    INSERT INTO emailing_queue (lib_schema, action, date_log, user_log)
VALUES (libSchema,action,date_log,user_log);
ELSE END CASE;
END;$BODY$ LANGUAGE plpgsql;
```

1- EXTRACTION POUR SEULEMENT QUELQUES ESPECES

```
    --On vide les listes
    TRUNCATE agregation.zz_log_liste_taxon;
    TRUNCATE agregation.zz_log_liste_taxon_et_infra;
```

```
    -- On recréé une liste de taxons pour lesquelles on veut des données
    INSERT INTO agregation.zz_log_liste_taxon (cd_ref,nom_valide)
    select cd_ref, nom_valide from ref.taxref_v7 where cd_nom in
('85474','610664','95829','160257','103139','104805','106742','106748','6106
02','109141','446978','161449');

    --Construction de la liste des infra-taxons
    SELECT * FROM hub_txinfra('agregation',7);
```

1 bis-EXTRACTION à PARTIR D'UNE LISTE CSV

```
    --On vide les listes précédentes
    TRUNCATE agregation.zz_log_liste_taxon;
    TRUNCATE agregation.zz_log_liste_taxon_et_infra;

    ----Import de la liste des taxons pour lesquelles on veut des données
    COPY agregation.zz_log_liste_taxon FROM
'F:\dump_agregation_alexis_desse\170323_liste_taxons_v7.csv' WITH csv header
delimiter ';' encoding 'LATIN1';
    --ou alors (attention la fonction est en encodage UTF8', delimiter ';')
    -- select * from hub_import_taxon('agregation', '/home/export_pgsql/',
'170323_liste_taxons_v7.csv')

    --- Construction de la liste des taxons et leurs infra taxons
    SELECT * FROM hub_txinfra('agregation',7);
```

2-DONNEES MANQUANTES?

```
---Vérification des taxons non présents dans taxref v7 et export de la liste
de ces taxons dans un fichier texte
    COPY (
    SELECT toto.* FROM
            (SELECT zz.cd_ref AS code_initial, zz.nom_valide AS nom_initial,
tx.cd_nom AS cdnom_v7, tx.cd_ref AS cdref_v7, tx.nom_complet AS
nom_complet_v7, tx.nom_valide AS nom_valide_v7
            FROM ref.taxref_v7 AS tx
            RIGHT JOIN agregation.zz_log_liste_taxon AS zz ON tx.cd_ref =
zz.cd_ref
            WHERE tx.cd_ref is null) AS toto
        LEFT join agregation.zz_log_liste_taxon ls
        ON toto.code_initial= ls.cd_ref
        ORDER BY toto.nom_initial
    ) TO
'F:\dump_agregation_alexis_desse\170323_liste_espece_non_taxrefv7.csv' WITH
csv header delimiter ';' encoding 'LATIN1';
```

3-RECUPERATION DE TOUTES LES OBSERVATIONS POUR LA LISTE DEMANDEE

```
    ---Récuperation des observations par maille 10 pour tous les taxons de
la liste + infra (si par maille 5 alors typ_geo='m5', si par commune alors
typ_geo='com'
    DROP TABLE IF EXISTS obs_maille10_taxons_demande ;
    CREATE TABLE obs_maille10_taxons_demande
    AS
    SELECT t2.*, rel.date_debut, rel.date_fin
    FROM
        (
        SELECT t1.cd_jdd, t1.cd_releve,
t1.cd_obs_mere,tx.cd_ref_demande,tx.nom_valide_demande,
t1.cd_ref,t1.nom_ent_ref, t1.cd_geo, t1.lib_geo
        FROM
            (SELECT obs.*, cd_geo, lib_geo
            FROM agregation.observation obs
            LEFT JOIN agregation.releve_territoire relt
            ON obs.cd_jdd=relt.cd_jdd AND obs.cd_releve=relt.cd_releve
            WHERE typ_geo='m10') t1
            JOIN agregation.zz_log_liste_taxon_et_infra tx
            ON t1.cd_ref = tx.cd_ref_cite
        ) t2
    LEFT JOIN agregation.releve rel
    ON t2.cd_jdd=rel.cd_jdd AND t2.cd_releve=rel.cd_releve
    ORDER BY cd_ref_demande ASC;
    --remarque: si pour un seul taxon ajouter "WHERE cd_ref_demande =
'102206'" avant le "order by"
```

4-CALCUL DE SYNTHESES POUR LA LISTE DEMANDEE

```
    --Nombre de mailles par taxon après 1990
    --A partir de la table des observations précédemment calculée faire:

    DROP TABLE IF EXISTS nbr_maille10_taxons_demande ;
    CREATE TABLE nbr_maille10_taxons_demande
    AS
    SELECT cd_ref_demande,nom_valide_demande, count (distinct cd_geo)
    FROM obs_maille10_taxons_demande
    WHERE date_debut>'1989-12-31' and date_fin>'1989-12-31'
    GROUP BY cd_ref_demande,nom_valide_demande
```

```
    --Carte de répartition des taxons
    --A partir de la table des observations précédemment calculée faire:

    DROP TABLE IF EXISTS repartition_maille10_taxons_demande ;
    CREATE TABLE repartition_maille10_taxons_demande
    AS
    SELECT cd_ref_demande,nom_valide_demande, cd_geo, max (date_fin), geom
```

```
    FROM obs_maille10_taxons_demande
    INNER JOIN ref.geo_maille10 on cd_geo=cd_sig
    WHERE date_debut>'1989-12-31' and date_fin>'1989-12-31'
    GROUP BY cd_ref_demande,nom_valide_demande, cd_geo, geom;

--      Pour voir le résultat
--  select * from repartition_maille10_taxons_demande;
--      Ou alors visualiser dans QGIS en utilisant un connecteur à la base
de données postgis
```

```
    ---Nombre d'observations par maille 10kmx10km toutes dates confondues

copy (
    SELECT i.nom_valide_demande, i.cd_ref_demande,e.cd_geo,
count(a.cd_obs_mere) as nb_obs, min(z.date_debut) as date_debut_min,
max(z.date_fin) as date_fin_max
    FROM agregation.observation a
    JOIN agregation.zz_log_liste_taxon_et_infra i on a.cd_ref =
i.cd_ref_cite
    JOIN agregation.releve z ON a.cd_releve = z.cd_releve AND a.cd_jdd =
z.cd_jdd
    JOIN agregation.releve_territoire e ON a.cd_releve = e.cd_releve AND
a.cd_jdd = e.cd_jdd
    JOIN agregation.releve_acteur r ON a.cd_releve = r.cd_releve AND
a.cd_jdd = r.cd_jdd
    WHERE typ_geo = 'm10'
    GROUP BY cd_geo, i.nom_valide_demande, i.cd_ref_demande)
    to  'F:\dump_agregation_alexis_desse\20170323_nbre_obs_par_maille10.csv'
CSV HEADER DELIMITER ';' encoding 'UTF8';
```

```
    ---Nombre d'observations par maille 5kmx5km toutes dates confondues

    copy (
    SELECT i.nom_valide_demande, i.cd_ref_demande,e.cd_geo,
count(a.cd_obs_mere) as nb_obs, min(z.date_debut) as date_debut_min,
max(z.date_fin) as date_fin_max
    FROM agregation.observation a
    JOIN agregation.zz_log_liste_taxon_et_infra i on a.cd_ref =
i.cd_ref_cite
    JOIN agregation.releve z ON a.cd_releve = z.cd_releve AND a.cd_jdd =
z.cd_jdd
    JOIN agregation.releve_territoire e ON a.cd_releve = e.cd_releve AND
a.cd_jdd = e.cd_jdd
    JOIN agregation.releve_acteur r ON a.cd_releve = r.cd_releve AND
a.cd_jdd = r.cd_jdd
    WHERE typ_geo = 'm5'
    GROUP BY cd_geo, i.nom_valide_demande, i.cd_ref_demande)
    to  'F:\dump_agregation_alexis_desse\20170323_nbre_obs_par_maille5.csv'
CSV HEADER DELIMITER ';' encoding 'UTF8';
```