– Base de données Renards –
Du labo au terrain au labo

Nicolas Casajus

30 mars 2018

Table des matières

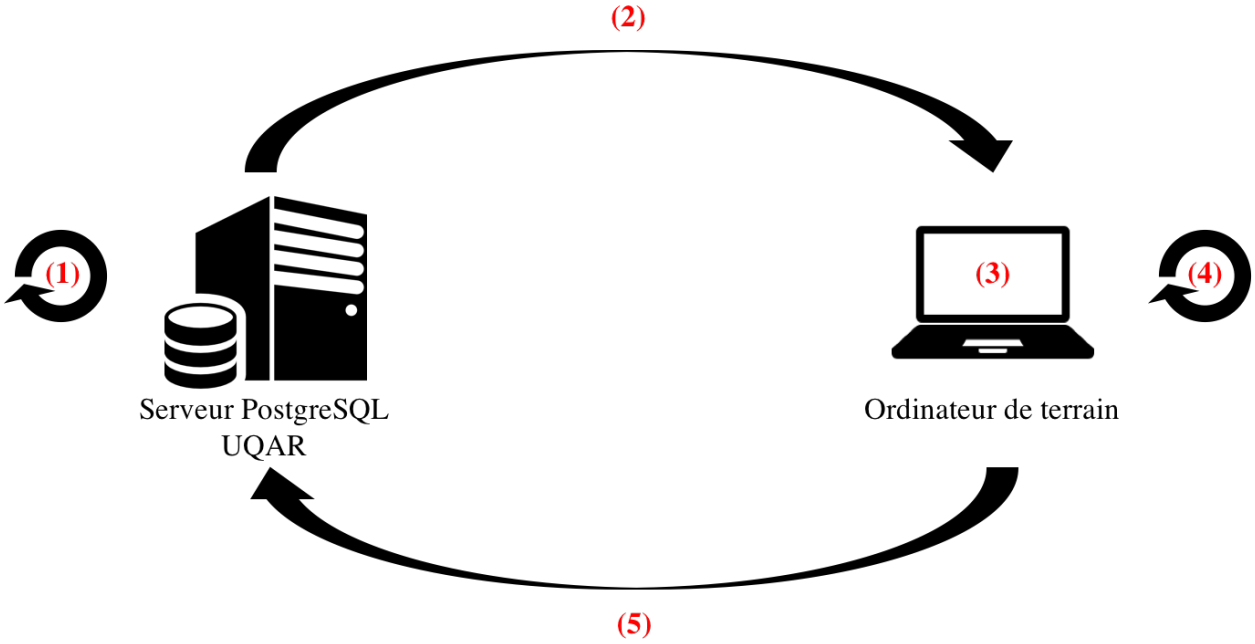

FIGURE 1 – Base de données Renards - Flux de travail

1 Serveur PostgreSQL de l'UQAR

Les données Renards sont organisées dans une base de données relationnelle PostgreSQL. Celle-ci est archivée sur le serveur PostgreSQL de l'UQAR administré par James Caveen (james_caveen@uqar.ca, bureau O-242). En tant qu'utilisateurs privilégiés, nous pouvons 1) nous connecter et 2) modifier le contenu de cette base de données. Par exemple, nous sommes autorisés à créer des (nouvelles) tables, des liens entre tables, à ajouter/supprimer de données/colonnes, etc. Cependant, nous ne pouvons ni renommer ni supprimer la base de données. Pour être plus précis, nous ne pouvons pas supprimer l'emplacement de la base de données mais nous pouvons supprimer son contenu (tables et données). Nous verrons plus loin que ceci a son importance.

Concernant la connexion au serveur de l'UQAR, deux choses importantes sont à savoir:

1. La connexion au serveur doit **impérativement** se faire depuis les bureaux du C-405. Nous n'avons l'autorisation de nous connecter au serveur que depuis certaines adresses IP (spécifiques à ces bureaux). Ainsi, il sera **impossible** de s'y connecter ailleurs depuis l'UQAR et surtout ailleurs dans le monde (donc de chez vous).
2. La connexion au serveur doit **impérativement** se faire en câble Ethernet et le Wi-Fi doit **obligatoirement** être coupé. Ceci est en lien avec les adresses IP autorisées.

Le Tableau 1 indique les paramètres de connexion au serveur PostgreSQL de l'**UQAR**.

TABLE 1: Paramètres de connexion au serveur PostgreSQL (UQAR)

Paramètre de connexion	**Valeur**
Hôte	srbd04.uqar.ca
Port	5432
Identifiant	casani01 (ou boldel01)
Mot de passe	2bchanged
Nom de la base de données	berteaux_renards_bylot

– IMPORTANT –

La base de données Renards est sauvegardée tous les jours sur le serveur de l'UQAR et les backups sont conservés pendant deux ans. Ainsi, si une erreur se produit, il sera toujours possible de revenir à une version précédente en contactant James Caveen.

La base de données Renards hébergée sur le serveur de l'UQAR est LA VERSION la plus à jour. Toutes les autres versions (ordinateurs personnels ou de terrain) ne sont que des copies.

2 Avant le terrain – Transfert de la base de données

La saisie de nouvelles données se faisant sur le terrain, il faut donc avoir une copie de la base de données sur l'ordinateur de terrain (un seul ordinateur dédié à la saisie). Ainsi, **avant de partir sur le terrain** (et depuis les bureaux du C-405), il faut installer une copie de la base de données sur l'ordinateur de terrain. Deux cas de figures peuvent se présenter.

— soit la base de données Renards n'a jamais été installée sur l'ordinateur de terrain (nouvel ordinateur ou disque dur formaté) [commencez à l'étape 2 ci-dessous],
— soit une version précédente de la base de données existe déjà sur l'ordinateur de terrain [commencez à l'étape 1 ci-dessous].

Pour installer une copie de la base de données sur l'ordinateur de terrain, ouvrez un invite de commandes (ou terminal) en cliquant sur le Bouton DÉMARRER et en recherchant **CMD**. Une fenêtre noire apparait: c'est

le terminal. Vous allez devoir y écrire les lignes de code ci-après (à l'exception de celles précédées d'un **#**). Mais, avant d'aller plus loin, voici les paramètres de connexion à PostgreSQL sur l'**ordinateur de terrain (Fox04)**.

TABLE 2: Paramètres de connexion à PostgreSQL (ordinateur)

Paramètre de connexion	**Valeur**
Identifiant	postgres
Mot de passe	12345
Nom de la base de données	berteaux_renards_bylot

Nous sommes fin prêts à **installer une copie de la base de données sur l'ordinateur de terrain**.

1. Si l'ordinateur de terrain contient une version précédente de la base de données, il faut au préalable la supprimer (ignorez cette étape si aucune version n'existe sur l'ordinateur)

```
# ETAPE 1 - Suppression d'une ancienne version de la base de données (si nécessaire)

dropdb -U postgres berteaux_renards_bylot

# Note: Remplacez 'postgres' par votre identifiant PostgreSQL
#       Pour l'ordinateur de terrain Fox 04, c'est 'postgres'
```

2. Ensuite, il faut se connecter au serveur de l'UQAR et exporter une copie de la base de données (`dump`). Pour cela, installez-vous dans un bureau du C-405 en filaire (désactivez le Wi-Fi) et tapez:

```
# ETAPE 2a - On se place sur le bureau de l'ordinateur de terrain (cd = change directory)

cd C:/Users/admin/Desktop

# Note: Remplacez 'admin' par votre nom d'utilisateur de l'ordinateur
#       Pour l'ordinateur de terrain Fox 04, c'est 'admin'


# ETAPE 2b - Exportation d'une copie de la base de données (depuis le serveur)

pg_dump -U casani01^
        -h srbd04.uqar.ca^
        -d berteaux_renards_bylot > berteaux_renards_bylot_yyyymmdd.sql

# Note: Un fichier SQL (copie de la base de données) a été créé sur le bureau
#       Adaptez 'yyyymmdd', mais ne modifiez rien d'autre
```

3. La dernière étape consiste à injecter cette copie de la base de données (fichier SQL) dans le système PostgreSQL de l'ordinateur de terrain. On procèdera en deux temps:

```
# ETAPE 3a - Création de l'emplacement de la base de données dans PostgreSQL

createdb -U postgres berteaux_renards_bylot

# Note: Remplacez 'postgres' par votre identifiant PostgreSQL
#       Pour l'ordinateur de terrain Fox 04, c'est 'postgres'
```

```
# ETAPE 3b - Injection de la copie dans l'emplacement créé

psql -U postgres^
     -d berteaux_renards_bylot < berteaux_renards_bylot_yyyymmdd.sql

# Note: Adaptez 'yyyymmdd', mais ne modifiez rien d'autre
```

Vous devez voir des lignes de code défiler: c'est que tout va bien. Il se peut que des erreurs apparaissent (`ERROR le rôle casani01 n'existe pas...` ou quelque chose du genre), ne n'est grave. Pourquoi? Simplement, parce que le fichier SQL (qui contient à la fois les instructions pour construire les tables ainsi que les données) est lu de manière séquentielle par PostgreSQL en suivant ces trois étapes:

i. Construction de la structure des tables
ii. Ajout des données
iii. Modification du propriétaire

C'est cette troisième étape qui génère une erreur, car le rôle (utilisateur) *casani01* n'existe pas sur l'ordinateur de terrain. Donc, le propriétaire de la base de données sur l'ordinateur reste **postgres**. Ainsi, vu que PostgreSQL fonctionne de manière séquentielle, les données ont bien été injectées.

REMARQUE: Vous pouvez suivre la procédure expliquée dans cette section pour installer une copie de la base de données sur votre propre ordinateur personnel.

3 Sur le terrain – Saisie des données

Afin de faciliter la saisie des données, une interface graphique a été développée sous Microsoft Access (fichier appelé **berteaux_renards_bylot.accdb**). Des formulaires de saisie ont été créés pour les principales tables de la base de données. Ainsi, tout nouvel enregistrement sera édité à l'aide de ces formulaires, même s'il est également possible de procéder à la saisie directement dans les tables (ce qui peut s'avérer dangereux en modifiant par inadvertance d'anciens enregistrements).

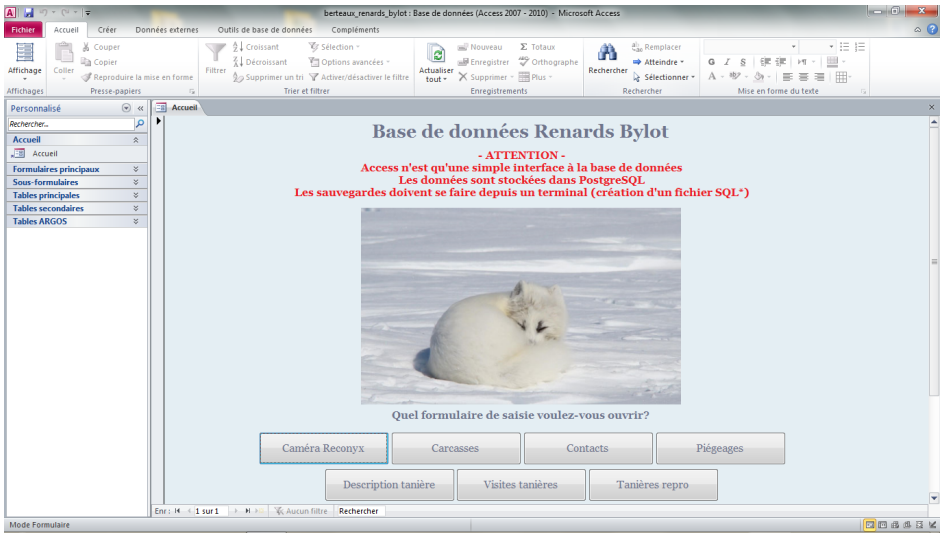

FIGURE 2 – Page d'accueil de l'interface de saisie de données

Le fichier Access est configuré pour reconnaitre et se connecter automatiquement à la base de données PostgreSQL. Seul prérequis: lorsque vous installez la base de données PostgreSQL sur l'ordinateur de terrain (procédure de la section précédente), celle-ci doit **impérativement** se nommer: **berteaux_renards_bylot**.

Et n'oubliez pas: le fichier Access n'est qu'une simple interface et **ne contient aucune donnée**.

4 Sur le terrain – Sauvegardes

Ce qui nous amène à la question des sauvegardes. Comment sauvegarder les données sur le terrain sur une base régulière? Si vous avez bien suivi, normalement vous avez la réponse. Il suffit d'adapter l'étape 2 de la procédure précédente pour effectuer un `dump`, non pas depuis le serveur de l'UQAR, mais depuis l'ordinateur de terrain. Ainsi, vous devrez ouvrir l'invite de commandes et entrer les lignes suivantes:

```
# On se place sur le bureau de l'ordinateur de terrain

cd C:/Users/admin/Desktop

# On exporte une copie de la base de données

pg_dump -U postgres -d berteaux_renards_bylot > berteaux_renards_bylot_yyyymmdd.sql
```

Ce n'est pas plus compliqué que ça. Un conseil: copiez le fichier SQL généré sur un disque dur externe. Gardez à l'esprit le proverbe *Ne pas mettre tous ses oeufs dans le même panier* qu'on pourrait traduire par *Ne pas mettre toutes ses données sur le même support.*

5 Après le terrain – Archivage des données

Lorsque toutes les données ont été saisies, il est important de procéder à l'archivage de la base de données sur le serveur PostgreSQL de l'UQAR. Cependant, nous allons devons contourner deux difficultés:

1. Comme les deux bases de données (celle du serveur et celle de l'ordinateur de terrain) contiennent des données similaires (celles des années passées), nous ne pouvons pas injecter directement la nouvelle base de données sur le serveur. Une solution, fastidieuse et requérant des connaissances en SQL, serait de sélectionner seulement les nouvelles données et de les injecter dans la base de données du serveur. Cependant, nous allons opter pour une solution plus simple et plus radicale: supprimer la base de données du serveur et la remplacer par l'intégralité de la nouvelle base de données. Ce qui nous amène à la seconde difficulté.
2. Nous ne sommes pas **Super administrateur** de la base de données du serveur, ce qui veut dire que nous ne pouvons pas la supprimer. Mais, nous pouvons supprimer son contenu. Faisons le parallèle avec un fichier Word: nous ne pouvons pas supprimer le fichier Word du disque dur, mais nous pouvons effacer son contenu. C'est la même chose ici.

Cependant, comme la base de données est construite sur le modèle relationnel, plusieurs tables sont liées entre elles. Ce qui va poser problème ici, concerne les relations *un à plusieurs*. Prenons par ex. les tables **Identité Renards** et **Contacts**. La première recense tous les individus capturés depuis le début du suivi du renard arctique à Bylot. Chaque ligne est un renard unique. La seconde contient l'ensemble des contacts avec les renards (captures, observations visuelles, caméras, etc.). Dans cette table, une ligne correspond à un contact d'un individu, mais un individu a pu être observé plusieurs fois (plusieurs contacts). Ainsi, on a une relation *un* (table **Identité Renards**) à *plusieurs* (table **Contacts**) basée sur le code couleur des individus. Ceci se traduit par le fait que la saisie du contact d'un nouvel individu ne pourra se faire dans la table **Contacts** que si cet individu est présent dans la table **Identité Renards**. Donc, pour supprimer ces tables, il faudra le faire dans l'ordre suivant: table **Contacts**, puis table **Identité Renards**.

La Figure 3 représente la base de données Renards en mode Entité-Relation: chaque ligne est un lien entre deux tables. Il y en a beaucoup... Donc trouver l'ordre de suppression des 60 tables peut être laborieux. C'est pourquoi un script R (nommé `kill-database.R`) est disponible pour exécuter cette tâche. Ce script peut être téléchargé à l'adresse https://github.com/chaireBioNorth/arcticfox-tools [1]

Ce script liste l'ensemble des 60 tables présentes dans la base de données dans leur ordre de suppression et envoie au serveur PostgreSQL l'instruction de les supprimer (`DROP TABLE` en SQL). Vous n'aurez qu'à indiquer dans le script les paramètres de connexion au serveur de l'UQAR [2] et coller l'ensemble des lignes de code dans la console R. À la fin de l'exécution du script, le contenu de la base de données du serveur aura été supprimé.

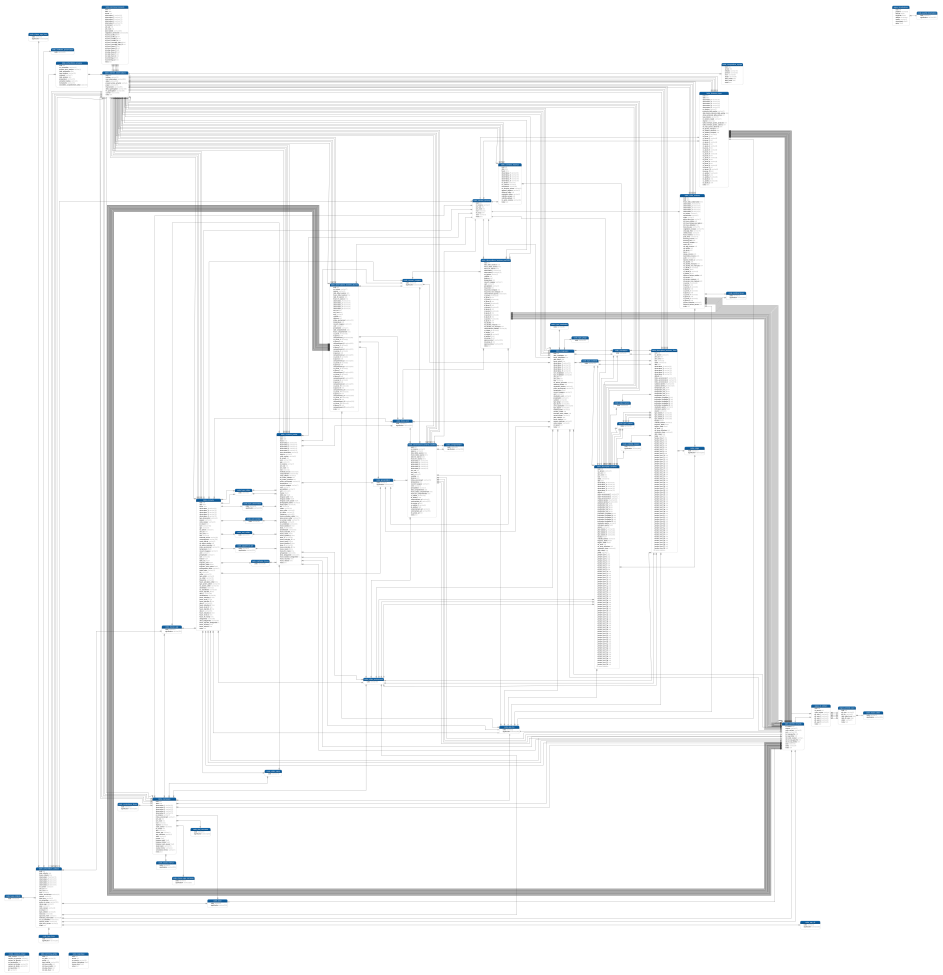

FIGURE 3 – Diagramme Entité-Relation de la base de données Renards

Il ne reste plus qu'à injecter la nouvelle base de données sur le serveur (depuis l'invite de commandes):

1. Un compte GitHub a été créé pour le labo: **chaireBioNorth**. Il est accessible à https://github.com/chaireBioNorth et contient des dépôts (*repositories*) dont le dépôt **arcticfox-tools** qui contient des scripts (par ex. il y a un script R pour générer les annexes du protocole Renards) et des documents utiles pour bien gérer les données Renards. Libre à vous de gérer ce compte comme il vous plait.
2. Comme le dépôt **arcticfox-tools** est un dépôt public, n'importe qui dans le monde peut y accéder. C'est pourquoi, les paramètres de connexion au serveur de l'UQAR ne sont pas présents dans le(s) script(s).

```
# ETAPE 1 - On se place sur le bureau de l'ordinateur de terrain

cd C:/Users/admin/Desktop


# ETAPE 2 - Exportation d'une copie de la base de données (depuis l'ordinateur)

pg_dump -U postgres -d berteaux_renards_bylot > berteaux_renards_bylot_yyyymmdd.sql


# ETAPE 3 - Archivage de la nouvelle base de données sur le serveur

psql -U casani01 -h srbd04.uqar.ca^
     -d berteaux_renards_bylot < berteaux_renards_bylot_yyyymmdd.sql
```

Le tour est joué: les données sont archivées. Ceci clôt l'ensemble du flux de travail avec la base de données Renards.

Bonus – Consultation des données

Tout ceci peut sembler abstrait, et vous vous demandez sûrement comment accéder aux données du serveur pour les analyser. Comme souvent, il y a plusieurs solutions et nous allons en voir trois.

Cependant, une **recommandation** doit être faite: obtenez une copie de la base de données (reportez-vous à la section **Avant le terrain – Transfert de la base de données**) et installez-la en local, c.-à-d. sur votre ordinateur. De cette manière, vous n'altèreraient pas les données archivées. Et si vous détectez des erreurs, faites-les suivre à la personne responsable de la base de données. OUI, ce serait bien qu'il n'y ait qu'une seule personne responsable...

Si vous n'avez pas installé le logiciel PostgreSQL sur votre ordinateur, veuillez-vous référer au document intitulé **Installation de PostgreSQL sous Windows**. C'est important !

Depuis pgAdmin

Le logiciel pgAdmin permet également de gérer les données PostgreSQL. Veuillez-vous référer au document intitulé **Installation de PostgreSQL sous Windows** pour découvrir son installation et sa configuration.

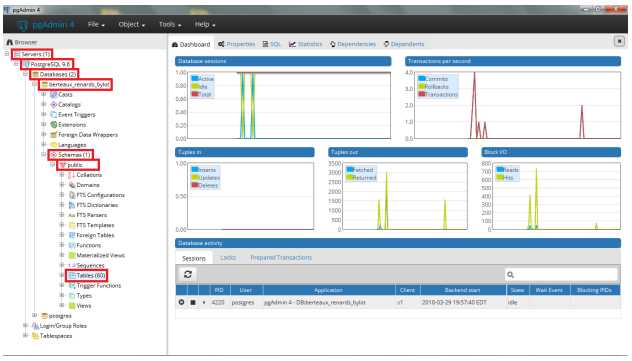

FIGURE 4 – Accès aux tables depuis pgAdmin

Les deux figures suivantes indiquent comment afficher les données d'une table.

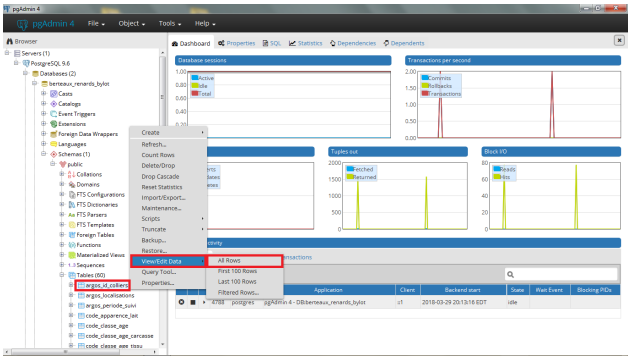

FIGURE 5 – Sélection de la table à afficher (clic droit pour afficher le menu)

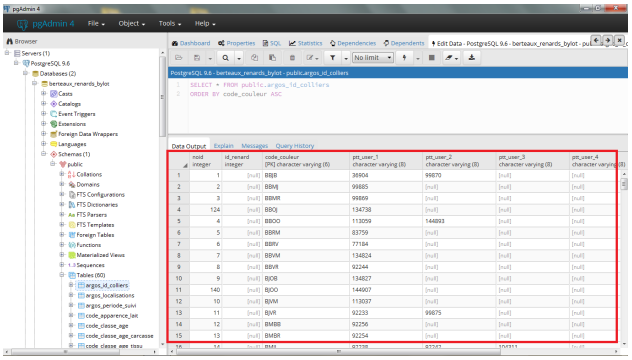

FIGURE 6 – Consultation des données de la table sélectionnée

Les deux figures suivantes indiquent comment exporter une table au format CSV.

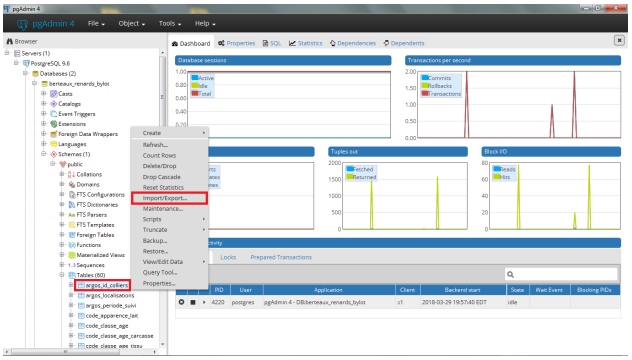

FIGURE 7 – Sélection de la table à exporter (clic droit pour afficher le menu)

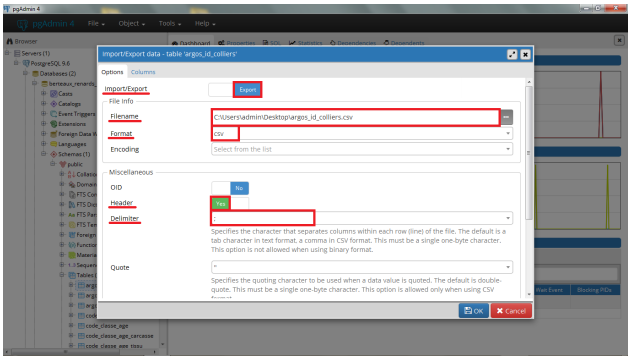

FIGURE 8 – Exportation d'une table au format CSV (informations requises)

Depuis Microsoft Access

Comme il a été dit précédemment, une interface de saisie des données a été développé sous Access. Cette interface permet également de consulter toutes les données stockées dans PostgreSQL. Vous n'avez qu'à 1) installer la base de données PostgreSQL sur votre ordinateur et 2) copier le fichier **berteaux_renards_bylot.accdb** (disponible sur https://github.com/chaireBioNorth/arcticfox-tools dans le dossier `interface/`). La connexion entre les deux logiciels se fera de manière automatique. Pour consulter une table, vous n'aurez qu'à double-cliquer sur celle-ci.

Depuis R

Un des principaux avantages de PostgreSQL est qu'il peut s'interfacer avec le logiciel R et ce sous n'importe quel système d'exploitation (contrairement à Microsoft Access). Un seul prérequis est nécessaire: installer le package `RPostgreSQL` qui permet d'effectuer la connexion entre les deux logiciels.

```r
# Installation du package <RPostgreSQL> (si nécessaire)
install.packages("RPostgreSQL")
```

La connexion entre R et la base de données Renards copiée sur votre ordinateur personnel se fera de la manière suivante:

```r
# Chargement du package <RPostgreSQL>
library(RPostgreSQL)


# Choix du pilote à utiliser
drv  <- dbDriver("PostgreSQL")


# Établissement de la connexion
mydb <- dbConnect(
  drv      = drv,
  user     = "postgres",
  password = "12345",
  dbname   = "berteaux_renards_bylot"
)
```

En utilisant la fonction `dbListTables()` de ce package, on peut afficher le nom des 60 tables stockées dans la base de données Renards.

```
# Nom des tables disponibles
tbnames <- dbListTables(mydb)


# On affiche le nom des 10 premières tables
tbnames[1:10]
```

```
##  [1] "codes_couleurs_dispo"      "table_lemming_grilles"
##  [3] "table_logistique"          "argos_id_colliers"
##  [5] "code_qualite_localisation" "argos_localisations"
##  [7] "code_statut_collier"       "argos_periode_suivi"
##  [9] "table_cameras_reconyx"     "code_classe_age_carcasse"
```

Pour importer des enregistrements dans R, il faudra cependant avoir quelques notions élémentaires en SQL et envoyer la requête à PostgreSQL grâce à R (on parle de *embedded SQL*). Mais, s'il existe bien un langage facile à appréhendrer, c'est bien le SQL. Regardons comment importer une table en utilisant la fonction `dbGetQuery()` qui envoie une requête et retourne le résultat de cette requête [3].

```
# Importation de tous les enregistrements de la table 'argos_periode_suivi'
argos <- dbGetQuery(mydb, "SELECT * FROM argos_periode_suivi;")


# On affiche les 8 premiers enregistrements
argos[1:8, ]
```

```
##   noid ptt_user ptt_cls date_debut_suivi date_fin_suivi statut
## 1   58    99885   99885       2010-07-23     2010-08-18      D
## 2  120   113059  113059       2013-05-31     2014-08-08      C
## 3    7    77184   77184       2007-08-17     2008-02-28      C
## 4  135   134827  134827       2014-05-19     2015-04-07      C
## 5   24    92233   92233       2009-06-09     2010-07-20      B
## 6   35    92256   92256       2009-07-26     2011-05-01      C
## 7   37    92242   92242       2010-05-31     2011-05-08      B
## 8   59    99878   99878       2010-07-23     2012-04-06      D
##                                                                notes
## 1                                  Mort determinee par analyse de mouvements
## 2                             Collier a emissions irregulieres apres cette date
## 3                      Collier arrete naturellement sans emissions irregulieres
## 4                      Collier arrete naturellement sans emissions irregulieres
## 5                Collier retire et change ; Pose sur un nouvel individu en 2011
## 6                      Collier arrete naturellement sans emissions irregulieres
## 7 Collier retire et change ; Emissions irregulieres jusqu'à la mi-Octobre 2010
## 8                                  Mort determinee par analyse de mouvements
```

3. Contrairement à la fonction `dbSendQuery()` qui elle ne fait qu'envoyer une requête ne retournant aucun résultat. C'est par exemple le cas lorsqu'on utilise l'instruction SQL `UPDATE` pour modifier un enregistrement.

Allons un peu plus loin et sélectionnons certaines colonnes, ou attributs (opération appelée **Projection** en algèbre relationnel) et certaines lignes, ou tuples/enregistrements (opération appelée **Restriction** en algèbre relationnel).

```
# Projection et restriction
argos <- dbGetQuery(mydb,
  "SELECT ptt_cls,date_debut_suivi,statut FROM argos_periode_suivi WHERE statut = 'D';"
)


# On affiche les 12 premiers enregistrements
argos[1:12, ]
```

```
##    ptt_cls date_debut_suivi statut
## 1    99885       2010-07-23      D
## 2    99878       2010-07-23      D
## 3   104325       2012-06-19      D
## 4   113043       2012-06-25      D
## 5    83752       2008-07-09      D
## 6   104305       2011-06-15      D
## 7    83754       2008-08-05      D
## 8   104317       2011-06-14      D
## 9    83747       2008-05-26      D
## 10  134838       2014-05-19      D
## 11   83758       2008-08-02      D
## 12   92251       2010-06-22      D
```

Ici, nous avons envoyé la requête suivante: *Sélectionne tous les enregistrements dont le statut est égal à D (colliers dont les renards sont morts) et ne retourne que les attributs N° de collier (ptt_cls), date de début de suivi des colliers et le statut.*

Voilà, ce n'est pas plus compliqué que ça.

Et n'oubliez pas de vous déconnecter de PostgreSQL:

```
# Déconnexion de PostgreSQL
lapply(dbListConnections(PostgreSQL()), DBI::dbDisconnect)
dbUnloadDriver(dbDriver("PostgreSQL"))
```