SQL embarqué en C

- Pour vous connecter à la base de données 'grtti', tapez dans un terminal :
  `psql -h asi-pg.insa-rouen.fr -U grtti grtti`

  Vous remplacerez le 'i' du nom d'utilisateur et de la base 'grtti' par le numéro de binôme qui vous aura été attribué par l'enseignant.
  Le mot de passe est de manière absolument pas sécurisée le nom d'utilisateur.

- Quelques commandes :

  ```
  \h aide mémoire des commandes
  \q pour terminer la connexion à la base
  q pour sortir de l'affichage d'une table
  ```

  En particulier la commande \i nomfichier.sql permet d'exécuter l'ensemble des requêtes présentes dans le fichier (mettre le fichier dans le répertoire d'où vous vous connectez à la base, ou bien spécifiez le chemin d'accès au fichier).
- Pour tout complément d'information concernant le SQL embarqué dans du C dans le cadre de postgresql, vous pouvez vous référer à la documentation en ligne : `http://docs.postgresqlfr.org/9.2/ecpg.html`
-

Nous allons utiliser la base réalisée lors du TP3 (Initiation à PL/POSTGRESQL) dont on rappelle le schéma ci-dessous.

PARCNAT (idZone VARCHAR(10), type VARCHAR(15), description VARCHAR(50), superficie REAL, pays VARCHAR(20), ouvertD DATE, ouvertF DATE, coutInter INTEGER)
ANIMAUX (noeAnim VARCHAR(10), espece VARCHAR(15), idZone VARCHAR(10), dateNais DATE)
VETO (noVeto VARCHAR(10), nom VARCHAR(15), prenom VARCHAR(15), etablissement VARCHAR(20), compEspece VARCHAR(15), dateDispoD DATE, dateDispoF DATE)
PLANNING (noInter VARCHAR(10), noVeto VARCHAR(10), idZone VARCHAR(10), dateInterD DATE, dateInterF DATE)

Par précaution, téléchargez les fichiers *suppressionTables.sql*, *creationTables.sql*, *tuples.sql* sur moodle et exécutez-les.

➲ Question 1 :

Téléchargez le fichier *appliGestionZoo.pgc* sur Moodle. Que signifient les lignes :

```
EXEC SQL INCLUDE sqlca;
EXEC SQL WHENEVER sqlerror sqlprint;
```

➲ Question 2 :

- Appliquez le préprocesseur **ecpg** sur ce fichier (commande **ecpg** *appliGestionZoo.pgc*). Un fichier *appliGestionZoo.c* est généré. Ouvrez le et observez le travail effectué par le préprocesseur.
- Compilez votre fichier *appliGestionZoo.c* via les commandes :

(1) `gcc -g -Wall -pedantic -c -I/usr/include/postgresql/ appliGestionZoo.c`
et

(2) `gcc -o appliGestionZoo appliGestionZoo.o -L/usr/lib -lpq -lpgtypes -lecpg`

- (1) est indispensable car le fichier .c généré via **ecpg** inclut les fichiers d'entête de l'installation Postgresql, qui n'est pas installé dans votre répertoire courant.
- (2) permet d'inclure la librairie libpq qui est l'interface de programmation pour les applications C avec PostgreSQL, libecpg librairie indispensable pour utiliser le préprocesseur ecpg, et libpgtypes permet entre autre d'utiliser le type DATE et les fonctions associées.

➲ Question 3 :

Complétez le fichier précédent de façon à pouvoir effectuer la connexion à la base de données grtti.

- La suite du TP a pour objectif de proposer à un utilisateur plusieurs fonctionalités pour agir sur la base de données. Ceci prendra la forme d'un menu qui apparaitra dès que l'utilisateur lance son programme.
- Toutes les requêtes sql utilisées seront à mettre en oeuvre via les méthodes liées au SQL dynamique vues en cours.
- Un exemple d'utilisation de l'ensemble du programme est disponible dans le fichier *exempleSortie.txt* sur Moodle.

➲ Question 4 :

Réalisez la fonctionalité *SupprimeAnimaux* qui permet de supprimer un tuple dans la table ANIMAUX en fonction d'une valeur de l'attribut *noeAnim* entrée par l'utilisateur.

➲ Question 5 :

Réalisez la fonctionalité *AjouteAnimaux* qui permet d'insérer un tuple dans la table ANIMAUX.

Attention à la gestion de la variable de type DATE !

- Vous aurez besoin de la fonction PGTYPESdate_from_asc(char *, NULL) qui prend une chaine de caractères et renvoie cette chaîne au format DATE.

➲ Question 6 :

Remarques :

- Vous aurez sans doute besoin d'utiliser la fonction PGTYPESdate_to_asc(DATE) qui prend une date et renvoie une chaîne de caractères.

Réalisez la fonctionnalité *AfficheTable* qui lorsqu'elle est choisie par l'utilisateur :

- liste l'ensemble des tables de la base : [Remarque 1] débrouillez vous pour ne voir afficher que les tables de la base qui nous intéressent ici (i.e. pas les tables systèmes), [Remarque 2] La table système qui contient cette information sous Postgresql est la table *pg_tables*. Pour définir la variable associée au nom d'une table : CHAR nomTable[NAMEDATALEN]
- demandez à l'utilisateur laquelle il souhaite voir afficher [Remarque] Considérez que l'usager demande la table ANIMAUX. En effet ici vous ne pourriez pas faire simplement une requête préparée du type 'Select * from ?', il vous faudrait préparer les requêtes pour chaque table.
- affichez cette table (ANIMAUX)
- Ré-écrivez cette fonctionnalité en mettant en oeuvre les informations fournies par sqlca pour votre condition d'arrêt.