INITIATION Á PL/POSTGRESQL

- 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).

- **En complément du cours**, la documentation suivante peut vous aider : `http://docs.postgresqlfr.org/9.1/`

- Vous rédigerez vos fonctions PLPSQL dans un fichier .sql.
- Pensez à mettre à jour votre fichier *suppressionTables.sql* au fur et à mesure du TP.

➲ **On reprend le même schéma de base de données que le TP2-création de tables(BD1)**
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)

➲ **Questions :**

1. Téléchargez les fichiers *suppressionTables.sql* et *creationTables.sql* sur moodle et exécutez les.
2. Téléchargez le fichier *tuplesBaseGrtti.sql* sur moodle, et exécuter le.
3. Créez un fichier *fonctionplsql.sql* que vous compléterez avec les différentes fonctions/vues des questions suivantes.
4. L'objectif est de créer la vue 'Salaires' qui affiche la liste des vétérinaires ayant effectué une intervention et le salaire qu'il leur a été versé. Dans ce but :
   (a) Créez la fonction 'calculSalaire' PL/PGSQL qui calcule le salaire relatif à l'intervention d'un vétérinaire (table **PLANNING**). Pour simplifier on considère qu'il n'y a qu'une seule intervention par vétérinaire dans **PLANNING**. Quelques indications :
   - Cette fonction prend en paramètre le coût d'une intervention (tarif journalier) et l'identifiant du vétérinaire.

- On suppose que le vétérinaire passé en paramètre apparaît dans la table **PLANNING** exactement une fois.
- Vous définirez une variable locale qui contiendra le nombre de jours d'intervention.
- Vous testerez la fonction avec la commande :

```
select calculSalaire(100, 'v1');
```

(b) Créez la vue 'Salaires' en utilisant la fonction précédente, vous afficherez le nom, prénom et le salaire pour chaque vétérinaire.

5. Définissez une fonction 'age' qui va parcourir la table **ANIMAUX** et afficher l'identifiant de l'animal, son espèce et son âge. Pour cela vous utiliserez un curseur qui va parcourir la table **ANIMAUX**. Quelques indications :
   - Vous définirez une variable locale vous permettant de récupérer une ligne de la table **ANIMAUX**.
   - Pour arêter la boucle de lecture, testez si une ligne est égale à NULL.

   Deux versions sont à réaliser :

   (a) Utilisez un type existant
   - Pour définir le type d'une ligne : TYPE%ROWTYPE
   - Vous affichez toutes les lignes

   (b) Créez votre propre type [1]
   - Il doit permettre de récupérer uniquement les champs nécessaires
   - N'affichez que les lignes relatives à l'espèce 'CHINCHILLA'

6. L'objectif est de pouvoir s'assurer lors d'un ajout à la table PLANNING que les dates d'interventions sont cohérentes **à la fois** avec celles d'ouverture du parc **et** avec la période de disponibilité des vétérinaires. Dans ce but :

   (a) Définissez une fonction 'test' sans paramètre d'entrée qui déclenche et affiche une erreur si la contrainte de cohérence des dates n'est pas respectée lors d'un ajout à la table PLANNING. Pour tester votre fonction vous devez définir le trigger, les opérations (a) et (b) sont donc à effectuer conjointement.
   - Pour simplifier on supposera que les dates de disponibilité des vétérinaires doivent être comprises entre les dates de début et de fin d'intervention (pas de chevauchement)
   - La syntaxe pour lancer/afficher une erreur (parcD et parcF sont des variables locales qui contiennent les dates d'ouverture et de fermeture du parc) :

```
RAISE EXCEPTION 'PB ouverture parc du % au % ', parcD, parcF;
```

   (b) Définissez le trigger testPlanning qui exécute la fonction test().

   (c) Vérifiez que votre trigger fonctionne correctement. Testez par exemple les insertions suivantes :

```
INSERT INTO PLANNING VALUES ('i3','v3','pn4','16/12/2012','10/01/2013');
INSERT INTO PLANNING VALUES ('i3','v3','pn4','14/02/2013','15/02/2013');
INSERT INTO PLANNING VALUES ('i3','v3','pn4','02/01/2013','10/01/2013');
INSERT INTO PLANNING VALUES ('i3','v3','pn4','14/12/2012','30/12/2012');
INSERT INTO PLANNING VALUES ('i3','v3','pn4','04/01/2013','12/01/2013');
```

1. Créer/utiliser un type
   Définition : CREATE TYPE nomType as (champ1 type1, champ2 type 2 ...) ;
   Déclaration : variable monType ;
   Accès à un champ : variable.champ1