IS2A3 – BDA – T.P. 1a

Résumé

Ce TP aborde les fonctions et les procédures stockées sous PostgreSQL ainsi que les triggers.
N'hésitez pas à consulter la documentation en ligne sur le site de PostgreSQL : https://www.postgresql.org/docs/13/index.html

Mise en place

— Créer une base de données postgres qui aura pour nom `bda_tp1a_`*votreNomLogin*.

```
$ createdb -U votreNomLogin [1] bda_tp1a_votreNomLogin
```

Rappel, pour exploiter le serveur postgres de l'école, positionnez la variable `PGHOST` :

```
export PGHOST=serveur-etu.polytech-lille.fr
```

— Se connecter à votre nouvelle base de données :

```
$ psql -U votreNomLogin -d bda_tp1a_votreNomLogin
```

Rappel, le mot de passe de base est "postgres".

— Vous écrirez toutes vos commandes SQL dans un fichier et vous importez votre fichier SQL dans la base de données :

```
# \i votreFichierSQL
```

I Création des tables

Une compagnie de transport urbain vous demande de créer une base données qui relie les bus de la compagnie et les stations de la commune.

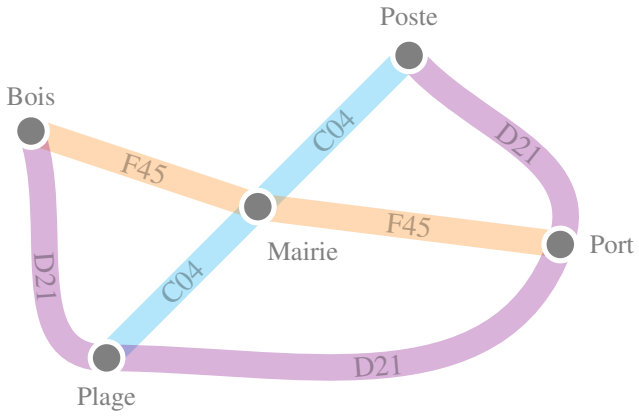
I.1. Créez une première table `bus` qui prendra deux champs : un entier `id` et un texte `nom`.

I.2. Créez une deuxième table `station` qui prendra deux champs : un entier `id` et un texte `nom`.

I.3. Créez une dernière table `passe` qui mettra en relation l'`id` d'un bus et l'`id` d'une station.

Aide : N'oubliez pas dans votre fichier de supprimer, dans l'ordre inverse de création, les tables (si elles existent) que vous allez créer ensuite .

II Peuplement de la table



II.1. Ci-dessus, vous avez le plan de la ville avec les stations et les bus. Peuplez la base de données en conséquence.

1. Cette option n'est pas obligatoire quand votre nom de login UNIX est le même que votre nom de login PostgreSQL.

III Fonctions

III.1. Utilisez la fonction SQL `max` pour trouver la plus grande `id` dans la table `bus`.

III.2. Créez une fonction SQL `next_bus_id()` qui vous donne un `id` disponible (qui n'est pas encore attribué) dans la table `bus`.

III.3. Réécrivez la même fonction `next_bus_id()` mais en PLPGSQL.

III.4. Écrivez une fonction `nb_stations` qui pour un `id` de `bus`, donne le nombre de stations par lesquelles ce bus passe. Afficher la liste des bus avec leur nombre de stations.
Aide : N'hésitez pas à tester directement dans votre base de données les requêtes SQL que vous allez ensuite utiliser dans vos fonctions et procédures.

III.5. Écrivez une fonction `liste_bus` qui pour un `id` de `station`, donne la liste des bus passant par cette station. Afficher la liste des stations avec leur liste de bus respective.

IV Procédures

IV.1. Créez une procédure `ajouter_bus` qui pour un `nom` et un `id`, ajoute une nouvelle ligne dans la table `bus`.

IV.2. Créez une procédure `ajouter_bus` qui pour un `nom`, ajoute une nouvelle ligne dans la table `bus` en prenant le prochain `id` disponible.
Remarque : On a créé ici deux procédures différentes avec le même nom mais avec des arguments différents. Cela est possible car la *surcharge* est possible en SQL.

IV.3. Modifiez la procédure `ajouter_bus` de IV.2 pour ajouter une erreur "le bus <nom_du_bus> existe deja", si le bus existe déjà.
Aide : Utilisez `raise exception` pour lever une erreur.

V Triggers

V.1. Créez un trigger se déclenchant à la création ou à la mise à jour d'un bus et qui donne le nombre de caractères de `nom` du `bus`.
Aide : Utilisez `raise notice` pour afficher un message.

V.2. Modifiez le trigger précédant pour qu'à la création et à la mise à jour d'un bus, seul les nom de bus commençant par une lettre et ayant deux chiffres ensuite soient acceptés.
Aide : Utilisez la fonction `textregexeq` pour comparer une chaine de caractères à une expression régulière, par exemple `textregexeq(ch,'^[0-9]*$')` teste si `ch` contient un chiffre.