Développement en PL/pgSQL

Fabien Coelho
Mines Paris – PSL

1 Introduction

Cette séance pratique vise à utiliser le langage PL/pgSQL proposé par PostgreSQL pour développer de nouvelles fonctionnalités.

Chaque exercice peut être réalisé dans un fichier séparé, qui crée les fonctions et extensions nécessaires et effectue des tests pour vérifier leur bon fonctionnement.

Le mode SQL de votre éditeur (vscode, emacs, gedit, vi, nano) peut vous aider à éditer vos fonctions dans un fichier texte SQL, qui redéfinit votre fonction `CREATE OR REPLACE` à chaque fois.

Pour tester, utiliser par exemple :

```
psql -h pagode < nom-du-fichier.sql
```

Le produit de vos efforts sera rendu via l'interface web habituelle.

Il est également possible d'utiliser cette interface pour le développement et les tests, en faisant attention que l'édition n'est pas sauvegardée si teste simplement avec le bouton *Exec SQL*.

2 Exercices

Fonctions simples

1 Réaliser une fonction `fthrt` qui calcule la racine cinquième d'un réel. Retourner `NULL` si l'entrée est `NULL`. Penser à traiter les nombres négatifs !

```
SELECT fthrt(-32.0); -- -2.0
```

2 Est-ce que `fthrt` fonctionne si on lui passe un entier ? Pourquoi ?

```
SELECT fthrt(32); -- ???
```

3 Développez une fonction `estPalindrome` qui teste si un mot est un palindrome.

```
SELECT estPalindrome('Hobbes'); -- FALSE
SELECT estPalindrome('Kayak'); -- TRUE
```

4 L'encodage ROT13 permet d'encoder des chaînes de caractères de manière super secrète. Développez une fonction `rot13` qui implémente cet encodage.

```
SELECT rot13('Fnyhg Pnyiva !'); -- 'Salut Calvin !'
```

5 Développez une fonction `email` pour générer automatiquement une adresse de messagerie en `prenom.nom@comics.net` pour un auteur.

```
SELECT a.email FROM auteur AS a LIMIT 1; -- 'Jean.Plantu@comics.net'
```

6 Réaliser une fonction `nrows` de comptage des lignes dans une table dont le *nom* est passé en argument. Penser au cas où une espace apparaît dans le nom de la table.

```
SELECT nrows('pg_user');
-- same result as SELECT COUNT(*) FROM pg_user;
```

Aggrégation moyenne géométrique

7 Développer une nouvelle **aggrégation** `GEOM_AVG` qui calcule la moyenne géométrique.

```
SELECT GEOM_AVG(annee) FROM oeuvre;
```

8 Que se passe-t-il si une valeur passé à `GEOM_AVG` est `NULL` ?

Opération ===

9 Développer une nouvelle opération `===` s'appliquant à deux chaînes de caractères qui dit si la seconde correspond à l'expression régulière SQL décrite dans la première chaîne (il s'agit donc d'un LIKE renversé !).

```
-- a === b  same as  b LIKE a:
SELECT 'A%' === 'Aspic'; -- TRUE
SELECT '%alv%' === 'Calvin'; -- TRUE
```

Nombres premiers

10 Développer une fonction `isPrime` qui teste si un entier est premier. On se contentera d'une version fonctionnelle non optimisée. On considérera 1 comme premier.

```
SELECT isPrime(2); -- TRUE
SELECT isPrime(6); -- FALSE
```

11 Développer une fonction `primes` qui retourne sous forme d'une relation la liste des nombres premiers à partir de un et inférieurs à un entier passé en arguments. Vous pouvez réutiliser la fonction précédente.

```
SELECT * FROM primes(11);
-- one column: 1, 2, 3, 5, 7, 11
```

Insertion d'un ouvrage

12 Développer une procédure `nouvelOuvrage` qui ajoute un ouvrage à partir de son titre, son année, sa collection et son éditeur. L'ouvrage sera supposé écrit en français. La collection et l'éditeur seront supposés déjà exister.

```
CALL nouvelOuvrage('L''Empire du Milieu', 2023, 'Asterix', 'Albert Rene');
```

Dernière modification d'un tuple

13 Développer une fonction trigger `keepLastUpdate` qui maintient à jour un attribut `lastEditor` et `lastUpdate` d'une relation en conservant automatiquement l'identitié et l'instant de la dernière modification de chaque tuple.

14 Comment associer la trigger `keepLastUpdate` à une table de manière pertinente ?

Tests anagrammiques

15 Créer une fonction `estAnagramme` qui teste si deux mots en minuscules sont des anagrammes l'un de l'autre.

```
SELECT estAnagramme('calvin', 'hobbes'); -- FALSE
SELECT estAnagramme('melimelos', 'sommeille'); -- TRUE
```

16 Étendre la fonction précédente pour ignorer la casse et les espaces.

```
SELECT estAnagramme('Le Marquis de Sade', 'Demasqua le desir'); -- TRUE
```