Développement en PL/pgSQL

Fabien Coelho
MINES ParisTech

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 (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

Fonction racine cinquième

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 cela fonctionne si on met un entier ? Pourquoi ?

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

Test palindromique

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

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

Fonction rot13

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 !'
```

Colonne virtuelle *email*

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'
```

Fonction comptage de ligne

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 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. Attention, 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
```

Parcours d'un attribut

12 Développer une fonction `attribut` qui parcourt un attribut d'une table sous forme texte, ces deux éléments étant passés en arguments (nom de la table, nom de l'attribut).

```
SELECT * FROM attribut('pg_user', 'usename');
-- same as SELECT usename FROM pg_user;
```

Domaine Euro

13 Définir et tester un nouveau domaine `Euro` comme un type `NUMERIC` ayant une précision d'au plus 2 chiffres après la virgule.

```
SELECT Euro '1.0'; -- ok
SELECT Euro '10.003'; -- error...
SELECT Euro '-1'; -- error
SELECT Euro 'deux'; -- error...
```

14 Faire une fonction de conversion `euro2text` de `Euro` vers `TEXT` qui ajoute le symbole de la monnaie. On réutilisera la définition du domaine `Euro`.

```
SELECT euro2text(Euro '54.32'); -- 54.32 €
SELECT euro2text(Euro '54.3'); -- 54.30 €
SELECT euro2text(Euro '54'); -- 54.00 €
```

15 Comment définir un cast **explicite** qui permette de convertir une `Euro` en `TEXT` ? Fonctionne-t-il ? On réutilisera le résultat de la question précédente.

```
SELECT CAST(Euro '10.0' AS TEXT); -- 10.00 €
```

Trigger blocage d'un tuple

16 Développer une fonction trigger `lock_tuple` de blocage d'un tuple par un attribut booléen `lock`. Si l'attribut est vrai, toute modification du tuple est bloquée. Sinon, les modifications sont autorisées.

```
SELECT id, nom, lock FROM fruit;
-- (1, 'Fraise', FALSE)
UPDATE fruit SET nom='Figue' WHERE id=1; -- okay
UPDATE fruit SET lock=TRUE WHERE id=1; -- okay
UPDATE fruit SET nom='Fraise' WHERE id=1;
-- Error, tuple is locked!
```

17 Comment associer ce de manière *pertinente* ce trigger à une relation ?

18 Comment malgré tout modifier le tuple si celui-ci a été bloqué ?

Trigger garde compte

19 Développer une fonction trigger `comptes` qui maintient dans une table externe le nombre total de tuples des tables auxquelles il est attaché. En argument de cette trigger, on trouvera le nom de la table de compte, le nom de l'attribut stockant le nom de la relation, et celui stockant le nombre de tuple.

```
SELECT total FROM les_comptes WHERE nom='fruit'; -- 12
INSERT INTO fruit VALUES(...);
SELECT total FROM les_comptes WHERE nom='fruit'; -- 13
```

20 Comment associer de manière pertinente l'opération, le niveau et le moment auxquels doit être attaché ce trigger ?

21 Comment être sûr de la validité de ce compte en cas de transactions opérant en parallèle ? Que penser de l'impact sur les performances ?

Tests anagrammiques

22 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
```

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

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