

Développement en PL/pgSQL

Fabien Coelho (fabien@coelho.net)
MINES ParisTech

Résumé

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

Chaque exercice sera réalisé dans un fichier séparé, qui créera les fonctions et extensions nécessaires et opérera des tests pour vérifier leur bon fonctionnement.

Le mode SQL de votre éditeur (emacs, gedit, vim) peut vous aider à éditer vos fonctions.

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

1 Fonction racine cinquième

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
```

Est-ce que cela fonctionne si on met un entier ? Pourquoi ?

2 Fonction comptage de ligne

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;
```

3 Aggrégation compte petits entiers

Développer une nouvelle **aggrégation** `smalls` qui compte le nombre de valeurs entières comprises entre 0 et 9.

```
SELECT smalls(usesysid::INTEGER) FROM pg_user;
```

Que se passe-t-il si une valeur est NULL ?

4 Opération ===

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
```

5 Nombres premiers

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
```

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.

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

6 Parcours d'un attribut

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

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

7 Domaine date en français

Définir et tester un nouveau domaine `DateFr` comme une chaîne de caractère décrivant une date en français sous la forme 20 mars 1970.

```
SELECT DateFr '9 novembre 1799'; -- ok...
SELECT DateFr '18 brumaire VIII'; -- error...
```

Faire une fonction de conversion `date2datefr` de `Date` vers `DateFr`.

```
SELECT Date2DateFr('1970-03-20'); -- 20 mars 1970
```

Faire une fonction de conversion `datefr2date` de `DateFr` vers `Date`.

```
SELECT DateFr2Date('20 mars 1970'); -- DATE '1970-03-20'
```

Comment définir un cast **explicite** qui permette de convertir une Date en DateFr ? Fonctionne-t-il ?

```
SELECT CAST(DATE '1799-11-09' AS DateFr); -- 9 novembre 1799
```

Comment définir un cast qui convertisse **automatiquement** une DateFr en Date ? Fonctionne-t-il ?

```
SELECT CAST(DateFr '20 mars 1970' AS Date); -- 1970-03-20
```

8 Trigger blocage d'un tuple

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!
```

Comment associer ce de manière pertinente ce trigger à une relation ?
Comment malgré tout modifier le tuple si celui-ci a été bloqué ?
Comment paramétrer le nom de l'attribut de blocage ?

9 Trigger garde compte

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
```

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

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