Bases de données avancées

TD n° 4 (PL/pgSQL) : Curseurs paramétrés, SQL dynamique

J. Darmont (https://eric.univ-lyon2.fr/jdarmont/), 03/10/23

Exercice 1 : Curseur paramétré implicite

1. Écrire une fonction de nom schema() qui retourne un ensemble de chaînes de caractères. Définir une variable chaîne de caractères nommée « nomTable » et lui affecter le nom d'une table de votre compte. Écrire le code PL/pgSQL permettant de retourner la liste de ses attributs à partir de la requête `SELECT column_name FROM information_schema.columns WHERE table_name = nomTable`.

2. Transformer la fonction schema() en faisant de nomTable un paramètre de la fonction.

Exercice 2 : Requête dynamique imbriquée dans un curseur statique

Reprendre la fonction cat() du TD n° 1 (Exercice 3). La transformer en une fonction catPlus() qui renvoie en plus du nom de chaque table de votre base de données leur nombre de n-uplets. Réutiliser l'enregistrement tNuplet(texte, nombre) du TD n° 2.

Exercice 3 : Curseur paramétré implicite et curseur dynamique

En reprenant une partie du code de la fonction schema() de l'Exercice 1, créer une nouvelle fonction deuxAtt() qui affiche les valeurs des deux premiers attributs (généralement la clé primaire et un attribut que l'on espère significatif) de la table passée en paramètre.

Indications :

- Au préalable, créer un type d'enregistrement nommé tDeuxAtt, qui contient deux champs att1 et att2 qui sont des chaînes de caractères. tDeuxAtt servira à retourner le résultat de la fonction deuxAtt().
- Lire les *noms* des deux premiers attributs de la table à l'aide d'un curseur paramétré implicite s'inspirant de la requête de l'Exercice 2. Stocker les noms dans un tableau. La clause LIMIT est autorisée (`SELECT … FROM … WHERE … LIMIT 2`). Elle permet de ne retourner que les deux premiers n-uplets du résultat de la requête.
- Exprimer la requête qui permet de récupérer les *valeurs* des deux premiers attributs sous la forme d'une chaîne de caractères stockée dans un enregistrement de type tDeuxAtt.
- Utiliser un curseur dynamique pour exécuter la requête, parcourir son résultat et afficher les valeurs des deux premiers attributs.

Résultat attendu pour la table EMP :

```
7369 SMITH
7499 ALLEN
7521 WARD
7654 MARTIN
7698 BLAKE
7782 CLARK
7839 KING
7844 TURNER
7900 JAMES
7934 MILLER
7566 JONES
7902 FORD
```

Correction

```
-- Exercice 1

CREATE OR REPLACE FUNCTION schema(nomTable VARCHAR) RETURNS SETOF VARCHAR AS $$
     DECLARE
          -- nomTable VARCHAR := 'emp';
          att VARCHAR;
     BEGIN
          FOR att IN SELECT column_name FROM information_schema.columns
          WHERE table_name = nomTable LOOP
               RETURN NEXT att;
          END LOOP;
          RETURN;
     END
$$ LANGUAGE plpgsql;

SELECT * from schema('emp');



-- Exercice 2

CREATE OR REPLACE FUNCTION catPlus() RETURNS SETOF tNuplet AS $$
      DECLARE
            nuplet RECORD;
            res tNuplet;
      BEGIN
            FOR nuplet IN SELECT tablename FROM pg_tables
                  WHERE tableowner = 'darmont'
            LOOP
                  res.texte := nuplet.tablename;
                  EXECUTE 'SELECT COUNT(*) FROM ' || nuplet.tablename
                        INTO res.nombre;
                  RETURN NEXT res;
            END LOOP;
            RETURN;
      END
$$ LANGUAGE plpgsql;

SELECT * from catPlus();



-- Exercice 3

CREATE TYPE tDeuxAtt AS (
     att1 VARCHAR,
     att2 VARCHAR
);
```

```
CREATE OR REPLACE FUNCTION deuxAtt(nomTable VARCHAR)
RETURNS SETOF tDeuxAtt AS $$
      DECLARE
            cursDyn REFCURSOR;
            att VARCHAR[]; -- Tableau des noms d'attributs
            a VARCHAR;
            i INT := 1;
            res tDeuxAtt;
            requete VARCHAR;
      BEGIN
            -- Récupération des attributs via un curseur paramétré implicite
            FOR a IN SELECT column_name FROM information_schema.columns
            WHERE table_name = nomTable LIMIT 2 LOOP
                  att[i] := a;
                  i := i + 1;
            end loop;

            -- Construction de la requête
            requete := 'SELECT ' || att[1] || ', ' || att[2] || ' FROM '
                  || nomTable;

            -- Parcours du curseur dynamique
            OPEN cursDyn FOR EXECUTE requete;
            FETCH cursDyn into res;
            WHILE FOUND LOOP
                  RETURN NEXT res;
                  FETCH cursDyn into res;
            END LOOP;
            CLOSE cursDyn;
            RETURN;
      END
$$ LANGUAGE plpgsql;

SELECT * from deuxAtt('emp');
```