M1 Informatique – Année 2018-2019
Bases de données avancées
TD n° 4 (PL/pgSQL) : SQL dynamique
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 27/08/18

Exercice 1 : 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.

Exercice 2 : Curseur paramétré

1. Écrire une fonction de nom schema() qui retourne un ensemble de chaînes de cractères. 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 3 : Curseur dynamique

En reprenant une partie du code de la fonction schema() de l'Exercice 2, 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 :

- Récupérer les *noms* des deux premiers attributs de la table à l'aide du curseur paramétré de l'Exercice 2.
- Exprimer la requête qui permet de récupérer les *valeurs* des deux premiers attributs sous la forme d'une chaîne de caractères stockée dans une variable.
- 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
```

Exercice 4 : Altération de schéma (fausse requête dynamique)

Écrire une fonction prenant en paramètre le nom d'une table et permettant de créer une vue contenant les noms de tous les attributs de cette table, ainsi que leur type. Le nom de la vue devra être de la forme ATT_nomDeLaTable. Vérifier le résultat.

Indication :

- Dans la table système `information_schema.columns`, le type des attributs est `data_type`.
- Dans les tables systèmes, toutes les chaînes de caractères (comme les noms de tables ou d'attributs) sont stockées en minuscules.

Correction

```
-- Exercice 1

CREATE TYPE tCatPlus AS (
     tablename VARCHAR,
     tablecount NUMERIC
);

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

SELECT * from catPlus();


-- Exercice 2

CREATE OR REPLACE FUNCTION schema(nomTable VARCHAR) RETURNS SETOF VARCHAR AS $$
     DECLARE
          cursAtt CURSOR(tbl VARCHAR) FOR
               SELECT column_name FROM information_schema.columns
               WHERE table_name = tbl;
          -- nomTable VARCHAR := 'emp';
          att VARCHAR;
     BEGIN
          OPEN cursAtt(nomTable);
          FETCH cursAtt into att;
          WHILE FOUND LOOP
               RETURN NEXT att;
               FETCH cursAtt into att;
          END LOOP;
          CLOSE cursAtt;
          RETURN;
     END
$$ LANGUAGE plpgsql;

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

-- Exercice 3

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

CREATE OR REPLACE FUNCTION deuxAtt(nomTable VARCHAR)
RETURNS SETOF tDeuxAtt AS $$
     DECLARE
          cursAtt CURSOR(tbl VARCHAR) FOR SELECT column_name
               FROM information_schema.columns WHERE table_name = tbl;
          cursDyn REFCURSOR;
          att tDeuxAtt;
          res tDeuxAtt;
          requete VARCHAR;
     BEGIN
          -- Récupération des attributs via le curseur paramétré
          OPEN cursAtt(nomTable);
          FETCH cursAtt into att.att1;
          FETCH cursAtt into att.att2;
          CLOSE cursAtt;
          -- Construction de la requête
          requete := 'SELECT ' || att.att1 || ', ' || att.att2 ||
               ' FROM ' || nomTable;
          -- Parcours du curseur dynamique
          OPEN cursDyn FOR EXECUTE requete;
          FETCH cursDyn into res;
          WHILE FOUND LOOP
               RETURN NEXT res;
               FETCH cursDyn into res;
          END LOOP;
          CLOSE cursDyn;
          RETURN;
     END
$$ LANGUAGE plpgsql;

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

-- Exercice 4

```
CREATE OR REPLACE FUNCTION vueAtt(nomTable VARCHAR) RETURNS VOID AS $$
     DECLARE
          requete VARCHAR;
     BEGIN
          requete := 'CREATE VIEW att_' || nomTable ||
               ' AS SELECT column_name, data_type FROM information_schema.columns
               WHERE table_name = ''' || LOWER(nomTable) || '''';
          EXECUTE requete;
     END
$$ LANGUAGE plpgsql;

SELECT vueAtt('emp');

SELECT * FROM att_emp;
```