TP 2 : PL/PGSQL

Sauf mention contraire, on considère les tables de HR

Rappel :

- L'appel d'une procédure PL/PGSQL peut se faire selon la syntaxe :

```
CALL NOM_PROCEDURE [(PARAMETRES)] ;
```

- L'appel d'une fonction PL/PGSQL peut se faire selon la syntaxe :

```
SELECT NOM_FONCTION [(PARAMETRES)] ;
```

Exercice 1 :

1) Créer une fonction CHARGES qui retourne la valeur des charges pour un salaire donné en paramètre. Les charges représentent 45% du salaire.

```
CREATE OR REPLACE FUNCTION CHARGES (SALAIRE FLOAT ) RETURNS FLOAT AS $$
BEGIN RETURN (SALAIRE*0.45);
END;
$$ LANGUAGE plpgsql;

SELECT CHARGES (1000);
```

Utiliser la fonction CHARGES pour afficher tous les montants des charges des salaires des employés actuellement en poste.

```
SELECT EMPLOYEE_ID, CHARGES(SALARY) AS "SALAIRE BRUT"
FROM EMPLOYEES;
```

2)

Ecrire une procédure PL/PGSQL qui insère un nouveau tuple dans LOCATIONS: le numéro de localisation est le successeur du numéro de localisation le plus élevé ; la rue et la ville sont donnés en paramètre. La RUE aura comme valeur par défaut 'INCONNU' ; la valeur par défaut de VILLE est 'PARIS'.

```
CREATE OR REPLACE PROCEDURE AJOUT_LOCAL(RUE LOCATIONS.street_address%TYPE
DEFAULT 'INCONNU', VILLE  LOCATIONS.CITY%TYPE DEFAULT 'PARIS')
LANGUAGE plpgsql
AS $$
DECLARE
NO_MAX INTEGER ;
BEGIN
SELECT MAX(location_id) INTO NO_MAX FROM LOCATIONS;
```

```
INSERT INTO LOCATIONS (location_id,street_address, city) VALUES (NO_MAX+1, RUE, VILLE);
COMMIT;
END; $$
```

Exécuter la procédure pour ajouter un local à PARIS, rue du Louvre et à NANCY, rue Saint-Jean. Vérifier que les ajouts se sont bien passés.

```
CALL AJOUT_LOCAL('Rue-test','Nancy' );
```

3) a- Ecrire un bloc PL/PGSQL anonyme permettant de répondre à cette question :

Est-ce que tous les employés du service 'Finance' gagnent plus de 10 000$?
Afficher un message clair en fonction de la situation.

```
do $$
DECLARE
num_emp INTEGER;
BEGIN
SELECT Count(employee_id) INTO num_emp
FROM Employees
WHERE salary > 10000 ;
IF num_emp = 0 THEN
raise notice 'aucun employé ne gagne plus de 10000$';
ELSE
raise notice ' employés gagnent plus de 10000$  : % ',  num_emp ;
END IF;
END ;
$$ LANGUAGE plpgsql;
```

b- Transformer ce bloc pour en faire une procédure nommée TEST_SERVICE permettant de répondre à la question étant donné un nom de service donné en paramètre. Dérouter les cas d'erreur (cas d'un nom de service inexistant) à l'aide d'une exception (Instruction RAISE EXCEPTION).

```
CREATE OR REPLACE FUNCTION test_service(nom_service  VARCHAR)  RETURNS FLOAT AS
$$
DECLARE
num_emp INTEGER;
test INTEGER ;
BEGIN
SELECT COUNT(department_id) INTO test
   FROM Departments
   WHERE department_name like nom_service ;
   IF test = 0 THEN
     RAISE EXCEPTION 'service_inexistant' ;
   ELSE
      SELECT Count(employee_id) INTO num_emp
      FROM Employees E, Departments D
      WHERE E.salary > 10000
      AND D.department_name like nom_service
      AND D.department_id = E.department_id ;
      IF num_emp = 0 THEN
```

```
raise notice 'aucun employé ne gagne plus de 10000$ dans le service % ', nom_service;
ELSE
raise notice ' % employés gagnent plus de 10000$ dans le service % ', num_emp, nom_service ;
RETURN num_emp;
      END IF ;
END IF ;
END ;
$$ LANGUAGE plpgsql;
```

c- Tester votre procédure avec les noms de services suivants : IT, ACCOUNTING, EXECUTIVE, FUN.

```
CALL test_service('Accounting');
```

4) Ecrire une procédure SERVICES_SANS_CHEF donnant la liste des services sans chef à partir de la table DEPARTMENTS de HR. Exécuter la procédure.

```
CREATE OR REPLACE PROCEDURE SERVI_SANS_CHEF () LANGUAGE plpgsql AS $$
DECLARE
Curs CURSOR FOR SELECT department_id AS id, department_name AS nom
FROM departments
WHERE manager_id IS NULL ;
curs_id departments.department_id%TYPE ;
curs_name departments.department_name%TYPE;
BEGIN
raise notice 'ddd';
OPEN Curs ;
LOOP
FETCH Curs INTO curs_id, curs_name ;
IF NOT FOUND THEN EXIT; END IF;
raise notice 'Nom du dep. % - DepID % ', curs_name,curs_id;
END LOOP ;
CLOSE Curs ;
END $$

CALL SERVI_SANS_CHEF ();
```

5) Ecrire une procédure SERVICES_SANS_CHEF_VILLE qui donne la liste des services sans chef localisés dans une ville donnée en paramètre. Exécuter la procédure avec les villes SEATTLE, SOUTHLAKE, TOKYO, ROMA.

```
CREATE OR REPLACE PROCEDURE SERVI_SANS_CHEF_VILLE (VILLE VARCHAR)
LANGUAGE PLPGSQL AS $$
DECLARE
CURS CURSOR FOR SELECT D.DEPARTMENT_ID AS ID, D.DEPARTMENT_NAME AS NOM
FROM DEPARTMENTS D, LOCATIONS L
WHERE MANAGER_ID IS NULL
AND D.LOCATION_ID = L.LOCATION_ID
AND UPPER(L.CITY) LIKE VILLE ;

CURS_ID DEPARTMENTS.DEPARTMENT_ID%TYPE ;
CURS_NAME DEPARTMENTS.DEPARTMENT_NAME%TYPE;
BEGIN
```

```
OPEN CURS ;
LOOP
FETCH CURS INTO CURS_ID, CURS_NAME ;
IF NOT FOUND THEN EXIT; END IF;
RAISE NOTICE 'NOM DU DEP. % - DEPID % ', CURS_NAME,CURS_ID;
END LOOP ;
CLOSE CURS ;
END $$

CALL SERVI_SANS_CHEF_VILLE ('ROMA');
```

6) **Pour les plus rapides :** Ecrire une procédure EFFECTIF_SERVICES_VILLE donnant pour chaque service localisé dans une ville donnée en paramètre, le nombre de personnes en poste actuellement dans ce service.

Exercice 2 : PL/SQL – Curseurs - Procédures

1- A partir des tables de HR, afficher toutes les informations concernant les 5 employés ayant le salaire le plus élevé (bloc anonyme).

```
do $$
DECLARE
N integer := 5;
C1 CURSOR FOR SELECT FIRST_NAME,LAST_NAME,SALARY FROM EMPLOYEES ORDER
BY SALARY DESC;
CO INTEGER :=0;
BEGIN
FOR I IN C1 LOOP
IF CO<N THEN
raise notice '%- % - %', I.FIRST_NAME,I.LAST_NAME, I.SALARY;
CO:=CO+1;
END IF;
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

2- Transformer le bloc de la question a- pour en faire une procédure TOP_N qui affiche les nom et prénom des N salariés les mieux payés. N est un paramètre IN de la procédure. Penser à dérouter les cas où N n'est pas un nombre strictement positif.

```
CREATE OR REPLACE PROCEDURE topk(k integer)
LANGUAGE plpgsql
AS $$
DECLARE
```

```
C1 CURSOR FOR SELECT FIRST_NAME,LAST_NAME,SALARY FROM EMPLOYEES ORDER
BY SALARY DESC;
CO INTEGER :=0;
BEGIN
FOR I IN C1 LOOP
IF CO<k THEN
raise notice '%- % - %', I.FIRST_NAME,I.LAST_NAME, I.SALARY;
CO:=CO+1;
END IF;
END LOOP;
END;  $$
```

3- Exécuter la procédure TOP_N pour N=5 puis N=11.

```
Appel de la procédure
CALL topk(6);
```