TP PL/SQL - schéma HR

Lyes Rahal

11 avril 2025

Lien GitHub du projet :
`https://github.com/Rahal-Lyes/plsql-cursor`

Table des matières

1 Partie A : Affichage des informations des employés

1.1 Version avec curseur explicite

```
DECLARE
    CURSOR c_emp IS
        SELECT E.FIRST_NAME, J.JOB_TITLE, J.MAX_SALARY
        FROM EMPLOYEES E
        JOIN JOBS J ON E.JOB_ID = J.JOB_ID;
    v_emp_rec c_emp%ROWTYPE;

BEGIN
    OPEN c_emp;
    LOOP
        FETCH c_emp INTO v_emp_rec;
        EXIT WHEN c_emp%NOTFOUND;
        DBMS_OUTPUT.put_line(v_emp_rec.first_name|| ' est un ' ||
   v_emp_rec.JOB_TITLE|| ' il touch u salaire  de ******' ||v_emp_rec.
   MAX_SALARY);
    END LOOP;
    CLOSE c_emp;
END;
/
```

1.2 Autre method ameliorer pour la question A

```
DECLARE
    CURSOR c_emp IS
        SELECT E.FIRST_NAME, J.JOB_TITLE, J.MAX_SALARY
        FROM EMPLOYEES E
        JOIN JOBS J ON E.JOB_ID = J.JOB_ID
        ORDER BY E.FIRST_NAME;
BEGIN
  FOR v_emp_rec IN c_emp LOOP
    DBMS_OUTPUT.put_line(v_emp_rec.first_name|| ' est un ' || v_emp_rec.
   JOB_TITLE|| ' il touch u salaire  de ****' ||v_emp_rec.MAX_SALARY);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(v_separator);
END;
/
```

2 Partie B : Fonction de calcul de salaire

```
1   CREATE OR REPLACE FUNCTION new_sal (
2     old_sal IN NUMBER,
3     pct     IN NUMBER
4  ) RETURN NUMBER
5  AS
6     new_salary EMPLOYEES.salary%TYPE;
7     pct_null_exc EXCEPTION;
8  BEGIN
9     IF pct IS NOT NULL THEN
10       new_salary := old_sal * (1 + pct);
11       RETURN new_salary;
12    ELSE
13       RAISE pct_null_exc;
14    END IF;
15
16 EXCEPTION
17    WHEN pct_null_exc THEN
18       DBMS_OUTPUT.PUT_LINE('La valeur de pct de l''employ  est NULL');
19       RETURN NULL;
20 END;
21 /
```

3 Partie C : Ajout de l'attribut `new_salary` de type `NUMBER` à la table `EMPLOYEES`

```sql
ALTER TABLE EMPLOYEES ADD new_salary  NUMBER(8,2);
```

4 Partie D : Procédure de mise à jour

```sql
  CREATE OR REPLACE PROCEDURE new_salary AS
  CURSOR emp_cursor IS
    SELECT employee_id, salary, commission_pct
    FROM EMPLOYEES;


  new_sal_value EMPLOYEES.salary%TYPE;

BEGIN

  FOR i IN emp_cursor LOOP
    new_sal_value := new_sal(i.salary, i.commission_pct);
    UPDATE EMPLOYEES
    SET new_salary = new_sal_value
    WHERE employee_id = i.employee_id;
  END LOOP;

  COMMIT;
END;
/
```

5 Partie E : Exécuter la procédure et montrer les résultats de son exécution (avant et après)

```
PL/SQL procedure successfully completed.

La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
La valeur de pct de l'employé est NULL
```

FIGURE 1 – L'exécution de la procédure new$_{s}alary$

6 Partie F : Afficher les informations sur les old et new salary sous cette forme)

```
DECLARE
CURSOR c_emp is SELECT E.FIRST_NAME,E.SALARY,E.NEW_SALARY
FROM EMPLOYEES E;
v_emp_rec c_emp %rowtype;

BEGIN
  OPEN c_emp;
  LOOP
   FETCH c_emp into v_emp_rec;
   exit when c_emp %notfound;

   DBMS_OUTPUT.put_line(v_emp_rec.first_name|| '      ,ancien salaire
*****' || v_emp_rec.SALARY|| '       nouveau salaire  ******' ||
v_emp_rec.NEW_SALARY);
  END LOOP;
  CLOSE c_emp;

  END;
  /

```

7 (g) Programme PL/SQL utilisant un curseur

Le programme suivant utilise un curseur pour afficher les informations suivantes pour chaque employé :

- Nom de l'employé
- Son poste (job)
- La date de début d'emploi
- L'expérience exprimée en nombre de jours (arrondi sans virgule)
- L'expérience exprimée en nombre de mois (arrondi sans virgule)

```
1
2  DECLARE
3  CURSOR c_employe IS SELECT E.FIRST_NAME,J.JOB_TITLE,JH.START_DATE,JH.
       END_DATE
4  FROM EMPLOYEES E
5  JOIN JOBS J ON j.JOB_ID=E.JOB_ID
6  JOIN JOB_HISTORY JH ON JH.JOB_ID=J.JOB_ID;
7
8
9  v_nom EMPLOYEES.FIRST_NAME %type;
10 v_job_title JOBS.JOB_TITLE %type;
11 v_start_date JOB_HISTORY.START_DATE%type;
12 v_end_date JOB_HISTORY.END_DATE%type;
13 v_experience_jours NUMBER;
14 v_experience_mois  NUMBER;
15 v_emp_rec c_employe%rowtype;
16 BEGIN
17   OPEN c_employe;
18   LOOP
19     FETCH c_employe into v_emp_rec;
20     exit when c_employe%notfound;
21     v_nom :=v_emp_rec.FIRST_NAME;
22     v_job_title :=v_emp_rec.JOB_TITLE;
23     v_start_date :=v_emp_rec.START_DATE;
24     v_end_date := v_emp_rec.END_DATE;
25   v_experience_jours := TRUNC(v_end_date - v_start_date);
26   v_experience_mois  := ROUND(MONTHS_BETWEEN(v_end_date, v_start_date));
27     dbms_output.PUT_LINE('Nom_employer '|| v_nom|| '| Son job '||
      v_job_title|| '| Date_debut  d emploi  '
28     ||v_start_date||'| Exp rience en jours '||v_experience_jours ||'
     Jours'|| '| Exp rience en mois '|| v_experience_mois || ' mois');
29     end loop;
30     CLOSE c_employe;
31
32
33 END;
34 /
35
```

8 Mise à jour de la date de prochaine promotion des employés

Objectif

Ce programme PL/SQL met à jour le champ `PROCHAIN_PROMO` pour chaque employé. La prochaine date de promotion est calculée comme la date à laquelle l'employé atteindra

la prochaine centaine de mois d'ancienneté.

— Exemple : Un employé ayant 278 mois d'ancienneté au 12/03/2024 aura pour prochaine date de promotion le 13/01/2026 (à 300 mois).

```
1
2  DECLARE
3    CURSOR c_employe IS
4      SELECT E.EMPLOYEE_ID,
5             E.FIRST_NAME,
6             NVL(MAX(JH.START_DATE), E.HIRE_DATE) AS START_DATE
7      FROM EMPLOYEES E
8      LEFT JOIN JOB_HISTORY JH ON JH.EMPLOYEE_ID = E.EMPLOYEE_ID
9      GROUP BY E.EMPLOYEE_ID, E.FIRST_NAME, E.HIRE_DATE;
10
11 BEGIN
12   FOR emp IN c_employe LOOP
13     DECLARE
14       v_months_worked   NUMBER;
15       v_next_centaine   NUMBER;
16       v_next_promo_date DATE;
17     BEGIN
18       -- Calcul de l'anciennet  en mois
19       v_months_worked := MONTHS_BETWEEN(SYSDATE, emp.START_DATE);
20
21       -- Prochaine centaine de mois atteinte
22       v_next_centaine := (FLOOR(v_months_worked / 100) + 1) * 100;
23
24       -- Date de la prochaine promotion
25       v_next_promo_date := ADD_MONTHS(emp.START_DATE, v_next_centaine);
26
27       -- Mise    jour de la date de prochaine promo
28       UPDATE EMPLOYEES
29       SET PROCHAIN_PROMO = v_next_promo_date
30       WHERE EMPLOYEE_ID = emp.EMPLOYEE_ID;
31
32       -- Affichage pour v rification
33       DBMS_OUTPUT.PUT_LINE(
34         'Employ  ' || emp.EMPLOYEE_ID ||
35         ' (' || emp.FIRST_NAME || ')' ||
36         ' | Anciennet  : ' || TRUNC(v_months_worked) || ' mois de
     travail    cette date : ' ||
37         TO_CHAR(SYSDATE, 'DD/MM/YYYY') ||
38         ' | Prochaine promo : ' || TO_CHAR(v_next_promo_date, 'DD/MM/
     YYYY')
39       );
40     END;
41   END LOOP;
42
43   COMMIT;
44 END;
45 /
```

Lien GitHub du projet : Voir le code source sur GitHub