Rendu du TPs de développement de base de données, le 25/01/2024

Fait par : EL WELY Sidi Mohamed

Encadrant : Myoung-Ah KANG

TP2

Après avoir effectuer la création de tables et les insertions demandées on commence les questions

Q1) Afficher à l'écran le nom, salaire, la commission de l'employé 'MILLER' ainsi que le nom du département dans lequel il travail.

```
SELECT e.ename AS "Nom de l'employé", e.sal AS "Salaire", e.comm AS "Commission", d.dname AS "Nom du département"
  2  FROM emp e
  3  JOIN dept d ON e.deptno = d.deptno
  4  WHERE e.ename = 'MILLER';

Nom de l'e    Salaire Commission Nom du d??part
---------- ---------- ---------- --------------
MILLER           1300            ACCOUNTING

SQL>
```

Q2)Insérer les 100 premiers numéros avec leurs parités :

```
 --insertion de parité

DECLARE
    col1 NUMBER(9,4);
    col2 NUMBER(9,4);
    charcol CHAR(55);
BEGIN
    -- boucle for
    FOR i IN 1..100 LOOP
        col1:=i;
        col2:=i*100;

        IF(MOD(i,2)=0) THEN
            charcol:=TO_CHAR(i) || 'est pair';
        ELSE
            Charcol := TO_CHAR(i) || 'est impair';


       END IF;
    INSERT INTO temp (num_col1, num_col2, char_col) VALUES (col1, col2,
charcol);

    DBMS_OUTPUT.PUT_LINE('100 lignes inserer');
    END LOOP;
END;
/
```

```
SQL> select * from temp;

  NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- ----------------------------------------------------
         1        100 1est impair
         2        200 2est pair
         3        300 3est impair
         4        400 4est pair
         5        500 5est impair
         6        600 6est pair
         7        700 7est impair
         8        800 8est pair
         9        900 9est impair
        10       1000 10est pair
        11       1100 11est impair
```

Q3) inserer le 5 employé le mieux payés

```
INSERT INTO temp (num_col1, num_col2, char_col)
  2  SELECT sal, empno, ename
  3  FROM (
  4       SELECT sal, empno, ename,
            ROW_NUMBER() OVER (ORDER BY sal DESC) AS salary_rank
  6       FROM emp
  7  ) sub
  8  WHERE salary_rank <= 5;

5 lignes creees.
```

Resultas :

```
---------- ---------- --------------
       100      10000 100est pair
      5000       7839 KING
      3500       7000 elwely
      3000       7788 SCOTT
      3000       7902 FORD
      2975       7566 JONES
```

Q4) les employés avec revenues supérieure à 200$

```
INSERT INTO temp (num_col1, num_col2, char_col)
SELECT sal + NVL(comm, 0) AS revenus_mensuels, empno, ename
  2    3  FROM emp
  4  WHERE sal + NVL(comm, 0) > 2000;

8 lignes creees.
```

Resultas :

```
      2975       7566 JONES
      2650       7654 MARTIN
      2850       7698 BLAKE
      2450       7782 CLARK
      3000       7788 SCOTT

 NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- -------------------------
      5000       7839 KING
      3000       7902 FORD
      3500       7000 elwely
```

Q5)inserer le premier employé qui a un sal supérieur à 4000$:

```
INSERT INTO temp (num_col1, num_col2, char_col)
  2  SELECT sal AS num_col1, empno AS num_col2, ename AS char_col
  3  FROM (
  4      SELECT sal, empno, ename,
           ROW_NUMBER() OVER (ORDER BY empno) AS emp_rank
  6      FROM emp
  7      WHERE sal > 4000
  8      AND empno < 7902
  9  ) sub
 10  WHERE emp_rank = 1;

1 ligne creee.
```

Résultats :

```
5000       7839 KING
```

TP3

Creation de procedures et fonctions :

Procedur createdeot_elwely :

```
create or replace procedure createdept_elwely(numero_dept dept.deptno%type,
dept_name dept.dname%type, localisation dept.loc%type)
is
temp int;
begin
  select count(*) into temp from dept where deptno=numero_dept;

  if (temp=0) then

    insert into dept values (numero_dept, dept_name, localisation);


  else
    raise_application_error(-20001, 'nmero_dept exist deja');
  end if;

end;
/
```

Fonction salok_elwely :

```
create or replace function salok_elwely(job SalIntervalle_elwely.job%type,
salaire number)
return number
is
res int;
inf SalIntervalle_elwely.lsal%type;
sup SalIntervalle_elwely.hsal%type;
begin
  res :=0;
  select lsal into inf from SalIntervalle_elwely where
SalIntervalle_elwely.job=job;

  select hsal into sup from SalIntervalle_elwely where
SalIntervalle_elwely.job=job;

  if ((salaire > inf) and (salaire < sup)) then

     res:=1;
  end if;
```

```
  return (res);
end;
/
```

Procedure raise_salary_elwely :

```
create or replace procedure raise_salary_elwely(emp_id emp.empno%type , amont
number)
IS
v_sal number;
v_job emp.job%type;
BEGIN
  --recuperer le salaire et le job de l'employeur
  select sal, job into v_sal, v_job
  FROM emp
  where empno=emp_id;

  --verifier si le salaire est dans l'intervalle correspondant
  if salok_elwely(v_job,v_sal)=1 then
     --augmenter le salaire
     update emp
     set sal = sal + amont
     where empno=emp_id;

     --afficher un message de succes
     DBMS_OUTPUT.PUT_LINE('erreur : Le salaire de l''employé ' || emp_id || '
a été augmenté de ' || amont || 'euros.');

  else
     DBMS_OUTPUT.PUT_LINE('erreur : Le salaire de l''employé ' || emp_id || '
ne peut pas etre augmenté de ' || amont || 'euros.');
  end if;
  exception
    when no_data_found then
       --l'employé n'existe pas
       DBMS_OUTPUT.PUT_LINE('erreur : Employé ' || emp_id || 'introuvable');
end;
/
```

Resultat :

```
SQL> start progpl/prog.sql

Procedure creee.


Fonction creee.


Procedure creee.
```

Creation du bloc pl/sql qui fait un backup de tous les tables de l’utilisateurs connecté :

```plsql
DECLARE
    v_table_name user_tables.table_name%TYPE;
BEGIN
    FOR tables_cursor IN (SELECT table_name FROM user_tables) LOOP
        v_table_name := tables_cursor.table_name;

        -- Vérifier si la table existe déjà avec le suffixe "_old"
        BEGIN
            EXECUTE IMMEDIATE 'DROP TABLE ' || v_table_name || '_old';
        EXCEPTION
            WHEN OTHERS THEN
                NULL; -- Ignorer l'erreur si la table n'existe pas encore
        END;

        -- Copier la table avec le suffixe "_old"
        EXECUTE IMMEDIATE 'CREATE TABLE ' || v_table_name || '_old AS SELECT *
FROM ' || v_table_name;
    END LOOP;
END;
/
```

Resultat :

```
SQL> select table_name from user_tables;

TABLE_NAME
-----------------------------------------------------------
EMP
DEPT
TEMP
SALINTERVALLE_ELWELY
ELWELY
ELWELY_OLD
EMP_OLD
DEPT_OLD
TEMP_OLD
SALINTERVALLE_ELWELY_OLD

10 lignes selectionnees.
```

Creation de Package ELWELY_PKG

```
CREATE OR REPLACE PACKAGE ELWELY_PKG AS
    -- Déclaration du curseur pour la procédure afficher_emp_elwely
    CURSOR emp_par_dep_elwely (p_deptno NUMBER) IS
        SELECT empno, ename
        FROM emp
        WHERE deptno = p_deptno;

    -- Déclaration de la procédure raise_salary_elwely
    PROCEDURE raise_salary_elwely (p_empno NUMBER, p_increase NUMBER);

    -- Déclaration de la procédure afficher_emp_elwely
    PROCEDURE afficher_emp_elwely (p_deptno NUMBER);
END ELWELY_PKG;
/
```

Resultat :

```
SQL> start progpl/package.sql

Package cree.
```

Creations de trigers :

```
--trigers Raise_elwely

CREATE OR REPLACE TRIGGER raise_elwely
BEFORE UPDATE OF sal ON emp
FOR EACH ROW
BEGIN
    IF :NEW.sal < :OLD.sal THEN
        RAISE_APPLICATION_ERROR(-20001, 'Le salaire d''un employé ne peut pas
diminuer.');
    END IF;
END;
/

--triger numdept_elwely

CREATE OR REPLACE TRIGGER numdept_elwely
BEFORE INSERT OR UPDATE OF deptno ON emp
FOR EACH ROW
BEGIN
    IF :NEW.deptno < 61 OR :NEW.deptno > 69 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Le numero de departement doit etre
entre 61 et 69.');
    END IF;
END;
/

--triger dept_elwely
CREATE OR REPLACE TRIGGER dept_elwely
BEFORE INSERT OR UPDATE OF deptno ON emp
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM dept WHERE deptno = :NEW.deptno;

    IF v_count = 0 THEN
        INSERT INTO dept (deptno, dname, loc) VALUES (:NEW.deptno, 'A SAISIR',
'A SAISIR');
    END IF;
END;
/

--triger noweek_elwely

CREATE OR REPLACE TRIGGER noweek_elwely
```

```
BEFORE INSERT OR UPDATE OR DELETE ON emp
DECLARE
    v_day VARCHAR2(20);
BEGIN
    -- Obtenir le jour de la semaine
    v_day := TO_CHAR(SYSDATE, 'DAY');

    -- Vérifier si c'est le samedi ou le dimanche
    IF v_day IN ('SAMEDI', 'DIMANCHE', 'JEUDI') THEN
        RAISE_APPLICATION_ERROR(-20003, 'Les modifications de la relation
employé sont interdites pendant le week-end (samedi , dimanche et jeudi(jour
de verification) ) .');
    END IF;
END;
/

--triger stats_elwely

CREATE OR REPLACE TRIGGER stat_elwely
AFTER INSERT OR UPDATE OR DELETE ON emp
FOR EACH ROW
DECLARE
    v_typemaj CHAR(7);
BEGIN
    -- Déterminer le type de modification (INSERT, UPDATE, DELETE)
    IF INSERTING THEN
        v_typemaj := 'INSERT';
    ELSIF UPDATING THEN
        v_typemaj := 'UPDATE';
    ELSIF DELETING THEN
        v_typemaj := 'DELETE';
    END IF;

    -- Mettre à jour le compteur de la table STATS_elwely
    UPDATE STATS_elwely
    SET NbMaj = NbMaj + 1, Date_derniere_Maj = SYSDATE
    WHERE TypeMaj = v_typemaj;
END;
/


--triger cheksal_elwely

CREATE OR REPLACE TRIGGER checksal_elwely
BEFORE UPDATE OF job ON emp
FOR EACH ROW
DECLARE
    v_new_sal NUMBER;
    v_lsal emp.sal%TYPE;
```

```
    v_hsal emp.sal%TYPE;
BEGIN
    -- Récupérer les valeurs de lsal et hsal pour le nouveau job depuis la
table SalIntervalle
    SELECT lsal, hsal INTO v_lsal, v_hsal
    FROM SalIntervalle_elwely
    WHERE job = :NEW.job;

    -- Si le nouveau job est "president", ne pas appliquer de changement de
salaire
    IF :NEW.job = 'president' THEN
        RETURN;
    END IF;

    -- Calculer le nouveau salaire avec l'augmentation de 100 euros
    v_new_sal := :NEW.sal + 100;

    -- Si le nouveau salaire est supérieur à hsal, affecter la valeur de hsal
    IF v_new_sal > v_hsal THEN
        :NEW.sal := v_hsal;
    -- Si le nouveau salaire est inférieur à lsal, affecter la valeur de lsal
    ELSIF v_new_sal < v_lsal THEN
        :NEW.sal := v_lsal;
    ELSE
        :NEW.sal := v_new_sal;
    END IF;
END;
/
```

Vérifications :

```
UPDATE emp
  2  SET sal = 700
  3  WHERE empno = 7369;
UPDATE emp
       *
ERREUR a la ligne 1 :
ORA-20001: Le salaire d'un employ?? ne peut pas diminuer.
ORA-06512: a "SMELWELY.RAISE_ELWELY", ligne 3
ORA-04088: erreur lors d'execution du declencheur 'SMELWELY.RAISE_ELWELY'
```

```
CREATE TABLE STATS_elwely (
  2      TypeMaj CHAR(7),
  3      NbMaj NUMBER,
  4      Date_derniere_Maj DATE
  5  );

Table creee.

INSERT INTO STATS_elwely (TypeMaj, NbMaj, Date_derniere_Maj) VALUES ('INSERT', 0, NULL);
INSERT INTO STATS_elwely (TypeMaj, NbMaj, Date_derniere_Maj) VALUES ('UPDATE', 0, NULL);

1 ligne creee.

SQL>
1 ligne creee.

SQL> INSERT INTO STATS_elwely (TypeMaj, NbMaj, Date_derniere_Maj) VALUES ('DELETE', 0, NULL);

1 ligne creee.
```

```
SQL> insert into emp values
  2  (63,'SMITH','CLERK',7902,to_date('25-01-2024','dd-mm-yyyy'),800,NULL,63);

1 ligne creee.

SQL> update emp set sal = 900 where empno = 63;

1 ligne mise a jour.

SQL> select * from stats_elwely;

TYPEMAJ      NBMAJ DATE_DER
------- ---------- --------
INSERT           1 25/01/24
UPDATE           1 25/01/24
DELETE           1 25/01/24
```

```
SQL> insert into emp values
  2  (1000,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
insert into emp values
            *
ERREUR a la ligne 1 :
ORA-20002: Le numero de departement doit etre entre 61 et 69.
ORA-06512: a "SMELWELY.NUMDEPT_ELWELY", ligne 3
ORA-04088: erreur lors d'execution du declencheur 'SMELWELY.NUMDEPT_ELWELY'
```

```
SQL> insert into emp values
  2  (63,'SMITH','CLERK',7902,to_date('25-01-2024','dd-mm-yyyy'),800,NULL,63);

1 ligne creee.

SQL> delete from emp where empno=63;

1 ligne supprimee.

SQL> slect * from dept where deptno=63;
SP2-0734: commande inconnue au debut de "slect * fr..." - le reste de la ligne est ignore.
SQL> select * from dept where deptno=63;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        63 A SAISIR       A SAISIR
```

Conclusion :

En conclusion, ce TP nous a permis de mettre en pratique plusieurs concepts fondamentaux de la programmation PL/SQL et de comprendre comment ils peuvent être utilisés pour développer des applications robustes et performantes sur Oracle. La maîtrise de PL/SQL est essentielle pour tout développeur travaillant avec Oracle Database, car cela permet de tirer pleinement parti des fonctionnalités et des performances de la base de données.