# Travaux Pratiques

## Mise-à-jour et integrité, transaction et cohérence, PL/SQL

Documentation technique d'Oracle.

Dans le premier TP, vous avez interrogé les tables Buveurs, Achats, Vins, Recoltes, Producteurs. Vous avez aussi créé des vues sur ces tables: Bons_buveurs, Buveurs_asec, Buveurs_achats, Buveurs_achats2, Q83pl. Vérifiez que ces vues existent avant de commencer ce TP :

```
SQL> set long 1024

SQL> set lines 200

SQL> set pages 66

SQL> describe USER_VIEWS

SQL> select view_name, text from USER_VIEWS
```

Si vous ne trouvez pas les vues, vous pouvez les créer à nouveau en reprenant les ordres de création de vues donnés dans le corrigé du TP .

### Première partie : Mise à jour à travers les vues.

Dans le modèle relationnel tables et vues sont des relations qui peuvent se manipuler sans faire de distinction à priori. C'est tout à fait vrai pour les manipulations de type SELECT. Ce ne l'est pas pour les INSERT, UPDATE, DELETE. Les deux exercices suivant vous permettront de le comprendre.

- Créer une table (notée rbb) de même schéma que bons_buveurs et qui contient les tuples sélectionnés dans bons_buveurs par :

```
CREATE TABLE RBB AS SELECT * FROM BONS_BUVEURS;
```

- Mise à jour de la table buveurs : mettre le type des buveurs à 'gros' si la somme des quantites bues est supérieure à 100.
- Comparer les contenus de la table rbb et la vue bons_buveurs.Qu'observez-vous? expliquez ce qui c'est passé.
- Insérez dans bons_buveurs un buveur qui porte votre nom et votre prénom, un numéro égale au max(nb) + 1 et un type 'moyen'. Est ce que ce buveur est visible à travers la vue ?
- Insérez dans la même vue un autre buveur de type 'petit'. Est ce que ce buveur est visible ? expliquez ce qui s'est passé.
- Détruisez puis recréez la vue bons_buveurs en spécifiant la clause `WITH CHECK OPTION` à la fin de la commande. Recommencez l'insertion d'un autre petit buveur. Est ce que ce buveur est maintenant visible ? expliquez ce que fait la clause ajoutée.
- Re-créer les vues buveurs_achats et buveurs_achats2 en précisant la clause `WITH CHECK OPTION`. Insérer un nouveau buveur à travers chacune de ces vues. Expliquez le comportement d'Oracle (vous pouvez consulter la documentation Oracle ).
- Insérez dans q83pl le tuple ('PARIS', 'Cotes du Jura', 300). Tentez de retirer les tuples tels que QTE_BUE>100. Expliquez le comportement d'Oracle

### Seconde partie : PL/SQL et procédures

Le PL/SQL d'Oracle permet la programmation de traitements complexes sur la base de données. Il permet aussi de spécifier des procédures et des fonctions qui seront stockées pour être exécutées aux niveau du SGBD. Nous allons voir dans cette partie du TP comment écrire ces programmee et comment les exécuter.

- Bloc PL/SQL, structures de contrôle et boucles.
  Tester l'effet du programme suivant :

```
DECLARE
    x  NUMBER := 1;
BEGIN
```

```
    WHILE x < 100 LOOP
          DELETE FROM ACHATS WHERE nv=x;
          x := x + 2;
    END LOOP;
END;
/
```

Annuler l'effet de ce programme par la commande `ROLLBACK;`

Tester l'effet du programme suivant, puis annuler encore une fois son effet :

```
BEGIN
    FOR i IN 1..100 LOOP
        IF MOD(i,2) = 0 THEN     -- i est pair
            UPDATE ACHATS SET QTE=QTE*2 WHERE nv=i;
        ELSE
            UPDATE ACHATS SET QTE=QTE/2 WHERE nv=i;
        END IF;
    END LOOP;
END;
/
```

Remarque: vous pouvez utiliser votre éditer préféré pour mémoriser le code de vos scripts dans des fichiers et les lancer à l'aide de la commande `START` *`nom_fichier`*.

- Utilisation de curseur, commande Fetch explicite.
  Créer une table temporaire comme suit :

```
CREATE TABLE TEMP (Vin NUMBER(3), description VARCHAR2(30),
                   commentaire VARCHAR2(30));
```

Tester le programme suivant, expliquer ce qu'il fait :

```
DECLARE
   cursor V_CUR is select * from vins
              where nv in (select nv from recoltes, producteurs
                           where region='Alsace' and producteurs.np=recoltes.np)
              order by nv;
   V_LIGNE VINS%ROWTYPE;
BEGIN
   open V_CUR;
   loop
       fetch V_CUR into V_Ligne;
       exit when V_CUR%NOTFOUND;
       IF V_Ligne.mill in (1976, 1978, 1983) THEN
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'Récolte exeptionelle !');
       ELSE
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'RAS !');
       END IF;
   end loop;
   close V_CUR;
END;
/
```

- Utilisation de curseur, commande Fetch implicite.
  Que fait le programme suivant ?

```
DECLARE
   cursor V_CUR is select * from vins
              where nv in (select nv from recoltes, producteurs
                           where region='Alsace' and producteurs.np=recoltes.np)
              order by nv;
BEGIN
   for V_Ligne in V_CUR loop
       IF V_Ligne.mill in (1976, 1978, 1983) THEN
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'Récolte exeptionelle !');
       ELSE
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'RAS !');
       END IF;
   end loop;
END;
/
```

- Procédures stockées. Changer le programme précedent pour en faire une procédure stockée :

```
CREATE OR REPLACE PROCEDURE choix_experts AS
--DECLARE
   cursor V_CUR is select * from vins
               where nv in (select nv from recoltes, producteurs
                            where region='Alsace' and producteurs.np=recoltes.np)
               order by nv;
BEGIN
   for V_Ligne in V_CUR loop
       IF V_Ligne.mill in (1976, 1978, 1983) THEN
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'Récolte exeptionelle !');
       ELSE
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'RAS !');
       END IF;
   end loop;
END;
/
```

Appeler cette procédure à l'aide de la commande :

```
EXECUTE choix_experts
OU
BEGIN
  choix_experts;
END;
/
```

Tester la fonction suivante :

```
CREATE OR REPLACE FUNCTION recherche_vins (r PRODUCTEURS.REGION%TYPE, annee VINS.MILL%TYPE) RETURN number AS
--DECLARE
--   r PRODUCTEURS.REGION%TYPE;
--   annee VINS.MILL%TYPE;
   cursor V_CUR is select * from vins
               where nv in (select nv from recoltes, producteurs
                            where region=r and producteurs.np=recoltes.np)
               order by nv;
   cpt number:=0;

BEGIN
   for V_Ligne in V_CUR loop
       IF V_Ligne.mill = annee THEN
          INSERT INTO TEMP
                VALUES (V_Ligne.nv, V_Ligne.cru || '(' || to_char(V_Ligne.mill) || ')', 'Récolte exeptionelle !');
          cpt:=cpt+1;
       END IF;
   end loop;
   RETURN cpt;
END;
/
```

- Appel de fonction et renvoi de message à l'utilisateur.

```
SET SERVEROUTPUT ON
EXECUTE dbms_output.enable(1000000);

DECLARE
  x number;
BEGIN
  x:=recherche_vins('Alsace',1983);
  dbms_output.new_line;
  dbms_output.put_line(x || ' vins selectionnes ');
END;
/
```

- Ecrire une procédure VSTAT qui permet de déterminer, pour un vin donné, les différentes régions de production, le nombre de producteurs par région et un classement des villes (lieu) de consommation (nombre de bouteilles vendus), année par année.