NGUYEN Duc Hau
ROBERT Quentin

BDD : Travail 3

Table des matières

Introduction :

Dans ce tp nous allons travailler les vues, les triggers, et les blocs pl sql. Et ce afin de pouvoir mieux gérer les bases de données.

Création d'une vue :

Permet d'afficher l'intégralité des informations client d'un seul coup (nom, prénom, numéro du passport, adresse, téléphone, nombre de réservation.

La requête est la suivante :

```
CREATE OR REPLACE VIEW FicheClient(nom, prenom, numpassport, adresse,
telephone, nbReservation )
    AS SELECT DISTINCT P.NOM, P.PRENOM, Passeport.NUMERO, '(' ||
T.CODE_PAYS || ') ' || T.CODE_REGION || '-' || T.NUMERO,A.APP || ' ' ||
A.NUMERO || ' ' || A.RUE || ' ' || A.VILLE , count(RP.PERSONNE)
       FROM Personne P, Telephone_Personne TP, Telephone T,
           Adresse_Personne AP, Adresse A,
           Passeport,
           Reservation_Personne RP, Reservation R,  Etat_Reservation ER
       WHERE P.ID = TP.PERSONNE AND TP.TELEPHONE = T.ID
             AND P.ID = AP.PERSONNE AND AP.ADRESSE = A.ID
             AND P.PASSEPORT = Passeport.ID
             AND P.ID = RP.PERSONNE AND RP.RESERVATION = R.ID AND R.ETAT =
ER.ID AND ER.LIBELLE LIKE '%Valide%'
       GROUP BY P.NOM, P.PRENOM, Passeport.NUMERO, T.CODE_PAYS ,
T.CODE_REGION , T.NUMERO , A.APP , A.NUMERO , A.RUE , A.VILLE;
/
```

Pour afficher le contenu de la vue il suffit de l'interroger comme une table :

```
SELECT * FROM FicheClient;
```

Oracle retourne alors des résultats (plus ou moins bien formatés) tel que :

F 95 P.O. Box 398, 8371 Amet Avenue Colchester 4

Mechanique plutôt pratique afin de faciliter l'utilisation de la bdd quand on utilise souvent une requête sur le même model.

Trigger

Permet de déclencher une action quand il y a modification des données d'une table donnée.

Le code est le suivant :

```
CREATE OR REPLACE TRIGGER AlertETSA
AFTER INSERT ON Reservation FOR EACH ROW
    DECLARE
        destUSA Vol%ROWTYPE;
    BEGIN
        SELECT V.* INTO destUSA
        FROM Reservation_Vol RV, Vol V, Ville_Desservie VD, Ville Vi, Pays
P
        WHERE RV.RESERVATION = :new.ID
              AND RV.VOL = V.ID
              AND V.AEROPORT_ARRIVE = VD.AEROPORT
              AND VD.VILLE = Vi.ID
              AND Vi.PAYS = P.ID
              AND P.NOM = 'USA';

        -- no exception == this record is in USA
```

```
        DBMS_OUTPUT.PUT_LINE('Il est necessaire au client de posseder l
ETSA');

        EXCEPTION
        -- No data found == not in USA
        WHEN NO_DATA_FOUND THEN NULL;
    END;
/
```

Dans le cas présent le trigger s'active sur les inserts uniquement et nous affiche, si la destination est les USA :
Il est necessaire au client de posseder l ETSA

RechercherVolsOuverts

Exemple d'appel :

```
EXECUTE CLient.RechercherVolsOuverts('Macerata', 'Abbotsford',
TO_DATE('01/01/2016', 'DD/mm/YYYY'), TO_DATE('01/01/2019', 'DD/mm/YYYY'));
```

Exemple de retour :
num vol: CA1520
num vol: AF2306

Le code, plutôt simple, est le suivant :

```
PROCEDURE RechercherVolsOuverts(ville1 Ville.nom%TYPE,
                                ville2 Ville.nom%TYPE,
                                datemin DATE,
                                datemax DATE) IS
    BEGIN
        FOR tcur IN (SELECT v.numero
                     FROM Vol v, Aeroport a1, ville_desservie vd1, ville
v1, aeroport a2, ville_desservie vd2, ville v2, Statut s
                     WHERE v1.nom = ville1 AND v2.nom = ville2 AND v.depart
>= datemin AND v.arrivee <= datemax AND v.aeroport_depart = a1.id AND a1.id
= vd1.aeroport AND vd1.ville = v1.id AND v.aeroport_arrive = a2.id AND
a2.id = vd2.aeroport AND vd2.ville = v2.id and s.id = v.statut and
s.libelle = 'Ouvert Reservation') LOOP
            DBMS_OUTPUT.PUT_LINE('num vol: ' || tcur.numero);
        END LOOP;

END RechercherVolsOuverts;
```

Il s'agit ici d'un simple select from where dont on parcourt les résultats pour les afficher comme on le souhaite, ici simplement par le numéro qui est un moyen simple, et concis, de définir un vol.

OuvrirVol

Exemple d'appel :

```
EXECUTE OuvrirVol('ifhihfz', TO_DATE('10/09/2017', 'DD/mm/YYYY'),
TO_DATE('25/09/2017', 'DD/mm/YYYY'), 'Cressa', 'Macerata', 'Compagnie 1');
```

Exemple de retour :

```
PL/SQL procedure successfully completed.
```

Le retour ne montrant pas la bonne execution du bloc pl sql on peut le vérifier à la main via :

```
SELECT *
FROM vol;
```

Retournant, entre autre :

```
    ID NUMERO        COMPAGNIE DEPART    ARRIVEE   AEROPORT_DEPART
---------- --------------- ---------- --------- --------- ---------------
AEROPORT_ARRIVE     STATUT
--------------- ----------
    17 ifhihfz           1 10-SEP-17 25-SEP-17       1
          2      1
```

Le code est :

```
PROCEDURE OuvrirVol (
        p_numero VOL.NUMERO%TYPE,
        p_date_depart VOL.DEPART%TYPE,
        p_date_arrivee VOL.ARRIVEE%TYPE,
        p_ville_depart Ville.NOM%TYPE,
        p_ville_arrivee Ville.NOM%TYPE,
        p_compagnie Compagnie_Aerienne.NOM%TYPE
    ) AS
        v_id_compagnie          Compagnie_Aerienne.ID%TYPE;
        v_aeroport_depart       Aeroport.ID%TYPE;
        v_aeroport_arivee       Aeroport.ID%TYPE;
        v_statut_ouvert         STATUT.ID%TYPE;
            depart_notfound         EXCEPTION;
            arrivee_notfound        EXCEPTION;
            compagnie_notfound      EXCEPTION;
    BEGIN

        -- Check if compagnie exist
        BEGIN
            SELECT ID into v_id_compagnie FROM Compagnie_Aerienne WHERE NOM
= p_compagnie;
            EXCEPTION
            WHEN NO_DATA_FOUND THEN
            raise compagnie_notfound;
        END;


        -- Check if ville depart exist
        BEGIN
            SELECT A.ID INTO v_aeroport_depart
            FROM Aeroport A, Ville_Desservie VD, VILLE V
            WHERE A.ID = VD.AEROPORT AND VD.VILLE = V.ID AND V.NOM =
p_ville_depart;
            EXCEPTION
            WHEN NO_DATA_FOUND THEN
            raise depart_notfound;
        END;

        -- Check if ville arrivee exist
        BEGIN
            SELECT A.ID INTO v_aeroport_arivee
            FROM Aeroport A, Ville_Desservie VD, VILLE V
            WHERE A.ID = VD.AEROPORT AND VD.VILLE = V.ID AND V.NOM =
p_ville_arrivee;
            EXCEPTION
```

```
            WHEN NO_DATA_FOUND THEN
            raise arrivee_notfound;
        END;

        SELECT ID INTO v_statut_ouvert FROM STATUT WHERE LIBELLE LIKE
'%Ouvert%';
        dbms_output.put_line('ID : ' || v_statut_ouvert);

        INSERT INTO VOL(NUMERO, COMPAGNIE, DEPART, ARRIVEE,
AEROPORT_DEPART, AEROPORT_ARRIVE, STATUT)
        VALUES (p_numero, v_id_compagnie, TO_DATE(p_date_depart,
'DD/mm/YYYY'), TO_DATE(p_date_arrivee, 'DD/mm/YYYY'), v_aeroport_depart,
v_aeroport_arivee, v_statut_ouvert);

        EXCEPTION
        WHEN depart_notfound THEN
        dbms_output.put_line('Votre ville de depart n existe pas ou ne
possede pas d aeroport');
        WHEN arrivee_notfound THEN
        dbms_output.put_line('Votre ville d arrivee n existe pas ou ne
possede pas d aeroport');
        WHEN compagnie_notfound THEN
        dbms_output.put_line('Compagnie non trouvee');
    END OuvrirVol;
```

Le code est un peu long mais il se dégage deux parties, une vérification des entrés utilisateurs, afin de trouver les id correspondants aux villes, ect… et une 2eme partie on on réalise l'insert en lui-même. On utilise le système d'exception pour signaler un problème à l'utilisation, plutôt qu'un affichage qui passe facilement inaperçu.

EffectuerReservation

Exemple d'appel :

```
EXECUTE Client.EffectuerReservation(1, 'zefzef', 'Business');
```

Exemple de retour :

```
PL/SQL procedure successfully completed.
```

De même le retour ne confirme pas le bon fonctionnement, on vérifie par une requête de type:

```
SELECT pl.vol,pl.classe,pl.place,pl.prix, Disponible
FROM Infos_Place p, Place pl, Classe c, Vol v
WHERE v.id = 8 AND v.id = pl.vol AND pl.classe = c.id AND pl.Place = p.id
AND p.Disponible = 'F';
```

Qui retourne ici :

```
    VOL     CLASSE     PLACE       PRIX D
---------- ---------- ---------- ---------- -
     8      1          2    125 F
```

Le code du bloc pl sql est :

```
PROCEDURE EffectuerReservation(
        ppersonne Personne.ID%TYPE,
        pvol      Vol.NUMERO%TYPE,
        pclasse   Classe.LIBELLE%TYPE
    ) IS
        idPlace Place%ROWTYPE;
```

```
        pasDePlace EXCEPTION;
    BEGIN

        SELECT
            pl.vol,
            pl.classe,
            pl.place,
            pl.prix
        INTO idPlace
        FROM Infos_Place p, Place pl, Classe c, Vol v
        WHERE c.libelle = pclasse AND v.numero = pvol AND v.id = pl.vol AND
pl.classe = c.id AND pl.Place = p.id AND p.Disponible = 'T';

        -- exception if no place available
        IF (idPlace.vol IS NULL)
        THEN
            RAISE pasDePlace;
        END IF;
        -- Réserver vol
        UPDATE Infos_Place
        SET Disponible = 'F'
        WHERE idPlace.Place = Infos_Place.id;
        INSERT INTO Reservation VALUES ((Select max(id)+1 from
Reservation), 'r5', TO_DATE('1/1/2017', 'DD/mm/YYYY'), 1);
        INSERT INTO Reservation_Vol VALUES (idPlace.vol, (Select max(id)
from Reservation));
        INSERT INTO Reservation_Personne VALUES (ppersonne, (Select max(id)
from Reservation));

        EXCEPTION
        WHEN pasDePlace THEN
        dbms_output.put_line('Pas de place sur ce vol');
    END EffectuerReservation;
```

On commence par chercher si on peut trouver une place sur le vol avec les informations données, si non une exception est levée, sinon on réalise les insert dans les tables concerné, ainsi qu'un update pour signaler la place comme prise.

ListerVolsPlusCherAVG

Exemple d'appel :

```
EXECUTE ListerVolsPlusChersAVG('Business');
```

Exemple de retour :

```
Prix 500 zfehgf 3 01-JAN-17 01-JAN-17 4 3 1
Prix 1000 eekuyn 3 01-JAN-17 01-JAN-17 4 3 1
Prix 666 tottoo 3 01-JAN-17 01-JAN-17 4 3 1
Prix 999 escale 2 01-JAN-17 01-JAN-17 6 4 1
Prix 550 Paris NY 2 01-JAN-17 01-JAN-17 5 4 1
```

Le code est :

```
CREATE OR REPLACE FUNCTION avgVol(p_classe Classe.LIBELLE%TYPE)
    RETURN PLS_INTEGER IS
    moyenne PLS_INTEGER;
    BEGIN
        SELECT AVG(P.PRIX) INTO moyenne
        FROM Place P, Vol V, Statut S, Classe C
```

```
        WHERE P.VOL = V.ID AND V.STATUT = S.ID AND S.LIBELLE LIKE
'%Ouvert%'
            AND P.CLASSE = C.ID AND C.LIBELLE = p_classe;
        RETURN moyenne;
    END;
/

CREATE OR REPLACE PROCEDURE ListerVolsPlusChersAVG (
    p_classe Classe.LIBELLE%TYPE
) AS
    moyenne PLS_INTEGER;
    BEGIN
        moyenne := avgVol(p_classe);
        FOR rec IN (
            SELECT V.*, P.Prix
            FROM Vol V, PLACE P, STATUT S, CLASSE C
            WHERE V.ID = P.VOL AND P.CLASSE = C.ID AND C.LIBELLE = p_classe
              AND V.STATUT = S.ID AND S.LIBELLE LIKE '%Ouvert%'
              AND P.Prix > moyenne
        ) LOOP
        dbms_output.put_line('Prix ' ||rec.Prix ||' '||rec.NUMERO || ' ' ||
rec.COMPAGNIE || ' ' || rec.DEPART || ' ' || rec.ARRIVEE
                                || ' ' || rec.AEROPORT_DEPART || ' ' ||
rec.AEROPORT_ARRIVE || ' ' || rec.STATUT);
        END LOOP;

    END;
/
```

Ici le code est un peu plus compliqué, nous avons utilisé une fonction pour calculer la moyenne du prix des vols. Après l'appel de cette fonction on parcourt les vols et on vérifie si leur prix est en dessous ou non de la moyenne. Si il est au-dessus on l'affiche. Ici on affiche le prix, les villes, et les dates car on ne sait pas forcement quel vol va ressortir et cela permet de l'identifier plus facilement.

ModifierAdresse

Exemple d'appel :

```
EXECUTE Client.ModifierAdresse (1,1,12, 'K','Ap #595-7594 Quam
Rd.','Strona','Q8V9H1');
```

Exemple de retour :

```
PL/SQL procedure successfully completed.
```

Pour en vérifier le bon fonctionnement ici :

```
Select * from adresse where id=1;
```

Qui affiche, avec tout le côté artistique d'oracle:

```
---------- ---------- -
RUE
--------------------------------------------------------------------------------
-----
VILLE
--------------------------------------------------------------------------------
-----
CODE_P       PAYS    PROVINCE
------ ---------- ----------
```

```
     1     12 K
Ap #595-7594 Quam Rd.
Strona
Q8V9H1       3     24
```

Le code de cette modification d'adresse est :

```
PROCEDURE ModifierAdresse (
        client      Personne.ID%TYPE,
        id_adresse  Adresse.ID%TYPE,
        p_numero      Adresse.NUMERO%TYPE,
        p_app         Adresse.APP%TYPE,
        p_rue         Adresse.RUE%TYPE,
        p_ville       Adresse.VILLE%TYPE,
        p_codepostal  Adresse.CODE_POSTAL%TYPE
    ) IS
    nbClient INTEGER;
    ppays Adresse.PAYS%TYPE;
    pprovince Adresse.PROVINCE%TYPE;
    BEGIN
        -- check if adress binded by 2 or more persons
        -- if adress binded by 2 or more, create new one and point to the
new
        -- if not, modify directly the adress
        -- Pour le présent on fait le minimum
        Select count(*) into nbClient from Adresse A, Adresse_Personne AP
where AP.Personne = client AND AP.Adresse = id_adresse;
        if (nbClient=1) then
            UPDATE Adresse
            SET NUMERO = NVL(p_numero, NUMERO),
                APP = NVL(p_app, APP),
                RUE = NVL(p_rue, RUE),
                VILLE = NVL(p_ville, VILLE),
                CODE_POSTAL = NVL(p_codepostal, CODE_POSTAL)
            WHERE ID = id_adresse;
        else
            Select pays, province into ppays, pprovince from Adresse A
where id_adresse = A.id;
            INSERT INTO Adresse(NUMERO, APP, RUE, VILLE, CODE_POSTAL, PAYS,
PROVINCE)
            VALUES (p_numero, p_app, p_rue, p_ville, p_codepostal, ppays,
pprovince);
            UPDATE Adresse_Personne AP SET Adresse = (select max(id) from
adresse) WHERE AP.adresse = id_adresse and AP.Personne = client;
        end if;
    END ModifierAdresse;
```

Ici deux cas à distinguer, si l'adresse correspond à une seule personne ou à plusieurs. Si c'est une seule personne il suffit de faire un update… Sinon il faut ajouter une nouvelle ligne dans Adresse puis update Adresse_Personne. Ce qui permet de ne pas modifier l'adresse de tout les clients.

Conclusion

Travail qui permet d’approfondir la gestion des bases de données avec trigger, vues, et procédures ! On note que les procédures sont un peu (beaucoup) lourdes à debug car les erreurs d’oracle sont... assez étranges.