PL/SQL

BDTN

SQL dynamique

Utilisé pour créer une instruction SQL dont on ne connait pas tous les éléments à l'avance

- Est une chaine de caractère placée dans un sous-programme
- Les noms de colonnes, conditions… peuvent varier
- Permet d'exécuter dans du code PL/SQL des instructions de définition de données

EXECUTE IMMEDIATE

```
CREATE OR REPLACE PROCEDURE drop_any_table(p_tablename VARCHAR2) IS
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE ' || p_tablename;
END
```

```
CREATE OR REPLACE PROCEDURE drop_any_table(p_tablename VARCHAR2) IS
 V_dynamic_stat VARCHAR2(50);
BEGIN
  V_dynamic_stat := 'DROP TABLE ' || p_tablename;
  EXECUTE IMMEDIATE v_dynamic_stat;
END;
```

SQL Dynamique

```
CREATE OR REPLACE PROCEDURE add_row( p_tablename VARCHAR2,
                                     p_id type_show_tsh.tsh_id%TYPE,
                                     p_name type_show_tsh.tsh_name%TYPE)
IS
BEGIN
EXECUTE IMMEDIATE 'INSERT INTO ' || p_tablename ||
  ' VALUES(' || p_id || ',''' || p_name || ''')';
END;
BEGIN
 add_row('type_show_tsh', 10, ‘Jazz');
END;
```

```
CREATE OR REPLACE PROCEDURE add_row(
                        p_tablename VARCHAR2,
                        p_id type_show_tsh.tsh_id%TYPE,
                        p_name type_show_tsh.tsh_name%TYPE)

IS

BEGIN

   EXECUTE IMMEDIATE 'INSERT INTO ' || p_tablename ||
                                ' VALUES(' || p_id || ',''' || p_name || ''')';

END;

/

BEGIN

 add_row('type_show_tsh', 10, 'Jazz');

END;
```

SQL Dynamique

Modifier les prix des spectacles

- Pourcentage en paramètre d'entrée

Selon un critère portant soit sur le spectacle soit sur le type de prix

- Colonne en paramètre
- Valeur de comparaison en paramètre

Ordre UPDATE

UPDATE has_price_hpr

SET hpr_seat_price = hpr_seat_price * ?

WHERE **??** = ?

```
CREATE OR REPLACE PROCEDURE raise_shw_price (
                 column_value IN NUMBER,
                hpr_column IN VARCHAR2,
                coef IN NUMBER)
 IS
   v_column VARCHAR2(30);
   sql_stmt  VARCHAR2(200);

BEGIN
…
```

```
SELECT column_name
   INTO v_column
   FROM user_tab_cols
   WHERE table_name = 'HAS_PRICE_HPR'
        AND column_name = UPPER(hpr_column);


 sql_stmt := 'UPDATE has_price_hpr
                  SET hpr_seat_price = hpr_seat_price * :1
                  WHERE ' || v_column || ' = :2';

 EXECUTE IMMEDIATE sql_stmt USING coef, column_value;
```

```
IF SQL%ROWCOUNT > 0 THEN
   DBMS_OUTPUT.PUT_LINE('Prices have been updated for: ' ||
                                          hpr_column ||
                                          ' = ' ||
                                         column_value);
END IF;

 EXCEPTION
 WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE ('Invalid Column: ' || hpr_column);
END raise_shw_price
```

Triggers (déclencheurs)

Un trigger (déclencheur) est un programme PL/SQL

- Qui se déclenche lors d'un évènement
- Condition d'exécution
- Est stocké

En particulier pour :

- Valider certaines contraintes
- Tracer les modifications réalisées sur les tables

Mais pas pour se substituer à des contraintes

- FOREIGN KEY
- CHECK

Evènements possibles

Instructions LMD

- Sur une table
- Sur une vue

Instructions LDD

- CREATE, ALTER

Evènements système

- Connexion d’un utilisateur, arrêt de la base

Exemple de trigger

```
CREATE OR REPLACE TRIGGER log_hpr_price_inc
  AFTER UPDATE OF hpr_seat_price ON has_price_hpr FOR EACH ROW
  WHEN (NEW.hpr_seat_price < OLD.hpr_seat_price)
BEGIN
  INSERT INTO log_price(l_user, l_date,l_hpr_id, l_shw_id,  l_new_price,
 l_old_price)
  VALUES(USER, SYSDATE, :NEW.hpr_tpr_id, :NEW.hpr_shw_id,
 :NEW.hpr_seat_price, :OLD.hpr_seat_price);
END;
```

Spécifier l’évènement

AFTER / BEFORE

Type d’évènement

- AFTER UPDATE
- BEFORE UPDATE
- AFTER INSERT OR UPDATE OR DELETE

Spécification table / colonne

- Uniquement pour les ordres UPDATE
  - ON fournisseur
  - ON emp
  - OF remise ON fournisseur
  - OF salary, com ON emp

Niveau ligne ou niveau instruction

FOR EACH ROW

- L’ordre SQL peut affecter plusieurs lignes
- Avec FOR EACH ROW = Le trigger se déclenche pour chaque ligne
- Trigger niveau ligne (row-level trigger)

Ne rien mentionner

- Trigger niveau instruction (statement-level trigger)

Référence aux anciennes et/ou nouvelles valeurs des lignes

Uniquement dans les triggers niveau ligne

INSERT

- :NEW seulement

DELETE

- :OLD seulement

UPDATE

- :OLD et/ou :NEW

WHEN

Optionnel

Permet de préciser une condition pour le déclenchement effectif

WHEN (NEW.hpr_seat_price < OLD.hpr_seat_price)

Tester le type d’évènement

Déclenchement sur plusieurs évènements

- INSERTING
- DELETING
- UPDATING

IF INSERTING THEN ….

```
CREATE OR REPLACE TRIGGER ck_hpr_price
  BEFORE INSERT OR UPDATE OF hpr_seat_price ON has_price_hpr FOR EACH ROW

BEGIN
    IF INSERTING THEN
        IF TO_CHAR(SYSDATE, 'DY') = 'DIM.' THEN
            RAISE_APPLICATION_ERROR(-20201, 'Pas d''insertion de prix le dimanche');
        END IF;
    ELSIF :NEW.hpr_seat_price < :OLD.hpr_seat_price THEN
        RAISE_APPLICATION_ERROR(-20202, ‘Update : Pas de baisse de prix');
   END IF;

END
```

```
UPDATE has_price_hpr
  set hpr_seat_price = hpr_seat_price - 1;
```

```
Erreur commençant à la ligne: 16 de la commande -
UPDATE has_price_hpr
  set hpr_seat_price = hpr_seat_price - 1
Rapport d'erreur -
Erreur SQL : ORA-20202: Pas de baisse de prix
ORA-06512: at "BDD1.CK_HPR_PRICE", line 7
ORA-04088: error during execution of trigger 'BDD1.CK_HPR_PRICE'
```

Nommer / Activer / Supprimer

- Nom : Faire apparaître le nom de la table
  - EMP_CK_SALARY
- Activer / désactiver
  - ALTER TRIGGER emp_ck_salary disable;
  - ALTER TRIGGER emp_ck_salary enable;
- Supprimer
  - DROP TRIGGER emp_ck_salary;
- Lister
  - SELECT trigger_name, status FROM USER_TRIGGERS;

Transactions

COMMIT, ROLLBACK

- Ne sont pas autorisés dans les triggers