Morgado-Samagaio Jonathan

PABD / TP12 Oracle

Table des matières

1. Introduction

Dans ce TP, nous allons voir comment utiliser les procédures stockées

2. Reprise des blocs PL/SQL en procédures stockées

Nous commençons par reprendre nos blocs des TPs précédents.

1) Ajouter une montagne :

```
--1.1.a
CREATE OR REPLACE PROCEDURE ajouterMonatagne(vNom Montagne.nom%TYPE, vAlt Montagne.altitude%TYPE, vCha
    n NUMBER;
    altError EXCEPTION;
    nomError EXCEPTION;

BEGIN
    --Vérification de l'existence de la montagne
    SELECT COUNT(*) INTO n
    FROM Montagne
    WHERE nom = vNom;
    IF (n > 0) THEN
        RAISE nomError;
    END IF;
    --Vérification de l'altitude
    IF (vAlt <= 0) THEN
        RAISE altError;
    END IF;
    --Insérer nouvelle montagne
    INSERT INTO Montagne(nm, nom, altitude, chaine)
    VALUES (seq_montagne.NEXTVAL, vNom, vAlt, vCha);
    DBMS_OUTPUT.PUT_LINE('Montagne '||vNom||' ajoutée.');
    --Valider (fin de commande)
    COMMIT;

EXCEPTION
    WHEN nomError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Montagne existante !');
    WHEN altError THEN
```

2) Ajouter un pays :

```
--1.1.b
CREATE OR REPLACE PROCEDURE ajouterPays(vNom Pays.Nom%TYPE, vCap Pays.Capitale%TYPE, vSup Pays.Superficie%
    vTestP NUMBER;
    vTestI3 NUMBER;
    vTestI2 NUMBER;
    vTestC NUMBER;
    supError EXCEPTION;
    popError EXCEPTION;
    nomPError EXCEPTION;
    iso3Error EXCEPTION;
    iso2Error EXCEPTION;
    contError EXCEPTION;

BEGIN
    --Vérifications dela superficie et de la population
    IF (vSup <= 0) THEN
        RAISE supError;
    END IF;
    IF (vPop <= 0) THEN
        RAISE popError;
    END IF;
    SELECT COUNT(*) INTO vTestP FROM Pays WHERE nom = vNom;
    SELECT COUNT(*) INTO vTestI3 FROM Pays WHERE iso3 = vIs3;
    SELECT COUNT(*) INTO vTestI2 FROM Pays WHERE iso2 = vIs2;
    SELECT COUNT(*) INTO vTestC FROM Pays WHERE nomc = vCon;
    IF vTestP >= 1 THEN
        RAISE nomPError;
    END IF;
    IF vTestI3 >= 1 THEN
        RAISE iso3Error;
    END IF;
    IF vTestI2 >= 1 THEN
    RAISE iso2Error;
    END IF;
    IF vTestC < 1 THEN
    RAISE contError;
    END IF;
    --Insertion du nouveau Pays
    INSERT INTO Pays (nom, capitale, superficie, population, iso3, iso2, nomc)
    VALUES (vNom, vCap, vSup, vPop, vIs3, vIs2, vCon);
    --Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Pays '||vNom||' ajouté.');
    --Validation
    COMMIT;
EXCEPTION
    WHEN supError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Superficie invalide !');
    WHEN popError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Population invalide !');
    WHEN nomPError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Pays existant !');
    WHEN iso3Error THEN
```

3) Ajouter une montagne a un pays :

```
--1.1.c
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE ajouterMontagnePays(vNomM Montagne.nom%TYPE, vAlt Montagne.altitude%TYPE, vCha
    vTestM NUMBER;
    vTestP NUMBER;
    altError EXCEPTION;
    nomMError EXCEPTION;
    nomPError EXCEPTION;

BEGIN
    --Vérification de l'altitude
    IF vAlt < 0 THEN
        RAISE altError;
    END IF;
    SELECT COUNT(*) INTO vTestM FROM Montagne WHERE nom = vNomM;
    SELECT COUNT(*) INTO vTestP FROM Pays WHERE nom = vNomP;
    --Vérification de l'existance de la montagne
    IF vTestM >= 1 THEN
        RAISE nomMError;
    END IF;
    --Verification de l'existance du pays
    IF vTestP < 1 THEN
        RAISE nomPError;
    END IF;
    --Ajoute de la montagne
    INSERT INTO Montagne(nm, nom, altitude, chaine)
    VALUES(seq_montagne.NEXTVAL, vNomM, vAlt, vChaine);
    DBMS_OUTPUT.PUT_LINE('Montagne'||vNomM||' ajoutée.');
    --Ajout de la montagne dans le pays
    INSERT INTO Localiser(nm, nomp)
    VALUES (seq_montagne.CURRVAL, vNomP);
    DBMS_OUTPUT.PUT_LINE('Montagne'||vNomM||' ajoutée dans'||vNomP||'.');
    --Validation
    COMMIT;
EXCEPTION
    WHEN altError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Altitude invalide !');
    WHEN nomMError THEN
        ROLLBACK;
```

4) Suppression d’un pays

```
--1.1.d
CREATE OR REPLACE PROCEDURE supprimerPays(vNom Pays.
    vTestP NUMBER;
    nomPError EXCEPTION;

BEGIN
    --Verification de l'existence du pays
    SELECT COUNT(*) INTO vTestP FROM Pays WHERE nom
    IF (vTestP < 1) THEN
        RAISE nomPError;
    END IF;
    --Suppression du pays
    DELETE FROM Localiser WHERE nomp = vNom;
    DELETE FROM Traverser WHERE nomp = vNom;
    DELETE FROM Frontiere WHERE nomf = vNom OR nomp
    DELETE FROM Pays WHERE nom = vNom;
    --VALIDATIONCOMMIT;
EXCEPTION
    WHEN nomPError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Pyas inexistant !');
    WHEN OTHERS THEN
```

Pour pouvoir appeler ces procédures, nous allons écrire (pour ajouterMontagne) :

```
--1.2
SET SERVEROUTPUT ON;

ACCEPT pNom PROMPT 'Saisie un nom de montag
ACCEPT pAlt PROMPT 'Saisir l altitude de la
ACCEPT pCha PROMPT 'Saisir la chaine de mon
```

On fait écrire à l'utilisateur tous les paramètres d'entrée et on appelle la commande avec EXEC suivit du nom de la procédure.

En testant pour chaque procédure, les erreurs sont bien soulevées comme pour le bloc sans procédure.

3. Nouvelle procédure stockée

Nous allons maintenant créer une procédure de 0. Nous allons créer une procédure permettant d'ajouter une frontière. Voici la procédure :

```
--2.1
CREATE OR REPLACE PROCEDURE AjouterFrontiere(vNomP Frontiere.nomp%TYPE, vNomF Frontiere.nomf%TY
    n NUMBER;
    nomPError EXCEPTION;
    nomFError EXCEPTION;
    frontError EXCEPTION;
    longError EXCEPTION;

BEGIN
    IF (vLong <= 0) THEN
        RAISE longError;
    END IF;
    SELECT COUNT(*) INTO n FROM Pays WHERE nom = vNomP;
    IF (n < 1) THEN
        RAISE nomPError;
    END IF;
    SELECT COUNT(*) INTO n FROM Pays WHERE nom = vNomF;
    IF (n < 1) THEN
        RAISE nomFError;
    END IF;
    SELECT COUNT(*) INTO n FROM Frontiere WHERE nomp = vNomP AND nomf = vNomF;
    IF (n > 0) THEN
        RAISE frontError;
    END IF;
    INSERT INTO Frontiere(nomp, nomf, longueur)
    VALUES (vNomP, vNomF, vLong);
    INSERT INTO Frontiere(nomp, nomf, longueur)
    VALUES (vNomF, vNomP, vLong);
    DBMS_OUTPUT.PUT_LINE('Frontiere ajoutée');
    COMMIT;
EXCEPTION
    WHEN nomPError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Pays Inexistant');
    WHEN nomFError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Frontiere inexistente');
    WHEN frontError THEN
        ROLLBACK;
```

Les erreurs étant similaire à des précédentes, elles fonctionnent correctement.

4. Conclusion

Dans ce TP, nous avons pu voir comment utiliser les procédures stockées en PL/SQL. Elles nous permettent de stockée un bout de code avec un nom et de l'appeler quand on veut.