Morgado-Samagaio Jonathan

PABD / TP9 Oracle

Table des matières

1. Introduction

Dans ce TP, nous allons voir comment utiliser les exceptions dans des blocs PL/SQL

2. Bloc PL /SQL d'ajout d'une montagne

On commence par reprendre un bloc d'un TP précédent et, celui de l'ajoute des montagnes et on y ajoute des exceptions. Voici le bloc :

```
--1.1
SET SERVEROUTPUT ON;

ACCEPT pNom PROMPT 'Saisie un nom de montagne :';
ACCEPT pAlt PROMPT 'Saisir l altitude de la montagne';
ACCEPT pCha PROMPT 'Saisir la chaine de montagne :';

DECLARE
    vNom Montagne.nom%TYPE := '&pNom';
    vAlt Montagne.Altitude%TYPE := &pAlt;
    vCha Montagne.Chaine%TYPE := '&pCha';
    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
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Altitude invalide !');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE_APPLICATION_ERROR(-20000, 'ERREUR IMPREVUE !');
END;
/
```

Nous pouvons voir l'apparition de l partie EXCEPTION avec des variables de ce type.

Si l'on test avec une altitude négative, nous obtenons comme retour :

```
END;
Altitude invalide !


Procédure PL/SQL terminée.
```

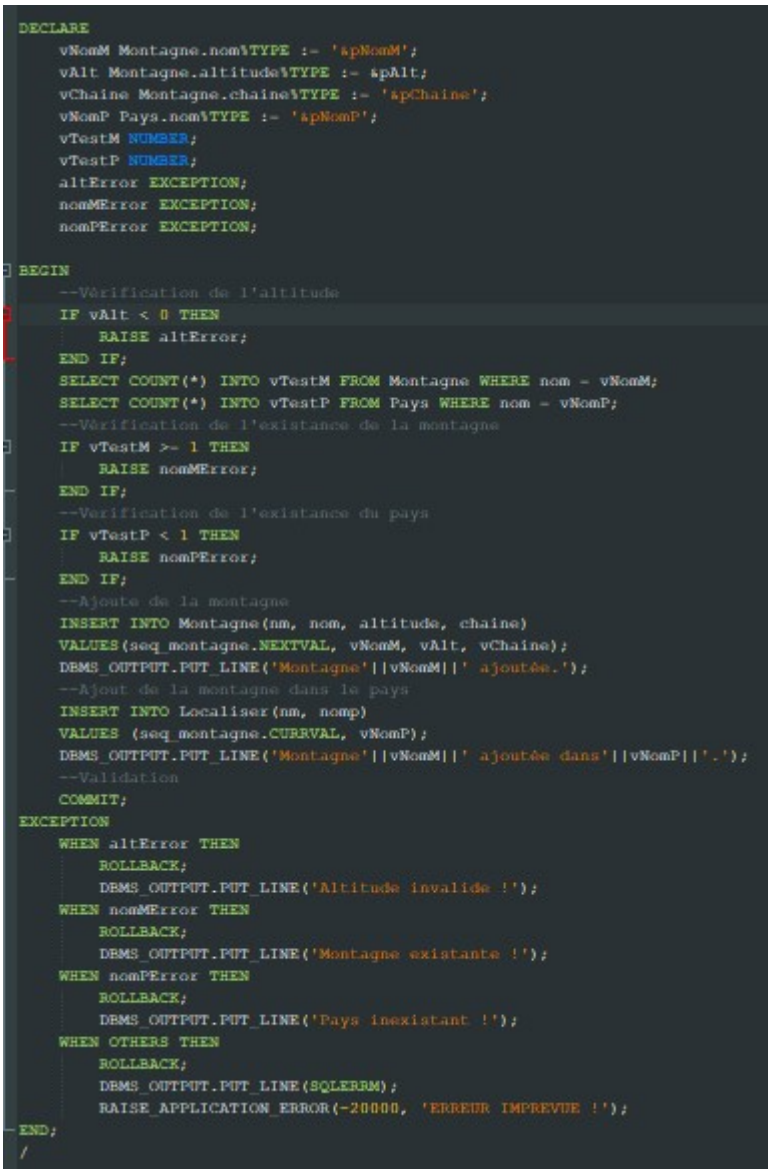
Avec un nom déjà existant :

```
END;
Montagne existante !


Procédure PL/SQL terminée.
```

3. Bloc PL /SQL d'ajout d'une montagne à un pays

Nous reprenons le bloc permettant d'ajouter une montagne à un pays. On ajoute maintenant les exceptions. Voici le code :



```
DECLARE
    vNomM Montagne.nom%TYPE := '&pNomM';
    vAlt Montagne.altitude%TYPE := &pAlt;
    vChaine Montagne.chaine%TYPE := '&pChaine';
    vNomP Pays.nom%TYPE := '&pNomP';
    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;
    --Vérification de l'existance du pays
    IF vTestP < 1 THEN
        RAISE nomPError;
    END IF;
    --Ajout 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;
        DBMS_OUTPUT.PUT_LINE('Montagne existante !');
    WHEN nomPError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Pays inexistant !');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE_APPLICATION_ERROR(-20000, 'ERREUR IMPREVUE !');
END;
/
```

La partie avec les ACCEPT n'a pas été mise dans la capture car le code est trop grand. Les tests avec l'altitude et le nom de la montagne ayant déjà été fait dans la question précédente, nous allons seulement tester le nom du pays. Si on donne un nom de pays inexistant, on obtient :

```
END;
Pays inexistant !


Procédure PL/SQL terminée.
```

4. Bloc PL/SQL d'ajout d'un pays

Nous reprenons encore une fois un ancien bloc pour l'ajout d'un pays auquel on va ajouter des exceptions. Voici le bloc :

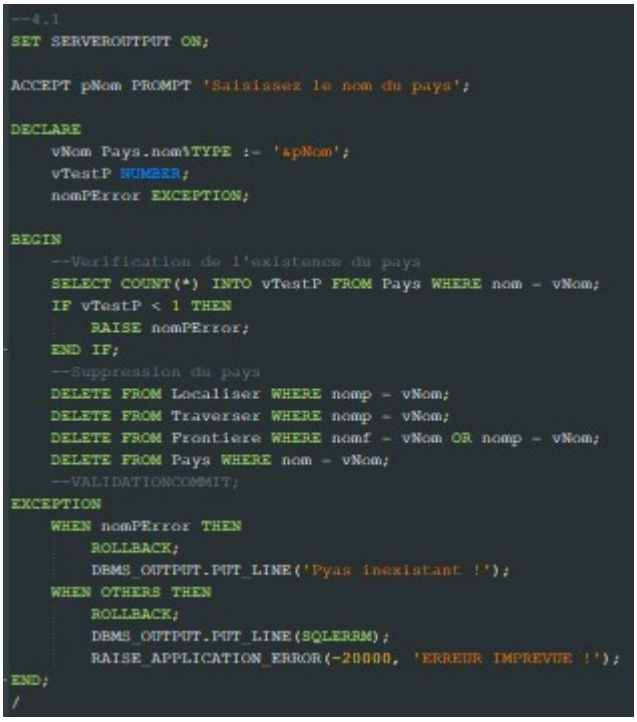
```sql
DECLARE
    --Affectations des paramètres aux variables locales
    vNom Pays.Nom%TYPE := '&pNom';
    vCap Pays.Capitale%TYPE := '&pCap';
    vSup Pays.Superficie%TYPE := &pSup;
    vPop Pays.Population%TYPE := &pPop;
    vIs3 Pays.Iso3%TYPE := '&pIs3';
    vIs2 Pays.Iso2%TYPE := '&pIs2';
    vCon Pays.NomC%TYPE := '&pCon';
    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
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Code iso3 invalide !');
    WHEN iso2Error THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Code iso2 invalide !');
    WHEN contError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Continent inexistant !');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE_APPLICATION_ERROR(-20000, 'ERREUR IMPREVUE !');
END;
```

Lorsque tester avec différents cas, nous obtenons bien les messages voulues.

5. Création d'un bloc PL/SQL de suppression d'un pays

Nous allons maintenant créer un bloc de toute pièce permettant la suppression d'un pays. Voici le code :



```
--4.1
SET SERVEROUTPUT ON;

ACCEPT pNom PROMPT 'Saisissez le nom du pays';

DECLARE
    vNom Pays.nom%TYPE := '&pNom';
    vTestP NUMBER;
    nomPError EXCEPTION;

BEGIN
    --Verification de l'existence du pays
    SELECT COUNT(*) INTO vTestP FROM Pays WHERE nom = vNom;
    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 = vNom;
    DELETE FROM Pays WHERE nom = vNom;
    --VALIDATIONCOMMIT;
EXCEPTION
    WHEN nomPError THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Pyas inexistant !');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RAISE_APPLICATION_ERROR(-20000, 'ERREUR IMPREVUE !');
END;
/
```

La vérification étant la même que pour les derniers blocs, nous n'allons pas faire de test dessus. Cependant, retirer un pays fonctionne correctement.

6. Conclusion

Dans ce TP, nous avons pu voir comment utiliser les EXCEPTIONS dans le bloc adapté. Nous avons vu les déclarations de variables de type exception et comment RAISE ces exceptions dans le bloc BEGIN. Nous avons ensuite vu comment les gérer dans le bloc EXCEPTIONS.