Morgado-Samagaio Jonathan

PABD / TP6 Oracle

Table des matières

1. Introduction

Dans ce TP, nous allons voir comment utiliser le PL/SQL. Cela permet de réduire les échanges réseaux entre le client et le serveur. Nous allons donc commencer sans prendre en compte les contraintes de nos tables.

2. Utilisation d'un bloc PL/SQL

1) On commence par exécuter le code fournis avec les bonnes valeurs :

```
--1.1
SET SERVEROUTPUT ON;
ACCEPT pNom PROMPT 'Saisir un nom de montagne: ';
ACCEPT pAlt PROMPT 'Saisir l''altitude de la montagne: ';
ACCEPT pCha PROMPT 'Saisir la chaîne de montagnes: ';

DECLARE
    vNom Montagne.Nom%TYPE  := '&pNom';
    vAlt Montagne.Altitude%TYPE  := &pAlt;
    vCha Montagne.Chaine%TYPE  := '&pCha';

BEGIN
    INSERT INTO Montagne(nm, nom, altitude, chaine)
    VALUES(seq_montagne.NEXTVAL, vNom, vAlt, vCha);
    DBMS_OUTPUT.PUT_LINE('Montagne'||vNom||' ajoutée.');
    COMMIT;
END;
/
```

Le bloc s'est exécuté avec succès :

```
MontagnePic d'Aneto ajoutée.


Procédure PL/SQL terminée.
```

2) On va maintenant vérifier le contenu de la table Montagne.

```
--1.2
SELECT *
FROM Montagne;
```

Nous pouvons voir que la montagne a bien été ajouter :

NM	NOM	ALTITUDE	CHAINE
261	Pic d'Aneto	3404	Pyrénées
1	Kilimandjaro - Kibo	5892	Vallée du grand rift
2	Mont Kenya	5199	Vallée du grand rift
3	Kilimandjaro - Mawenzi	5149	Vallée du grand rift
4	Mont Stanley	5109	Rwenzori
5	Mont Speke	4890	Rwenzori
6	Mont Baker	4844	Rwenzori
7	Mont Emin	4798	Rwenzori
8	Mont Gessi	4715	Rwenzori
9	Mont Méru	4565	Vallée du grand rift
10	Ras Dashan	4550	Plateaux d'Éthiopie

3) On va maintenant essayer d'exécuter à nouveau le bloc avec les mêmes valeurs. Le code s'exécute sans problèmes et si l'on regarde la table Montagne :

NM	NOM	ALTITUDE	CHAINE
261	Pic d'Aneto	3404	Pyrénées
262	Pic d'Aneto	3404	Pyrénées
1	Kilimandjaro - Kibo	5892	Vallée du grand

Nous pouvons voir que la montagne a été ajouter une seconde fois. En effet, nous ne gérons pas les contraintes donc on peut ajouter deux fois la même montagne.

3. Création d'un bloc PL/SQL

1) Nous allons créer un bloc permettant d'ajouter un pays :

```
--2.1
ACCEPT pNom PROMPT 'Saisir le nom du pays : ';
ACCEPT pCap PROMPT 'Saisir la capitale du pays : ';
ACCEPT pSup PROMPT 'Saisir la superficie du pays : ';
ACCEPT pPop PROMPT 'Saisir la population du pays : ';
ACCEPT pIs3 PROMPT 'Saisir le code ISO3 du pays : ';
ACCEPT pIs2 PROMPT 'Saisir le code ISO2 du pays : ';
ACCEPT pCon PROMPT 'Saisir le contnent du pays : ';

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';

BEGIN
    --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;
END;
/
```

2) Nous essayons maintenant d'ajouter un nouveau pays avec les valeurs fournis. La commande s'exécute sans problèmes.

3) Nous vérifions maintenant le contenu de la table :

NOM	CAPITALE	SUPERFICIE
Gaule	Lugdunum	494000
Émirats arabes unis	Abou Dabi	82880
Nigeria	Abuja	923768
Ghana	Accra	238540
Éthiopie	Addis-Abeba	1127127
Jordanie	Amman	89342
Pays-Bas	Amsterdam	41530
Turquie	Ankara	783562
Madagascar	Antananarivo	587041
Érythrée	Asmara	121320
Paraguay	Asuncion	406752

Nous pouvons voir que le pays a bien été ajouté.

4. Modification d'un bloc PL/SQL

1) Nous allons commencer par modifier le bloc PL/SQL de création de pays pour pouvoir ajouter deux conditions pour tester si la superficie et la population données sont négatives.

```
--3.1
ACCEPT pNom PROMPT 'Saisir le nom du pays: ';
ACCEPT pCap PROMPT 'Saisir la capitaledu pays: ';
ACCEPT pSup PROMPT 'Saisir la superficiedu pays: ';
ACCEPT pPop PROMPT 'Saisir la populationdu pays: ';
ACCEPT pIs3 PROMPT 'Saisir le code ISO3du pays: ';
ACCEPT pIs2 PROMPT 'Saisir le code ISO2du pays: ';
ACCEPT pCon PROMPT 'Saisir le contnentdu pays: ';

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';

BEGIN
    --Vérifications dela superficie et de la population
    IF (vSup <= 0) THEN
        RAISE_APPLICATION_ERROR(-20001, 'Superficie négative');
    END IF;
    IF (vPop <= 0) THEN
        RAISE_APPLICATION_ERROR(-20002, 'Population négative')
    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;
END;
/
```

2) Nous allons maintenant tester de rajouter un pays avec une superficie négative. Nous obtenons le message d'erreur suivant :

```
Rapport d'erreur -
ORA-20001: Superficie négative
ORA-06512: à ligne 14
```

Avec une population négative :

```
Rapport d'erreur -
ORA-20002: Population négative
ORA-06512: à ligne 17
```

Nous obtenons bien les bon message d’erreurs.

5. Conclusion

Dans ce TP, nous avons pu voir comment créer et modifier des blocs PL/SQL pour limiter les échanges réseaux. Nous avons vu comment demander à l’utilisateur de rentrer des valeurs et comment les vérifier.