Aharouite Ahmed

PABD / TP10 Oracle

Table des matières

1. Introduction

Ce TP porte sur les mécanismes de gestion d'erreurs appelés EXCEPTIONS systèmes (ou prédéfinies) déclenchées par le SGBD.

Nous utiliserons également le mécanisme de PRAGMA. Pour mémoire

- 2290 : code erreur système d'une contrainte CHECK
- 2291 : code erreur système d'une contrainte FOREIGN KEY (référence inconnue)
- 2292 : code erreur système d'une contrainte FOREIGN KEY (référence utilisée)

Un bloc PL/SQL est alors formé comme suit :

```
DECLARE
     -- variables
BEGIN
     -- traitements

EXEPTION
     -- traitements des erreurs

END;
/
```

2. Manipulation de la base de données

2.1. Modification du bloc PL /SQL d'ajout d'un pays

(1) Reprendre le bloc PL/SQL permettant d'ajouter un pays. Modifiez le bloc de telle sorte qu'il soit constitué uniquement d'exceptions systèmes prenant en charge les erreurs suivantes.

```
DECLARE
-- variables
vCio Pays.cio%TYPE := '&pCio';
vNom Pays.nom%TYPE := '&pNom';
n NUMBER := 0;
PAYS_EXISTANT EXCEPTION;
PRAGMA EXCEPTION_INIT(PAYS_EXISTANT, -00001);

BEGIN
-- traitements
    INSERT INTO Pays (cio, nom) VALUES (vCio, vNom);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Pays '||vCio||' '||vNom||' '||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('Pays enregistré');
COMMIT;

EXCEPTION
-- traitements des erreurs
WHEN PAYS_EXISTANT THEN
    DBMS_OUTPUT.PUT_LINE('Le pays existe déjà');

END;
/
```

(2) Testez votre bloc PL/SQL avec les différents cas possibles prévus.

```
vCio Pays.cio%TYPE := 'FRA';
vNom Pays.nom%TYPE := 'France';
```

```
Le pays existe déjà

Procédure PL/SQL terminée.
```

```
vCio Pays.cio%TYPE := 'NEW';
vNom Pays.nom%TYPE := 'NewPays';
```

```
Pays NEW NewPays  ajouté.
Pays enregistré

Procédure PL/SQL terminée.
```

2.2. Modification du bloc PL/SQL d'ajout d'un joueur

(1) Reprendre le bloc PL/SQL permettant d'ajouter un joueur. Modifiez le bloc de telle sorte qu'il soit constitué uniquement d'exceptions systèmes prenant en charge les erreurs suivantes.

```
ACCEPT pPre PROMPT 'Saisir le prénom du joueur : ';
ACCEPT pNom PROMPT 'Saisir le nom du joueur : ';
ACCEPT pGen PROMPT 'Saisir le genre du joueur : ';
ACCEPT pNat PROMPT 'Saisir la nationalité du joueur : ';
ACCEPT pVns PROMPT 'Saisir la ville de naissance du joueur : ';
ACCEPT pPns PROMPT 'Saisir le pays de naissance du joueur : ';
ACCEPT pDns PROMPT 'Saisir la date de naissance du joueur : ';

DECLARE
vNj Joueur.nj%TYPE := SEQ_JOUEUR.nextval;
vPre Joueur.pre%TYPE := '&pPre';
vNom Joueur.nom%TYPE := '&pNom';
vGen Joueur.gen%TYPE := '&pGen';
vNat Joueur.nat%TYPE := '&pNat';
vVns Joueur.vns%TYPE := '&pVns';
vPns Joueur.pns%TYPE := '&pPns';
vDns Joueur.dns%TYPE := '&pDns';
n NUMBER := 0;
p NUMBER := 0;
GENRE_INVALIDE EXCEPTION;
PRAGMA EXCEPTION_INIT (GENRE_INVALIDE, -2290);
PAYS_INCONNU EXCEPTION;
PRAGMA EXCEPTION_INIT (PAYS_INCONNU, -2291);

BEGIN
-- Insertion du nouveau Joueur
INSERT INTO Joueur (nj, pre, nom, gen, nat, vns, pns, dns) VALUES (vNj, vPre, vNom, vGen, vNat, vVns, vPns, vDns);
-- Affichage d'un message
DBMS_OUTPUT.PUT_LINE('Joueur '||vNj||' '||vPre||' '||vNom||' '||vGen||' '||vNat||' '||vVns||' '||vPns||' '||vDns||' ajouté.');
DBMS_OUTPUT.PUT_LINE('Joueur enregistré');
-- Validation
COMMIT;

EXCEPTION
-- traitements des erreurs
WHEN GENRE_INVALIDE THEN
    DBMS_OUTPUT.PUT_LINE('Genre invalide');
WHEN PAYS_INCONNU THEN
    DBMS_OUTPUT.PUT_LINE('Pays saisi inconnu');

END;
/
```

(2) Testez votre bloc PL/SQL avec les différents cas possibles prévus

```
Joueur 372 AHMED AHAROUITE H FRA TOULOUSE FRA 20/09/01 ajouté.
Joueur enregistré


Procédure PL/SQL terminée.
```

```
nouveau :DECLARE
vNj Joueur.nj%TYPE := SEQ_JOUEUR.nextval;
vPre Joueur.pre%TYPE := 'AHMED';
vNom Joueur.nom%TYPE := 'AHAROUITE';
vGen Joueur.gen%TYPE := 'M';
vNat Joueur.nat%TYPE := 'FRA';
vVns Joueur.vns%TYPE := 'TOULOUSE';
vPns Joueur.pns%TYPE := 'FRA';
vDns Joueur.dns%TYPE := '20/09/01';
```

```
Genre invalide


Procédure PL/SQL terminée.
```

```
nouveau :DECLARE
vNj Joueur.nj%TYPE := SEQ_JOUEUR.nextval;
vPre Joueur.pre%TYPE := 'AHMED';
vNom Joueur.nom%TYPE := 'AHAROUITE';
vGen Joueur.gen%TYPE := 'H';
vNat Joueur.nat%TYPE := 'M';
vVns Joueur.vns%TYPE := 'TOULOUSE';
vPns Joueur.pns%TYPE := 'M';
vDns Joueur.dns%TYPE := '20/09/01';
```

```
Pays saisi inconnu


Procédure PL/SQL terminée.
```

Ici même, deux cas d'erreurs sont pris en charges :

- Nationalité inconnue
- Pays de naissance inconnu

2.3. Création d'un bloc PL/SQL d'ajout d'un match

(1) Reprendre le bloc PL/SQL permettant d'ajouter un match de simple ou de double. Modifiez le bloc de telle sorte qu'il soit constitué d'exceptions systèmes lorsque cela est possible prenant en charge les erreurs suivantes.

```
ACCEPT pNt PROMPT 'Saisir le numéro du tableau : ';
ACCEPT pTour PROMPT 'Saisir le tour : ';
ACCEPT pNj PROMPT 'Saisir le numéro des joueurs : ';

DECLARE
FK_EXCEPTION EXCEPTION;
PRAGMA EXCEPTION_INIT (FK_EXCEPTION, -2291);
NB_JOUEUR EXCEPTION;
PRAGMA EXCEPTION_INIT (NB_JOUEUR, -2000);
vNm Match.nm%TYPE := SEQ_MATCH.nextval;
vNt Match.nt%TYPE := '&pNt';
vTour Match.tour%TYPE := '&pTour';
vNj1 Match.nj1%TYPE;
vNj2 Match.nj2%TYPE;
vNj1b Match.nj1%TYPE;
vNj2b Match.nj2%TYPE;
i NUMBER := 0;
a NUMBER ;
nb NUMBER := 0;
vNj VARCHAR(30) := '&pNj';

BEGIN
FOR a in (SELECT regexp_substr(vNj, '[^-]+', 1, level) AS colonne FROM DUAL CONNECT BY regexp_substr(vNj, '[^-]+', 1, level) IS NOT NULL) LOOP
    IF i = 0 THEN
        vNj1 := a.colonne;
    ELSIF i = 1 THEN
        vNj2 := a.colonne;
    ELSIF i = 2 THEN
        vNj1b := a.colonne;
    ELSE
        vNj2b := a.colonne;
    END IF;
    i := i+1;
END LOOP;
```

```
-- traitements
IF i = 2 AND vNt IN ('SM','SD')  THEN
    -- Insertion du nouveau match
    INSERT INTO Match (nm, nt, tour, nj1, nj2) VALUES (vNm, vNt, vTour, vNj1, vNj2);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Match '||vNm||' '||vNt||' '||vTour||' '||vNj1||' '||vNj2||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('match enregistré');
ELSIF i = 4 AND vNt IN ('DM','DD','DX') THEN
    -- Insertion du nouveau match
    INSERT INTO Match (nm, nt, tour, nj1, nj2, nj1b, nj2b) VALUES (vNm, vNt, vTour, vNj1, vNj2, vNj1b, vNj2b);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Match '||vNm||' '||vNt||' '||vTour||' '||vNj1||' '||vNj2||' '||vNj1b||' '||vNj2b||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('match enregistré');
ELSIF (i != 2 AND vNt IN ('SM','SD')) OR (i != 4 AND vNt IN ('DM','DD','DX')) THEN
    RAISE NB_JOUEUR;
END IF;
-- Validation
COMMIT;

EXCEPTION
-- traitements des erreurs
WHEN FK_EXCEPTION THEN
    IF vNt NOT IN ('SM','SD','DM','DD','DX') THEN
        DBMS_OUTPUT.PUT_LINE('Numéro de tableau invalide');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Joueur inconnu');
    END IF;
WHEN NB_JOUEUR THEN
      DBMS_OUTPUT.PUT_LINE('Nombre de joueurs invalide');
END;
/
```

(2) Testez votre bloc PL/SQL avec les différents cas possibles prévus.

```
vNt Match.nt%TYPE := 'MM';
vTour Match.tour%TYPE := 'final';

vNj VARCHAR(30) := '1-2';
```

```
Procédure PL/SQL terminée.
```

Cela ne fonctionne pas… Je ne comprends pas.

```
vNt Match.nt%TYPE := 'SD';
vTour Match.tour%TYPE := 'finale';

vNj VARCHAR(30) := '1-2';
```

```
Match 328 SD finale 1 2 ajouté.
match enregistré

Procédure PL/SQL terminée.
```

```
vNt Match.nt%TYPE := 'SD';
vTour Match.tour%TYPE := 'finale';

vNj VARCHAR(30) := '1-2-3-4-5';
```

```
Nombre de joueurs invalide


Procédure PL/SQL terminée.
```

```
vNt Match.nt%TYPE := 'SM';
vTour Match.tour%TYPE := 'finale';

vNj VARCHAR(30) := '1-596';
```

```
Joueur inconnu

Procédure PL/SQL terminée.
```

3. Conclusion

Ce TP10 a pour but de mettre en application ce que nous avons déjà vu au cours des TD et TP précédents, c'est à dire la manipulation de base de données grâce au langage SQL (Structured Query Langage) et le fonctionnement de blocs PL/SQL.

Durant ce TP, on a pu étudier le fonctionnement des exceptions de blocs PL/SQL . Nous avons pu déclarer des variables, utiliser des saisies (valeurs entrées au clavier), créer des exceptions et vérifier les cas d'erreurs.

Ce TP10 nous a permis d'approfondir nos connaissances sur la gestion des bases de donnés en PL-SQL, plus précisément sur l'utilisation d'exceptions.