Aharouite Ahmed

PABD / TP07 Oracle

Table des matières

1. Introduction

Ce TP traite du développement de blocs PL/SQL. On suppose que les valeurs entrées au clavier sont correctes ; il n'est donc pas demandé de vérifier les différents cas d'erreurs.
On s'intéresse ici à la mise en place de traitements transactionnels simples. Vous devrez utiliser les instructions de base du langage PL/SQL

1/ les conditions

```
IF <condition> THEN …
ELSIF <condition> THEN …
ELSIF <condition> THEN …
(…)
ELSE …
END IF;
```

2/ les boucles

```
FOR <variable> IN (<requete>) LOOP
 …
END LOOP;
```

2. Manipulation de la base de données

2.1. Création d'un bloc PL /SQL d'ajout d'un match de simple

(1) Créer le bloc PL/SQL conforme aux spécifications suivantes.

```
SET SERVEROUTPUT ON;

ACCEPT pNt PROMPT 'Saisir le numéro du tableau : ';
ACCEPT pTour PROMPT 'Saisir le tour : ';
ACCEPT pNj1 PROMPT 'Saisir le numéro du joueur 1 : ';
ACCEPT pNj2 PROMPT 'Saisir la numéro du joueur 2 : ';

DECLARE
FK_MATCH_NT EXCEPTION;
PRAGMA EXCEPTION_INIT (FK_MATCH_NT, -2291);
vNm Match.nm%TYPE := SEQ_MATCH.nextval;
vNt Match.nt%TYPE := '&pNt';
vTour Match.tour%TYPE := '&pTour';
vNj1 Match.nj1%TYPE := '&pNj1';
vNj2 Match.nj2%TYPE := '&pNj2';

BEGIN
-- Erreur
IF vnt != 'SM' AND vnt != 'SD' THEN RAISE FK_MATCH_NT;
ELSE
    -- Insertion du nouveau match
    INSERT INTO Match VALUES (vNm, vNt, vTour, vNj1, null, vNj2, null, null, null, null, null, null,
    null, null, null, null, null, null, null, null, null, null, null, null, null, null);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Match '||vNm||' '||vNt||' '||vTour||' '||vNj1||' '||vNj2||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('match enregistré');
    -- Validation
    COMMIT;
END IF;

EXCEPTION
WHEN FK_MATCH_NT THEN
      DBMS_OUTPUT.PUT_LINE('numéro de tableau invalide');
      DBMS_OUTPUT.PUT_LINE ('Le numéro du tableau doit être SM ou SD');
END;
/
```

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

```
vNt Match.nt%TYPE := 'DD';
vTour Match.tour%TYPE := 'finale';
vNj1 Match.nj1%TYPE := '12';
vNj2 Match.nj2%TYPE := '14';
```

```
numéro de tableau invalide
Le numéro du tableau doit être SM ou SD


Procédure PL/SQL terminée.
```

```
vNt Match.nt%TYPE := '';
vTour Match.tour%TYPE := 'finale';
vNj1 Match.nj1%TYPE := '12';
vNj2 Match.nj2%TYPE := '14';
```

```
vNt Match.nt%TYPE := 'SD';
vTour Match.tour%TYPE := 'finale';
vNj1 Match.nj1%TYPE := '12';
vNj2 Match.nj2%TYPE := '14';
```

```
Match 299 SD finale 12 14 ajouté.
match enregistré


Procédure PL/SQL terminée.
```

2.2. Création d'un bloc PL/SQL d'ajout d'un match simple ou double

(1) En vous inspirant du bloc PL/SQL précédent, programmez un nouveau bloc permettant l'ajout d'un nouveau match soit de simple, soit de double.

```
SET SERVEROUTPUT ON;

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

DECLARE
FK_MATCH_NT EXCEPTION;
PRAGMA EXCEPTION_INIT (FK_MATCH_NT, -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;
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;
    ELSIF i = 3 THEN
        vNj2b := a.colonne;
    END IF;
    i := i+1;
END LOOP;
-- Erreur
IF vNt NOT IN ('SM', 'SD', 'DM', 'DD', 'DX') OR i NOT IN (2,4) THEN
    RAISE FK_MATCH_NT;
ELSIF i = 2 AND vNt NOT IN ('SM','SD') THEN
    RAISE FK_MATCH_NT;
ELSIF i = 4 AND vNt NOT IN ('DM','DD','DX') THEN
    RAISE FK_MATCH_NT;
END IF;
```

```
-- Erreur
IF vNt NOT IN ('SM', 'SD', 'DM', 'DD', 'DX') OR i NOT IN (2,4) THEN
    RAISE FK_MATCH_NT;
ELSIF i = 2 AND vNt NOT IN ('SM','SD') THEN
    RAISE FK_MATCH_NT;
ELSIF i = 4 AND vNt NOT IN ('DM','DD','DX') THEN
    RAISE FK_MATCH_NT;
END IF;
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');
END IF;
-- Validation
COMMIT;

EXCEPTION
WHEN FK_MATCH_NT THEN
      DBMS_OUTPUT.PUT_LINE('Numéro de tableau invalide');
END;
/
```

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

```
Match 301 SM finale 1 2 ajouté.
match enregistré
```

```
Procédure PL/SQL terminée.
```

```
Match 302 DD finale 1 2 3 4 ajouté
match enregistré
```

```
Procédure PL/SQL terminée.
```

```
vNt Match.nt%TYPE := '';
vTour Match.tour%TYPE := '';
vNj1 Match.nj1%TYPE;
vNj2 Match.nj2%TYPE;
vNj1b Match.nj1%TYPE;
vNj2b Match.nj2%TYPE;
i NUMBER := 0;
a NUMBER;
vNj VARCHAR(30) := '';
```

```
Numéro de tableau invalide
```

```
Procédure PL/SQL terminée.
```

```
vNt Match.nt%TYPE := 'DX';
vTour Match.tour%TYPE := 'finale';
vNj1 Match.nj1%TYPE;
vNj2 Match.nj2%TYPE;
vNj1b Match.nj1%TYPE;
vNj2b Match.nj2%TYPE;
i NUMBER := 0;
a NUMBER;
vNj VARCHAR(30) := '';
```

```
Numéro de tableau invalide
```

```
Procédure PL/SQL terminée.
```

(3) Vérifiez le contenu de la table Match.

```
SELECT * FROM Match WHERE nm IN (301, 302);
```

	NM	NT	TOUR	NJ1	NJ1B	NJ2	NJ2B	J1SSET1	J1STIE1	J1SSET2	J1STIE
1	301	SM	finale	1	(null)	2	(null)	(null)	(null)	(null)	(nul
2	302	DD	finale	1	3	2	4	(null)	(null)	(null)	(nul

Il y a bien le contenu.

3. Conclusion

Ce TP07 nous a permis d'approfondir nos connaissances sur la gestion des bases de donnés en PL-SQL, plus précisément sur l'utilisation des boucles « FOR » et les conditions « IF ».