Aharouite Ahmed

PABD / TP12 Oracle

Table des matières

1. Introduction

Dans ce TP nous utilisons les mécanismes des procédures stockées.

2. Manipulation de la base de données

2.1. Reprise des blocs PL/SQL en procédures stockées

a. ajouter un pays

(1) Reprendre les blocs PL/SQL développés dans les TP précédents et les transformer en procédures stockées avec paramètres en entrée.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE ajoueterPays (pCio Pays.cio%TYPE, pNom Pays.nom%TYPE) AS

PAYS_EXISTANT EXCEPTION;
PRAGMA EXCEPTION_INIT(PAYS_EXISTANT,-20010);
n NUMBER;

BEGIN
SELECT COUNT(*) INTO n FROM Pays WHERE pCio = cio;
IF (n=1) THEN
    RAISE PAYS_EXISTANT;
ELSE
    INSERT INTO Pays (cio, nom) VALUES (pCio, pNom);
    DBMS_OUTPUT.PUT_LINE('Pays '||pCio||' '||pNom||' ajouté.');
END IF;
COMMIT;
EXCEPTION
    WHEN PAYS_EXISTANT THEN
        DBMS_OUTPUT.PUT_LINE('Le pays existe déja');
END;
/

ACCEPT vCio PROMPT 'Saisir le cio du pays : ';
ACCEPT vNom PROMPT 'Saisir le nom du pays : ';
EXECUTE ajoueterPays('&vCio', '&vNom');
```

(2) Testez vos procédures stockées avec tous les cas (erreur et sans erreur)

```
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.
```

b. ajouter un joueur

(1)

```
CREATE OR REPLACE PROCEDURE ajoueterJoueur (pPre Joueur.pre%TYPE, pNom Joueur.nom%TYPE, pGen Joueur.gen%TYPE,
pNat Joueur.nat%TYPE, pVns Joueur.vns%TYPE, pPns Joueur.pns%TYPE, pDns Joueur.dns%TYPE) AS

vNj Joueur.nj%TYPE := SEQ_JOUEUR.nextval;
p NUMBER;
GENRE_INVALIDE EXCEPTION;
PRAGMA EXCEPTION_INIT(GENRE_INVALIDE,-2290);
PAYS_INCONNU EXCEPTION;
PRAGMA EXCEPTION_INIT(PAYS_INCONNU,-2291);

BEGIN
SELECT COUNT(*) INTO p FROM Pays WHERE cio IN (pNat,pPns);
IF (pGen IN ('H','F') AND p=1) THEN
    --Insertion du joueur
    INSERT INTO Joueur (nj, pre, nom, gen, nat, vns, pns, dns) VALUES (vNj, pPre, pNom, pGen, pNat, pVns, pPns, pDns);
    DBMS_OUTPUT.PUT_LINE('Joueur '||vNj||' '||pPre||' '||pNom||' '||pGen||' '||pNat||' '||pVns||' '||pPns||' '||pDns||' ajouté.');
ELSE
    IF pGen NOT IN ('H','F') THEN
        RAISE GENRE_INVALIDE;
    END IF;
    IF (p = 0) THEN
        RAISE PAYS_INCONNU;
    END IF;
END IF;
-- Validation
COMMIT;
EXCEPTION
    WHEN PAYS_INCONNU THEN
        DBMS_OUTPUT.PUT_LINE('Joueur '||vNj||' '||pPre||' '||pNom||' '||pGen||' '||pNat||' '||pVns||' '||pPns||' '||pDns||' non ajouté.');
        DBMS_OUTPUT.PUT_LINE('CAUSE : Pays saisi inconnu');
    WHEN GENRE_INVALIDE THEN
        DBMS_OUTPUT.PUT_LINE('Joueur '||vNj||' '||pPre||' '||pNom||' '||pGen||' '||pNat||' '||pVns||' '||pPns||' '||pDns||' non ajouté.');
        DBMS_OUTPUT.PUT_LINE('CAUSE : Genre invalide');
END;
/
```

(2)

```
SET SERVEROUTPUT ON;
ACCEPT vPre PROMPT 'Saisir le prénom du joueur : ';
ACCEPT vNom PROMPT 'Saisir le nom du joueur : ';
ACCEPT vGen PROMPT 'Saisir le genre du joueur : ';
ACCEPT vNat PROMPT 'Saisir la nationalité du joueur : ';
ACCEPT vVns PROMPT 'Saisir la ville de naissance du joueur : ';
ACCEPT vPns PROMPT 'Saisir le pays de naissance joueur : ';
ACCEPT vDns PROMPT 'Saisir le date de naissance joueur : ';

EXECUTE ajoueterJoueur('&vPre', '&vNom', '&vGen', '&vNat', '&vVns', '&vPns', '&vDns');
```

```
Elément Procedure AJOUETERJOUEUR compilé

Joueur 424 Ahmed Aharouite H FRA Toulouse FRA 20/09/01 ajouté.
```

```
Joueur 426 Ahmed Aharouite M FRA Toulouse FRA 20/09/01 non ajouté.
CAUSE : Genre invalide
```

```
Joueur 427 Ahmed Aharouite H SQL Toulouse SQL 20/09/01 non ajouté.
CAUSE : Pays saisi inconnu


Procédure PL/SQL terminée.
```

c. ajouter un match

(1)

```
CREATE OR REPLACE PROCEDURE ajoueterMatch (pNt Match.nt%TYPE, pTour Match.tour%TYPE, pNj VARCHAR) AS

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;
vNj1 Match.nj1%TYPE;
vNj2 Match.nj2%TYPE;
vNj1b Match.nj1%TYPE;
vNj2b Match.nj2%TYPE;
i NUMBER := 0;
a NUMBER ;
nb NUMBER := 0;

BEGIN
FOR a in (SELECT regexp_substr(pNj, '[^-]+', 1, level) AS colonne FROM DUAL CONNECT BY regexp_substr(pNj, '[^-]+', 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 pNt IN ('SM','SD')  THEN
    -- Insertion du nouveau match
    INSERT INTO Match (nm, nt, tour, nj1, nj2) VALUES (vNm, pNt, pTour, vNj1, vNj2);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Match '||vNm||' '||pNt||' '||pTour||' '||vNj1||' '||vNj2||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('match enregistré');
ELSIF i = 4 AND pNt IN ('DM','DD','DX') THEN
    -- Insertion du nouveau match
    INSERT INTO Match (nm, nt, tour, nj1, nj2, nj1b, nj2b) VALUES (vNm, pNt, pTour, vNj1, vNj2, vNj1b, vNj2b);
    -- Affichage d'un message
    DBMS_OUTPUT.PUT_LINE('Match '||vNm||' '||pNt||' '||pTour||' '||vNj1||' '||vNj2||' '||vNj1b||' '||vNj2b||' ajouté.');
    DBMS_OUTPUT.PUT_LINE('match enregistré');
ELSIF (i != 2 AND pNt IN ('SM','SD')) OR (i != 4 AND pNt IN ('DM','DD','DX')) THEN
    RAISE NB_JOUEUR;
END IF;
-- Validation
COMMIT;

EXCEPTION
-- traitements des erreurs
    WHEN FK_EXCEPTION THEN
        DBMS_OUTPUT.PUT_LINE('Numéro de tableau invalide');
    WHEN NB_JOUEUR THEN
        DBMS_OUTPUT.PUT_LINE('Nombre de joueurs invalide');
END;
/
```

(2)

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

EXECUTE ajoueterMatch ('&pNt' , '&pTour' , '&pNj');
```

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

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

```
Joueur inconnu


Procédure PL/SQL terminée.
```

```
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.
```

d. afficher matchs

(1)

```
CREATE OR REPLACE PROCEDURE afficherMatchs (pPre Joueur.pre%TYPE, pNom Joueur.nom%TYPE) AS

CURSOR cl IS SELECT nt, tour, nj1, nj2, ('('||J1SSET1||' - '||J2SSET1||')') as SET1,
('('||J1SSET2||' - '||J2SSET2||')') as SET2, ('('||J1SSET3||' - '||J2SSET3||')') as SET3,
('('||J1SSET4||' - '||J2SSET4||')') as SET4, ('('||J1SSET5||' - '||J2SSET5||')') as SET5 FROM Match;
vNj Joueur.nj%TYPE;
vPre2 Joueur.pre%TYPE;
vNom2 Joueur.nom%TYPE;

BEGIN
SELECT nj INTO vNj FROM Joueur WHERE pPre = pre AND pNom = nom;
DBMS_OUTPUT.PUT_LINE('Match de : ' || pPre || ' ' || pNom );
FOR clLigne IN cl LOOP
    IF clLigne.nt IN ('SM', 'SD') THEN
        IF clLigne.nj1 = vNj THEN
            SELECT pre, nom INTO vPre2, vNom2 FROM Joueur WHERE clLigne.nj2 = nj;
            DBMS_OUTPUT.PUT_LINE(clLigne.tour||' : '||vPre2||' '||vNom2||' '||clLigne.SET1||' '||
            clLigne.SET2||' '||clLigne.SET3||' '||clLigne.SET4||' '||clLigne.SET5);
        END IF;
        IF clLigne.nj2 = vNj THEN
            SELECT pre, nom INTO vPre2, vNom2 FROM Joueur WHERE clLigne.nj1 = nj;
            DBMS_OUTPUT.PUT_LINE(clLigne.tour||' : '||vPre2||' '||vNom2||' '||clLigne.SET1||' '||
            clLigne.SET2||' '||clLigne.SET3||' '||clLigne.SET4||' '||clLigne.SET5);
         END IF;
    END IF;
END LOOP;
COMMIT;
END;
/
```

(2)

```
Match de : JO-WILFRIED TSONGA
1er tour : PETER GOJOWCZYK (7 - 6) (6 - 1) (4 - 6) (6 - 3) ( - )
2ème tour : KEI NISHIKORI (4 - 6) (6 - 4) (6 - 4) (6 - 4) ( - )


Match de : RAFAEL NADAL
1er tour : YANNICK HANFMANN (2 - 6) (1 - 6) (3 - 6) ( - ) ( - )
2ème tour : YANNICK MADEN (1 - 6) (2 - 6) (4 - 6) ( - ) ( - )
3ème tour : DAVID GOFFIN (1 - 6) (3 - 6) (6 - 4) (3 - 6) ( - )
1/8 de finale : JUAN IGNACIO LONDERO (2 - 6) (3 - 6) (3 - 6) ( - ) ( - )
1/4 de finale : KEI NISHIKORI (1 - 6) (1 - 6) (3 - 6) ( - ) ( - )
1/2 finale : ROGER FEDERER (3 - 6) (4 - 6) (2 - 6) ( - ) ( - )
finale : DOMINIC THIEM (3 - 6) (7 - 5) (1 - 6) (1 - 6) ( - )
```

2.2. Nouvelle procédure stockée

(1) Créer une nouvelle procédure stockée SupprimerJoueur conforme aux spécifications suivantes.

```
CREATE OR REPLACE PROCEDURE SupprimerJoueur (pNj Joueur.nj%TYPE) AS

j NUMBER;
JOUEUR_INCONNU EXCEPTION;
PRAGMA EXCEPTION_INIT(JOUEUR_INCONNU,-2291);

BEGIN
SELECT COUNT(*) INTO j FROM Joueur WHERE nj = pNj;
IF (j = 1) THEN
    --Supprésion du joueur
    DELETE FROM Match WHERE pNj IN (nj1, nj2, nj1b, nj2b);
    DELETE FROM Joueur WHERE nj = pNj;
    DBMS_OUTPUT.PUT_LINE('Joueur '||pNj||' supprimé.');
ELSE
    RAISE JOUEUR_INCONNU;
END IF;
-- Validation
COMMIT;
EXCEPTION
    WHEN JOUEUR_INCONNU THEN
        DBMS_OUTPUT.PUT_LINE('CAUSE : Numéro du joueur saisi inconnu');
END;
/
```

(2) Testez votre nouvelle procédure

```
SELECT * FROM Joueur WHERE nj=424;
```

	NJ	PRE	NOM	GEN	CLA	NAT	VNS	PNS	DNS
1	424	Ahmed	Aharouite	H	(null)	FRA	Toulouse	FRA	20/09/01

```
EXECUTE afficherMatchs('Ahmed', 'Aharouite');
```

```
Match de : Ahmed Aharouite
finale : NOVAK DJOKOVIC ( - ) ( - ) ( - ) ( - ) ( - )
finale : HENRI LAAKSONEN ( - ) ( - ) ( - ) ( - ) ( - )
```

```
SET SERVEROUTPUT ON;
ACCEPT vNj PROMPT 'Saisir le numéro du joueur : ';

EXECUTE SupprimerJoueur('&vNj');
```

```
Joueur 424 supprimé.


Procédure PL/SQL terminée.
```

Le joueur n'existe plus dans les tables Match et Joueur….

3. Conclusion

Ce TP12 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 procédures stockées . Nous avons pu reprendre les blocs PL/SQL développés dans les TP précédents et les transformer en procédures stockées avec paramètres en entrée et créer une nouvelle procédure stockée SupprimerJoueur conforme aux spécifications suivantes.

Une procédure stockée est une collection précompilée d'instructions (Transact-SQL) stockée sous un nom et traitée comme une unité. Les procédures stockées de SQL Server permettent de gérer celui-ci et d'afficher les informations sur les bases de données et les utilisateurs. Une procédure est également un objet de base de données, regroupant un ensemble de commandes qui effectuent une tâche particulière le plus souvent, récurrente. A la différence d'une fonction, une procédure stockée ne renvoie pas de valeur.

Ce TP12 nous a permis d'approfondir nos connaissances sur la gestion des bases de donnés en PL-SQL, plus précisément sur l'utilisation de procédures stockées.