

30/11/2024

# TP4 Base de données Réparties

**Création des Traitements Reparties**

Mamadou Baïlo BARRY & Eylül ALPTEKIN

M1 MIAGE

UNIVERSITE TOULOUSE 3 – PAUL SABATIER

# Sommaire

1. Introduction .....	3
2. Programmation des traitements sur fragments.....	3
➤ Test de la procédure <i>AfficherMedaille</i> :.....	3
➤ Test de la procédure <i>AjouterDiscipline</i> : .....	3
➤ Reprogrammation de la procédure <i>AjouterDiscipline</i> : .....	3
➤ Test de la procédure modifiée à partir des trois sites : .....	4
3. Vue globale active par déclencheurs « INSTEAD OF » .....	5
➤ Création des déclencheurs « INSTEAD OF » :.....	5
• Pour la vue <i>Discipline</i> .....	5
• Pour la vue <i>Athlete</i> .....	6
• Pour la vue <i>Pratiquer</i> .....	8
➤ Test de la procédure <i>AjouterAthlete</i> (du paquetage <i>GEST_JO</i> ) :.....	9
4. Conclusion.....	9

## 1. Introduction

Après la répartition de la base de données centralisée sur les trois bases (etupre, etusec et etuter), un nouveau problème se pose avec les traitements centralisés initiaux opérationnels sur le système repart. Ainsi l'objectif de ce TP nous incite à faire des manipulations pour rendre ces traitements centralisés initiaux opérationnels en utilisant deux stratégies :

- Soit reprogrammer directement les procédures stockées sur fragments,
- Soit rendre les vues globales actives en fonction des ordres LMD nécessaires au bon fonctionnement des procédures stockées transactionnelles par la mise en place de « triggers instead of ».

## 2. Programmation des traitements sur fragments

### ➤ Test de la procédure *AfficherMedaille* :

La procédure *AfficherMedaille* fonctionne correctement sans modification. Elle fonctionne parce qu'elle n'utilise que des SELECT d'interrogation sur les vues et non des INSERT, UPDATE ou DELETE.

### ➤ Test de la procédure *AjouterDiscipline* :

La procédure *AjouterDiscipline* ne fonctionne pas parce qu'elle n'utilise pas que des SELECT mais aussi des INSERT pour insérer les données dans la base.

### ➤ Reprogrammation de la procédure *AjouterDiscipline* :

```
CREATE OR REPLACE PACKAGE Gestion_JO_Rep IS
    PROCEDURE AjouterDiscipline(
        p_nomd Discipline.nomd%TYPE,
        p_typed Discipline.typed%TYPE,
        p_genred Discipline.genred%TYPE,
        p_cds Sport.cds%TYPE
    );
END;
/

CREATE OR REPLACE PACKAGE BODY Gestion_JO_Rep IS
    PROCEDURE AjouterDiscipline(
        p_nomd Discipline.nomd%TYPE,
        p_typed Discipline.typed%TYPE,
        p_genred Discipline.genred%TYPE,
        p_cds Sport.cds%TYPE
    ) IS
        -- Déclaration des variables
        nbb NUMBER;
        erreur_fk EXCEPTION;
        PRAGMA EXCEPTION_INIT(erreur_fk, -2221); -- Code d'erreur sur les clés
étrangères
        erreur_ck EXCEPTION;
        PRAGMA EXCEPTION_INIT(erreur_ck, -2290); -- Code d'erreur sur les contraintes
        n NUMBER;
```

```

BEGIN
    -- Ajouter une nouvelle discipline
    SELECT SEQ_DISCIPLINE.NEXTVAL INTO n FROM DUAL;
    IF (p_genred = 'M') THEN
        INSERT INTO Discipline_1@db1_etupre
        VALUES(n, p_nomd, p_typed, p_cds);
    ELSIF(p_genred='W') THEN
        INSERT INTO Discipline_2@db1_etusec
        VALUES(n, p_nomd, p_typed, p_cds);
    ELSIF(p_genred IN ('X', 'O')) THEN
        INSERT INTO Discipline_3@db1_etuter
        VALUES(n, p_nomd, p_typed, p_genred, p_cds);
    END IF;

    -- Affichage du message d'ajout de la Discipline
    DBMS_OUTPUT.PUT_LINE('Discipline <' || p_nomd || '> ajoutée avec succès');

    -- Validation de l'ajout
    COMMIT;

EXCEPTION
    -- Gestion des erreurs
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Discipline ' || p_nomd || 'existante !');
    WHEN erreur_fk THEN
        DBMS_OUTPUT.PUT_LINE('Code sport inconnu !'); -- Erreur sur le code du
Sport
    WHEN erreur_ck THEN
        IF (SQLERRM LIKE '%CK_DISCIPLINE_TYPED%') THEN
            DBMS_OUTPUT.PUT_LINE('Type de Discipline inconnu !'); --Erreur sur
Le type de la Discipline
        ELSIF (SQLERRM LIKE '%CK_DISCIPLINE_GENRED%') THEN
            DBMS_OUTPUT.PUT_LINE('Genre de Discipline inconnu !'); --Erreur sur
Le genre de la Discipline
        END IF;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM); -- Autres erreurs
    END AjouterDiscipline;
END Gestion_JO_Rep;
/

```

➤ Test de la procédure modifiée à partir des trois sites :

```

-- Sur la site ETUPRE
SET SERVEROUTPUT ON;
EXEC Gestion_JO_Rep.AjouterDiscipline('Sauter très haut', 'P', 'M', 'ATH');

-- Sur la site ETUSEC
SET SERVEROUTPUT ON;

```

```
EXEC Gestion_JO_Rep.AjouterDiscipline('Courir vite', 'P', 'W', 'ARC');

-- Sur la site ETUTER
SET SERVEROUTPUT ON;
EXEC Gestion_JO_Rep.AjouterDiscipline('Cogner tres fort', 'P', 'X', 'BOX');
```

```
Discipline <Sauter très haut> ajoutée avec succès
```

```
Procédure PL/SQL terminée.
```

```
Discipline <Courir vite> ajoutée avec succès
```

```
Procédure PL/SQL terminée.
```

```
Discipline <Cogner tres fort> ajoutée avec succès
```

```
Procédure PL/SQL terminée.
```

### 3. Vue globale active par déclencheurs « INSTEAD OF »

#### ➤ Création des déclencheurs « INSTEAD OF » :

##### • Pour la vue *Discipline*

```
CREATE OR REPLACE TRIGGER t_iud_Discipline
INSTEAD OF INSERT OR DELETE OR UPDATE
ON Discipline
FOR EACH ROW
BEGIN
    -- Insertion des données dans la vue
    IF (INSERTING) THEN
        IF (:NEW.genred='M') THEN
            INSERT INTO Discipline_1@dbl_etupre
            VALUES (:NEW.ndd, :NEW.nomd, :NEW.typed, :NEW.cds);
        ELSIF (:NEW.genred='W') THEN
            INSERT INTO Discipline_2@dbl_etusec
            VALUES (:NEW.ndd, :NEW.nomd, :NEW.typed, :NEW.cds);
        ELSIF (:NEW.genred IN ('X', 'O')) THEN
            INSERT INTO Discipline_3@dbl_etuter
            VALUES (:NEW.ndd, :NEW.nomd, :NEW.typed, :NEW.genred, :NEW.cds);
        END IF;
    END IF;

    -- Suppression des données de la vue
    IF (DELETING) THEN
        IF (:NEW.genred='M') THEN
```

```

        DELETE FROM Discipline_1@dbl_etupre
        WHERE ndd = :OLD.ndd;
    ELSIF(:NEW.genred='W') THEN
        DELETE FROM Discipline_2@dbl_etusec
        WHERE ndd = :OLD.ndd;
    ELSIF(:NEW.genred IN ('X', 'O')) THEN
        DELETE FROM Discipline_3@dbl_etuter
        WHERE ndd = :OLD.ndd;
    END IF;
END IF;

-- Mise à jour des données dans la vue
IF (UPDATING) THEN
    IF(:NEW.genred='M') THEN
        UPDATE Discipline_1@dbl_etupre
        SET ndd = :NEW.ndd,
            nomd = :NEW.nomd,
            typed = :NEW.typed,
            cds = :NEW.cds
        WHERE ndd = :OLD.ndd;
    ELSIF(:NEW.genred='W') THEN
        UPDATE Discipline_2@dbl_etusec
        SET ndd = :NEW.ndd,
            nomd = :NEW.nomd,
            typed = :NEW.typed,
            cds = :NEW.cds
        WHERE ndd = :OLD.ndd;
    ELSIF(:NEW.genred IN ('X', 'O')) THEN
        UPDATE Discipline_3@dbl_etuter
        SET ndd = :NEW.ndd,
            nomd = :NEW.nomd,
            typed = :NEW.typed,
            cds = :NEW.cds
        WHERE ndd = :OLD.ndd;
    END IF;
END IF;
END;
/

```

- Pour la vue *Athlete*

```

CREATE OR REPLACE TRIGGER t_iud_Athlete
INSTEAD OF INSERT OR DELETE OR UPDATE
ON Athlete
FOR EACH ROW
BEGIN
    -- Gestion des INSERTIONS
    IF (INSERTING) THEN
        -- Redirection vers le fragment correspondant
    
```

```
        IF (:NEW.genre = 'M') THEN
            INSERT INTO Athlete_1@db1_etupre
            VALUES (:NEW.nda, :NEW.ncomplet, :NEW.taille, :NEW.poids, :NEW.daten,
:NEW.villen, :NEW.paysn, :NEW.cio);
        ELSIF (:NEW.genre = 'F') THEN
            INSERT INTO Athlete_2@db1_etusec
            VALUES (:NEW.nda, :NEW.ncomplet, :NEW.taille, :NEW.poids, :NEW.daten,
:NEW.villen, :NEW.paysn, :NEW.cio);
        END IF;
    END IF;

-- Gestion des SUPPRESSIONS
IF (DELETING) THEN
    -- Suppression dans le fragment correspondant
    IF (:OLD.genre = 'M') THEN
        DELETE FROM Athlete_1@db1_etupre
        WHERE nda = :OLD.nda;
    ELSIF (:OLD.genre = 'F') THEN
        DELETE FROM Athlete_2@db1_etusec
        WHERE nda = :OLD.nda;
    END IF;
END IF;

-- Gestion des MISES À JOUR
IF (UPDATING) THEN
    IF (:NEW.genre = 'M') THEN
        UPDATE Athlete_1@db1_etupre
        SET nda = :NEW.nda,
            ncomplet = :NEW.ncomplet,
            taille = :NEW.taille,
            poids = :NEW.poids,
            daten = :NEW.daten,
            villen = :NEW.villen,
            paysn = :NEW.paysn,
            cio = :NEW.cio
        WHERE nda = :OLD.nda;
    ELSIF (:NEW.genre = 'F') THEN
        UPDATE Athlete_2@db1_etusec
        SET nda = :NEW.nda,
            ncomplet = :NEW.ncomplet,
            taille = :NEW.taille,
            poids = :NEW.poids,
            daten = :NEW.daten,
            villen = :NEW.villen,
            paysn = :NEW.paysn,
            cio = :NEW.cio
        WHERE nda = :OLD.nda;
    END IF;
```

```

    END IF;
END;
/

```

- Pour la vue *Pratiquer*

```

CREATE OR REPLACE TRIGGER t_iud_Pratiquer
INSTEAD OF INSERT OR DELETE OR UPDATE
ON Pratiquer
FOR EACH ROW
DECLARE
    v_genred CHAR(1);
BEGIN
    SELECT genred INTO v_genred
    FROM Discipline
    WHERE ndd = :NEW.ndd;

    -- Gestion des INSERTIONS
    IF (INSERTING) THEN
        IF (v_genred = 'M') THEN
            INSERT INTO Pratiquer_1@dbl_etupre (nda, ndd)
            VALUES (:NEW.nda, :NEW.ndd);
        ELSIF (v_genred = 'W') THEN
            INSERT INTO Pratiquer_2@dbl_etusec (nda, ndd)
            VALUES (:NEW.nda, :NEW.ndd);
        ELSIF (v_genred IN ('X', 'O')) THEN
            INSERT INTO Pratiquer_3@dbl_etuter (nda, ndd)
            VALUES (:NEW.nda, :NEW.ndd);
        END IF;
    END IF;

    -- Gestion des SUPPRESSIONS
    IF (DELETING) THEN
        IF (v_genred = 'M') THEN
            DELETE FROM Pratiquer_1@dbl_etupre
            WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
        ELSIF (v_genred = 'W') THEN
            DELETE FROM Pratiquer_2@dbl_etusec
            WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
        ELSIF (v_genred IN ('X', 'O')) THEN
            DELETE FROM Pratiquer_3@dbl_etuter
            WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
        END IF;
    END IF;

    -- Gestion des MISE A JOUR
    IF (UPDATING) THEN
        IF (v_genred = 'M') THEN

```



```

        UPDATE Pratiquer_1@db1_etupre
        SET nda = :NEW.nda,
            ndd = :NEW.ndd
        WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
    ELSIF (v_genred = 'W') THEN
        UPDATE Pratiquer_2@db1_etusec
        SET nda = :NEW.nda,
            ndd = :NEW.ndd
        WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
    ELSIF (v_genred IN ('X', 'O')) THEN
        UPDATE Pratiquer_3@db1_etuter
        SET nda = :NEW.nda,
            ndd = :NEW.ndd
        WHERE nda = :OLD.nda AND ndd = :OLD.ndd;
    END IF;
END IF;
END;
/

```

➤ **Test de la procédure AjouterAthlete (du paquetage GEST\_JO) :**

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.AjouterAthlete(6581672, 'BARRY', 'Mamadou bailo', 'M', 172, 64.5,
TO_DATE('1999-11-09', 'YYYY-MM-DD'), 'Mamou', 'Guinea', 'GUI');

```

Athlète BARRY Mamadou bailo ajouté avec succès

Procédure PL/SQL terminée.

	NDA	NCOMPLET	GENRE	TAILLE	POIDS	DATEN	VILLEN	PAYSN	CIO
1	6581672	BARRY Mamadou bailo	M	172	64.5	09/11/99	Mamou	Guinea	GUI

## 4. Conclusion

En conclusion, ce TP nous a permis de savoir comment résoudre le problème des traitements centralisés initiaux (INSERT, DELETE et UPDATE) en utilisant soit reprogrammation des procédures (qui marche bien mais n'est pas une bonne pratique) ou les triggers INSTEAD OF.