

09/10/2024

TP2 Base de données Réparties

Création des Traitements Centralisés

Mamadou Baïlo BARRY

M1 MIAGE
UNIVERSITE TOULOUSE 3 – PAUL SABATIER

Sommaire

1. Introduction	3
2. Création du packaging (Scripts et résultats).....	3
➤ Procédure <i>AjouterDiscipline</i>:	3
➤ Procédure <i>AjouterAthlete</i> :	5
➤ Procédure <i>PratiquerUneDiscipline</i> :	8
➤ Procédure <i>PratiquerDesDisciplines</i> :	11
➤ Procédure <i>AfficheMedaille</i> :	14
3. Conclusion	19

1. Introduction

L'objectif de ce TP est de nous faire une révision sur le langage PL/SQL d'Oracle afin de mettre en place un paquetage et automatiser les traitements avec des procédures.

2. Création du paquetage (Scripts et résultats)

➤ Procédure *AjouterDiscipline*:

```
CREATE OR REPLACE PACKAGE Gest_JO 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 Gest_JO 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

    BEGIN
        -- Ajouter une nouvelle discipline
        INSERT INTO Discipline(ndd, nomd, typed, genred, cds)
        VALUES(SEQ_DISCIPLINE.NEXTVAL, p_nomd, p_typed, p_genred, p_cds);

        -- 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 '%CHK_DISCIPLINE_TYPED%') THEN
            DBMS_OUTPUT.PUT_LINE('Type de Discipline inconnu !'); --Erreur sur
Le type de la Discipline
        ELSIF (SQLERRM LIKE '%CHK_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 Gest_JO;
/

SET SERVEROUTPUT ON;
EXEC Gest_JO.AjouterDiscipline('Sauter très haut', 'P', 'M', 'ATH');

```

Résultats de l'exécution de la requête :

```

Elément Package GEST_JO compilé

Elément Package Body GEST_JO compilé

Discipline <Sauter très haut> ajoutée avec succès

Procédure PL/SQL terminée.

```

Vérification de la gestion des erreurs :

- Pour le code du sport

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.AjouterDiscipline('Sauter très haut', 'P', 'M', 'FRA');

```

- Résultat

```

ORA-02291: integrity constraint (BRM2996A.FK_DISCIPLINE_CDS) violated - parent key not found

Procédure PL/SQL terminée.

```

- Pour le type de la Discipline

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.AjouterDiscipline('Sauter très haut', 'B', 'M', 'ATH');

```

- Résultat

```
Type de Discipline inconnu !
```

```
Procédure PL/SQL terminée.
```

- Pour le genre de la Discipline

```
SET SERVEROUTPUT ON;
```

```
EXEC Gest_J0.AjouterDiscipline('Sauter très haut', 'T', 'Z', 'ATH');
```

- Résultat

```
Genre de Discipline inconnu !
```

```
Procédure PL/SQL terminée.
```

➤ Procédure *AjouterAthlete* :

```
CREATE OR REPLACE PACKAGE Gest_J0 IS
  PROCEDURE AjouterAthlete(
    p_nda Athlete.nda%TYPE,
    p_nom VARCHAR2,
    p_prenom VARCHAR2,
    p_genre Athlete.genre%TYPE,
    p_taille Athlete.taille%TYPE,
    p_poids Athlete.poids%TYPE,
    p_daten Athlete.daten%TYPE,
    p_villen Athlete.villen%TYPE,
    p_paysn Athlete.paysn%TYPE,
    p_nationalite Pays.cio%TYPE
  );
END Gest_J0;
/

CREATE OR REPLACE PACKAGE BODY Gest_J0 IS
  PROCEDURE AjouterAthlete(
    p_nda Athlete.nda%TYPE,
    p_nom VARCHAR2,
    p_prenom VARCHAR2,
    p_genre Athlete.genre%TYPE,
    p_taille Athlete.taille%TYPE,
    p_poids Athlete.poids%TYPE,
    p_daten Athlete.daten%TYPE,
    p_villen Athlete.villen%TYPE,
    p_paysn Athlete.paysn%TYPE,
    p_nationalite Pays.cio%TYPE
  ) IS
    -- Déclaration des variables
    v_ncomplet Athlete.ncomplet%TYPE;
    erreur_fk EXCEPTION;
```

```

PRAGMA EXCEPTION_INIT(erreur_fk, -2291); -- Code d'erreur sur les clés
étrangères
erreur_ck EXCEPTION;
PRAGMA EXCEPTION_INIT(erreur_ck, -2290); -- Code d'erreur sur les contraintes

BEGIN
    -- Concaténation du nom et du prénom de l'Athlète avec le nom et la première
    lettre du prenom en majuscule
    -- puis le reste en du prenom en minuscule
    v_ncomplet := UPPER(p_nom) || ' ' || UPPER(SUBSTR(p_prenom, 1, 1)) ||
LOWER(SUBSTR(p_prenom, 2));

    -- Ajout d'un nouveau Athlète
    INSERT INTO Athlete(nda, ncomplet, genre, taille, poids, daten, villen,
paysn, cio)
    VALUES(p_nda, v_ncomplet, p_genre, p_taille, p_poids, p_daten, p_villen,
p_paysn, p_nationalite);

    -- Affichage du message d'ajout
    DBMS_OUTPUT.PUT_LINE('Athlète ' || v_ncomplet || ' ajouté avec succès');

    -- Validation de l'ajout
    COMMIT;

EXCEPTION
    -- Gestion des erreurs
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Ce numéro dAthlète existe déjà'); -- Erreur sur le
numéro de l'Athlète
    WHEN erreur_fk THEN
        DBMS_OUTPUT.PUT_LINE('Nationalité de lAthlète inconnu. '); -- Erreur sur
la nationalité de l'Athlète
    WHEN erreur_ck THEN
        IF (SQLERRM LIKE '%CHK_ATHLETE_GENRE%') THEN
            DBMS_OUTPUT.PUT_LINE('Genre de lAthlète invalide. '); -- Erreur sur
le genre de l'Athlète
        ELSEIF (SQLERRM LIKE '%CHK_ATHLETE_TAILLE%') THEN
            DBMS_OUTPUT.PUT_LINE('Taille de lAthlète invalide. '); -- Erreur sur
la taille de l'Athlète
        ELSEIF (SQLERRM LIKE '%CHK_ATHLETE_POIDS%') THEN
            DBMS_OUTPUT.PUT_LINE('Poids de lAthlète invalide. '); -- Erreur sur
le poids de l'Athlète
        END IF;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM); -- Autres erreurs
    END AjouterAthlete;
END Gest_J0;
/

```

```
SET SERVEROUTPUT ON;  
EXEC Gest_JO.AjouterAthlete(6581675, 'BARRY', 'Mamadou Bailo', 'M', 172, 64.5,  
TO_DATE('1999-11-09', 'YYYY-MM-DD'), 'Mamou', 'Guinea', 'GUI');
```

Résultats de l'exécution de la requête :

```
Elément Package GEST_JO compilé  
  
Elément Package Body GEST_JO compilé  
  
Athlète BARRY Mamadou bailo ajouté avec succès  
  
Procédure PL/SQL terminée.
```

NDA	NCOMPLET	GENRE	TAILLE	POIDS	DATEN	VILLEN	PAYSN	CIO
6581675	BARRY Mamadou bailo	M	172	64.5	09/11/99	Mamou	Guinea	GUI

Vérification de la gestion des erreurs :

- Pour le numéro de l'Athlète

```
SET SERVEROUTPUT ON;  
EXEC Gest_JO.AjouterAthlete(6581675, 'BARRY', 'Mamadou Bailo', 'M', 172, 64.5,  
TO_DATE('1999-11-09', 'YYYY-MM-DD'), 'Mamou', 'Guinea', 'GUI');
```

- Résultat

```
Ce numéro d'Athlète existe déjà  
  
Procédure PL/SQL terminée.
```

- Pour la nationalité de l'Athlète

```
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', 'GIU');
```

- Résultat

```
Nationalité de l'Athlète inconnu.  
  
Procédure PL/SQL terminée.
```

- Pour le genre de l'Athlète

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

- Résultat

```
Genre de l'Athlète invalide.
```

```
Procédure PL/SQL terminée.
```

- Pour la taille de l'Athlète

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

- Résultat

```
Taille de l'Athlète invalide.
```

```
Procédure PL/SQL terminée.
```

- Pour le poids de l'Athlète

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

- Résultat

```
Poids de l'Athlète invalide.
```

```
Procédure PL/SQL terminée.
```

➤ Procédure *PratiquerUneDiscipline* :

```
CREATE OR REPLACE PACKAGE Gest_JO IS
  PROCEDURE PratiquerUneDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_nom_discipline Discipline.nomd%TYPE
  );
END Gest_JO;
/
```

```
CREATE OR REPLACE PACKAGE BODY Gest_JO IS
  PROCEDURE PratiquerUneDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_nom_discipline Discipline.nomd%TYPE
  ) IS
    -- Déclaration des variables
    v_nda Athlete.nda%TYPE;
    v_cds Sport.cds%TYPE;
    v_ndd Discipline.ndd%TYPE;
    erreur_athlete EXCEPTION;
    erreur_sport EXCEPTION;
    erreur_discipline EXCEPTION;
```



```
BEGIN
  -- Vérification de l'existence de l'Athlète
  BEGIN
    SELECT nda INTO v_nda
    FROM Athlete
    WHERE ncomplet = p_nom_athlete;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE erreur_athlete;
  END;

  -- Vérification de l'existence du Sport
  BEGIN
    SELECT cds INTO v_cds
    FROM Sport
    WHERE noms = p_nom_sport;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE erreur_sport;
  END;

  -- Vérification de l'existence de la discipline
  BEGIN
    SELECT ndd INTO v_ndd
    FROM Discipline
    WHERE nomd = p_nom_discipline
    AND cds = v_cds;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE erreur_discipline;
  END;

  -- Ajouter La pratique de La Discipline par L'Athlète
  INSERT INTO Pratiquer(nda, ndd)
  VALUES(v_nda, v_ndd);

  -- Affichage du message de création de La pratique
  DBMS_OUTPUT.PUT_LINE('L''Athlète ' || p_nom_athlete || ' pratique la
discipline ' || p_nom_discipline);

  -- Validation
  COMMIT;

EXCEPTION
  -- Gestion des erreurs
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Cet Athlète pratique déjà la discipline.');
```

```
  WHEN erreur_athlete THEN
```

```

        DBMS_OUTPUT.PUT_LINE('Athlète inconnu !');
    WHEN erreur_sport THEN
        DBMS_OUTPUT.PUT_LINE('Sport inconnu !');
    WHEN erreur_discipline THEN
        DBMS_OUTPUT.PUT_LINE('Discipline inconnu !');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
    END PratiquerUneDiscipline;
END Gest_JO;
/

SET SERVEROUTPUT ON;
EXEC Gest_JO.PratiquerUneDiscipline('BARRY Mamadou bailo', 'Basketball', 'Men');

```

Résultats de l'exécution de la requête :

```

L'Athlète BARRY Mamadou bailo pratique la discipline Men

Procédure PL/SQL terminée.

```

Vérification de la gestion des erreurs :

- Pour le nom de l'Athlète

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.PratiquerUneDiscipline('DIAKITE Fode', 'Basketball', 'Men');

```

- Résultat

```

Athlète inconnu !

Procédure PL/SQL terminée.

```

- Pour le nom du Sport

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.PratiquerUneDiscipline('BARRY Mamadou bailo', 'Courir', 'Men');

```

- Résultat

```

Sport inconnu !

Procédure PL/SQL terminée.

```

- Pour le nom de la Discipline

```

SET SERVEROUTPUT ON;
EXEC Gest_JO.PratiquerUneDiscipline('BARRY Mamadou bailo', 'Basketball', 'Sauter
très haut');

```

- Résultat

```
Discipline inconnu !
```

```
Procédure PL/SQL terminée.
```

- Pour le fait que l'Athlète pratique déjà la Discipline

```
SET SERVEROUTPUT ON;
```

```
EXEC Gest_JO.PratiquerUneDiscipline('BARRY Mamadou bailo', 'Basketball', 'Men');
```

- Résultat

```
Cet Athlète pratique déjà la discipline.
```

```
Procédure PL/SQL terminée.
```

➤ Procédure *PratiquerDesDisciplines* :

```
CREATE OR REPLACE PACKAGE Gest_JO IS
  PROCEDURE PratiquerUneDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_nom_discipline Discipline.nomd%TYPE,
    p_valider BOOLEAN DEFAULT TRUE
  );

  PROCEDURE PratiquerDesDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_listes_noms_disciplines VARCHAR2
  );
END Gest_JO;
/

CREATE OR REPLACE PACKAGE BODY Gest_JO IS
  ----- Modification de la procédure PratiquerUneDiscipline -----
  PROCEDURE PratiquerUneDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_nom_discipline Discipline.nomd%TYPE,
    p_valider BOOLEAN DEFAULT TRUE
  ) IS
    -- Déclaration des variables
    v_nda Athlete.nda%TYPE;
    v_cds Sport.cds%TYPE;
    v_ndd Discipline.ndd%TYPE;
    erreur_athlete EXCEPTION;
    erreur_sport EXCEPTION;
    erreur_discipline EXCEPTION;

  BEGIN
```

```
-- Vérification de l'existence de l'Athlète
BEGIN
    SELECT nda INTO v_nda
    FROM Athlete
    WHERE ncomplet = p_nom_athlete;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE erreur_athlete;
END;

-- Vérification de l'existence du Sport
BEGIN
    SELECT cds INTO v_cds
    FROM Sport
    WHERE noms = p_nom_sport;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE erreur_sport;
END;

-- Vérification de l'existence de la discipline
BEGIN
    SELECT ndd INTO v_ndd
    FROM Discipline
    WHERE nomd = p_nom_discipline
    AND cds = v_cds;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE erreur_discipline;
END;

-- Ajouter la pratique de la Discipline par l'Athlète
INSERT INTO Pratiquer(nda, ndd)
VALUES(v_nda, v_ndd);

-- Affichage du message de création de la pratique
DBMS_OUTPUT.PUT_LINE('L''Athlète ' || p_nom_athlete || ' pratique la
discipline ' || p_nom_discipline);

-- Validation
IF p_valider THEN
    COMMIT;
END IF;

EXCEPTION
    -- Gestion des erreurs
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Cet Athlète pratique déjà la discipline.');
```

```

    WHEN erreur_athlete THEN
        DBMS_OUTPUT.PUT_LINE('Athlète inconnu !');
    WHEN erreur_sport THEN
        DBMS_OUTPUT.PUT_LINE('Sport inconnu !');
    WHEN erreur_discipline THEN
        DBMS_OUTPUT.PUT_LINE('Discipline inconnu !');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END PratiquerUneDiscipline;

-----Procédure PratiquerDesDisciplines -----
PROCEDURE PratiquerDesDiscipline(
    p_nom_athlete Athlete.ncomplet%TYPE,
    p_nom_sport Sport.noms%TYPE,
    p_listes_noms_disciplines VARCHAR2
) IS
    -- Déclaration des variables
    v_nom_discipline Discipline.nomd%TYPE;
    v_position NUMBER := 1;

BEGIN
    -- Extraction du premier nom de discipline
    v_nom_discipline := REGEXP_SUBSTR(p_listes_noms_disciplines, '^[^-]+', 1,
v_position);

    -- Boucle WHILE pour chaque discipline de la liste
    WHILE v_nom_discipline IS NOT NULL LOOP
        PratiquerUneDiscipline(
            p_nom_athlete => p_nom_athlete,
            p_nom_sport => p_nom_sport,
            p_nom_discipline => v_nom_discipline,
            p_valider => FALSE
        );

        v_position := v_position + 1;
        -- Extraction du prochain nom de discipline
        v_nom_discipline := REGEXP_SUBSTR(p_listes_noms_disciplines, '^[^-]+', 1,
v_position);
    END LOOP;

    -- Validation
    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END PratiquerDesDiscipline;

```

```

END Gest_JO;
/

SET SERVEROUTPUT ON;
EXEC Gest_JO.PratiquerDesDiscipline('BARRY Mamadou bailo', 'Football', 'Men-Women');

```

Résultats de l'exécution de la requête :

```

Elément Package GEST_JO compilé

Elément Package Body GEST_JO compilé

L'Athlète BARRY Mamadou bailo pratique la discipline Men
L'Athlète BARRY Mamadou bailo pratique la discipline Women

Procédure PL/SQL terminée.

```

Vérification de la gestion des erreurs :

Pour la gestion des erreurs, c'est identique à la procédure *PratiquerUneDiscipline*.

➤ Procédure *AfficheMedaille* :

```

CREATE OR REPLACE PACKAGE Gest_JO IS
    PROCEDURE AfficherMedaille(
        p_cio Pays.cio%TYPE
    );
END Gest_JO;
/

CREATE OR REPLACE PACKAGE BODY Gest_JO IS
    PROCEDURE AfficherMedaille(
        p_cio Pays.cio%TYPE
    ) IS
        -- Déclarations des variables
        v_ncomplet Pays.ncomplet%TYPE;
        v_total_medailles NUMBER := 0;

        -- Exception personnalisée
        pays_inconnu EXCEPTION;

    BEGIN
        -- Vérification de l'existence du pays et récupération du nom complet
        BEGIN
            SELECT ncomplet INTO v_ncomplet
            FROM Pays
            WHERE cio = p_cio;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                RAISE pays_inconnu;
        END;
    END AfficherMedaille;
END Gest_JO;

```

```

END;

-- Affichage du code CIO et du nom complet du pays
DBMS_OUTPUT.PUT_LINE('Pays #' || p_cio || '# : ' || v_ncomplet);

-- Récupération des sports aux quels a participé un pays (en équipes ou en
individuel)
FOR sports_recuperes IN (
    SELECT DISTINCT s.cds, s.noms
    FROM Sport s
    JOIN Discipline d ON d.cds = s.cds
    LEFT JOIN GAGNER_E g_e ON g_e.ndd = d.ndd
    LEFT JOIN GAGNER_I gi ON gi.ndd = d.ndd
    LEFT JOIN Athlete a ON (gi.nda = a.nda AND a.cio = p_cio)
    WHERE g_e.cio = p_cio OR a.cio = p_cio
    ORDER BY s.cds
) LOOP
    -- Affichage du sport
    DBMS_OUTPUT.PUT_LINE('* ' || sports_recuperes.cds || ' ' ||
sports_recuperes.noms);

    -- Récupération des des Discipline des Sports pratiques
    FOR disciplines_recuperees IN (
        SELECT DISTINCT d.ndd, d.nomd,
            (SELECT COUNT(*) FROM GAGNER_E g_e WHERE g_e.ndd = d.ndd AND
g_e.cio = p_cio) as medaille_equipe,
            (SELECT COUNT(*) FROM GAGNER_I gi
            JOIN Athlete a ON gi.nda = a.nda
            WHERE gi.ndd = d.ndd AND a.cio = p_cio) as
medaille_individuelle
        FROM Discipline d
        LEFT JOIN GAGNER_E g_e ON g_e.ndd = d.ndd
        LEFT JOIN GAGNER_I gi ON gi.ndd = d.ndd
        LEFT JOIN Athlete a ON (gi.nda = a.nda AND a.cio = p_cio)
        WHERE d.cds = sports_recuperes.cds
        AND (g_e.cio = p_cio OR a.cio = p_cio)
        ORDER BY d.ndd
    ) LOOP
        -- Calcul du nombre total de médailles pour cette discipline
        IF disciplines_recuperees.medaille_equipe > 0 OR
disciplines_recuperees.medaille_individuelle > 0 THEN
            -- Affichage de la discipline et des médailles
            DBMS_OUTPUT.PUT_LINE(' ' || LPAD(disciplines_recuperees.ndd,
3) || ' ' ||
                                RPAD(disciplines_recuperees.nomd, 30) ||
                                LPAD(TO_CHAR(disciplines_recuperees.medaille
_equipe + disciplines_recuperees.medaille_individuelle), 3));

```

```

-- Mise à jour du total des médailles
v_total_medailles := v_total_medailles +
disciplines_recuperees.medaille_equipe +
disciplines_recuperees.medaille_individuelle;
    END IF;
    END LOOP;
END LOOP;

-- Affichage du nombre total de médailles
IF v_total_medailles > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nombre total de médailles : ' ||
v_total_medailles);
ELSE
    DBMS_OUTPUT.PUT_LINE('Aucune médaille pour ce pays.');
```

END IF;

EXCEPTION

WHEN pays_inconnu THEN

DBMS_OUTPUT.PUT_LINE('Erreur : Pays inconnu (code CIO : ' || p_cio ||

'))');

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Erreur inattendue : ' || SQLERRM);

END AfficherMedaille;

END Gest_JO;

/

SET SERVEROUTPUT ON;

EXEC Gest_JO.AfficherMedaille('FRA');

Résultats de l'exécution de la requête sans erreurs :

Elément Package GEST_JO compilé

Elément Package Body GEST_JO compilé

Pays #FRA# : France

* ARC Archery

2 Women's Individual 1

3 Men's Team 1

* ATH Athletics

54 Women's 100m Hurdles 1

* BKB Basketball

75 Men 1

76 Women	1
* BKG Breaking	
94 B-Boys	1
* BK3 Basketball 3x3	
77 Men	1
* BMF Cycling BMX Freestyle	
113 Men's Park	1
* BMX Cycling BMX Racing	
115 Men	3
* BOX Boxing	
81 Men's 51kg	1
83 Men's 63.5kg	1
87 Men's +92kg	1
* CRD Cycling Road	
118 Men's Road Race	2
* CSL Canoe Slalom	
96 Men's Kayak Single	1
98 Men's Canoe Single	1
101 Women's Kayak Cross	1
* CTR Cycling Track	
126 Men's Omnium	1
* EQU Equestrian	
142 Eventing Team	1
146 Jumping Team	1
* FBL Football	
160 Men	1
* FEN Fencing	
148 Women's Épée Individual	1
150 Women's Sabre Individual	2
151 Men's Épée Individual	1
154 Women's Épée Team	1
158 Men's Foil Team	1
159 Men's Sabre Team	1
* HBL Handball	

165	Women	1
* JUD Judo		
168	Women -48 kg	1
169	Women -52 kg	1
170	Women -57 kg	1
171	Women -63 kg	1
174	Women +78 kg	1
175	Men -60 kg	1
177	Men -73 kg	1
179	Men -90 kg	1
181	Men +100 kg	1
182	Mixed Team	1
* MPN Modern Pentathlon		
185	Women's Individual	1
* MTB Cycling Mountain Bike		
116	Men's Cross-country	1
117	Women's Cross-country	1
* RU7 Rugby Sevens		
203	Men	1
* SAL Sailing		
210	Women's Kite	1
212	Women's Skiff	1
* SHO Shooting		
224	25m Pistol Women	1
* SRF Surfing		
238	Men	1
239	Women	1
* SWM Swimming		
245	Women's 1500m Freestyle	1
257	Men's 50m Freestyle	1
266	Men's 200m Breaststroke	1
268	Men's 200m Butterfly	1
269	Men's 200m Individual Medley	1
270	Men's 400m Individual Medley	1

```
273 Men's 4 x 100m Medley Relay    1
* TKW Taekwondo
280 Men -58kg                      1
287 Women +67kg                    1
* TRI Triathlon
295 Women's Individual             1
296 Men's Individual               1
* TTE Table Tennis
275 Men's Singles                  1
278 Men's Team                     1
* VVO Volleyball
298 Men                            1
```

Nombre total de médailles : 64

Procédure PL/SQL terminée.

Résultats de l'exécution de la requête avec erreurs :

- Code :

```
SET SERVEROUTPUT ON;
EXEC Gest_JO.AfficherMedaille('BMS');
```

- Résultat :

```
Erreur : Pays inconnu !
```

```
Procédure PL/SQL terminée.
```

3. Conclusion

En conclusion, ce TP nous a permis de renforcer notre connaissance en PL/SQL en implémentant des procédures allant de simples à complexes, tout en gérant les cas d'erreurs.