Ingénierie des base de données
TD 4 : Langage PL/SQL
Responsable du cours : Dr. Mariem Mahfoudh & Dr. Ines Zouari

Exercice 1 :

Soit la table RES(NO).
Écrire un bloc PL/SQL qui insère les chiffres de 1 à 100 dans cette table.

Correction

```
DECLARE
nb NUMBER := 1 ;
BEGIN
LOOP
INSERT INTO RES VALUES(nb) ;
nb = nb + 1 ;
EXIT WHEN nb > 100 ;
END LOOP
END
```

Exercice 2 :

On considère la table suivante :

`PILOTE(Matricule, Nom, Ville, Age, Salaire).`

Écrire un programme PL/SQL qui calcule la moyenne des salaires des pilotes dont l'âge est entre 30 et 40 ans.

Correction

```
DECLARE
CURSOR curseur1 IS SELECT salaire FROM pilote
WHERE (Age >= 30 AND Age <=40);
salairePilote Pilote.Salaire%TYPE;
sommeSalaires NUMBER(11,2) := 0;
moyenneSalaires NUMBER(11,2);

BEGIN
```

```
OPEN curseur1;
LOOP
FETCH curseur1 INTO salairePilote;
EXIT WHEN (curseur1%NOTFOUND OR curseur1%NOTFOUND IS NULL);
sommeSalaires := sommeSalaires + salairePilote;
END LOOP;
moyenneSalaires := sommeSalaires / curseur1%ROWCOUNT;
CLOSE curseur1;
DBMS_OUTPUT.PUT_LINE('Moyenne salaires (pilotes de 30 <E0> 40 ans) : ' ||
moyenneSalaires);
END;
```

Exercice 3 :

Des clubs de ski souhaitent informatiser la gestion des compétitions. Pour cela, ils utilisent une base de données dans laquelle les relations suivantes sont définies (clés primaires soulignés, étrangères en italique) :

Skieur (ski number(4) , nom_skieur varchar2(100) ,
prénom varchar2(100), spécialité varchar2(100),
nom_stat varchar2(100))

Cette relation décrit un skieur : le numéro, le nom, le prénom d'un skieur, sa spécialité (slalom, descente,...) ainsi que le nom de sa station d'origine.

Station (nom_stat varchar2(100), alt_stat number(4) ,
pays_stat varchar2(100), capacité number(4))

Cette relation donne le nom, l'altitude, le pays et la capacité d'une station de ski (ou le nombre maximum de personnes pouvant y héberger en plus des autochtones).

Compétition (comp number(4) , date_comp date , *nom_stat* varchar2(100),
nb_participants number(4) , nb_inscrits number(4))

Cette relation donne le numéro, la date d'une compétition, le nom de la station où elle s'est déroulée, le nombre de participants (compétiteurs) ainsi que le nombre total de personnes attendues pour la compétition.

Classement (*comp , ski*, rang number(4))

Cette relation donne le numéro d'une compétition, le numéro d'un skieur ainsi que son classement dans cette compétition (1 pour le premier, 2 pour le deuxième...).

Travail demandé

1. Est-ce que la capacité de la station d'accueil de la compétition numéro 1234 est atteinte ? Si oui, afficher un message de confirmation. Sinon, afficher le nombre de places d'hébergement restantes. En cas de dépassement de capacité, afficher un message d'alerte avec le surplus d'inscriptions. Ecrire un bloc PL/SQL.

Correction

```
SET SERVEROUPUT ON
DECLARE
diff competition.#compet%TYPE;
BEGIN
SELECT s.capacité - c.nb_inscrits INTO diff
FROM COMPETITION c, STATION s
WHERE #compet = 1234 AND c.nom_stat=s.nom_stat;
IF diff IS NULL THEN
   DBMS_OUTPUT .put_line ('Attribut capacité et/ou nb_inscrits
non définis') ;
ELSE IF diff < 0 THEN
   DBMS_OUTPUT .put_line ('Capacité dépassée de :' || DIFF) ;
ELSE IF DIFF = 0 THEN
   DBMS_OUTPUT .put_line ('Capacité atteinte') ;
ELSE IF DIFF > 0 THEN
   DBMS_OUTPUT .put_line ('Capacité pas encore atteinte. Il reste
' || DIFF || ' places.') ;
END IF; END IF ; END IF ; END IF ;
END;
```

2. (a) Ecrire une procédure PL/SQL qui insère un nouveau classement dans la table Classement. La procédure reçoit en entrée un numéro de compétition, un numéro de skieur et un rang.

   (b) Identifier le cas d'erreur lié aux paramètres de la procédure et le dérouter à l'aide d'exceptions. Penser à valider la transaction en cas de succès de l'insertion (COMMIT).

   Correction

```
CREATE OR REPLACE PROCEDURE NEW_RANK (NOC NUMBER,
NOS NUMBER, RANG NUMBER)
AS
PARAM_NULL EXCEPTION ;
SKIEUR_INEX EXCEPTION ;
RANG_OUT EXCEPTION ;
NS INTEGRER ; NP INTEGER ; -- NP : NOMBRE DE PARTICPANTS A LA COMPET
```

```
BEGIN
IF (NOC IS NULL) OR (NOS IS NULL) OR (RANK IS NULL)THEN
   RAISE PARAM_NULL; END IF;
SELECT COUNT(*) INRO NS FROM SKIEUR WHERE NO_SKIEUR=NOS;
IF NS=0 THEN
   RAISE SKIEUR_INEX; END IF;
SELECT NB_PARTICIPANTS INTO NP FROM COMPETITION WHERE NO_COMP = NOC;
IF (RANG <0) OR (RANG > NP) THEN
   RAISE RANG_OUT; END IF;
INSERT INTO CLASSEMENT VALUES (NOC,NOS,RANG);
COMMIT;
EXCEPTION
WHEN PARAM_NULL THEN
    RAISE_APLLICATION_ERROR ('UN DES TROIS PARAMETRES DE LA PROCEDUR
WHEN SKIEUR_INEX THEN
   RAISE_APLLICATION_ERROR ('AUCUN SKIEUR AVEC CE NUMERO');
WHEN NO_DATA_FOUND THEN
   RAISE_APLLICATION_ERROR ('AUCUNE COMPETION AVEC CE NUMERO');
WHEN RANG_OUT
   THEN RAISE_APLLICATION_ERROR ('CLASSEMENT INCORRECT');
END NEW_RANK ;
```

3. Ecrire une fonction PL/SQL qui retourne le nom de la station d'origine d'un skieur dont on donne le numéro. Cas d'erreur : si numéro est null ou non répertorié. nota bene : Eviter d'afficher des messages à partir d'une fonction. Préférer l'envoi d'un code d'erreur d'application (raise_application_error).

   correction

```
CREATE OR REPLACE FUNCTION NOM_STAT (NOS INTEGRER)
RETURN VARCHAR
NOS_NULL EXCEPTION;
NOM VARCHAR(100);
IS
BEGIN
IF NOS IS NULL THEN RAISE NOS_NULL; END IF;
SELECT NOM_STATION INTO NOM
FROM SKIEUR WHERE NO_SKIEUR = NOS;
RETURN NOM
EXCEPTION
WHEN NO_DATA_FOUND THEN
   RAISE_APLLICATION_ERROR ('VALEUR DE PARAMETRE INCONNUE
DE LA FONCTION NOM_STAT') ;
WHEN NOS_NULL THEN
    RAISE_APLLICATION_ERROR ('AUCUN SKIEUR AVEC CE NUMERO');
END ;
```

Exercice 4

Considérons le schéma de la BD cinéma (clés soulignées, clés étrangères en italique)

Cinéma (nomCiné, adresse, téléphone)

Film (titre, pays, année, genre, réalisateur)

Salle (noSalle, *nomCiné*, nbPlaces, dolby)

Programme (*nomCiné, noSalle*, titre, dateDébAffiche, dateFinAffiche, nbEntrées)

Distribution (acteur, *titre*)

Travail demandé

Répondre aux questions suivantes dans le langage PL/SQL.

1. Est-ce que l'acteur Réno a joué dans tous les films réalisés par Besson ? Afficher un message clair dans chacun des cas.

   Correction :

```
DECLARE
N INTEGER ; -- nombre de films réalisés par Besson
et dans lesquels Réno n'a pas joué
BEGIN
SELECT COUNT(*) INTO N
FROM FILM F
WHERE F.RÉALISATEUR = 'BESSON'
AND NOT EXISTS (SELECT *
FROM DISTRIBUTION D
WHERE D.ACTEUR = 'RÉNO'
AND D.TITRE = F.TITRE))).
IF N > 0 THEN
DBMS_OUTPUT.PUT_LINE ('Non : Réno n'a pas joué
dans tous les films de Besson') ;
ELSE
DBMS_OUTPUT.PUT_LINE ('Oui : Réno a joué dans tous les films de Besson'
END IF ;
END ;
```

2. Afficher le titre et le réalisateur des 5 meilleurs films en termes de nombre d'entrées. On considérera tous les films (même ceux qui sont encore à l'affiche).

   Correction

```
DECLARE
CURSOR C IS (SELECT F.TITRE, F.REALISATEUR, SUM(P.NBENTREES) TOTAL -- S
FROM PROGRAMME P, FILM F WHERE F.TITRE=P.TITRE
GROUP BY F.TITRE
ORDER BY TOTAL DESC);
T_FILM C%ROWTYPE;
BEGIN
OPEN C;
IF C%ISOPEN THEN
FETCH C INTO T_FILM;
WHILE (C%FOUND AND C%ROWCOUNT < 6) LOOP
/* tant qu'il y a des tuples et qu on en a traité moins de 6 */
DBMS_OUTPUT.PUT_LINE ('LE FILM '|| T_FILM.TITRE || 'REALISE PAR ' ||
T_FILM.REALISATEUR || ' EST DANS LE TOP FIVE');
FETCH C INTO T_FILM;
END LOOP;
ELSE
DBMS_OUTPUT.PUT_LINE ('ERREUR LORS DE Ll'OUVERTURE DU CURSEUR C');
END IF;
CLOSE C;
END ;
```

3. Ecrire une fonction nommée score_moyen qui retourne, pour une ville donnée, le nombre moyen d'entrées par film.

Correction

```
CREATE OR REPLACE FUNCTION SCORE_MOYEN (VILLE VARCHAR2)
RETURN INTEGER
IS
TOTAL INTEGER;
NB_FILM INTEGER;
MOYENNE INTEGER;
PAS_DE_FILM EXCEPTION;
BEGIN
SELECT SUM(NBENTREES),COUNT(DISTINCT TITRE) INTO TOTAL,NB_FILM
FROM PROGRAMME P, CINEMA C
WHERE P.NOMCINE=C.NOMCINE AND C.ADRESSE LIKE '%'||VILLE||'%' ;
IF (NB_FILM = 0) THEN
RAISE PAS_DE_FILM;
ELSE MOYENNE:=TOTAL/NB_FILM; RETURN(MOYENNE);
END IF;
EXCEPTION
WHEN PAS_DE_FILM THEN RAISE_APPLICATION_ERROR('PAS DE FILM DANS LA VILI
END;
```

4. (a) Ecrire une procédure nommée palmarès donnant pour chaque film, qui n'est plus à l'affiche, une appréciation tenant compte du nombre de spectateurs ayant vu ce film dans

une ville donnée (paramètre de la procédure) en comparaison par rapport à une valeur moyenne qui sera le second paramètre de la procédure (et qui vaudra 10 000 par défaut).

Correction

```
CREATE PROCEDURE PALMARES (VILLE VARCHAR2,MOYENNE INTEGER
DEFAULT 10000)
AS
V_TITRE PROGRAMME.TITRE%TYPE;
V_TOTAL INTEGER ;
CURSOR C IS (SELECT P.TITRE, SUM(P.NBENTRÉES) AS TOTAL
FROM PROGRAMME P, CINÉMA C
WHERE UPPER(C.ADRESSE) LIKE '%'||VILLE||'%'
AND P.NOMCINÉ=C.NOMCINÉ
GROUP BY P.TITRE);
BEGIN
OPEN C;
IF C%ISOPEN THEN
LOOP
FETCH C INTO V_TITRE, V_TOTAL;
EXIT WHEN C%NOTFOUND; -- sortie de la boucle
IF V_TOTAL < 0.95*MOYENNE THEN
DBMS_OUTPUT.PUT_LINE ('LE FILM '|| V_TITRE || ' A FAIT FLOP');
ELSE IF V_TOTAL > 1.05*MOYENNE THEN
DBMS_OUTPUT.PUT_LINE ('LE FILM '|| V_TITRE || ' A ETE UN SUCCES');
ELSE
DBMS_OUTPUT.PUT_LINE ('LE FILM '|| V_TITRE || ' A ETÉ MOYEN');
END IF ;
END IF ;
END LOOP;
ELSE
DBMS_OUTPUT.PUT_LINE ('ERREUR LORS DE Ll'OUVERTURE DU CURSEUR C');
END IF;
CLOSE C;
END PALMARES;
```

(b) Exécuter la procédure palmarès pour la ville de Kairouan et une moyenne de 10 000.

Correction

```
PALMARES ('Kairouan'); -- le second paramètre prend sa valeur par dé
ou
PALMARES (VILLE => 'Kairouan') ;
```

(c) Exécuter palmarès pour Tunis et une moyenne de 50 000.

Correction

```
PALMARES ('Tunis', 50000);
ou
PALMARES (VILLE => 'Tunis', MOYENNE => 50000) ;
```

(d) Exécuter palmarès pour Sousse et comme moyenne le nombre moyen d'entrées par film pour la ville de Sousse

Correction

```
PALMARES (VILLE =>'Sousse', MOYENNE => SCORE_MOYEN('Sousse'));
```