Exercices Corrigés PL/SQL TD SQL Oracle

Oracle 0

Exercice 1 :

Parmi les déclarations de variables suivantes, déterminer celles qui sont incorrectes :

A - DECLARE  
v\_id NUMBER(4);  
Correcte

B - DECLARE  
v\_x,v\_y,v\_z VARCHAR2(10);  
Incorrecte : un seul identifiant par ligne

C - DECLARE  
v\_date\_naissance DATE NOT NULL;  
Incorrecte : une valeur NOT NULL doit être initialisée

D - DECLARE  
v\_en\_stock BOOLEAN := 1;  
Incorrecte : 1 n’est pas une valeur booléenne

E - DECLARE  
emp\_record emp\_record\_type;  
Incorrecte : EMP\_RECORD\_TYPE doit être déclaré

F - DECLARE  
TYPE type\_table\_nom IS TABLE OF VARCHAR2(20)  
INDEX BY BINARY\_INTEGER;  
dept\_table\_nom type\_table\_nom;  
Correcte

Exercice 2 :

1 - Créer un bloc PL/SQL pour insérer un nouveau département dans la table DEPARTEMENTS

a) Utiliser la séquence DEPT\_ID\_SEQ pour générer un numéro de département. Créer un paramètre pour le nom du département. Laisser le numéro de région à NULL.

Fichier p2q1.sql

ACCEPT p\_dept\_nom PROMPT ‘Entrer un nom de département : ‘  
BEGIN  
INSERT INTO departements(id, nom, region\_id)  
VALUES (dept\_id\_seq.NEXTVAL, ‘&p\_dept\_nom’, NULL);  
COMMIT;  
END;  
/

b) Exécuter le bloc PL/SQL avec la valeur “Santé” pour le nom du département.

SQL> start p2q1  
Entrer un nom de département : Santé  
PL/SQL procedure successfully completed.

c) Afficher le nouveau département créé

SQL> SELECT \* FROM departements  
2 WHERE nom = 'Santé';  
ID NOM REGION\_ID  
--------- ------------------------- ---------  
82 Santé

2 - Créer un bloc PL/SQL pour supprimer le département créé précédemment

a) Créer un paramètre pour le numéro de département. Faire afficher à l’écran le nombre de lignes affectées.

Fichier p2q2.sql  
ACCEPT p\_dept\_id PROMPT 'Entrer un numéro de département : '  
VARIABLE g\_mess VARCHAR2(30)  
DECLARE  
v\_resultat NUMBER(2);  
BEGIN  
DELETE FROM departements  
WHERE id = &p\_dept\_id;  
v\_resultat := SQL%ROWCOUNT;  
:g\_mess := TO\_CHAR(v\_resultat)||' ligne(s) supprimée(s).';  
COMMIT;  
END;  
/  
PRINT g\_mess

b) Tester le bloc. Que se passe-t-il si on saisit un numéro de département qui n’existe pas?

SQL> start p2q2  
Entrer un numéro de département : 134  
PL/SQL procedure successfully completed.  
G\_MESS  
--------------------------------  
0 ligne(s) supprimée(s).

Si on saisit un numéro de département qui a des employés?

SQL> start p2q2  
Entrer un numéro de département : 31  
DECLARE  
\*  
ERROR at line 1:  
ORA-02292: integrity constraint (BOHEZ.EMPLOYES\_DEPT\_ID\_FK) violated - child record found  
ORA-06512: at line 4

c) Tester le bloc avec le département Santé (82)

SQL> start p2q2  
Entrer un numéro de département : 82  
PL/SQL procedure successfully completed.  
G\_MESS  
--------------------------------  
1 ligne(s) supprimée(s).

d) Vérifier que le département n’existe plus

SQL> SELECT \*  
2 FROM departements  
3 WHERE id = 82;  
no rows selected

Exercice 3 :

1 - Créer un bloc PL/SQL permettant de mettre à jour le pourcentage de commission d’un employé en fonction du total de ses ventes

Cet exercice nécessite la suppression de la contrainte sur la colonne commission de la table EMPLOYES :

SQL> ALTER TABLE employes  
2 DROP CONSTRAINT employes\_commission\_ck;

a) Créer un paramètre qui reçoit un numéro d’employé

Trouver la somme totale de toutes les commandes traitées par cet employé  
Mettre à jour le pourcentage de commission de l’employé :  
- si la somme est inférieure à 100,000 passer la commission à 10  
- si la somme est comprise entre 100,000 et 1,000,000 inclus passer la commission à 15  
- si la somme excède 1,000,000 passer la commission à 20  
- si aucune commande n’existe pour cet employé, mettre la commission à 0  
Valider la modification (commit)

Fichier p3q1.sql.

ACCEPT p\_id PROMPT 'Entrer un numéro de vendeur : '  
DECLARE  
v\_somme\_total NUMBER(11,2);  
v\_comm employes.commission%TYPE;  
BEGIN  
SELECT SUM(total)  
INTO v\_somme\_total  
FROM commandes  
WHERE vendeur\_id = &p\_id;

IF v\_somme\_total < 100000 THEN  
v\_comm := 10;  
ELSIF v\_somme\_total <= 1000000 THEN  
v\_comm := 15;  
ELSIF v\_somme\_total > 1000000 THEN  
v\_comm := 20;  
ELSE   
v\_comm := 0;   
END IF;  
UPDATE employes

SET commission = v\_comm  
WHERE id = &p\_id;  
COMMIT;  
END;  
/

b) Tester le bloc et visualiser les résultats (on teste avec les employés 1,11,12,14)

SQL> SELECT id, commission  
2 FROM employes  
3 WHERE id IN (1,11,12,14);  
ID COMMISSION  
--------- ----------  
14 10  
12 15  
11 20  
1 0

2 - Créer un bloc PL/SQL qui boucle pour chaque région (le numéro des régions va de 1 à 5) afin de modifier le code de solvabilité de tous les clients. Ne pas valider (pas de commit).

Si le numéro de région est pair mettre la solvabilité à EXCELLENTE (même si elle l’est déjà), sinon mettre à BONNE pour les numéros de région impairs.

Une fois les lignes modifiées, trouvez combien de lignes ont été mises à jour.

Afficher les résultats suivants en fonction du nombre de lignes modifiées :

Si moins de trois lignes ont été modifiées, afficher : ‘Moins de trois lignes ont été modifiées pour la région x’ (x étant le numéro de la région).  
Sinon afficher : ‘y lignes ont été modifiées pour la région x’(y étant le nombre de lignes modifiées).

Annuler les modifications (rollback).   
Fichier p3q2.sql.

VARIABLE g\_mess VARCHAR2(500)  
DECLARE  
v\_sortie VARCHAR2(500);  
v\_modifies NUMBER(2);  
v\_solvable VARCHAR2(25);  
c\_peu CONSTANT VARCHAR2(100)   
:= 'Moins de 3 lignes ont été modifiées pour la région ';

BEGIN  
FOR i IN 1..5 LOOP

IF MOD(i,2) <> 0 THEN  
v\_solvable := 'EXCELLENTE';  
ELSE   
v\_solvable := 'BONNE';  
END IF;

UPDATE clients  
SET solvabilite = v\_solvable  
WHERE region\_id = i;  
v\_modifies := SQL%ROWCOUNT;

IF v\_modifies < 3 THEN  
v\_sortie := v\_sortie||c\_peu||TO\_CHAR(i)||CHR(10);  
ELSE  
v\_sortie := v\_sortie||TO\_CHAR(v\_modifies)||  
' lignes ont été modifiées pour la région '||TO\_CHAR(i)||CHR(10);  
END IF;  
END LOOP;  
:g\_mess := v\_sortie;  
END;  
/  
PRINT g\_mess

Exercice 4 :

Créer un bloc PL/SQL qui détermine les employés de plus haut salaire.

a) Créer pour cet exercice une nouvelle table pour stocker les employés et leurs salaires

SQL> CREATE TABLE meilleurs  
2 (nom VARCHAR2(25),  
3 salaire NUMBER(11,2));

b) Utiliser un paramètre pour prendre une valeur n en entrée pour identifier les n meilleurs.

Ecrire une boucle WHILE avec curseur pour récupérer le nom et salaire des n meilleurs employés selon leur salaire dans la table EMPLOYES

Enregistrer les noms et salaires dans la table MEILLEURS.

On suppose qu’aucun employé n’a le même salaire qu’un autre.

c) Tester le bloc avec différents cas tels que n=0 ou n supérieur au nombre total d’employés (25).

Vider la table MEILLEURS après chaque test.

Fichier p4q1.sql.

ACCEPT p\_n PROMPT 'Entrer une valeur numérique : '  
DECLARE  
CURSOR emp\_cursor IS  
SELECT nom, salaire  
FROM employes  
WHERE salaire IS NOT NULL  
ORDER BY salaire DESC;  
emp\_record emp\_cursor%ROWTYPE;  
BEGIN  
OPEN emp\_cursor;  
FOR i IN 1..&p\_n  
LOOP   
FETCH emp\_cursor INTO emp\_record;  
INSERT INTO meilleurs(nom, salaire)  
VALUES (emp\_record.nom,emp\_record.salaire);  
END LOOP;  
CLOSE emp\_cursor;  
COMMIT;  
END;  
/  
SELECT nom,TO\_CHAR(salaire,'fm$9,999,999') salaire FROM meilleurs;  
TRUNCATE TABLE meilleurs

Test du bloc avec n=4

SQL> start p4q1  
Entrer une valeur numérique : 4  
PL/SQL procedure successfully completed.  
NOM SALAIRE  
----------------------------- -------------  
Velasquez $2,500  
Ropeburn $1,550  
Nguyen $1,525  
Sedeghi $1,515  
Table truncated.

Test du bloc avec n=0

SQL> start p4q1  
Entrer une valeur numérique : 0  
PL/SQL procedure successfully completed.  
no rows selected  
Table truncated.

Test du bloc avec n=30

SQL> START p4q1  
Entrer une valeur numérique : 30  
PL/SQL procedure successfully completed.  
NOM SALAIRE  
------------------------- -----------  
Velasquez $2,500  
Ropeburn $1,550  
Nguyen $1,525  
Sedeghi $1,515  
Giljum $1,490   
Ngao $1,450  
Quick-To-See $1,450  
Dumas $1,450  
Nagayama $1,400  
Maduro $1,400  
Magee $1,400  
Havel $1,307  
Catchpole $1,300  
Menchu $1,250  
Urguhart $1,200  
Nozaki $1,200  
Biri $1,100  
Schwartz $1,100  
Smith $940  
Dancs $860  
Markarian $850  
Chang $800  
Patel $795  
Patel $795  
Newman $750  
Newman $750  
Newman $750  
Newman $750  
Newman $750  
Newman $750  
30 rows selected.  
Table truncated.

Exercice 5 :

Modifier le bloc PL/SQL fourni pour gérer les exceptions.

Le traitement essaie de mettre à jour des numéros de région pour des départements existants.

a) Charger le fichier p5qa.sql.

ACCEPT p\_dept\_id PROMPT 'Numéro de département : '  
ACCEPT p\_nom\_region PROMPT 'Nom de région : '  
VARIABLE g\_mess VARCHAR2(50)  
DECLARE  
v\_region\_id regions.id%TYPE;  
BEGIN  
SELECT id  
INTO v\_region\_id  
FROM regions  
WHERE UPPER(nom)=UPPER('&p\_nom\_region');  
UPDATE departements  
SET region\_id=v\_region\_id  
WHERE id = &p\_dept\_id;  
:g\_mess := 'Le département : '||TO\_CHAR(&p\_dept\_id)||  
’ est affecté à la région '||TO\_CHAR(v\_region\_id);  
COMMIT;  
END;  
/  
PRINT g\_mess

b) Exécuter le bloc avec comme valeur 50 pour le numéro de département et US pour le nom de région.

SQL> start p5qa  
Numéro de département : 50  
Nom de région : US  
DECLARE  
\*  
ERROR at line 1:  
ORA-01403: no data found  
ORA-06512: at line 5  
G\_MESS  
-----------------------------------------------------------------

c) Sauvegarder le fichier p5qa.sql sous le nom p5q1.sql. Modifier p5q1.sql pour écrire un traitement d’exception pour l’anomalie constatée afin de passer un message à l’utilisateur lorsque la région spécifiée n’existe pas.

ACCEPT p\_dept\_id PROMPT 'Numéro de département : '   
ACCEPT p\_nom\_region PROMPT 'Nom de région : '  
VARIABLE g\_mess VARCHAR2(50)  
DECLARE  
v\_region\_id regions.id%TYPE;  
BEGIN  
SELECT id  
INTO v\_region\_id  
FROM regions  
WHERE UPPER(nom)=UPPER('&p\_nom\_region');  
UPDATE departements  
SET region\_id=v\_region\_id  
WHERE id = &p\_dept\_id;  
:g\_mess := 'Le département : '||TO\_CHAR(&p\_dept\_id)||  
’ est affecté à la région '||TO\_CHAR(v\_region\_id);  
COMMIT;  
EXCEPTION  
WHEN NO\_DATA\_FOUND THEN  
ROLLBACK;  
:g\_mess := '&p\_nom\_region'||' : région inexistante.';  
END;  
/  
PRINT g\_mess  
SQL> start p5q1  
Numéro de département : 50  
Nom de région : US  
PL/SQL procedure successfully completed.  
G\_MESS  
---------------------------------------------------------------------  
US : région inexistante.

d) Exécuter le bloc avec comme valeur 31 pour le numéro de département et Asie pour le nom de région.

SQL> START p5q1  
Numéro de département : 31  
Nom de région : Asie  
DECLARE  
\*   
ERROR at line 1:  
ORA-00001: unique constraint (BOHEZ.DEPARTEMENTS\_NOM\_ET\_REGION\_UK) violated  
ORA-06512: at line 9  
G\_MESS  
---------------------------------------------------------------------------

e) Ecrire un traitement d’exception pour l’anomalie constatée afin de passer un message à l’utilisateur lorsque la région spécifiée est déjà référencée par un département du même nom

ACCEPT p\_dept\_id PROMPT 'Numéro de département : '  
ACCEPT p\_nom\_region PROMPT 'Nom de région : '  
VARIABLE g\_mess VARCHAR2(80)  
DECLARE  
v\_dept VARCHAR2(20);  
v\_region\_id regions.id%TYPE;  
BEGIN  
SELECT id  
INTO v\_region\_id  
FROM regions  
WHERE UPPER(nom)=UPPER('&p\_nom\_region');  
UPDATE departements  
SET region\_id=v\_region\_id  
WHERE id = &p\_dept\_id;  
:g\_mess := 'Le département : '||TO\_CHAR(&p\_dept\_id)||  
’ est affecté à la région '||TO\_CHAR(v\_region\_id);  
COMMIT;  
EXCEPTION  
WHEN NO\_DATA\_FOUND THEN  
ROLLBACK;  
:g\_mess := '&p\_nom\_region'||' : région inexistante.';   
WHEN DUP\_VAL\_ON\_INDEX THEN  
ROLLBACK;  
SELECT nom||' (N°'||TO\_CHAR(id)||')'  
INTO v\_dept  
FROM departements  
WHERE region\_id = v\_region\_id  
AND nom =   
(SELECT nom  
FROM departements  
WHERE id = &p\_dept\_id);  
:g\_mess := 'Il existe déjà un département '||v\_dept||  
' pour la région '||'&p\_nom\_region';  
END;  
/  
PRINT g\_mess

SQL> start p5q1  
Numéro de département : 31  
Nom de région : Asie  
PL/SQL procedure successfully completed.  
G\_MESS  
--------------------------------------------------------------------  
Il existe déjà un département Ventes (N°34) pour la région Asie

f) Exécuter le bloc avec comme valeur 99 pour le numéro de département et Europe pour le nom de la région  
SQL> start p5q1  
Numéro de département : 99  
Nom de région : Europe  
PL/SQL procedure successfully completed.  
G\_MESS  
----------------------------------------------------------------------------------  
Le département : 99 est affecté à la région 5

Anomalie constatée : la procédure ne remarque pas que le département 99 n’existe pas.

g) Ecrire un traitement d’exception pour l’anomalie constatée afin de passer un message à l’utilisateur lorsque le numéro de département spécifié n’existe pas.

Rappel : penser à utiliser l’attribut SQL%NOTFOUND et déclencher une exception manuellement

ACCEPT p\_dept\_id PROMPT 'Numéro de département : '  
ACCEPT p\_nom\_region PROMPT 'Nom de région : '  
VARIABLE g\_mess VARCHAR2(80)  
DECLARE  
v\_dept VARCHAR2(20);  
v\_region\_id regions.id%TYPE;  
e\_count EXCEPTION;  
BEGIN  
SELECT id  
INTO v\_region\_id  
FROM regions  
WHERE UPPER(nom)=UPPER('&p\_nom\_region');  
UPDATE departements  
SET region\_id=v\_region\_id  
WHERE id = &p\_dept\_id;  
IF SQL%NOTFOUND THEN  
RAISE e\_count;  
END IF;  
:g\_mess := 'Le département : '||TO\_CHAR(&p\_dept\_id)||  
' est affecté à la région '||TO\_CHAR(v\_region\_id);  
COMMIT;  
EXCEPTION  
WHEN NO\_DATA\_FOUND THEN  
ROLLBACK;  
:g\_mess := '&p\_nom\_region'||' : région inexistante.';  
WHEN DUP\_VAL\_ON\_INDEX THEN  
ROLLBACK;  
SELECT nom||' (N°'||TO\_CHAR(id)||')'   
INTO v\_dept  
FROM departements  
WHERE region\_id = v\_region\_id  
AND nom =   
(SELECT nom  
FROM departements  
WHERE id = &p\_dept\_id);  
:g\_mess := 'Il existe déjà un département '||v\_dept||  
' pour la région '||'&p\_nom\_region';  
WHEN e\_count THEN  
ROLLBACK;  
:g\_mess := 'Le département numéro '||TO\_CHAR(&p\_dept\_id)||' n''existe pas.';  
END;  
/  
PRINT g\_mess

SQL> start p5q1  
Numéro de département : 99  
Nom de région : Europe  
PL/SQL procedure successfully completed.  
G\_MESS