M1 IDSM-Kharkiv – Année 2018-2019
Bases de données avancées
Exercices pratiques (3) : PL/SQL – Interrogation de données
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 16/07/2018

Aide-mémoire PL/SQL

Bloc PL/SQL		`DECLARE`
		`  -- Déclarations : types, curseurs, constantes,`
		`  -- variables, sous-programmes`
		`BEGIN`
		`  -- Instructions du programme principal`
		`EXCEPTION`
		`  -- Traitement des erreurs à l'exécution`
		`END;`

Déclaration de variable		`nom_variable TYPE_VARIABLE;`
Affectation		`nom_variable := valeur;` `SELECT attribut INTO nom_variable FROM table;`

Tests		`IF condition1 THEN`
		`  -- Instructions`
		`ELSIF condition2 THEN -- (Optionnel)`
		`  -- Instructions`
		`ELSE -- (Optionnel)`
		`  -- Instructions`
		`END IF;`
		`CASE nom_variable`
		`  WHEN valeur 1 THEN -- Instructions`
		`  WHEN valeur 2 THEN -- Instructions`
		`  ELSE  -- Instructions`
		`END CASE;`

Boucles		`FOR iterateur IN [REVERSE] min..max LOOP`
		`  -- Instructions`
		`END LOOP;`
		`WHILE condition LOOP`
		`  -- Instructions`
		`END LOOP;`
		`LOOP -- Répéter jusqu'à`
		`  -- Instructions`
		`  EXIT WHEN condition`
		`END LOOP;`

Curseurs	- *Déclarer*	`CURSOR nom_curseur IS requete_SQL;`

	- *Utiliser*	`FOR nuplet IN nom_curseur LOOP`
		`  -- Instructions`
		`  -- Ex. nom_variable := nuplet.attribut;`
		`END LOOP;-- NB : nuplet est de type nom_curseur%ROWTYPE`

Exceptions	- *Déclarer*	`nom_exception EXCEPTION;`
	- *Lever*	`RAISE nom_exception;`
	- *Traiter*	`WHEN nom_exception THEN -- Instructions;`
Documentation PL/SQL		http://docs.oracle.com/database/121/LNPLS/

Exercice 1 : Requêtes simples et exception

Utiliser de nouveau Oracle SQL Live et la SQL Worksheet.

1. Créer de nouveau la table EMP avec le script SQL http://eric.univ-lyon2.fr/~jdarmont/docs/emp.sql.

2. Écrire un bloc PL/SQL anonyme qui permet de :

- compter le nombre total de n-uplets dans la table EMP et stocker le résultat dans une variable ;
- compter le nombre d'employés dont la fonction (JOB) est MANAGER dans la table EMP et stocker le résultat dans une deuxième variable ;
- calculer la proportion (en pourcentage), stocker le résultat dans une troisième variable et afficher le résultat à l'écran.

3. Exécuter le programme !

3. Inclure dans le programme précédent une exception pour détecter si la table EMP est vide, c'est-à-dire que le nombre total de n-uplets dans EMP est égal à zéro. Dans ce cas, déclencher une erreur fatale `RAISE_APPLICATION_ERROR` (on ne peut pas permettre une division par zéro). Tester avec la procédure suivante :

1. valider les mises à jour précédentes à l'aide de la commande SQL `COMMIT` ;
2. effacer le contenu de la table EMP (`DELETE FROM emp`) ;
3. exécuter le bloc PL/SQL ;
4. annuler l'effacement de la table EMP à l'aide de la commande SQL `ROLLBACK`.

Exercice 2 : Curseur implicite

1. Créer de nouveau la table DEPT et la vue EMPDIR à partir du script SQL : https://eric.univ-lyon2.fr/~jdarmont/docs/dept-empdir.sql.

2. Écrire un bloc PL/SQL anonyme permettant d'afficher votre catalogue système (liste des tables et des vues de votre compte, disponible avec la vue système TAB) sous la forme :

L'objet UNE_TABLE est de type TABLE.
L'objet UNE_AUTRE_TABLE est de type TABLE.
L'objet UNE_VUE est de type VIEW.
…

Indication : Exécuter avant la requête `SELECT * FROM TAB` pour voir le contenu de la vue système TAB.

Exercice 3 : Curseur explicite

1. Créer la table DEMO_PRODUCT_INFO à partir du script SQL : https://eric.univ-lyon2.fr/~jdarmont/docs/demo_product_info.sql.

2. Écrire un bloc PL/SQL anonyme qui affiche le nom (PRODUCT_NAME) et le prix (LIST_PRICE) des 5 produits les plus chers de la table DEMO_PRODUCT_INFO.

Indications :

- Définir un curseur qui liste les produits par ordre de prix décroissant.
- Effectuer un parcours explicite du curseur qui fonctionne tant que le nombre de n-uplets lus est inférieur ou égal à 5.

Exercice 4 : Liste et curseur implicite

1. Créer la table NOTATION à partir du script SQL téléchargeable à l'adresse suivante : https://eric.univ-lyon2.fr/~jdarmont/docs/notation.sql.

2. Dans un bloc PL/SQL anonyme, définir un type liste (TABLE) de réels qui contiendra les notes de contrôle continu (NOTECC) de la table NOTATION.

3. Définir une variable (collection) de votre type et l'initialiser vide.

4. Dans la section de code, charger en mémoire dans la collection les notes contenues dans la table NOTATION à l'aide d'un curseur implicite. Afficher les notes au fur et à mesure. Tester ! Ce traitement aurait-il été possible avec un tableau ?

5. Calculer et afficher la moyenne des notes contenues dans la liste.

6. Vérifier que votre calcul est bon en affichant le résultat d'une requête SQL utilisant la fonction AVG().

7. Afficher à l'écran l'indice du premier élément de la collection, l'indice du dernier élément et le nombre d'éléments dans la collection.

8. Supprimer les éléments d'indice 1, 10 et 16, puis répéter la question 7 (copier/coller).

9. Afficher tous les éléments de la collection.

Exercice 5 : Enregistrement et curseur paramétré

1. Dans un bloc PL/SQL anonyme, définir un type enregistrement (RECORD) avec les champs suivants :
- nom, du même type que l'attribut ENAME de la table EMP ;
- fonction, du même type que l'attribut JOB de la table EMP ;
- salaire_tot, du même type que l'attribut SAL de la table EMP.

2. Définir une variable (un enregistrement) de votre type.

3. Définir un curseur paramétré permettant de lister le nom (ENAME), la fonction (JOB) et le salaire total (SAL + COMM) des employés d'un département (NODEPT) de la table EMP.

Indication : Utiliser la fonction NVL[1] pour éviter les problèmes d'addition de valeurs NULLes.

4. Dans la section de code, parcourir le curseur paramétré pour le département n° 30, stocker le résultat dans la variable enregistrement et l'afficher au format « NOM (FONCTION) : SALAIRE_TOT € ».

[1] http://docs.oracle.com/database/121/SQLRF/functions130.htm#SQLRF00684

Correction

```
-- Exercice 1

DECLARE
     ntot INTEGER;       -- Nombre total d'employés
     nman INTEGER;       -- Nombre de managers
     pman REAL;          -- Proportion de managers
     personne EXCEPTION; -- Exception : pas d'employés

BEGIN
     SELECT COUNT(*) INTO ntot FROM emp;
     IF (ntot = 0) THEN
          RAISE personne;
     END IF;

     SELECT COUNT(*) INTO nman FROM emp WHERE job = 'MANAGER';
     pman := 100 * nman / ntot;
     DBMS_OUTPUT.PUT_LINE('Proportion de managers = ' || pman || ' %');

EXCEPTION
     WHEN personne THEN
          RAISE_APPLICATION_ERROR(-20500, 'La table EMP est vide !');

END;


-- Exercice 2

DECLARE
     CURSOR catalogue IS SELECT tname, tabtype FROM tab;
     ligne catalogue%ROWTYPE;

BEGIN
     FOR ligne IN catalogue LOOP
          DBMS_OUTPUT.PUT_LINE('L''objet ' || ligne.tname || ' est de type ' ||
                               ligne.tabtype || '.');
     END LOOP;
END;


-- Exercice 3

DECLARE
     CURSOR ranking IS SELECT product_name, list_price FROM demo_product_info
                       ORDER BY list_price DESC;
     p ranking%ROWTYPE;

BEGIN
     OPEN ranking;
     FETCH ranking INTO p; -- Premier produit
     WHILE ranking%FOUND AND ranking%ROWCOUNT <= 5 LOOP
          DBMS_OUTPUT.PUT_LINE(ranking%ROWCOUNT || ') ' || p.product_name ||
                               ' : ' || p.list_price || ' €');
          FETCH ranking INTO p; -- Produit suivant
     END LOOP;
     CLOSE ranking;
END;


-- Exercice 4

DECLARE
     TYPE ListeReels IS TABLE OF REAL;
     notes ListeReels := ListeReels();
```

```
     CURSOR notes_stockees IS SELECT notecc FROM notation
          WHERE notecc IS NOT NULL;
     nuplet notes_stockees%ROWTYPE;
     i INTEGER;
     moy REAL := 0;

BEGIN
     FOR nuplet IN notes_stockees LOOP
          notes.EXTEND;
          notes(notes.COUNT) := nuplet.notecc;
          DBMS_OUTPUT.PUT_LINE('notes(' || notes.count || ') = ' ||
                               nuplet.notecc);
     END LOOP;

     FOR i IN 1..notes.COUNT LOOP
          moy := moy + notes(i);
     END LOOP;
     DBMS_OUTPUT.PUT_LINE('Moyenne = ' || moy / notes.COUNT);
     SELECT AVG(notecc) INTO moy FROM notation;
     DBMS_OUTPUT.PUT_LINE('AVG(notecc) = ' || moy);

     DBMS_OUTPUT.PUT_LINE('Indice premier  : ' || notes.FIRST);
     DBMS_OUTPUT.PUT_LINE('Indice dernier  : ' || notes.LAST);
     DBMS_OUTPUT.PUT_LINE('Nombre éléments : ' || notes.COUNT);

     notes.delete(1);
     notes.delete(10);
     notes.delete(16);
     DBMS_OUTPUT.PUT_LINE('Indice premier  : ' || notes.FIRST);
     DBMS_OUTPUT.PUT_LINE('Indice dernier  : ' || notes.LAST);
     DBMS_OUTPUT.PUT_LINE('Nombre éléments : ' || notes.COUNT);

     i := notes.FIRST;
     WHILE i <= notes.LAST LOOP
          DBMS_OUTPUT.PUT_LINE('notes(' || i || ') = ' || notes(i));
          i := notes.NEXT(i);
     END LOOP;
END;
```

-- Exercice 5

```
DECLARE
     TYPE Employe IS RECORD(
          nom emp.ename%TYPE,
          fonction emp.job%TYPE,
          salaire_tot emp.sal%TYPE);
     e Employe;
     CURSOR c(nod NUMBER) IS
          SELECT ename, job, sal + NVL(comm, 0)
          FROM emp
     WHERE deptno = nod;

BEGIN
     OPEN c(30);
     FETCH c INTO e;
     WHILE c%FOUND LOOP
          DBMS_OUTPUT.PUT_LINE(e.nom || ' (' || e.fonction || ') : ' ||
                               e.salaire_tot || ' €');
          FETCH c INTO e;
     END LOOP;
     CLOSE c;
END;
```