Université Lumière – Lyon 2, Faculté de Sciences Économiques et de Gestion
Licence IUP ISEA – Année 1999-2000
Exercices – PL/SQL
J. Darmont (*jerome.darmont@univ-lyon2.fr*), 16/03/2000

Exercice 1

2001, il est temps de passer à l'Euro pour l'entreprise de VPC dont la base de données CLIENT-COMMANDE-PRODUIT-FOURNISSEUR sert d'exemple dans le cours. Écrire un programme PL/SQL permettant de construire, à partir de la table PRODUIT, une table PRODUIT2000 telle que :

- la désignation des produits soit écrite en majuscules ;
- le prix unitaire en francs des produits soit converti en Euros. Le prix en Euros devra être entier (arrondir au supérieur).

Cas particuliers à traiter :

- Si la table PRODUIT est vide, la table PRODUIT2000 devra contenir uniquement le tuple (0,'Pas de produit', NULL, NULL).
- Si un prix de la table produit est NULL, son prix en Euros doit être 0.

Indications :

- Considérer que la structure de la table PRODUIT2000 a déjà été créée.
- Tester si la table PRODUIT est vide. Si ce n'est pas le cas, y accéder séquentiellement à l'aide d'un curseur, effectuer les transformations sur les champs et stocker le résultat dans la table PRODUIT2000.
- Utiliser les fonctions SQL*Plus `UPPER`, `TRUNC` et `NVL`.

Exercice 2

Pour tenter d'établir une corrélation, on souhaite connaître la différence de quantité moyenne entre les commandes successivement enregistrées dans la table COMMANDE de la base de données CLIENT-COMMANDE-PRODUIT-FOURNISSEUR. La table COMMANDE est remplie de commandes valuées (c'est-à-dire, pour lesquelles l'attribut QUANTITE n'est pas NULL) ou non. Les commandes non valuées ne sont pas à prendre en compte. Écrire un programme PL/SQL permettant de calculer la différence de quantité moyenne entre les commandes.

Cas particuliers à traiter :

- La table COMMANDE contient moins de deux commandes valuées.

Indications :

- Créer un curseur contenant les quantités de commandes valuées.
- Lire la première quantité puis, pour toutes les quantités suivantes, cumuler la valeur absolue de *quantité courante – quantité précédente* (fonction `ABS`).
- Afficher le résultat dans une exception à l'aide de la procédure `RAISE_APPLICATION_ERROR`. Utiliser l'opérateur de concaténation `||` et la fonction `TO_CHAR`.

Correction Exercice 1

```
-- Creation de la table PRODUIT2000 a partir de la table PRODUIT

DECLARE
  euro CONSTANT REAL:=6.55957;
  nbprod NUMBER(3);
  aucun_produit EXCEPTION;
  CURSOR acces IS
    SELECT numprod, desi, prixuni, numfour FROM produit;
  prod acces%ROWTYPE;
  newdesi produit.desi%TYPE;
  newprix produit.prixuni%TYPE;

BEGIN
  -- Compte des produits
  SELECT COUNT(*) INTO nbprod FROM PRODUIT;

  -- Si pas de produits, exception
  IF nbprod=0 THEN
    RAISE aucun_produit;
  END IF;

  -- Acces sequentiel a la table PRODUIT
  -- Remplissage de la table PRODUIT2000
  FOR prod IN acces LOOP
    newdesi:=UPPER(prod.desi);
    newprix:=NVL(prod.prixuni,0);
    IF newprix<>0 THEN
      newprix:=TRUNC(newprix/euro)+1;
    END IF;
    INSERT INTO produit2000
      VALUES(prod.numprod,newdesi,newprix,prod.numfour);
    END LOOP;

  -- Validation de la transaction
  COMMIT;

EXCEPTION
  WHEN aucun_produit THEN
    INSERT INTO produit2000
      VALUES(0,'Pas de produit',NULL,NULL);

END;
.
/
```

Correction Exercice 2

```
-- DIFFERENCE MOYENNE DE QUANTITE ENTRE LES COMMANDES

DECLARE
  CURSOR valuees IS
    SELECT quantite FROM commande WHERE quantite IS NOT NULL;
  cde valuees%ROWTYPE;
  prec REAL; -- quantite precedente
  cour REAL; -- quantite courante
  cumul REAL;
  moyenne REAL;
  ncv INTEGER;
  n INTEGER;
  pas_assez EXCEPTION;
  resultat EXCEPTION;

BEGIN
  -- Test nombre de commandes valuees
  SELECT COUNT(*) INTO ncv FROM commande WHERE quantite IS NOT NULL;
  IF ncv<2 THEN
    RAISE pas_assez;
  END IF;

  -- Acces 1er tuple
  OPEN valuees;
  FETCH valuees INTO cde;
  prec:=cde.quantite;

  -- Acces aux suivants et cumul
  cumul:=0;
  n:=0;
  LOOP
    FETCH valuees INTO cde;
    EXIT WHEN valuees%NOTFOUND;
    cour:=cde.quantite;
    cumul:=cumul+ABS(cour-prec);
    n:=n+1;
    prec:=cour;
  END LOOP;
  CLOSE valuees;

  -- Calcul et affichage de la moyenne
  moyenne:=cumul/n;
  RAISE resultat;

EXCEPTION
  -- Erreur
  WHEN pas_assez THEN
    RAISE_APPLICATION_ERROR(-20501,'Pas assez de commandes');
  -- Resultat
  WHEN resultat THEN
    RAISE_APPLICATION_ERROR(-20500,'Moyenne = '||TO_CHAR(moyenne));

END;
.
/
```