Objectif : Mettre en œuvre pratiquement vos connaissances en langage PL/SQL d'Oracle.

Exercice 1

2001, il est temps de passer à l'Euro pour l'entreprise de VPC dont la base de donn éesCLIENT-
COMMANDE-PRODUIT-FOURNISSEUR sert d'exemple dans le cours. Écrire u n programme PL/SQL permettant de construire, à partir de la table PRODUIT, une tablePRODUIT2000 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 devraêtre entier (arrondir au supérieur).

Cas particuliers à traiter :

- Si la table PRODUIT est vide, la table PRODUIT2000 devra contenir uniquem entle 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 accéderséquentiellement à l'aide d'un curseur, effectuer les transformations sur les c hamps 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 moyenneentre les
commandes successivement enregistrées dans la table COMMANDE de la base ded onnées CLIENT-
COMMANDE-PRODUIT-FOURNISSEUR. La table COMMANDE estremplie de com mandes va- luées (c'est-à-dire, pour lesquelles l'attribut QUANTITE n'est pasNULL) ou non. Les commandes non valuées ne sont pas à prendre en compte. Écrire unpro gramme PL/SQL permettant de calculer la diffé- rence de quantité moyenne entre lescommandes.

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 v aleurabsolue de *quantité courante – quantité précédente* (fonction ABS).
- Afficher le résultat dans une exception à l'aide de la procédureRAISE_APPLIC ATION_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;
```

Objectif : Mettre en œuvre pratiquement vos connaissances en langage SQL PLUS d'Oracle.

Exercice 3

Sur la base de données exemple du cours (CLIENT-COMMANDE-PRODUIT-FOURNISSEUR), formuler avec le langage SQL*Plus les requêtes suivantes.

1) Désignation et prix unitaire de tous les produits.
2) Désignation des produits de prix inférieur à 100 F.
3) Nom des clients qui ont commandé le produit n° 1.
4) Nom des clients qui ont commandé au moins un produit de prix supérieur à 500 F.
5) Nom des clients qui n'ont pas commandé le produit n° 1.
6) Numéro des clients qui ont commandé tous les produits.
7) Numéro des clients qui ont commandé tous les produits commandés par le client n° 2.

Exercice 4

Soit le schéma relationnel de la base FABRICATION.

CLIENT (NOC, NOM, ADRESSE)
SERVICE (NOS, INTITULE, LOCALISATION)
PIECE (NOP, DESIGNATION, COULEUR, POIDS) clés primaires
COMMANDE (NOP, NOS, NOC, QUANTITE) clés étrangères
Formuler en SQL*Plus les commandes de création de la structure de cette base, puis exprimer les requêtes suivantes.

1) Donner pour chaque service le poids de la pièce commandée de couleur bleue la plus pesante.
2) Donner le poids moyen des pièces commandées pour chacun des services "Promotion".
3) Donner les pièces de couleur bleue qui sont commandées par plus de trois services différents.
4) Donner le maximum parmi les totaux des quantités des pièces commandées par les différents services.

Correction Exercice 3

```
1) SELECT Desi, PrixUni
   FROM Client ;

2) SELECT Desi
   FROM Client
   WHERE PrixUni < 100 ;

3) SELECT DISTINCT Nom
   FROM Client C1, Commande C2
   WHERE C1.NumCli = C2.NumCli
   AND NumProd = 1 ;

4) SELECT DISTINCT Nom
   FROM Client C1, Commande C2, Produit P
   WHERE C1.NumCli = C2.NumCli
   AND C2.NumProd = P.NumProd
   AND PrixUni > 500 ;

5) SELECT NumCli
   FROM Client C1
   WHERE NOT EXISTS (
```

```
   SELECT *
   FROM Commande C2
   WHERE C2.NumCLi = C1.NumCli
   AND NumProd = 1) ;

6)   SELECT NumCLi
FROM Client C1
WHERE NOT EXISTS ( SELECT *
FROM Produit P WHERE NOT EXISTS (
SELECT *
FROM Commande C2
WHERE C2.NumCli = C1.NumCli
AND C2.NumProd = P.NumProd)) ;

7) SELECT Nom
   FROM Client C0
   WHERE NOT EXISTS (
     SELECT *
     FROM Commande C1
     WHERE NumCli = 2
     AND NOT EXISTS (
       SELECT *
       FROM Commande C2
       WHERE C2.NumCli = C0.NumCli
       AND C2.NumProd = C1.NumProd)) ;
```

Correction Exercice 4

```
CREATE TABLE CLIENT (NOC NUMBER(3),
                     NOM VARCHAR(40),
                     ADRESSE VARCHAR(100),
                      CONSTRAINT PRICLI PRIMARY KEY (NOC));

CREATE TABLE SERVICE (NOS NUMBER(3),
                      INTITULE VARCHAR(30),
                      LOCALISATION VARCHAR(100),
                      CONSTRAINT PRISER PRIMARY KEY (NOS));

CREATE TABLE PIECE (NOP NUMBER(3),
                    DESIGNATION VARCHAR(30),
                    COULEUR VARCHAR(20),
                    POIDS NUMBER(5,2),
                    CONSTRAINT PRIPIE PRIMARY KEY (NOP));

CREATE TABLE COMMANDE (NOP NUMBER(3),
                       NOS NUMBER(3), NOC NUMBER(3),
                       QUANTITE NUMBER(3),
                      CONSTRAINT PRICOM PRIMARY KEY (NOP, NOS, NOC),
                      CONSTRAINT ETRPIE FOREIGN KEY (NOP)
                      REFERENCES PIECE(NOP),
                      CONSTRAINT ETRSER FOREIGN KEY (NOS)
                      REFERENCES SERVICE(NOS)
                     CONSTRAINT ETRCLI FOREIGN KEY (NOC)
                     REFERENCES CLIENT(NOC));

1)  SELECT INTITULE, MAX(POIDS)
    FROM SERVICE S, COMMANDE C, PRODUIT P
```

```
   WHERE S.NOS=C.NOS
   AND C.NOP=P.NOP
   AND COULEUR='bleu' GROUP BY INTITULE ;

2) SELECT AVG(POIDS)
   FROM SERVICE S, COMMANDE C, PRODUIT P
   WHERE S.NOS=C.NOS
   AND C.NOP=P.NOP
   AND INTITULE='Promotion'
   GROUP BY S.NOS ;

3) SELECT P.NOP
   FROM PRODUIT P
  WHERE COULEUR='bleu'
  AND 3 <
  (SELECT COUNT(DISTINCT NOS)
  FROM COMMANDE C
  WHERE C.NOP=P.NOP) ;

4) SELECT MAX(SUM(QUANTITE))
   FROM COMMANDE
   GROUP BY NOS ;
```

Objectif : Mettre en œuvre pratiquement vos connaissances en langage SQL PLUS d'Oracle.

Exercice 1

Une base de données ancienne, gérée par M. Dupont, aujourd'hui à la retraite, doit être réorganisée et mise en troisième forme normale (3FN). Pour cela, il faut déterminer les dépendances fonctionnelles entre les attributs de cette base.

On supposera que vous avez accès à toutes les données de M. Dupont. Créer à l'aide de SQL*Plus une vue ATTRIBUTS permettant de lister tous les attributs de toutes les tables de la base ainsi que leur type (sans doublon). Utiliser pour cela le catalogue du système. La mise en 3FN devant être effectuée par quelqu'un d'autre, octroyer à tous les utilisateurs le droit d'accéder en lecture à la vue ATTRI- BUTS.

Exercice 2

Soit le schéma relationnel de la base de données « pilotes-avions-vols ».

PILOTE (PLNUM, PLNOM, PLPRENOM, VILLE, SALAIRE) AVION (AVNUM, AVNOM, CAPACITE, LOCALISATION)
VOL (VOLNUM, PLNUM, AVNUM, VILLEDEP, VILLEARR, HEUREDEP, HEUREARR)

Exprimer les requêtes suivantes en SQL*Plus.

1) Liste de tous les vols.

2) Nom, prénom et ville de tous les pilotes, par ordre alphabétique.

3) Nom, prénom et salaire des pilotes dont le salaire est supérieur à 20 000 F.

4) Numéro et nom des avions localisés à Paris.

5) Caractéristiques (AVNUM, AVNOM, CAPACITE, LOCALISATION) des avions localisés dans la même ville que le pilote Tanguy.

6) Caractéristiques (VOLNUM, VILLEDEP, VILLEARR, HEUREDEP, HEUREARR, AVNOM, PLNOM) du vol numéro 714.

7) Nom, prénom et numéro de vol des pilotes affectés à un vol.

8) Numéro et nom des avions affectés à des vols.

9) Nombre total de vols.

10) Somme des capacités par type (nom) d'avion.

11) Moyenne des durées des voyages.

Correction Exercice 1

```
CREATE VIEW ATTRIBUTS AS
SELECT DISTINCT COLUMN_NAME, DATA_TYPE
FROM USER_TAB_COLUMNS ATTR, ALL_TABLES TABL WHERE ATTR.TABLE_NAME=TABL.TABLE_NAME
AND OWNER='DUPONT';

GRANT SELECT ON ATTRIBUTS TO PUBLIC;
```

Correction Exercice2

1) select * from vol;

2) select plnom, plprenom, ville from pilote order by plnom, plprenom;

3) select plnom, plprenom, salaire from pilote where salaire>20000;

4) select avnum, avnom from avion where localisation='Paris';

5) select avnum, avnom, capacite, localisation from avion a, pilote p where a.localisation=p.ville and plnom='Tanguy';

6) select volnum, villedep, villearr, heuredep, heurearr, avnom, plnom from avion a, pilote p, vol v where p.plnum=v.plnum and a.avnum=v.avnum and volnum=714;

7) select plnom, plprenom, avnum from pilote p, vol v where v.plnum=p.plnum;

8) select distinct a.avnum, avnom from avion a, vol v where a.avnum=v.avnum;

9) select count(*) from vol;

10) select avnom, sum(capacite) from avion group by avnom;

11) select avg(heurearr-heuredep) from vol;