Université Henri Poincaré nancy 1
FST, Département Informatique

TP 1
SQL*Plus et SQL (Requêtes d'interrogation)

Il faudra garder une trace du travail fait en TP : pensez à sauvegarder dans des fichiers vos requêtes ainsi que les résultats. **Créer un répertoire par séance de TP** dans lequel sera sauvegardé le travail effectué dans chaque séance.

Préambule : Lancement du client SQL*Plus

1. ~~Sous Linux, ouvrir un terminal et se connecter par **ssh** à la machine **panoramix.depinfo.uhp-nancy.fr** (même login et mot de passe que sous **idefix**). Idem sous Windows, mais en utilisant **putty**.~~

2. ~~Lancer le client SQL*plus pour vous connecter au serveur Oracle par la commande **sqlplus**~~

   Votre compte sous Oracle :
   Login-Oracle : login qui vous est affecté
   Mot de passe Oracle : celui associé au login qui vous est affecté

3. Une fois connecté au serveur Oracle, vous pouvez modifier votre mot de passe Oracle par la commande : **ALTER USER** login-Oracle **IDENTIFIED BY** nouveau_mot_de_passe

A. Initiation à sql*plus

- Une fois connecté, positionner la variable _EDITOR à votre éditeur de texte préféré. Par exemple : DEFINE _EDITOR = emacs
- Editer dans un fichier nommé ESSAI.SQL la requête permettant de trouver la liste des numéros d'employés de la table EMPLOYEES appartenant à HR : EDIT essai.sql
- Après sauvegarde (CTRL-X CTRL-S, sous emacs) et fermeture de l'éditeur (CTRL-X CTRL-C, sous emacs), exécuter le contenu du fichier essai : **START** essai.sql
- Si votre requête est correcte, vous voyez s'afficher la liste des résultats. Sinon, ré-éditer le fichier essai et modifier la requête puis ré-exécuter.
- Rediriger les sorties dans le fichier **result.res** (les résultats sont aussi affichés à l'écran) : **SPOOL** result.res
- Exécuter le contenu du fichier essai : **START** essai.sql
- Retourner à l'affichage sur écran : **SPOOL OFF**

- Vérifier le contenu du fichier **result.res** (au niveau système d'exploitation)
- Taper directement une requête sous **sqlplus** puis l'exécuter. Par exemple

  SELECT LAST_NAME FROM HR.EMPLOYEES;
- Sauvegarder le contenu du tampon (dernière requête entrée) dans le fichier ESSAI2.SQL : **SAVE** ESSAI2.SQL **CREATE**
- Vérifier le contenu du fichier ESSAI2.SQL (au niveau système d'exploitation)
- Taper directement une seconde requête sous **sqlplus** puis l'exécuter. Par exemple

  SELECT FIST_NAME FROM HR.EMPLOYEES;
- Ajouter le contenu du tampon (dernière requête entrée) dans le fichier ESSAI2.SQL : **SAVE** ESSAI2.SQL **APPEND**
- Vérifier le contenu du fichier ESSAI2.SQL (au niveau système d'exploitation)

B. Requêtes d'interrogation des tables de HR

Ecrire puis exécuter les requêtes permettant de répondre aux questions suivantes sur les tables de HR. Pour cela, créer un fichier pour chaque requête nommé B1, B2, … et un fichier pour chaque résultat nommé RB1, RB2, ….). Le nombre de réponses attendu est donnée entre parenthèses.

Le schéma des tables HR a été donné en TD. Ne pas oublier pas qu'elles appartiennent à l'utilisateur HR (préfixer le nom des tables par **HR.**)

1- Afficher les noms des employés dont le salaire est supérieur à 10 000 $. (15)

2- Afficher les noms des employés dont la date d'embauche est comprise entre 17/02/97 et 30/10/1997. (21)

3- Afficher les noms des employés commençant par la lettre 'J'. (2)

4- Afficher les noms des employés dont le nom contient deux fois la lettre 'a'. (10)

5- Afficher les noms des employés dont le numéro du chef est 114. (5)

6- Afficher les noms des services dont le numéro du chef est 114 ou qui n'ont pas de chef. (1)

7- Afficher les noms des services qui ne sont pas localisés à 'Seattle'. (6)

8- Afficher les noms des employés et leur commission. Afficher 'pas de commission' dans la deuxième colonne lorsque c'est le cas. Ordonner la liste par rapport aux noms. (107)

Deux fonctions sont utilisables :

- La fonction **NVL (exp1, exp2)**
  - **a.** retourne **exp1** si **exp1** ne vaut pas null
  - **b.** retourne **exp2** si **exp1** vaut null

Il faut que les types de **exp1** et **exp2** soient compatibles.

- La fonction **TO_CHAR (exp [,format])** convertit l'expression numérique **exp** (dont on peut préciser le format) en chaîne de caractères.

9- Afficher les noms des employés, par ordre alphabétique avec la première lettre seulement en majuscules, qui ont un salaire supérieur au salaire moyen. (51)

10- Afficher les noms des employés qui ont au moins une personne sous leurs ordres. (18)

11- Afficher les noms des services dans lesquels il n'y a aucun employé. (16)

12- Donner le nombre d'employés pour chaque poste. (19)

13- Afficher pour tous les employés qui ont été embauchés avant 'Weiss', leur nom, l'adresse et la ville de leur service. (24)

14- Pour chaque poste, donner le nombre d'employés dont le salaire se trouve entre le minimum et le maximum des salaires prévus pour ce poste. (19)

15- Pour chaque poste, donner la somme des salaires lorsque cette somme est inférieure à 10000$. (5)

C. Consultation du dictionnaire

1- Retrouver le schéma de la table DICTIONARY (commande **describe**)

2- Quel est le nombre de tables et de vues du dictionnaire de données d'Oracle

3- Retrouver les noms des tables (et vues) du dictionnaire de données d'Oracle (à éviter, vu le résultat de la question précédente)

4- Trouver les noms des tables (et vues) qui traitent des indexes

5- Retrouver le schéma de la table ALL_TABLES et de ALL_TAB_COLUMNS.

6- Retrouver, de deux façons différentes, la liste des noms d'attributs de la table JOBS de l'utilisateur HR.

Schémas des tables de HR

Dans la version installée du serveur de données Oracle, il existe un utilisateur nommé HR (Human Resources) propriétaire d'un ensemble de tables accessibles à tout le monde dont les schémas sont les suivants :

EMPLOYEES (**EMPLOYEE_ID**, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT,
MANAGER_ID, *DEPARTMENT_ID*)

COUNTRIES (**COUNTRY_ID**,COUNTRY_NAME,*REGION_ID*)

DEPARTMENTS
(DEPARTMENT_ID,DEPARTMENT_NAME,*MANAGER_ID*,LOCATION_ID)

JOBS **(JOB_ID**,JOB_TITLE,MIN_SALARY,MAX_SALARY)

JOB_HISTORY
(**EMPLOYEE_ID**,**START_DATE**,END_DATE,*JOB_ID*,*DEPARTMENT_ID*)

LOCATIONS
(**LOCATION_ID**,STREET_ADDRESS,POSTAL_CODE,CITY,STATE_PROVINCE,
COUNTRY_ID)

REGIONS (**REGION_ID**, REGION_NAME)

Les schémas plus complets sont donnés ci-dessous. Ils sont obtenus à l'aide de la commande DESCRIBE suivi du nom d'une table (ou d'une vue) :

Table EMPLOYEES

Nom	NULL ?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

-- Pays: **Table COUNTRIES**

Nom	NULL ?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

-- Départements de l'entreprise : **Table DEPARTMENTS**

Nom	NULL ?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

-- Descriptif des postes: **Table JOBS**

Nom	NULL ?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

-- Historique des postes occupés par des employés : **Table JOB_HISTORY**

Nom	NULL ?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

-- Adresses: **Table LOCATIONS**

Nom	NULL ?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

-- Noms (en clair) des régions : **Table REGIONS**

Nom	NULL ?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

Corrigé (éléments)

A. Consultation du dictionnaire

1- Retrouver le schéma de la table DICTIONARY

```
DESCRIBE  DICTIONARY ;
Ou DESC DICT ;
```

2- Quel est le nombre de tables et de vues du dictionnaire de données d'Oracle

SELECT COUNT(*) FROM DICT

3- Retrouver les noms des tables (et vues) du dictionnaire de données d'Oracle (**à éviter**)

```
SELECT     TABLE_NAME
FROM       DICTIONARY;
```

4- Retrouver le schéma de la table ALL_TABLES et ALL_TAB_COLUMNS du dictionnaire.

```
DESCRIBE    ALL_TABLES ;
DESC        ALL_TAB_COLUMNS;
```

5- Retrouver , de deux façons différentes, la liste des noms d'attributs de la table JOBS de l'utilisateur HR.

* Méthode 1 (utilisation d'une table du dico):

```
SELECT     COLUMN_NAME
FROM       ALL_TAB_COLUMNS
WHERE      OWNER='HR'   AND    TABLE_NAME='JOBS';
```

* Méthode 2 (utiliser la fonction DESCRIBE offerte par le client)

```
DESC  HR.JOBS ;
```

B. Requêtes d'interrogation des tables de HR

a)
```
SELECT LAST_NAME
FROM HR.EMPLOYEES
WHERE SALARY > 10000 ;
```

```
LAST_NAME
-------------------------
King
Kochhar
De Haan
 Raphaely
 Higgins
```

15 ligne(s) sélectionnée(s).

b) SELECT LAST_NAME
FROM HR.EMPLOYEES
WHERE HIRE_DATE >= '17-FEV-1997' AND HIRE_DATE <= '30-OCT-1997' ;

LAST_NAME

Austin
Chen
Sciarra
Tobias
Bull
Chung
Dilly
Everett
Fay

21 ligne(s) sélectionnée(s).

c) SELECT LAST_NAME
FROM HR.EMPLOYEES
WHERE LAST_NAME LIKE 'J%';

LAST_NAME

Johnson
Jones

d) SELECT LAST_NAME
FROM HR.EMPLOYEES
WHERE LAST_NAME LIKE '%A%A%' ;

LAST_NAME

De Haan
Pataballa
Banda
Sarchand
10 ligne(s) sélectionnée(s).

e) SELECT LAST_NAME
FROM HR.EMPLOYEES
WHERE MANAGER_ID = 114 ;

LAST_NAME

Khoo
Baida
Tobias
Himuro
Colmenares

f) SELECT DEPARTMENT_NAME

```
FROM HR.DEPARTMENTS
where manager_id = 114 or manager_id IS null;
```

```
DEPARTMENT_NAME
------------------------------
Purchasing
```

g)
```
SELECT DEPARTMENT_NAME
FROM HR.DEPARTMENTS
MINUS
SELECT DEPARTMENT_NAME
FROM HR.DEPARTMENTS D, HR.LOCATIONS L
WHERE L.LOCATION_ID = D.LOCATION_ID
AND L.CITY = 'Seattle';
```

```
DEPARTMENT_NAME
------------------------------
Human Resources
IT
Marketing
Public Relations
Sales
Shipping
6 ligne(s) sélectionnée(s).
```

h)
```
SELECT LAST_NAME,
   NVL(TO_CHAR(COMMISSION_PCT, '90.99'), 'PAS DE COMMISSION')
 FROM HR.EMPLOYEES ;
```

```
LAST_NAME                 COMMISSION
------------------------- ----------------
 Grant                     0.15
Johnson                    0.10
Taylor                    pas de commission
Fleaur                    pas de commission
Sullivan                   pas de commission
Geoni                     pas de commission
Sarchand                   pas de commission
Bull                      pas de commission
Dellinger                  pas de commission
Cabrio                    pas de commission
....
107 ligne(s) sélectionnée(s).
```

j)
```
SELECT INITCAP(LAST_NAME) AS NOM
FROM HR.EMPLOYEES
WHERE SALARY > (SELECT AVG(SALARY) FROM HR.EMPLOYEES)
ORDER BY LAST_NAME;
```

```
NOM
-------------------------
Faviet
```

Fox
Fripp
Gietz
Marvins
Mavris
Mcewen
Olsen
...
51 ligne(s) sélectionnée(s).

k)
```
SELECT    DISTINCT E2.LAST_NAME
FROM      HR.EMPLOYEES E1, HR.EMPLOYEES E2
WHERE     E1.MANAGER_ID = E2.EMPLOYEE_ID;
```

LAST_NAME

Cambrault
De Haan
Errazuriz
Fripp
Greenberg
Hartstein
Higgins
Russell
Vollman
Weiss
Zlotkey
…
18 ligne(s) sélectionnée(s).

l)
```
SELECT DEPARTMENT_NAME
FROM HR.DEPARTMENTS D
WHERE NOT EXISTS(SELECT * FROM HR.EMPLOYEES E
      WHERE D.DEPARTMENT_ID = E.DEPARTMENT_ID);
```

DEPARTMENT_NAME

Treasury
Corporate Tax
Recruiting
Payroll
…
16 ligne(s) sélectionnée(s).

m)
```
SELECT  COUNT(DISTINCT EMPLOYEE_ID), JOB_TITLE
FROM HR.EMPLOYEES E, HR.JOBS J
WHERE J.JOB_ID = E.JOB_ID
GROUP BY JOB_TITLE  ;
```

COUNT(EMPLOYEE_ID) JOB_TITLE
---
5 Accountant
1 Accounting Manager
1 Administration Assistant

2 Administration Vice President
1 Finance Manager
1 Human Resources Representative
1 Marketing Manager
1 Marketing Representative
1 President
…
19 ligne(s) sélectionnée(s).

n)
```
SELECT E.LAST_NAME, L.STREET_ADDRESS, L.CITY
FROM HR.EMPLOYEES E, HR.LOCATIONS L, HR.DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID AND D.LOCATION_ID =
L.LOCATION_ID AND E.HIRE_DATE < (SELECT HIRE_DATE FROM HR.EMPLOYEES
E WHERE E.LAST_NAME = 'WEISS') ;
```

```
LAST_NAME            STREET_ADDRESS
-------------------- ----------------------------------------
CITY
------------------------------
Rajs                 2011 Interiors Blvd
South San Francisco

King                 Magdalen Centre, The Oxford Science Park
Oxford

Sully                Magdalen Centre, The Oxford Science Park
Oxford

Sarchand             2011 Interiors Blvd
South San Francisco

Bell                 2011 Interiors Blvd
South San Francisco

...

24 ligne(s) sélectionnée(s).
```

o)
```
SELECT JOB_TITLE, COUNT(E.EMPLOYEE_ID)
FROM HR.EMPLOYEES E,  HR.JOBS J
WHERE E.JOB_ID = J.JOB_ID AND E.SALARY < J.MAX_SALARY AND E.SALARY
> J.MIN_SALARY  GROUP BY JOB_TITLE;
```

```
JOB_TITLE                          COUNT(E.EMPLOYEE_ID)
---------------------------------- --------------------
Accountant                                            4
Accounting Manager                                    1
Administration Assistant                              1
Administration Vice President                         2
Finance Manager                                       1
Human Resources Representative                        1
Marketing Manager                                     1
Marketing Representative                              1
```

President 1
Programmer 5
…

19 ligne(s) sélectionnée(s).

p)

```
SELECT J.JOB_TITLE, SUM(E.SALARY)
FROM HR.EMPLOYEES E, HR.JOBS J
WHERE E.JOB_ID = J.JOB_ID
GROUP BY J.JOB_TITLE HAVING SUM(E.SALARY) <=10000;
```

JOB_TITLE	SUM(E.SALARY)
Administration Assistant	4400
Human Resources Representative	6500
Marketing Representative	6000
Public Accountant	8300
Public Relations Representative	10000

Université Henri Poincaré
UFR STMIA – Dept Informatique

TP 2
SQL LDD et Contraintes

/* On souhaite créer les schémas des relations suivantes :

1- Créer les schémas de ces relations. */

/* Ces schémas sont complétés par les contraintes d'intégrité suivantes :

- les attributs soulignés sont les clés primaires des relations.
- les attributs en italique sont les clés étrangères
- le triplet (**nopers, noposte,noservice**) a une valeur **unique** pour chaque tuple d'AFFECTATION ; **noservice** peut être null contrairement aux deux premiers attributs
- l'attribut **nochef** dans SERVICE contient un numéro de personne (attribut **nopers** de PERSONNE)
- les valeurs des attributs suivants ne sont pas obligatoires : **prenom**, **salaire**, **nochef**, **rue**, **salairemin**, **salairemax**, **finservice** (et AFFECTATION.noservice)
- la **ville** par défaut dans la table LOCALISATION est 'Paris'
- pour une affectation donnée, la date de fin de service (**finservice**) est postérieure à celle de début de service (**debservice**)

*/

```
create table PERSONNE (
        nopers number(6) primary key,
        prenom varchar2(20),
        nom2fam varchar2(25)   not null   )

create table SERVICE (
        noservice number(4) primary key,
        nomservice varchar2(30),
        noloc number(6) references LOCALISATION(noloc),
        nochef  number(6)  references PERSONNE(nopers)      )

create table LOCALISATION (
        noloc number(6) primary key,
        rue varchar2(40) not null,
        ville varchar2(30)   default 'Paris' )

create table POSTE (noposte varchar2(10) primary key, nomposte varchar2(35))

create table SALAIRE (noposte varchar2(10), noservice number(4,0),
        salairemin number(6), salairemax number(6)
        constraint CleSalaire primary key(noposte, noservice))
```

create table AFFECTATION (
nopers number(6) not null primary key,
noposte varchar2(10) not null,
noservice number(4),
debservice date,
finservice date,
salaire number(8) ,
Constraint unicite (nopers, noposte, noservice) unique,
Constraint SurDate check (debservice < finservice))

2- Retrouver quelques informations sur le schéma de la table PERSONNE

(a) En interrogeant la tables USER_ TAB_COLUMNS du dictionnaire, afficher le nom de chaque colonne (ou attribut), son type, la possibilté de valeur NULL (cf. attribut NULLABLE de la table USER_ TAB_COLUMNS).

(b) (b) Retrouver les mêmes informations en utilisant la fonction DESCRIBE.

3- Insérer dans les tables créées des données provenant de certaines tables HR

(a) Insérer dans la table PERSONNE les informations utiles concernant tous les employés de la table EMPLOYEES de l'utilisateur HR (107 tuples). *N.B.* : first_name correspond au prénom.

INSERT INTO PERSONNE (select v« LES BONNES COLONNES DE LA RELATION HR.EMPLOYEES » FROM HR.EMPLOYEES)

Idem pour les questions ci-dessous

(b) Peupler les tables SERVICE, LOCALISATION et POSTE à partir des tuples des tables HR.DEPARTMENTS, HR.LOCATIONS et HR.JOBS.

(c) Initialiser la table SALAIRE à partir du produit cartésien des tables HR.JOBS et SERVICE. On supposera , dans un premier temps, que les salaires minimum et maximum ne dépendent que du numéro du poste (et pas du service) .

(d) Peupler la table AFFECTATION à partir de la table HR.EMPLOYEES en initialisant à NULL la date de fin de service.

4- Mettre à jour ponctuellement les tables

(a) Augmenter de 2% le salaire minimum pour tous les postes du service 'IT Support'.

UPDATE SALAIRE set salairemin = 1.02*salairemin
FROM SALAIRE s1, service s2, poste p
Where upper(p.nomservice) like upper('IT Support')
and p.noservice = s1.noservice and s1.noposte = p.noposte

(b) Supprimer de la table SALAIRE les tuples concernant les postes dont le nom commence par 'Sales' dans les services dont le nom ne comporte pas 'sales'.

Similaire à la question précédente avec DELETE FROM ... WHERE ...
Avec upper(nomposte) like upper('%sales%)
And upper(nomservice) not like upper(%sales%')

(c) Supprimer le poste intitulé ‘Purchasing Manager’ de la table POSTE. Que se passe-t-il ? Renoncer à la mise à jour.

5- Modifier le schéma de certaines tables

(a) Ajouter un attribut **obsolète** à la table POSTE qui aura deux valeurs possibles **V** ou **F** selon qu’un poste est obsolète ou non. La valeur par défaut sera **F**.

ALTER TABLE POSTE ADD(obsolete char(1) check obsolete in(‘V’, ‘F’) DEFAULT ‘F’

(b) Reconsidérer la question 4.c- après cette modification du schéma de POSTE : modifier les tables AFFECTATION, SALAIRE et POSTE de façon à pourvoir tenir compte de la suppression de cette fonction dans l’entreprise. Il faut garder les postes devenus obsolètes, mettre un terme aux affectations les concernant et supprimer les salaires relatifs aux postes obsolètes. Vérifier.

(c) Ajouter un attribut COURRIEL à la table PERSONNE dont ‘confidentiel’ sera la valeur par défaut. Vérifier.

Voir (a)

(d) Renseigner l’attribut COURRIEL à partir de l’attribut EMAIL de la table HR.EMPLOYEES en complétant chaque valeur de email par la chaîne ‘@technissimo.fr’. Vérifier.

UPDATE PERSONNE set COURRIEL = HR.EMPLOYEES.EMAIL || ‘@technissiomo.fr’

6- Supprimer le schéma de toutes les tables créées.

Université de Lorraine
Campus Sciences, FST, Département informatique

TP PL/SQL

Sommaire :

I. Initiation à PL/SQL

1. Ecrire un bloc PL/SQL qui affiche
   'Bonjour, je suis' ***Votre Login Oracle***
   'Nous sommes le : ' ***Date du jour*** (sous la forme **N°-du-jour Nom-du-mois Année**)

```
set serveroutput on;
begin
  dbms_output.put_line('Bonjour');
  dbms_output.put_line('Je suis : '|| user);
  dbms_output.put_line('Nous sommes le : ' || to_char(sysdate,'dd month yyyy'));
end;
```

2. Tableaux et itérations : Créer un tableau et
   - le remplir par les valeurs 2, 4, 6, …, 20
   - imprimer son contenu dans l'ordre inverse (20, 18, …, 2) sous la forme T[i] = valeur.

```
set serveroutput on;
declare
  type type2tableau is
       table of int not null
       index by binary_integer;
  T type2tableau;
begin
  for i in 0..10
  loop
     T(i) := 2*i;
  end loop;
  for j in reverse 0..10
  loop
    dbms_output.put_line('T['||j||'] = ' || T(j));
  end loop;
end;
```

3. Une table *produit* et une table *stock* existent ; elles sont la propriété de *nacerb* et tout utilisateur a le droit de les interroger. Leur schéma est le suivant :

   SQL> desc produit

Name	Null?	Type
PROD#	NOT NULL	NUMBER(38)
LIBELLE		VARCHAR2(20)
PU	NOT NULL	FLOAT(126)

SQL> desc stock

Name	Null?	Type
PROD#	NOT NULL	NUMBER(38)
DEP#	NOT NULL	NUMBER(38)
QTE	NOT NULL	NUMBER(38)

3.1. Créer les mêmes tables avec leur contenu.

SQL> create table produit as (select * from nacerb.produit) ;

SQL> create table stock as (select * from nacerb.stock);

3.2. Ajouter une colonne TotalEnStock à la relation produit :

```
SQL> alter table produit add (Total_Stock int);
```

3.3. Instancier cette colonne par la quantité totale en stock pour chaque produit

```
SQL> update produit set
  2  Total_Stock = (select sum(qte) from stock s
  3  where s.prod# = produit.prod#
  4  group by s.prod#);

2000 ligne(s) mise(s) à jour.
```

3.3. Programmer et mettre au point un bloc anonyme qui permet d'obtenir les 3 produits ayant les plus grandes quantités en stock.

```
-- Bloc affichant les 3 produits les plus vendus
-- NE PAS OUBLIER: set serveroutput on;
declare
  lib    produit.libelle%type;
--  prix   produit.pu%type;
  ventes produit.Total_Des_Ventes%type;
  cursor C is
   select libelle, Total_Des_Ventes
   from produit order by Total_Des_Ventes desc;
begin
  dbms_output.put_line('--- Begin');
  open C;
  if C%ISOPEN
  then
    dbms_output.put_line('--- ISOPEN = Vrai');
   for i in 1..3
    loop
      dbms_output.put_line('--- Avant fetch');
      fetch C into lib, ventes;
      if C%found
      then
         dbms_output.put_line('Produit: '|| lib);
         dbms_output.put_line('vendu en : '|| ventes || ' exemplaires.');
```

```
      else
         dbms_output.put_line('Relation vide; Nbr de tuples = '||i);
      end if;
     end loop;
   else
     dbms_output.put_line('Erreur sur open curseur');
   end if;
   close C;
end;
```

3.4. Transformer le bloc de la question précédente en une procédure *Top_stock* qui permet d'obtenir les n produits *les plus présents en stock*, n étant un paramètre de la procédure.

```
create or replace procedure i_Produit_Min (i IN int)
/*-------------------------------------------------*/
/*  Sortir les i produits ayant les plus hauts niveaux  en stock */
/*  Exemple d'invocation: exec i_Produit_Min(3);      */
/*-------------------------------------------------*/
as
NomProd produit.libelle%TYPE;
Qte stock.qte%TYPE;
j number;
cursor MonCurseur is
  select s.prod#, sum(qte) "Total"
  from produit p, stock s
  where p.prod# = s.prod#
  group by s.prod#
  order by sum(qte) desc;
begin
  open MonCurseur;
  j := i;
  fetch MonCurseur into NomProd, Qte;
  while (j > 0) and MonCurseur%FOUND
  loop
    DBMS_OUTPUT.PUT_LINE('Somme des stocks du produit ' || NomProd || ': '
                  || TO_CHAR(Qte, '99999999'));
    fetch MonCurseur into NomProd, Qte;
    j := j - 1;
  end loop;
  close MonCurseur;
end;
```

II. Comprendre le mode de passage de paramètres

1. Ecrire et mettre au point une procédure P ayant 3 paramètres x1, x2, x3 passés respectivement en mode IN, INOUT et OUT. Cette procédure réalise les actions suivantes :
   - Affiche les valeurs des paramètres sous la forme :
     Valeurs des paramètres en entrée de P :
     - **x1 (in) = Valeur de x1**
     - **x2 (in out) = Valeur de x2**

- **x3 (out) = Valeur de x3**

- Incrémente x1 de 1, x2 de 10 et affecte 300 à x3
- Affiche les valeurs des paramètres sous la forme :
  Valeurs des paramètres en sortie de P :
  - **x1 (in) = Valeur de x1**
  - **x2 (in out) = Valeur de x2**
  - **x3 (out) = Valeur de x3**

```
create or replace procedure P(x1 in int, x2 in out int, x3 out int)
/*-- Test des modes de passage des parametres   --*/
as
begin
dbms_output.put_line('x1 (in) = ' || x1 ||' (valeur au debut de P)');
dbms_output.put_line('x2 (in out) = ' || x2 ||' (valeur au debut de P)');
dbms_output.put_line('x3 (out) = ' || x3 ||' (valeur au debut de P)');
-- x1 := x1 + 1;
-- PLS-00363: expression 'X1' ne peut être utilisée comme cible
-- d'affectation
x2 := x2 + 10;
x3 := 200;
dbms_output.put_line('x1 (in) = ' || x1 ||' (valeur a la fin de P)');
dbms_output.put_line('x2 (in out) = ' || x2 ||' (valeur a la fin de P)');
dbms_output.put_line('x3 (out) = ' || x3 ||' (valeur a la fin de P)');
end;
```

2. Ecrire un bloc PL/SQL qui appelle la procédure P avec des paramètres Y1, Y2, Y3 et qui affiche les valeurs de Y1, Y2, Y3 avant et après l'appel à P sous la même forme que P. Commenter et démontrer les résultats.

```
declare
  Y1 int; Y2 int; Y3 int;
begin
  Y1 := 1; Y2 := 20; Y3 := 0;
  dbms_output.put_line('Y1 (in) = ' || Y1 ||' (valeur avant appel de P)');
  dbms_output.put_line('Y2 (in out) = ' || Y2 ||' (valeur avant appel de P)');
  dbms_output.put_line('Y3 (out) = ' || Y3 ||' (valeur avant appel de P)');
  dbms_output.put_line('-- Le bloc appelle la procedure P :');
  P(Y1, Y2, Y3);
  dbms_output.put_line('-- retour dans le bloc apres appel de la procedure P :');
  dbms_output.put_line('Y1 (in) = ' || Y1 ||' (valeur au retour de P)');
  dbms_output.put_line('Y2 (in out) = ' || Y2 ||' (valeur au retour de P)');
  dbms_output.put_line('Y3 (out) = ' || Y3 ||' (valeur au retour de P)');
end;
```

Exécution :

```
--  Trace de l'exécution (Ne pas oublier set serveroutput on)

Y1 (in) = 1 (valeur avant appel de P)
Y2 (in out) = 20 (valeur avant appel de P)
Y3 (out) = 0 (valeur avant appel de P)
-- Le bloc appelle la procedure P :
x1 (in) = 1 (valeur au debut de P)
x2 (in out) = 20 (valeur au debut de P)
x3 (out) =  (valeur au debut de P)
x1 (in) = 1 (valeur a la fin de P)
x2 (in out) = 30 (valeur a la fin de P)
```

```
x3 (out) = 200 (valeur a la fin de P)
-- retour dans le bloc apres appel de la procedure P :
Y1 (in) = 1 (valeur au retour de P)
Y2 (in out) = 30 (valeur au retour de P)
Y3 (out) = 200 (valeur au retour de P)

Procédure PL/SQL terminée avec succès.
```