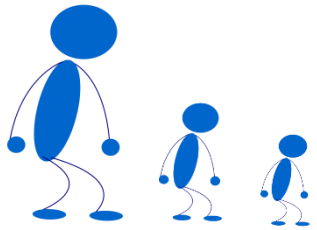
LANGAGE PL/SQL

PROCEDURAL LANGUAGE/SQL



AVANT-PROPOS

Création des tables utilisées

Au cours des différents exercices, les tables suivantes sont utilisées : aeroport, vol, avion, voyage et resultat.
Pour créer ces tables et insérer des enregistrements, exécutez le script nommé *LabPLSQL_00 - CreationTable*.
⊶ Ressource : Phase 1\ LabPLSQL_00 - CreationTable.sql

Structure des tables utilisées

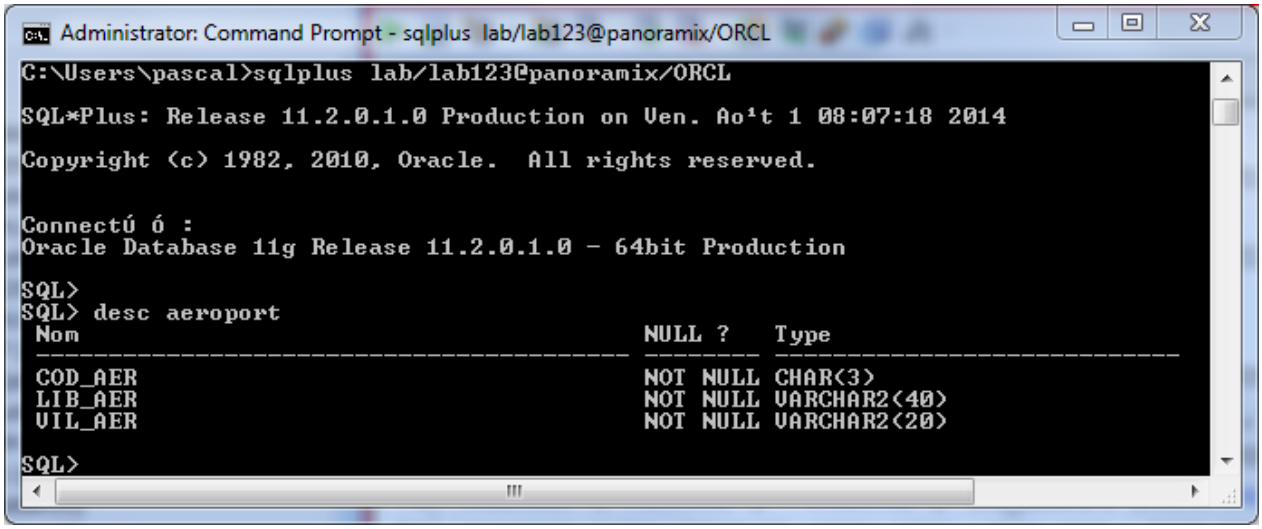
Pour visualiser la structure des tables, vous pouvez utiliser, sous SQL Developer ou SQL*Plus, la commande desc.

Connexion au serveur panoramix avec SQL*Plus

```
C:\> sqlplus lab/lab123@panoramix/ORCL
```

```
SQL> desc aeroport
```

Nom	Null ?	Type
COD_AER	NOT NULL	CHAR(3)
LIB_AER	NOT NULL	CHAR(20)
VIL_AER	NOT NULL	CHAR(20)



SQL> desc vol

Nom	Null ?	Type
NUM_VOL	NOT NULL	CHAR(6)
COD_AER_DEP	NOT NULL	CHAR(3)
COD_AER_ARR	NOT NULL	CHAR(3)

SQL> desc avion

Nom	Null ?	Type
NUM_AVI	NOT NULL	CHAR(5)
TYP_AVI	NOT NULL	CHAR(10)
CAP_AVI	NOT NULL	NUMBER(3,0)

SQL> desc voyage

Nom	Null ?	Type
NUM_VOY	NOT NULL	CHAR(6)
NUM_VOL	NOT NULL	CHAR(6)
DAT_VOY	NOT NULL	DATE
NUM_AVI	NOT NULL	CHAR(5)
NOM_COM	NOT NULL	CHAR(15)
PLA_RES	NOT NULL	NUMBER(3,0)

SQL> desc resultat

Nom	Null ?	Type
NO		NUMBER(2)
LIBELLE		VARCHAR2(60)
VALEUR		NUMBER(9,2)

Note : La colonne NO est utilisée pour insérer le numéro de l'exercice.

Contenu des tables utilisées

SQL> select * from aeroport ;

COD_AER	LIB_AER	VIL_AER
CDG	Aéroport Charles de Gaulle	Roissy-en-France
ABL	Aéroport de Blagnac	Blagnac
AMG	Aéroport de Mérignac	Mérignac

SQL> select * from vol;

NUM_VOL	COD_AER_DEP	COD_AER_ARR
VOL001	CDG	ABL
VOL002	ABL	CDG
VOL003	AMG	CDG

SQL> select * from avion;

NUM_AVI	TYP_AVI	CAP_AVI
AV001	CONCORDE	250
AV002	AIRBUS	250
AV003	BOEING	350

SQL> select * from voyage;

NUM_VOY	NUM_VOL	DAT_VOY	NUM_AVI	NOM_COM	PLA_RES
VOY001	VOL001	01-AUG-14	AV001	CAPITAINE1	100
VOY002	VOL002	12-APR-10	AV002	CAPITAINE2	200
VOY003	VOL003	12-JUN-10	AV002	CAPITAINE3	250
VOY004	VOL003	12-AUG-10	AV003	CAPITAINE4	250

EXERCICES

Exercice 1 : multiplication

Exercice 1.1

- Codez un bloc anonyme PL/SQL qui permet de calculer et d'afficher une table de multiplication. La table de multiplication à utiliser est demandée à l'utilisateur suite à l'exécution du bloc anonyme. Utilisez la table RESULTAT pour stocker les valeurs.

  Note : pour la saisie d'une valeur, utilisez la commande ACCEPT de SQL*Plus ou les variables de substitution de la forme &<nom>.

Exemple :

Quelle table de multiplication ? : 3

Résultat attendu pour la table de multiplication de 3.

LIBELLE	VALEUR
1*3	3
2*3	6
3*3	9
4*3	12
5*3	15
6*3	18
7*3	21
8*3	24
9*3	27
10*3	30

Exercice 1.2

- Codez un bloc anonyme PL/SQL qui permet d'effectuer une multiplication entre un multiplicande et un multiplicateur et d'afficher le résultat obtenu. Exécutez le bloc anonyme.

Exercice 1.3

- Transformez le bloc anonyme PL/SQL en une procédure sans paramètre. Exécutez la procédure à l'aide d'un bloc anonyme.

Exercice 1.4

- Transformez la procédure sans paramètre en une procédure avec 2 paramètres en entrée (multiplicande et multiplicateur). Exécutez la procédure à l'aide d'un bloc anonyme.

Exercice 1.5

- Transformez la procédure avec 2 paramètres en entrée (multiplicande et multiplicateur) en une procédure avec 2 paramètres en entrée (multiplicande et multiplicateur) et 1 paramètre en sortie (résultat de la multiplication). Exécutez la procédure à l'aide d'un bloc anonyme.

Exercice 1.6

- Transformez la procédure précédente en une fonction avec 2 paramètres en entrée (multiplicande et multiplicateur) et une valeur de retour (résultat de la multiplication). Exécutez la fonction à l'aide d'un bloc anonyme.

Exercice 2

- Codez un bloc anonyme PL/SQL qui permet de saisir en entrée le numéro d'un voyage et mettre à jour le nombre de places réservées suivant les conditions suivantes :

⌦ Si le type d'avion est 'CONCORDE', augmentez le nombre de places de 50 et insérez informations suivantes dans la table RESULTAT :

NO	*no de LAB*
LIBELLE	*numéro du voyage* 'a été augmenté de 50 places'
VALEUR	*nouveau nombre de places réservées*

⌦ Si le type d'avion est 'AIRBUS', augmentez le nombre de places de 100 et insérez les informations suivantes dans la table RESULTAT :

NO	*no de LAB*
LIBELLE	*numéro du voyage* 'a été augmenté de 100 places'
VALEUR	*nouveau nombre de places réservées*

⌦ Dans les autres cas, insérez les informations suivantes dans la table RESULTAT :

NO	*no de LAB*
LIBELLE	*numero du voyage* : avion de type *type d'avion*
VALEUR	*nombre de places réservées*

Exercice 3

- Codez un bloc anonyme PL/SQL qui permet :
  - d'extraire les lignes N et N+1 d'une table voyage triée dans l'ordre chronologique. Le numéro (N) est entré par l'utilisateur suite à l'exécution du bloc anonyme.
  - de mémoriser dans la table RESULTAT les 2 lignes extraites. Les informations mémorisées dans la table RESULTAT sont :
    - le champ NO qui contient le numéro de l'exercice,
    - le champ LIBELLE qui contient la concaténation du numéro du voyage et la date du voyage.

Exemple :

Soit la table suivante triée par ordre chronologique :

VOY002	VOL002	12/04/10	AV002	CAPITAINE2	300
VOY003	VOL003	12/06/10	AV002	CAPITAINE3	250
VOY004	VOL003	12/08/10	AV003	CAPITAINE4	250
VOY001	VOL001	18/12/14	AV001	CAPITAINE1	250

Résultat attendu avec la valeur N = 2 saisie par l'utilisateur.

NO	LIBELLE
3	VOY003 12/06/10
3	VOY004 12/08/10

Exercice 4

- Codez un bloc anonyme PL/SQL qui permet de stocker dans un tableau le contenu de la table AEROPORT.

Etape 1 :
Créer un type nommé par exemple type_aeroport qui est de type TABLE.
Spécifier que ce type (type_aeroport) est une table d'enregistrements similaires aux enregistrements de la table AEROPORT.
Spécifier que cette table est indicée par une variable numérique (INDEX BY BINARY_INTEGER).

```
TYPE type_aeroport IS TABLE OF AEROPORT%ROWTYPE INDEX BY BINARY_INTEGER;
```

Etape 2 :
Déclarer une variable qui est du type déclaré (type_aeroport).
tableau_aeroport type_aeroport;

Etape 3 :
Déclarer un curseur qui permet de sélectionner tous les enregistrements de la table AEROPORT.

Etape 4 :
Parcourir le curseur (avec par exemple un FOR) et mémoriser dans le tableau (tableau_aeroport) chaque ligne lue.

- Affichez ensuite le contenu des colonnes libellé et ville du tableau à l'aide du package DBMS_OUTPUT.

Etape 1 :
Parcourir le tableau avec une boucle WHILE et les commandes NEXT, FIRST et LAST.

Exercice 5

- Codez un bloc anonyme PL/SQL qui permet de stocker dans un tableau les numéros de voyage, de vol, le type d'avion et l'aéroport de départ.

- Affichez dans un second temps le contenu du tableau.

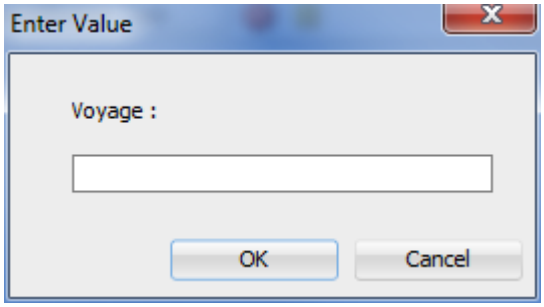
Exercice 6

- Codez un bloc anonyme PL/SQL qui permet de créer un nouveau voyage et gérer les erreurs suivantes :
  - Nombre de places réservées supérieur à la capacité.
  - Date de voyage antérieure à la date du jour.

Les messages d'erreur sont mémorisés dans le champ LIBELLE de la table RESULTAT.
Le champ NO de la table RESULTAT contient le numéro de l'exercice (6).

NO	LIBELLE
6	Date de voyage incorrecte
6	Problème de place

Suite à l'exécution de la procédure, les informations suivantes sont demandées :
Voyage :
Vol :
Date :
Avion :
Commandant :
Places :



- Modifiez votre code afin de prendre en compte l'intégrité référentielle et gérer les erreurs suivantes :
  - Problème de numéro de vol
  - Problème de numéro d'avion

- Testez les différents cas possibles :
  - Date de voyage incorrecte
  - Nombre de places incorrect par rapport à la capacité de l'avion
  - Numéro de vol n'existe pas
  - Numéro d'avion n'existe pas
  - Numéro de voyage existe déjà
  - Toutes les données sont correctes ⇨ insertion d'un voyage dans la table VOYAGE

Exercice 7

- Modifiez le fichier de l'exercice précédent pour gérer les erreurs suivantes sous la forme d'une procédure nommée valider_voyage().

La procédure prend 3 paramètres :
- 2 paramètres en entrée :
  - Numéro qui permet de savoir si la validation concerne le numéro de voyage, de vol ou d'avion.

- Numéro qui correspond à un numéro de voyage, de vol ou d'avion.
- 1 paramètre en sortie qui correspond au message.

Cette procédure peut retourner trois messages différents :

- Voyage existant
- Vol inconnu
- Avion inconnu

Les messages d'erreur sont mémorisés dans le champ LIBELLE de la table RESULTAT.
Le champ NO de la table RESULTAT contient le numéro de l'exercice (7).

```
NO    LIBELLE
----  ----------------------
7     Date de voyage incorrecte
7     Voyage existant
7     Vol inconnu
7     Avion inconnu
```

- Comme pour l'exercice précédent, testez les différents cas possibles.

Exercice 8

- Codez un bloc anonyme PL/SQL qui permet d'écrire le contenu de la table VOYAGE dans un fichier texte nommé voyages.txt.

- Au préalable, vous devez créer un objet Oracle de type Directory. Connectez-vous en tant qu'utilisateur sys et exécutez les commandes SQL ci-dessous :

```
-- Création de l'objet temp_dir
create or replace directory temp_dir as 'c:\tmp';
-- Accorder les droits de lecture et d'écriture à l'utilisateur lab
grant read, write on directory temp_dir to lab;
```

temp_dir = nom de l'objet Oracle créé. Cet objet sera utilisé dans votre script pour définir le répertoire de sauvegarde du fichier.
c:\tmp = répertoire sur lequel pointe l'objet et dans lequel sera sauvegardé le fichier voyages.txt.
lab = nom du compte Oracle.

Vous devez utiliser un curseur pour sélectionner l'ensemble des enregistrements de la table

VOYAGE.
Pour la gestion du fichier, vous devez utiliser le package UTL_FILE et ses fonctions FOPEN, PUT_LINE et FCLOSE.

Vous devez gérer les exceptions suivantes :

- UTL_FILE.INVALID_PATH
- UTL_FILE.INVALID_MODE
- UTL_FILE.INVALID_OPERATION

Ci-dessous, un exemple de fichier voyages.txt :

```
VOY001:VOL001:01-AUG-14:AV001:AV001:CAPITAINE1    :100
VOY002:VOL002:12-APR-10:AV002:AV002:CAPITAINE2    :200
VOY003:VOL003:12-JUN-10:AV002:AV002:CAPITAINE3    :250
VOY004:VOL003:12-AUG-10:AV003:AV003:CAPITAINE4    :250
VOY007:VOL001:12-DEC-14:AV001:AV001:BOB           :32
VOY012:VOL001:12-DEC-14:AV001:AV001:BOB           :100
```

- Créez une autre procédure qui permet d'afficher le contenu de ce fichier texte à l'écran.

Pour la gestion du fichier, vous devez utiliser le package UTL_FILE et ses fonctions FOPEN, GET_LINE et FCLOSE.
Vous devez gérer les exceptions suivantes :

- UTL_FILE.INVALID_PATH
- UTL_FILE.INVALID_MODE
- UTL_FILE.INVALID_OPERATION

Exercice 9

- Créez une procédure nommée afficher_table qui permet d'afficher à l'écran toutes les lignes d'une table pour au maximum deux colonnes : une colonne de 8 caractères maximum et une colonne de 6 caractères maximum.

Votre procédure doit avoir en entrée les paramètres suivants :

- nom des colonnes,
- nom de la table.

Vous devez utiliser un curseur pour sélectionner l'ensemble des enregistrements de la table.
Vous devez utiliser le package DBMS_SQL et ses fonctions OPEN_CURSOR, PARSE, FETCH_ROWS, CLOSE_CURSOR, EXECUTE, DEFINE_COLUMN et COLUMN_VALUE.

- Créez un bloc anonyme pour tester votre procédure afficher_table.

```
BEGIN
afficher_table('NUM_VOY, NOM_COM', 'VOYAGE');
END;
```

Exercice 10

Vous souhaitez afficher le classement des concurrents après une épreuve sportive de course à pied. Vous disposez des données suivantes :

- liste des concurrents,
- résultat (temps réalisé) obtenu par chaque concurrent.

Vous souhaitez mémoriser le rang (1er, 2ème, 3ème,…) et le nom de chaque concurrent, ainsi que l'écart de son temps avec le meilleur temps.

1. Créez une table CONCURRENT (NUMERO, NOM) qui contient la liste des noms des concurrents et une table RESULTAT (NUMERO_CONC, TEMPS) associant à chaque concurrent le temps qu'il a réalisé. Une valeur NULL pour le temps signifie un abandon du concurrent.

2. Peuplez les tables CONCURRENT et RESULTAT comme ci-dessous.

CONCURRENT		RESULTAT	
NUMERO	NOM	NUMERO_CONC	TEMPS
1	A	1	3.25
2	B	2	3.15
3	C	3	NULL
4	D	4	3.16
5	E	5	3.14
6	F	6	3.15
7	G	7	3.09
8	H	8	3.10
9	I	9	3.10
0	J	0	3.19

3. Créez une table CLASSEMENT (RANG, NOM, ECART) qui contiendra le classement des concurrents, du premier au dernier. Les concurrents ayant abandonné (valeur NULL dans le

champ TEMPS de la table RESULTAT) ne doivent pas y figurer.

4. Codez un bloc anonyme PL/SQL permettant de peupler la table CLASSEMENT.
Gérez, de préférence, les ex æquo, qui doivent être de même rang.

A partir des données insérées, le résultat attendu est présenté ci-dessous.

RANG	NOM	ECART
1	G	0
2	H	0.01
2	I	0.01
3	E	0.05
4	B	0.06
4	F	0.06
5	D	0.07
6	J	0.1
7	A	0.16

Exercice 11

- Créez une fonction récursive nommée calculer_factorielle qui permet de calculer la factorielle d'un entier.

Factorielle de n = le produit des nombres entiers strictement positifs inférieurs ou égaux à n
ex : 10! = 1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 × 9 × 10 = 3 628 800

Algorithme récursif

```
Fonction calculer_factorielle (n)
   SI n = 0
     Renvoyer 1
   SINON
     Renvoyer n * calculer_factorielle (n - 1)
   FIN SI
FIN fonction
```

- Créez un bloc anonyme pour tester votre fonction calculer_factorielle.