



# ***Les ateliers PL/SQL version 4.1***

Chaque module est accompagné d'un ou plusieurs ateliers qui portent le même numéro.

Sur le site [www.bizoi.fr](http://www.bizoi.fr), vous pourrez trouver à partir de septembre 2014 tous les six mois une nouvelle version des ateliers avec des exercices et QCM supplémentaires. Vous pourrez trouver également des nouveaux modules qui compléteront le livre, en téléchargement libre.

Vous pouvez dialoguer avec l'auteur en lui écrivant à l'adresse : [razvan@bizoi.fr](mailto:razvan@bizoi.fr) ou directement sur le site [www.bizoi.fr](http://www.bizoi.fr).

## Atelier 1.1 Présentation de l'environnement

---



### Questions

1. Une table peut-elle avoir plusieurs clés primaires ?
2. Une table peut-elle avoir une contrainte unique si elle possède déjà une clé primaire ?
3. Une table qui possède une clé étrangère est-elle une table enfant ou une table parent ?
4. Que signifie LMD ?
5. Que signifie LDD ?
6. Quels sont les types d'instructions qui ne peuvent être exécutés en PL/SQL ?
7. Quels sont les avantages du langage PL/SQL par rapport au SQL ?
8. Pour configurer le client, lequel de ces fichiers utilisez-vous?
  - A. init.ora
  - B. sqlnet.ora
  - C. listener.ora
  - D. tnsnames.ora
9. Quel est le répertoire où se trouvent les fichiers de configuration ?
  - A. %ORACLE\_HOME%\admin\network
  - B. %ORACLE\_HOME%\network\admin
  - C. %ORACLE\_HOME%\net90\admin

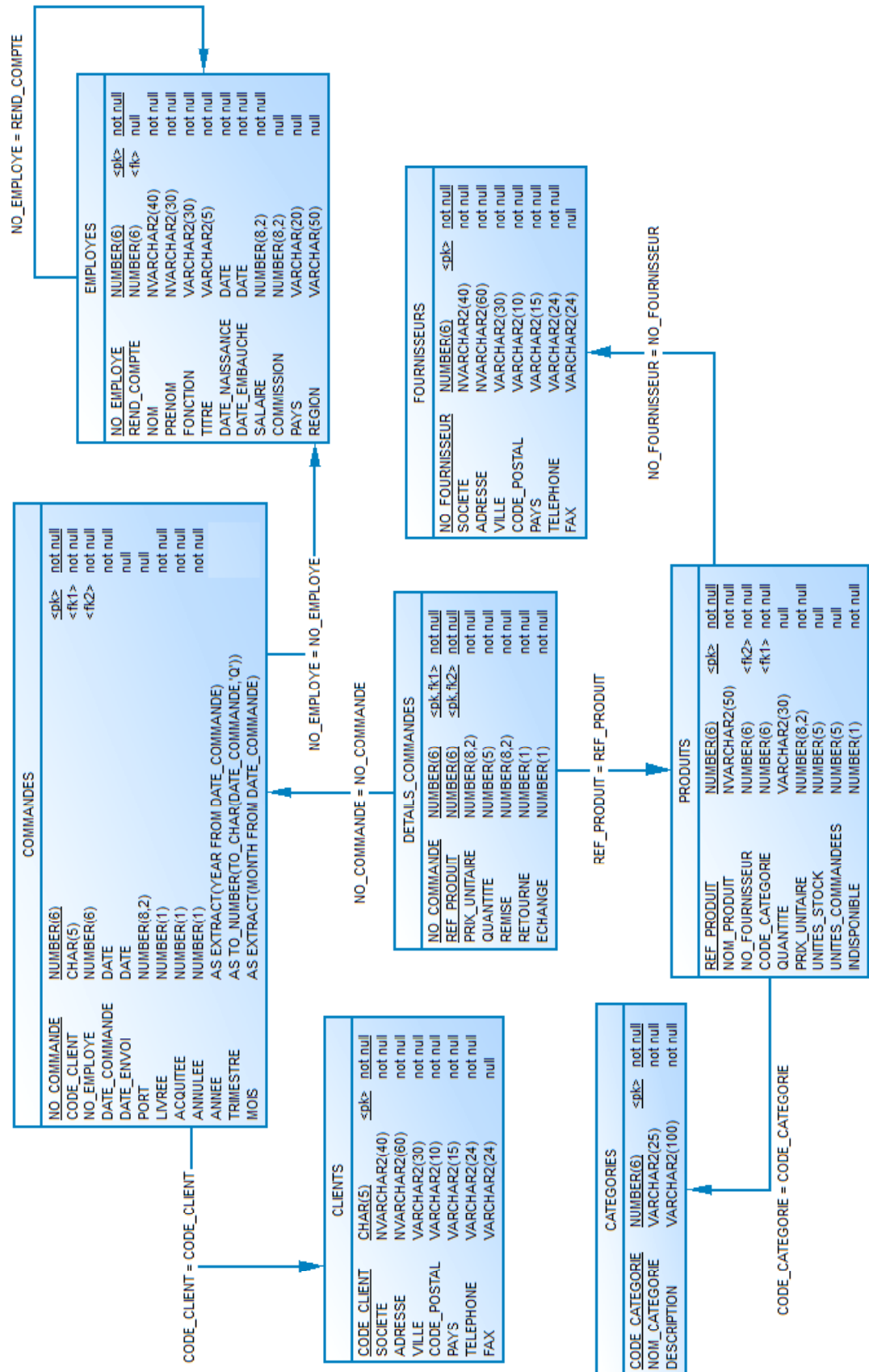
### Exercice n° 1 Installation

Installez Oracle XE sur votre machine en tenant compte de votre système d'exploitation.

### Exercice n° 2 Les tables utilisées pour les ateliers

Sachant que le symbole <pk> signifie clé primaire et <fk> la clé étrangère.

**Quelles sont les tables en relation parent enfant ?**



## Atelier 1.2 Les outils SQL\*Plus



### Questions

1. Quel est l'outil que vous retrouvez sur chaque serveur de base de données installée ?
  - A. SQL\*Plus.
  - B. iSQL\*Plus.
  - C. SQL\*Plus Worksheet
  - D. Oracle Enterprise Manager.
2. SQL\*Plus est-il un langage ou un environnement ?
3. Pour utiliser iSQL\*Plus sur une machine distante, avez-vous besoin d'installer le client Oracle ?
4. Quelle est la commande qui vous permet de vous connecter ?
5. Dans la syntaxe de démarrage de SQL\*Plus, pouvez-vous lancer l'exécution d'un script ?
6. Quelle est la commande qui vous permet de stocker dans un fichier tout ce qui est affiché à l'écran ?
7. Dans l'environnement SQL\*Plus, peut-on exécuter des commandes du système d'exploitation ?
8. Citez trois types de paramètres de mise en forme des résultats des requêtes.
9. Quelle est la commande qui vous permet de décrire la structure d'une vue ?

### Exercice n° 1 Préparer le poste de développement

Installez le schéma des exemples pour les ateliers en respectant la démarche suivante :

```
C:\>dir Oracle12cSQL_PLSQL.zip
Le volume dans le lecteur C n'a pas de nom.
Le numéro de série du volume est BC79-154D

Répertoire de C:\

12/08/2011  20:34          6 787 143 Oracle11gSQL_PLSQL.zip

C:\>unzip Oracle12cSQL_PLSQL.zip
Archive:  Oracle12cSQL_PLSQL.zip
  creating: Oracle12cSQL_PLSQL/
  inflating: Oracle12cSQL_PLSQL/DeleteEnvStagiaireXE.sql
  inflating: Oracle12cSQL_PLSQL/InitEnvEtoileXE.sql
  inflating: Oracle12cSQL_PLSQL/InitEnvStagiaireXE.sql
  creating: Oracle12cSQL_PLSQL/stagiaire/
  inflating: Oracle12cSQL_PLSQL/stagiaire/CATEGORIES.DAT
  inflating: Oracle12cSQL_PLSQL/stagiaire/CLIENTS.DAT
  inflating: Oracle12cSQL_PLSQL/stagiaire/COMMANDES.DAT
  inflating: Oracle12cSQL_PLSQL/stagiaire/COMMANDES_2009.DAT
```

```

inflating: Oracle12cSQL_PLSQL/stagiaire/DETAILS_COMMANDES.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/DETAILS_COMMANDES_2009.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/DIM_TEMPS.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/EMPLOYES.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/FOURNISSEURS.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/PRODUITS.DAT
inflating: Oracle12cSQL_PLSQL/stagiaire/STATISTIQUES.DAT

```

```
C:\>cd Oracle12cSQL_PLSQL
```

```
C:\Oracle12cSQL_PLSQL>dir
```

```
Répertoire de C:\Oracle12cSQL_PLSQL
```

```

12/08/2011  21:52    <REP>          .
12/08/2011  21:52    <REP>          ..
12/08/2011  21:52                550 DeleteEnvStagiaireXE.sql
12/08/2011  21:51                1 725 InitEnvEtoileXE.sql
12/08/2011  20:33                30 681 InitEnvStagiaireXE.sql
12/08/2011  20:33    <REP>          stagiaire

```

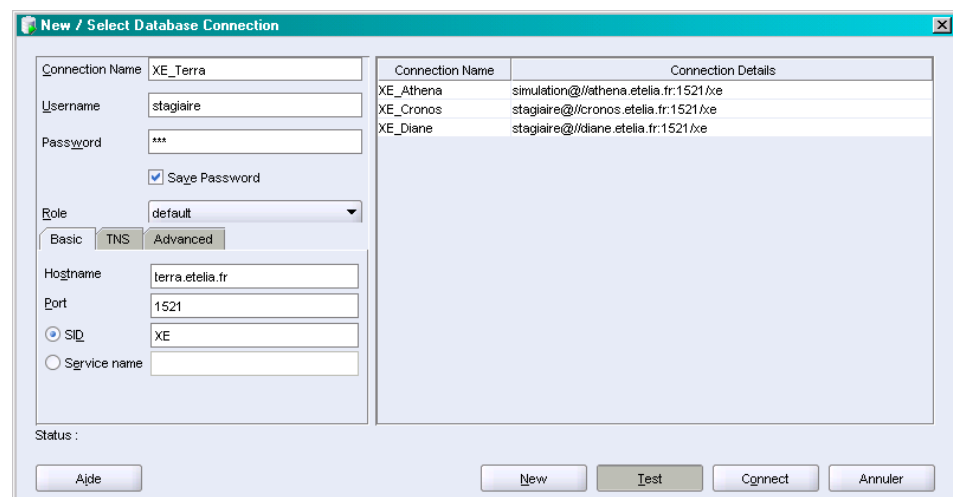
```
C:\Oracle12cSQL_PLSQL>sqlplus /nolog @InitEnvStagiaireXE.sql
```

Téléchargez et Installez l'outil SQL Developer.

## Exercice n° 2 Connexion

Démarrez SQL\*Plus, en ligne de commande, avec le nom d'utilisateur du schéma exemples « **STAGIAIRE** » et son mot de passe « **PWD** ».

Démarrez SQL Developer et paramétrez la connexion à la base de données.



## Exercice n° 3 Environnement SQL\*Plus

En utilisant SQL\*Plus en ligne de commande, redirigez les sorties vers un fichier et exécutez les commandes suivantes :

- Décrivez la table « **COMMANDES** » ;
- Déconnectez-vous de la base de données sans sortir du SQL\*Plus ;

- Décrivez de nouveau la table « **COMMANDES** ». Que remarquez-vous ?
- Connectez vous ;
- Affichez l'utilisateur courant ;
- Arrêtez la redirection des sorties vers le fichier ;
- Sans quitter l'environnement, listez le fichier que vous venez de créer.

#### Exercice n°4 Générer des scripts SQL

Connectez-vous à SQL\*Plus, redirigez les sorties vers le fichier « **DESC\_ALL.SQL** » et exécutez les commandes suivantes :

- Interrogez la vue catalogue à l'aide de la syntaxe suivante :

```
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SELECT 'DESC ' || TABLE_NAME FROM CAT
WHERE TABLE_TYPE = 'TABLE' ;
```

- Maintenant vous pouvez arrêter la redirection des sorties vers le fichier et exécuter le script ainsi conçu.

## Atelier 2.1 Bases du langage PL/SQL



### Questions

1. Quelles sont les sections qui font partie d'un bloc ?
2. Quel est le rôle de la section « **DECLARE** » ?
3. Quelles sont les syntaxes incorrectes ?
  - A. `declare begin NULL;begin NULL;begin NULL;end;end;end;`
  - B. `declare NULL;begin NULL;begin NULL;end;end;end;`
  - C. `declare begin NULL;begin NULL;begin NULL;end;end;`
  - D. `declare begin NULL;begin begin NULL;end;end;end;`
  - E. `declare begin NULL;begin NULL;begin NULL;end;NULL;end;NULL;end;`
4. Quel est le symbole de fin d'instruction en PL/SQL ?
  - A. .
  - B. :
  - C. ;
  - D. !
5. Quelles sont les syntaxes qui représentent des commentaires en PL/SQL ?
  - A. `/* Commentaire */`
  - B. `-- Commentaire --`
  - C. `' Commentaire '`
  - D. `" Commentaire "`
6. Quelle est la signification la syntaxe suivante :  
« **PRAGMA AUTONOMOUS\_TRANSACTION** » ?

### Exercice n°1 La présentation du PL/SQL

Créez un bloc PL/SQL qui affiche la description suivante :

Utilisateur : STAGIAIRE aujourd'hui est le 17 juillet 2006

Retrouvez le script créé pour l'Atelier 13 dans l'exercice 2, la mise à jour du modèle étoile permettent d'alimenter les quatre tables DIM\_EMPLOYES, DIM\_PRODUITS, DIM\_CLIENTS et à la fin INDICATEURS. Utilisez ce script pour créer un bloc PL/SQL qui effectue la mise à jour.

Utilisant les propriétés d'un bloc PL/SQL, vous devez effectuer la série des opérations suivantes :

- Augmenter les salaires des représentants de 10%.
- Insérer une nouvelle catégorie de produits avec le nom et la description suivante :  
'Produits cosmétiques'. Faites en sorte que l'insertion soit permanente.

- Annuler la modification de la table EMPLOYES.
- Vérifier que la nouvelle catégorie soit toujours en place.



## Atelier 3.1 Les variables



### Questions

1. Quelles sont les déclarations invalides ?
  - A. `nom_varA                   NUMBER(8) DEFAULT 10 ;`
  - B. `nom_var1, nom_var2 DATE;`
  - C. `nom_var                    VARCHAR2(20) NOT NULL ;`
  - D. `nom_var                    BOOLEAN := 1;`
  - E. `nom_var                    BINARY_INTEGER;`
  - F. `2nom_var                   BINARY_INTEGER;`
  - G. `a$nom_varG                DATE := '01/01/2006';`
  - H. `B#a$nom_var               DATE NOT NULL := SYSDATE;`
  - I. `nom_varI                    NUMBER(3) := 123.45678;`
  - J. `nom_var                    NUMBER(3) := 1234.5678;`
  - K. `nom_varK    CONSTANT NUMBER(12,3) := 1234.5678;`
2. Quel est le résultat de la requête suivante ?

```
SQL> declare
2   utilisateur varchar2(50) := '1 :'|USER;
3   begin
4     declare
5       utilisateur varchar2(50) := '2 :'|USER;
6     begin
7       declare
8         utilisateur varchar2(50) := '3 :'|USER;
9       begin
10        dbms_output.put_line( utilisateur);
11      end;
12    end;
13  end;
14  /
```

- A. '1 :STAGIAIRE'
  - B. '2 :STAGIAIRE'
  - C. '3 :STAGIAIRE'
3. Quelles sont les syntaxes correctes ?
  - A. `declare v_1 NUMBER(8,2) := 2500;`  
`begin v_1 = v_1 * 2; end;`
  - B. `declare v_1 date;`  
`begin v_1 := sysdate; end;`
  - C. `declare v_1 constant date;`  
`begin v_1 := sysdate; end;`

```
D. declare v_1 constant date := sysdate;  
    begin null; end;
```

```
E. declare v_1 NUMBER := v_2; begin null; end;
```

### Exercice n°1      La déclaration des variables

Créez un bloc PL/SQL dans lequel vous déclarez les variables de la question 24.1-1 les points : A, G, I, K. Affichez les informations stockées dans ces variables.

Déclarez une variable de liaison de type « **VARCHAR2** ». Créez un premier bloc qui alimente la variable avec la valeur de l'utilisateur courant concaténée avec la date du jour. Créez un deuxième bloc qui affiche la variable.

## Atelier 3.2 Les variables

### Questions



1. Quelles sont les déclarations invalides ?
  - A. `declare SUBTYPE Numeral IS NUMBER(1,0); v_1 Numeral; begin v_1 := 1; end;`
  - B. `declare SUBTYPE v_1 IS TIMESTAMP; begin v_1:= SYSTIMESTAMP; end;`
  - C. `declare v_1 DIM_TEMPS.JOUR%TYPE; begin v_1:= SYSDATE; end;`
  - D. `declare v_1 DIM_TEMPS%ROWTYPE; begin v_1.JOUR:= SYSDATE; end;`
  - E. `declare v_1 DIM_TEMPS%ROWTYPE; begin v_1:= SYSDATE; end;`
  - F. `declare TYPE var IS VARRAY(3) OF NVARCHAR2(30); v_1 var:= var('BIZOI', 'FABER'); begin null; end;`
  - G. `declare TYPE var IS RECORD ( A VARCHAR2(3) := 'AA', B VARCHAR2(3) := 'BB'); v_1 var; begin null; end;`
  - H. `declare TYPE var IS TABLE OF DATE INDEX BY BINARY_INTEGER; v_1 var; begin v_1(1):=sysdate; end;`
  - I. `declare v_1 DIM_TEMPS.JOUR%TYPE := ADD_MONTHS( TRUNC(SYSDATE, 'MONTH'), 1); begin null; end;`
  - J. `declare v_1 CLIENTS%ROWTYPE; begin v_1.CODE_CLIENT := 'AA'; v_1.SOCIETE := 'BB'; end;`
  - K. `declare TYPE var IS TABLE OF DATE INDEX BY VARCHAR2(2); v_1 var; begin null; end;`
2. Quelles est le type de retour de chaque expression suivante :
  - A. `256*2 + EXTRACT(YEAR FROM SYSDATE)`
  - B. `1024 || SYSDATE || USER`
  - C. `SYSDATE > '01/07/2006'`
  - D. `2.5*2.5/0f + 10`
  - E. `INSTR('QUANTITE', 'T')*256`
  - F. `SYSDATE - ROUND(TRUNC(MOD(1600,10), -1), 2)`
  - G. `2.5D*256+10`
  - H. `2.5f/0 || USER`
  - I. `SYSDATE + 3070 + 2.5f`

### Exercice n°1 Les variables composées

A partir des syntaxes de la question 24.2-1 écrivez les blocs suivants :

- L'option D, remplacez 'SYSDATE' par l'option I de la question 24.2.2. Initialisez tous les champs de l'enregistrement utilisant la date déjà affectée 'v\_1.JOUR' et insérez les dans la table 'DIM\_TEMPS' en validant la transaction directement dans le bloc.
- Modifiez les blocs des options F, G, I de la question 24.2-1 pour permettre l'affichage des variables déclarées.
- Modifiez le bloc de l'option H de la question 24.2-1, pour permettre d'alimenter le premier poste du tableau avec la date du jour et le deuxième poste avec le lendemain. Affichez les deux postes du tableau.

## Atelier 4.1 Les ordres SQL dans PL/SQL

### Questions



1. Sachant que les expressions doivent remplacer les trois points dans le bloc suivant, quelles sont les expressions invalides ?

```
declare
  v_1 EMPLOYES%ROWTYPE;
  TYPE TAB IS TABLE OF EMPLOYES%ROWTYPE
                                INDEX BY BINARY_INTEGER;
  t_1 TAB;
begin
  SELECT * INTO v_1 FROM EMPLOYES WHERE NO_EMPLOYE = 5;

end;
/
```

- A. SELECT \* INTO v\_1 FROM EMPLOYES WHERE NO\_EMPLOYE = 5;
  - B. UPDATE EMPLOYES SET ROW = v\_1 WHERE NO\_EMPLOYE = 5;
  - C. SELECT count(\*) INTO v\_1 FROM EMPLOYES WHERE 1 = 2;
  - D. SELECT \* INTO v\_1 FROM EMPLOYES WHERE 1 = 2;
  - E. SELECT \* INTO v\_1 FROM EMPLOYES;
  - F. SELECT \* BULK COLLECT INTO t\_1 FROM EMPLOYES;
  - G. v\_1.NO\_EMPLOYE:=100;INSERT INTO EMPLOYES VALUES v\_1;
  - H. v\_1.NO\_EMPLOYE:=100;INSERT INTO EMPLOYES  
VALUES ( v\_1.NO\_EMPLOYE, v\_1.REND\_COMPTE, v\_1.NOM,  
v\_1.PRENOM, v\_1.FONCTION, v\_1.TITRE, v\_1.DATE\_NAISSANCE,  
v\_1.DATE\_EMBAUCHE, v\_1.SALAIRE, v\_1.COMMISSION);
2. Sachant que les variables ont été déclarées auparavant, qu'elles sont du bon type et au bon endroit, quels sont les ordres de mise à jour incorrects ?
- A. INSERT INTO CATEGORIES VALUES ( 9,'Fruits',  
'Fruits') RETURNING ROWID INTO v\_rowid;
  - B. INSERT INTO CATEGORIES VALUES ( 9,'Fruits',  
'Fruits') RETURNING \* INTO v\_cat;
  - C. UPDATE COMMANDES SET PORT = PORT \* 1.05  
RETURNING NO\_COMMANDE BULK COLLECT INTO v\_comm;
  - D. DELETE CATEGORIES WHERE ROWID = v\_1  
RETURNING CODE\_CATEGORIE INTO v\_cat;
  - E. UPDATE CATEGORIES SET NOM\_CATEGORIE = DESCRIPTION  
WHERE ROWID = v\_1 RETURNING NOM\_CATEGORIE  
INTO v\_cat;
  - F. UPDATE COMMANDES SET PORT = PORT \* 1.05  
RETURNING NO\_COMMANDE INTO v\_comm;
  - G. DELETE INDICATEURS RETURNING ROWID  
BULK COLLECT INTO v\_rowid;

```
H. DELETE CATEGORIES RETURNING CODE_CATEGORIE  
   INTO v_cat;
```

## **Exercice n°1      Les ordres SQL dans PL/SQL**

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- Effacez les enregistrements des commandes de l'année 2009.
- Affichez le client, l'adresse et le numéro de téléphone du client qui a le CODE\_CLIENT='PARIS'. Effacez les enregistrements du client dans la table INDICATEURS.
- Modifiez le produit numéro 8 en le rendant disponible 'INDISPONIBLE' := 0 et rajoutant 200 unités en stock. Affichez le nom du fournisseur et le nom de la catégorie de ce produit. Effacez les enregistrements du produit dans la table INDICATEURS.
- Affichez les deux employés encadrés par 'Buchanan'. Augmentez les frais de port de '10%' pour toutes les commandes passées par ces deux employés dans l'année '2011', la modification doit être faite dans la table INDICATEURS. Affichez mensuellement pour l'année '2011' les cumuls des frais de port et des quantités.

## Atelier 4.2 Les ordres SQL dans PL/SQL

### Questions



1. Quel est l'affichage suite à l'exécution de ce bloc ? Argumentez votre réponse.

```
SQL> declare
2   v_sql_dynamique VARCHAR2(200) := 'CREATE TABLE SAV_CAT AS ' || '
3       SELECT * FROM CATEGORIES WHERE 1=2' ;
4   v_count NUMBER(5);
5   begin
6       EXECUTE IMMEDIATE v_sql_dynamique;
7       SELECT COUNT(*) INTO v_count FROM CATEGORIES;
8       dbms_output.put_line( 'Enregistrements : ' || v_count);
9       INSERT INTO SAV_CAT SELECT * FROM CATEGORIES
10      SELECT COUNT(*) INTO v_count FROM SAV_CAT;
11      dbms_output.put_line( 'Enregistrements : ' || v_count);
12  end;
13  /
```

- A. Enregistrements : 9 9  
 B. Enregistrements : 9 0  
 C. Enregistrements : 9  
 D. Enregistrements :  
 E. ERREUR à la ligne 10 : ...
2. Pour laquelle de ces exécutions, 'v\_sql' ne peut pas être un bloc PL/PLSQ ?
- A. EXECUTE IMMEDIATE v\_sql USING v\_1, v\_2  
 RETURNING BULK COLLECT INTO v\_tab;
- B. EXECUTE IMMEDIATE v\_sql USING  
 IN v\_1, IN v\_2, OUT v\_3;
- C. EXECUTE IMMEDIATE v\_sql USING v\_1, v\_2;
- D. EXECUTE IMMEDIATE v\_sql;

### Exercice n°1 Les ordres SQL dynamiques

Pour des besoins d'analyse, on a besoin d'une table pour recenser toutes les ventes, créée chaque jour. La structure de la table est identique à celle de la table VENTES\_CLIENS\_2011. Elle doit avoir le nom fourni par l'expression suivante :

```
'VENTES_' || TO_CHAR(SYSDATE, 'YYYYMMDD')
```

Une fois créée, vous devez l'alimenter avec les enregistrements des ventes de l'année '2011'.

Octroyez les privilèges de lecture pour tous les utilisateurs de la base et créez un synonyme public, avec le même nom, pour cette table.

## Atelier 5.1 Les structures de contrôle

---

### Questions



1. Quelles sont les instructions de contrôle structurellement invalides ?
  - A. `if CONDITION then EXPRESSION end if;`
  - B. `if CONDITION then EXPRESSION elsif CONDITION then EXPRESSION else EXPRESSION end if;`
  - C. `if CONDITION then EXPRESSION else if CONDITION then EXPRESSION else EXPRESSION end if; end if;`
  - D. `if CONDITION then EXPRESSION else if CONDITION then EXPRESSION else EXPRESSION end if;`
  - E. `if CONDITION then EXPRESSION else EXPRESSION endif;`
  - F. `case EXPRESSION when 1 then EXPRESSION when 2 then EXPRESSION else EXPRESSION end case;`
  - G. `case EXPRESSION when 1 then EXPRESSION when 2 then EXPRESSION else EXPRESSION endcase;`
  - H. `case EXPRESSION when CONDITION then EXPRESSION else EXPRESSION end case;`
  - I. `case when CONDITION then EXPRESSION when CONDITION then EXPRESSION else EXPRESSION end case;`
  - J. `case when CONDITION then EXPRESSION when 1 then EXPRESSION else EXPRESSION end case;`
2. Quelles sont les instructions de contrôle structurellement invalides ?
  - A. `while CONDITION loop  
CONDITION:= NOT CONDITION; end loop;`
  - B. `while CONDITION  
CONDITION:= NOT CONDITION; end loop;`
  - C. `while CONDITION loop  
CONDITION:= NOT CONDITION; endloop;`
  - D. `loop exit; end loop;`
  - E. `whileloop exit; end loop;`
  - F. `loop exit; when CONDITION; end loop;`
  - G. `loop exit when CONDITION; end loop;`
  - H. `<<B01>>loop exit B01 when CONDITION; end loop;`
  - I. `for i in 1..3 loop NULL; end loop;`
  - J. `for i in 1 3 loop NULL; end loop;`
  - K. `for i in 1..3 NULL; end loop;`
  - L. `for i in 1..3 loop NULL; endloop;`
  - M. `forall i in 1..3 ORDRE_DML;`



```
N. forall i in 1..3 loop ORDRE_DML; end loop;
```

## Exercice n°1 Les structures conditionnelles

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- Pour les commandes de l'année '2011' augmentez les frais de port de '10%' pour tous les clients étrangers et diminuez les frais de port de '5%' pour les clients français. Contrôlez le nombre des enregistrements modifiés et si vous avez modifié des enregistrements, validez la transaction.
- Augmentez le salaire de l'employé numéro 3 si le salaire de l'employé est inférieur à la moyenne des salaires des employés qui ont la même FONCTION. Modifiez également la commission du même employé si la commission est inférieure à la moyenne des commissions des employés qui ont la même FONCTION, on lui attribue la moyenne comme commission. Contrôlez le nombre des enregistrements modifiés et si vous avez modifié des enregistrements, validez la transaction.

## Exercice n°2 Les structures itératives

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- Affichez les chiffres de 1 à 10 comme dans le modèle suivant :

```
Le numéro 1 est impair
Le numéro 2 est pair
Le numéro 3 est impair
Le numéro 4 est pair
Le numéro 5 est impair
Le numéro 6 est pair
Le numéro 7 est impair
Le numéro 8 est pair
Le numéro 9 est impair
Le numéro 10 est pair
```

- Déclarez un tableau de type NUMBER de dix postes, et deux boucles : une qui affecte le tableau avec les valeurs de 1 à 9 et une autre qui affiche le tableau à partir du dernier élément affecté.
- Augmentez les salaires de '10%' pour tous les employés encadrés par 'Buchanan'. Augmentez la remise accordée par ces employés de '1%' ('REMISE + .01') pour toutes les commandes de l'année '2011'. Effacez tous les enregistrements de leurs commandes de la table INDICATEURS.
- Augmentez le prix unitaire de '10%' des produits de la catégorie 3. Mettez à jour les prix unitaires des produits pour les commandes de l'année '2011'.

## Atelier 6.1 Les curseurs

### Questions



```

SQL> declare
  2     CURSOR c_1 IS SELECT * FROM PRODUITS;
  3     v_1 c_1%ROWTYPE;
  4 begin
  5  /*A*/if c_1%FOUND then
  6         dbms_output.put_line('/*A*/ %FOUND'); end if;
  7  /*B*/if c_1%ISOPEN then
  8         dbms_output.put_line('/*B*/ %ISOPEN'); end if;
  9  /*C*/if c_1%NOTFOUND then
 10         dbms_output.put_line('/*C*/ %NOTFOUND');end if;
 11  /*D*/if c_1%ROWCOUNT >0 then
 12         dbms_output.put_line('/*D*/ %ISOPEN'); end if;
 13     open c_1;
 14  /*E*/if c_1%FOUND then
 15         dbms_output.put_line('/*E*/ %FOUND'); end if;
 16  /*F*/if c_1%ISOPEN then
 17         dbms_output.put_line('/*F*/ %ISOPEN'); end if;
 18  /*G*/if c_1%NOTFOUND then
 19         dbms_output.put_line('/*G*/ %NOTFOUND');end if;
 20  /*H*/if c_1%ROWCOUNT = 0 then
 21         dbms_output.put_line('/*H*/ %ROWCOUNT'); end if;
 22     loop
 23         fetch c_1 into v_1;
 24         exit when c_1%NOTFOUND;
 25     end loop;
 26  /*I*/if c_1%FOUND then
 27         dbms_output.put_line('/*I*/ %FOUND'); end if;
 28  /*J*/if c_1%ISOPEN then
 29         dbms_output.put_line('/*J*/ %ISOPEN'); end if;
 30  /*K*/if c_1%NOTFOUND then
 31         dbms_output.put_line('/*K*/ %NOTFOUND');end if;
 32  /*L*/if c_1%ROWCOUNT >0 then
 33         dbms_output.put_line('/*L*/ %ROWCOUNT'); end if;
 34     close c_1;
 35  /*M*/if c_1%FOUND then
 36         dbms_output.put_line('/*M*/ %FOUND'); end if;
 37  /*N*/if c_1%ISOPEN then
 38         dbms_output.put_line('/*N*/ %ISOPEN'); end if;
 39  /*O*/if c_1%NOTFOUND then
 40         dbms_output.put_line('/*O*/ %NOTFOUND');end if;
 41  /*P*/if c_1%ROWCOUNT >0 then
 42         dbms_output.put_line('/*P*/ %ISOPEN'); end if;
 43 end;
 44 /

```

1. Quelles sont les instructions qui génèrent une erreur due à une lecture des attributs du curseur ?
2. Si on efface les instructions qui génèrent une erreur, quelles sont les autres instructions qui valident la condition et affichent la chaîne avec leur lettre et leur attribut ?

### **Exercice n°1      Les curseurs explicites**

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- À l'aide d'un curseur explicite, insérez les enregistrements correspondants dans la table du modèle étoile `QUANTITES_CLIENTS`. Avant d'insérer, effacez tous les enregistrements.
- À l'aide d'un curseur explicite, insérez les enregistrements correspondants dans la table du modèle étoile `VENTES_MOIS` uniquement pour l'année passée en argument au curseur. Le bloc doit utiliser une variable de substitution pour alimenter une variable PL/SQL. Avant d'insérer, effacez tous les enregistrements de l'année qui a été passée en argument.
- À l'aide d'un curseur explicite, affichez l'employé, la fonction à partir de la table du modèle étoile `DIM_EMPLOYES`.

## Atelier 6.2 Les curseurs

---

### Exercice n°1 Les boucles et les curseurs

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- À l'aide d'un curseur et de la boucle « **FOR** », affichez les clients, leur ville et leur pays à partir de la table du modèle étoile DIM\_CLIENTS.
- À l'aide d'un curseur et de la boucle « **FOR** », insérez les enregistrements correspondants dans la table du modèle étoile VENTES\_CLIENTS uniquement pour l'année passée en argument au curseur. Le bloc doit utiliser une variable de substitution pour alimenter une variable PL/SQL. Avant d'insérer, effacez tous les enregistrements de l'année qui a été passée en argument.
- En utilisant deux curseurs déclarés directement dans la boucle « **FOR** », affichez les clients et commandes pour les clients qui payent un port supérieur à trois fois la moyenne des commandes pour la même année.

### Exercice n°2 Les curseurs en mise à jour

Créez le bloc PL/SQL qui permet de mettre les frais de port à zéro pour toutes les commandes de clients qui habitent dans la même ville que les fournisseurs des produits achetés pour uniquement l'année passée en argument au curseur. Les modifications sont effectuées dans la table COMMANDES du modèle relationnel mais en même temps vous devez effacer les enregistrements de ces commandes dans la table INDICATEURS du modèle étoile.

### Exercice n°3 Les variables curseurs

Créez le bloc PL/SQL qui permet d'afficher à partir du modèle étoile l'une des tables :

- VENTES\_CLIENTS\_2009,
- VENTES\_CLIENTS\_2010 ou
- VENTES\_CLIENTS\_2011

Dynamiquement, suivant l'année passée en argument, vous testez que la table existe et vous affichez tous les enregistrements de la table. Si la table n'existe pas, vous affichez tous les enregistrements de la table VENTES\_CLIENTS pour l'année qui a été passée en argument.

## Atelier 7.1 Les exceptions

### Questions



```

SQL> declare
2      EXCEPTION_1 EXCEPTION;
3      EXCEPTION_2 EXCEPTION;
4      EXCEPTION_3 EXCEPTION;
5      EXCEPTION_4 EXCEPTION;
6      EXCEPTION_5 EXCEPTION;
7  begin --bloc 1
8      begin --bloc 2
9          begin --bloc 3
10             begin --bloc 4
11                 begin --bloc 5
12                     RAISE EXCEPTION_5;
13                     dbms_output.put_line( 'Suite traitements bloc 5. ');
14                     exception--exception bloc 5
15                     when EXCEPTION_5 then
16                         dbms_output.put_line( 'Exception EXCEPTION_5 ');
17                         RAISE EXCEPTION_4;
18                     end;--end bloc 5
19                     dbms_output.put_line( 'Suite traitements bloc 4. ');
20                     exception--exception bloc 4
21                     when EXCEPTION_4 then
22                         dbms_output.put_line( 'Exception EXCEPTION_4 ');
23                         RAISE EXCEPTION_3;
24                     end;--end bloc 4
25                     dbms_output.put_line( 'Suite traitements bloc 3. ');
26                     exception--exception bloc 3
27                     when EXCEPTION_3 then
28                         dbms_output.put_line( 'Exception EXCEPTION_3 ');
29                         RAISE EXCEPTION_2;
30                     end;--end bloc 3
31                     dbms_output.put_line( 'Suite traitements bloc 2. ');
32                     exception--exception bloc 2
33                     when EXCEPTION_2 then
34                         dbms_output.put_line( 'Exception EXCEPTION_2 ');
35                         RAISE EXCEPTION_1;
36                     end;--end bloc 2
37                     dbms_output.put_line( 'Suite traitements bloc 1. ');
38                     exception--exception bloc 1
39                     when EXCEPTION_1 then
40                         dbms_output.put_line( 'Exception EXCEPTION_1 ');
41                     when OTHERS then
42                         dbms_output.put_line( 'Une autre erreur. ');
43                     end;--end bloc 1
44 /

```

1. Quel est l'affichage effectué par le bloc suite au traitement ?

A.

```
Exception EXCEPTION_5  
Suite traitements bloc 4.  
Exception EXCEPTION_4  
Suite traitements bloc 3.  
Exception EXCEPTION_3  
Suite traitements bloc 2.  
Exception EXCEPTION_2  
Suite traitements bloc 1.  
Exception EXCEPTION_1
```

B.

```
Exception EXCEPTION_5  
Suite traitements bloc 4.  
Suite traitements bloc 3.  
Suite traitements bloc 2.  
Suite traitements bloc 1.
```

C.

```
Exception EXCEPTION_5  
Exception EXCEPTION_4  
Exception EXCEPTION_3  
Exception EXCEPTION_2  
Exception EXCEPTION_1
```

D.

```
Suite traitements bloc 5.  
Suite traitements bloc 4.  
Suite traitements bloc 3.  
Suite traitements bloc 2.  
Suite traitements bloc 1.
```

E.

```
Une autre erreur.
```

## Exercice n°1 La gestion des exceptions

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- Récupérez dans trois variables PL/SQL les valeurs saisies à l'aide des variables de substitution. Les variables représentent le code d'une catégorie, le numéro d'un fournisseur et la référence d'un produit qui doivent être effacés. Ainsi vous effacez un enregistrement dans la table CATEGORIES, un enregistrement de la table FOURNISSEURS et un enregistrement de la table PRODUITS. Il faut enchaîner les traitements dans plusieurs blocs de sorte que si une de ces commandes n'aboutit pas, les suivantes seront exécutées quand même.
- Ecrire le code permettant de générer et de traiter chacune des exceptions suivantes :  
CASE\_NOT\_FOUND,  
CURSOR\_ALREADY\_OPEN,  
DUP\_VAL\_ON\_INDEX,  
INVALID\_CURSOR,  
INVALID\_NUMBER,  
NO\_DATA\_FOUND,  
TOO\_MANY\_ROWS,

VALUE\_ERROR,  
ZERO\_DIVIDE

Suivant les choix à l'exécution, par la saisie d'une variable de substitution, vous exécutez le code erroné qui passe directement à la section de l'exception choisie du programme qui affiche le nom de l'exception.

## **Exercice n°2      Les exceptions anonymes**

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- A chaque fois qu'on efface une commande saisie par l'utilisateur, gérez l'exception « ORA-02292: violation de contrainte (.) d'intégrité - enregistrement fils existant », en effaçant tous les enregistrements de la table DETAILS\_COMANDES correspondantes.

## **Exercice n°2      Les exceptions utilisateur**

Créez le bloc PL/SQL qui permet d'effectuer les opérations :

- Modifiez le salaire d'un employé pour le NO\_EMPLOYE saisi à l'exécution. Contrôlez que le salaire ne soit pas inférieur au salaire actuel ; si c'est le cas, lancez une exception.
- Permettre de saisir les informations d'un employé et de les insérer dans la table EMPLOYES. Si l'âge de l'employé est inférieur à 18 ans, n'insérez pas et lancez une exception.

## Atelier 8.1 Les sous-programmes

### Questions



```
SQL> declare
2   PROCEDURE ProcedureA IS
3   begin
4       dbms_output.put_line( 'ProcedureA');
5       ProcedureB;
6   end;
7   PROCEDURE ProcedureB IS
8   begin
9       dbms_output.put_line( 'ProcedureB');
10  end;
11 begin
12     ProcedureA;
13 end;
14 /
```

1. Le bloc précédent peut-il être compilé ? justifiez votre réponse.
2. Quelles sont les syntaxes invalides ?
  - A. PROCEDURE p1( a\_1 NUMBER ) IS declare a\_1 := 0; end;
  - B. PROCEDURE p1( a\_1 IN OUT NUMBER ) IS declare a\_1 := 0; end;
  - C. PROCEDURE p1( a\_1 OUT NUMBER ) IS declare a\_1 := 0; end;
  - D. PROCEDURE p1( a\_1 IN NUMBER ) IS declare a\_1 := 0; end;

```
SQL> CREATE OR REPLACE PROCEDURE
2   p1( a_1 NUMBER := 0, a_2 NUMBER := 0, a_3 NUMBER := 0 ) IS
3   begin
4       dbms_output.put_line( a_1||' '||a_2||' '||a_3);
5   end;
6   /
```

3. Quelles sont les syntaxes qui affichent la chaîne suivante '1 2 0' ?
  - A. exec p1;
  - B. exec p1(1,2,3);
  - C. exec p1(1,2);
  - D. exec p1(3);
  - E. exec p1(a\_3 => 3, a\_1 => 1);
  - F. exec p1(a\_2 => 3, a\_1 => 1);
  - G. exec p1(a\_3 => 0, a\_1 => 1, a\_2 => 2);
  - H. exec p1(a\_1 => 3, a\_2 => 2, a\_3 => 1);



I. `exec pl(a_1 => 3, a_2 => 0, a_3 => 0);`

J. `exec pl(a_2 => 2, a_1 => 1, a_3 => 0);`

## Exercice n°1 Les fonctions

Créez une fonction qui permet d'effectuer les opérations suivantes :

- A partir d'une catégorie des produits passée comme argument, la fonction doit retourner VRAI si l'enregistrement existe dans la table CATEGORIES, et FALSE dans le cas contraire.
- A partir du numéro du fournisseur passé comme argument, la fonction doit retourner VRAI si l'enregistrement existe dans la table FOURNISSEURS, et sinon FALSE.
- A partir du nom de l'employé et des deux dates passés comme arguments, retrouvez le nombre des contrats saisis dans la table COMMANDES.
- A partir du numéro de la commande, retournez le montant de la commande.
- A partir du nom de la table passée comme argument et de la valeur de la clé pour un enregistrement, valeur passée sous forme de chaîne de caractères. Contrôlez que le nom de la table existe, retrouvez le nom de la colonne de clé primaire et créez une requête dynamique qui vérifie que l'enregistrement existe. La fonction doit retourner VRAI si l'enregistrement existe, sinon FALSE.

## Exercice n°2 Les procédures

Créez une procédure qui permet d'effectuer les opérations suivantes:

- A partir du numéro de commande passé en argument, retirez des unités en stocks toutes les quantités de produits de la commande. S'il n'y a pas assez des unités en stock, commandez la différence, en modifiant la valeur des unités commandées.
- A partir du numéro d'une catégorie de produits, contrôlez les produits en stock pour la catégorie si la quantité des unités en stock est supérieure à la moyenne de la catégorie, sinon commandez deux fois la différence arrondie au dixième supérieur.
- A partir du numéro d'une catégorie de produits, validez l'arrivée des produits commandés, ainsi rajoutez aux quantités des unités en stock les quantités des unités commandées et mettez les unités commandées à zéro.
- Saisir toutes les informations nécessaires pour insérer un produit dans la table PRODUITS. Utilisez la fonction de contrôle d'existence d'un enregistrement précédemment créée pour vérifier toutes les contraintes de clé primaire nécessaires. Créez trois procédures, la première qui a comme argument un enregistrement du même type que la table produits, la deuxième avec une liste arguments représentant les champs de la table produits et la troisième qui effectue le traitement. Les deux premières procédures doivent avoir le même nom, le traitement est effectué par la troisième. La première et la deuxième procédures appellent la troisième. Ainsi la première et la deuxième procédure ne sont qu'un moyen de diversifier les possibilités d'accès à la troisième procédure.
- Dans les quatre tables DIM\_EMPLOYES, DIM\_PRODUITS, DIM\_CLIENTS et à la fin INDICATEURS du modèle étoile, mettez à jour les enregistrements et si tel est le cas insérer tous les enregistrements manquants.

- Effacez d’abord les enregistrements, puis insérez les dans les sept tables du modèle étoile :  
QUANTITES\_CLIENTS,  
VENTES\_CLIENTS,  
VENTES\_ANNEES ,  
VENTES\_MOIS  
VENTES\_CLIENTS\_2009  
VENTES\_CLIENTS\_2010  
VENTES\_CLIENTS\_2011

## Atelier 9.1 Les packages

### Questions



```
SQL> CREATE OR REPLACE PACKAGE GererProduit
2      CURSOR c_produit RETURN PRODUITS%ROWTYPE
3      IS SELECT REF_PRODUIT, NOM_PRODUIT, NO_FOURNISSEUR,
4          CODE_CATEGORIE, QUANTITE, PRIX_UNITAIRE,
5          UNITES_STOCK, UNITES_COMMANDEES, INDISPONIBLE
6          FROM PRODUITS;
7      ...
```

1. Le package précédent peut-il être compilé ? Justifiez votre réponse.

```
SQL> CREATE OR REPLACE PACKAGE PkgSurcharge
2  AS
3      PROCEDURE NomProcedure01( a_arg01 VARCHAR2);
4      PROCEDURE NomProcedure01( a_arg02 VARCHAR2);
5      PROCEDURE NomProcedure02( a_arg IN VARCHAR2);
6      PROCEDURE NomProcedure02( a_arg OUT VARCHAR2);
7      ...
```

2. Le package précédent peut-il être compilé ? Justifiez votre réponse.

### Exercice n°1 Les packages

Créez un package pour la gestion des employés avec ces caractéristiques :

- Une fonction qui contrôle l'existence d'un employé dans la table EMPLOYES à partir du numéro de l'employé.
- Une procédure de suppression d'un employé.
- Une procédure d'augmentation du salaire pour un employé. La procédure comporte deux arguments ; le premier est le numéro de l'employé, qui doit être contrôlé, et le deuxième est le montant de l'augmentation. Si le montant est égal à zéro l'employé se voit attribuer la moyenne des salaires.
- Une procédure d'insertion d'un employé dans la table EMPLOYES. Il faut contrôler que le supérieur hiérarchique existe déjà dans la table. L'âge de l'employé doit être supérieur à 18 ans. Vous pouvez utiliser une constante pour stocker l'âge minimum. Il faut également contrôler si l'employé n'existe pas déjà dans la table.
- Pour les tests du package, créez un script SQL qui vous permette de saisir les informations pour l'ajout d'un employé, l'augmentation et la suppression.

## Atelier 10.1 Les déclencheurs

### Questions



1. Quels sont les déclencheurs qui peuvent être compilés ?
  - A. `CREATE OR REPLACE TRIGGER T1 BEFORE UPDATE ON CATEGORIES BEGIN :new.DESCRPTION := 'DESCRIPTION'; END T1;`
  - B. `CREATE OR REPLACE TRIGGER T2 AFTER UPDATE ON CATEGORIES BEGIN :new.DESCRPTION := 'DESCRIPTION'; END T2;`
  - C. `CREATE OR REPLACE TRIGGER T3 BEFORE UPDATE ON CATEGORIES FOR EACH ROW BEGIN :new.DESCRPTION := 'DESCRIPTION'; END T3;`
  - D. `CREATE OR REPLACE TRIGGER T4 AFTER UPDATE ON CATEGORIES FOR EACH ROW BEGIN :new.DESCRPTION := 'DESCRIPTION'; END T4;`
  - E. `CREATE OR REPLACE TRIGGER T5 BEFORE INSERT ON CATEGORIES FOR EACH ROW BEGIN :old.DESCRPTION := 'DESCRIPTION'; END T5;`
  - F. `CREATE OR REPLACE TRIGGER T6 AFTER INSERT ON CATEGORIES FOR EACH ROW BEGIN :old.DESCRPTION := 'DESCRIPTION'; END T6;`
  - G. `CREATE OR REPLACE TRIGGER T8 AFTER INSERT ON CATEGORIES FOR EACH ROW DECLARE a NUMBER; BEGIN a:= :old.DESCRPTION; END T8;`
  - H. `CREATE OR REPLACE TRIGGER T9 BEFORE INSERT ON CATEGORIES FOR EACH ROW DECLARE a NUMBER; BEGIN a:= :old.DESCRPTION; END T9;`

### Exercice n°1 Les déclencheurs d'instruction

Créez un déclencheur sur la table `PRODUITS` qui empêche l'insertion d'un enregistrement ou la mise à jour des produits en stock de la table `PRODUITS` si un de produits suivants ' 2 , 4 , 6 , 8 ' a déjà un stock de 700 unités.

Créez un déclencheur sur la table `EMPLOYES` qui empêche toute opération si elle ne s'effectue pendant les heures de travail.

### Exercice n°2 Les déclencheurs d'enregistrement

Créez une séquence qui commence avec le dernier numéro d'employé se trouvant dans la table `EMPLOYES`. Ensuite, créez un déclencheur qui initialise le `NO_EMPLOYE` avec la valeur suivante de la séquence s'il n'a pas été renseigné. Dans le même déclencheur, testez si le champ `REND_COMPTE` est correctement initialisé, avec une valeur d'un employé existant, sinon vous l'initialisez avec la même valeur que `NO_EMPLOYE`.

Créez deux tables `PRODUITS_POUBELLE` et `PRODUITS_ARCHIVE` qui ont la même description que la table `PRODUITS` avec deux colonnes de plus, `UTILISATEUR` et `DATE_SYSTEME`. Créez un déclencheur sur la table `PRODUITS` qui archive dans la table `PRODUITS_POUBELLE` tous les produits effacés et dans la table `PRODUITS_ARCHIVE` tous les produits modifiés.

Créez une table `PRODUITS_INSERT` qui contient trois champs : `REF_PRODUIT`, `UTILISATEUR` et `DATE_SYTEM`. Modifiez le déclencheur précédemment créé pour pouvoir insérer dans la table `PRODUITS_INSERT` tous les `REF_PRODUIT` avec les informations correspondantes sur l'utilisateur et date de création.

## Atelier 11.1 L'approche objet



### Questions

1. Quels sont les déclencheurs qui peuvent être compilés ?
  - A. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 CLIENTS.TELEPHONE%TYPE, a2 VARCHAR2(24));
  - B. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 NUMBER(1), a2 ROWID);
  - C. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 NUMBER(1));
  - D. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 LONG, a2 VARCHAR2(24));
  - E. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 NUMBER(1), a2 BOOLEAN);
  - F. CREATE OR REPLACE TYPE T1 IS OBJECT  
  ( a1 NUMBER(1), a2 VARCHAR2(24));

```
SQL> CREATE OR REPLACE TYPE T1 IS OBJECT
2  ( a1 NUMBER(1), a2 NUMBER(1),
3    CONSTRUCTOR FUNCTION T1( a1 NUMBER) RETURN SELF AS RESULT,
4    MEMBER PROCEDURE m1);
5  /
```

Type créé.

```
SQL> CREATE OR REPLACE TYPE BODY T1
2  AS
3    CONSTRUCTOR FUNCTION T1( a1 NUMBER) RETURN SELF AS RESULT IS
4    BEGIN
5      SELF.a1 := a1;SELF.a2 := a2; RETURN;
6    END T1;
7
8    MEMBER PROCEDURE m1 IS
9    BEGIN
10     dbms_output.put_line('Objet T1 a1 :'||a1||' a2 :'||a2);
11   END m1;
12 END;
13 /
```

Corps de type créé.

2. Pour l'objet T1 créé avec la syntaxe précédente, quels sont les blocs qui peuvent être compilés ?
  - A. DECLARE v1 T1; BEGIN v1.a1 := 1; v1.m1;END;
  - B. DECLARE v1 T1; BEGIN v1.m1; END;
  - C. DECLARE v1 T1 := T1(1,2); BEGIN v1.m1;END;

- D. `DECLARE v1 T1 := T1(1); BEGIN v1.m1;END;`
- E. `DECLARE v1 T1; BEGIN v1 := T1(1,2);v1.m1;END;`
- F. `DECLARE v1 T1; BEGIN v1 := T1(1);v1.m1;END;`
- G. `DECLARE v1 T1; BEGIN v1 := T1(1,2,3);v1.m1;END;`
3. Pour les mêmes choix que la question précédente, quels sont les blocs qui affichent la chaîne suivante 'Objet T1 a1 :1 a2 :1' ?
4. Pour les mêmes choix que la question précédente, quels sont les blocs qui affichent la chaîne suivante 'Objet T1 a1 :1 a2 :2' ?

## Exercice n°1 Les types d'objet

Créez un type 'DetCommObj' qui reprend à partir de la description de la table DETAILS\_COMMANDES les colonnes suivantes :

- REF\_PRODUIT
- PRIX\_UNITAIRE
- QUANTITE
- REMISE

Créez un type de tableau imbriqué 't\_DetCommObj' qui stocke des objets de type 'DetCommObj'.

Créez un objet 'CommandeObj' qui reprend la description complète de la table COMMANDES. L'objet doit contenir également un attribut de type tableau imbriqué 't\_DetCommObj'.

Créer un constructeur 'CommandeObj' pour permettre l'initialisation de l'objet uniquement avec les informations sur le numéro de commande, code client et numéro de l'employé. La date de la commande si elle n'est pas renseignée est la date du jour. Pour la date de l'envoi et les frais de port, il faut donner la possibilité de les renseigner si non les deux auront la valeur « **NULL** ».

Créez une méthode 'AjoutDeProduit' qui insère un détail de commande dans le tableau imbriqué.

Créez une méthode 'MontantCommande' qui renvoie le montant de la commande.

Créez une méthode 'EnvoisDeCommande' qui effectue la modification des unités en stock et des unités commandés, retirez des unités en stocks toutes les quantités de produits de la commande. S'il n'y a pas assez d'unités en stock, commandez la différence, en modifiant la valeur des unités commandées.

Pour finir, créez une méthode 'AfficheCommande' qui effectue l'affichage de l'ensemble des informations stockées dans l'objet.

## Exercice n°2 Le stockage de types d'objet

Pour pouvoir stocker des enregistrements de type 'CommandeObj' dans une table, vous devez créer la méthode « **MAP** » qui retourne la clé de la commande en occurrence NO\_COMMANDE.

Créez une table 'TCommandeObj' qui reprend entièrement la description de l'objet, n'oubliez pas de mentionner l'alias pour le tableau imbriqué et la clé primaire qui reprend la colonne NO\_COMMANDE.

Ecrivez un bloc PL/SQL qui permet de lire tous les enregistrements des tables COMMANDES et DETAIL\_COMMANDES et les insère dans la nouvelle table 'TCommandeObj'.



## Atelier 12.1 Les packages intégrés

### Exercice n°1 DBMS\_OUTPUT

Créez deux blocs PL/SQL et en utilisant les propriétés du paramètre `SERVEROUTPUT`, le premier bloc retrouve le nom du premier produit et l'insère le tampon interne et le deuxième bloc utilise ce nom pour augmenter de 10% les unités en stock.

### Exercice n°2 UTL\_FILE

Créez un répertoire 'UTL\_FILE\_REPERTOIRE', l'objet de correspondance avec un répertoire physique sur le disque du serveur.

Créez un fichier et stockez tous les enregistrements des clients.

Attention si vous travaillez avec ORACLE XE, vous devez d'abord exécuter les commandes suivantes :

```
'SQLPLUS / AS SYSDBA'
```

```
@%ORACLE_HOME%\RDBMS\Admin\utlfile.sql
```

### Exercice n°3 DBMS\_JOB

Placez les deux procédures de mise à jour du modèle étoile dans la file d'attente des travaux. La fréquence d'exécution de ces deux traitements doit être hebdomadaire.

### Exercice n°3 DBMS\_METADATA

Créez un script dynamique qui recense la structure du schéma stagiaire.