

**Ecrire des instructions
exécutables**

Objectifs

A la fin de ce chapitre, vous pourrez :

- **comprendre l'utilité de la section exécutable**
- **utiliser correctement les identificateurs**
- **écrire des instructions dans la section exécutable**
- **décrire les règles des blocs imbriqués**
- **exécuter et tester un bloc PL/SQL**
- **utiliser des conventions de codage**

Syntaxe et remarques relatives aux blocs PL/SQL

- Les instructions peuvent s'étendre sur plusieurs lignes
- Les unités lexicales se répartissent en plusieurs catégories :
 - délimiteurs
 - identificateurs
 - littéraux
 - commentaires

Identificateurs

- **Peuvent contenir jusqu'à 30 caractères**
- **Doivent commencer par une valeur alphabétique**
- **Peuvent contenir des valeurs numériques, des traits de soulignement, ainsi que les signes dollar et dièse**
- **Ne doivent pas contenir de caractères tels que les traits d'union et les barres obliques, ni d'espaces**
- **Ne doivent pas porter le même nom qu'une colonne de table de la base de données**
- **Ne doivent pas correspondre à des mots réservés**

Syntaxe et remarques relatives aux blocs PL/SQL

- **Littéraux**
 - Les littéraux de type caractère et date doivent être mis entre apostrophes.

```
v_name := 'Henderson';
```
 - Les nombres peuvent correspondre à des valeurs simples ou à une notation scientifique.
- Une barre oblique (/) permet d'exécuter le bloc PL/SQL dans un fichier script ou dans certains outils tels que *iSQL*PLUS*.


Commenter le code

- Faire précéder les commentaires monolignes de deux tirets (--)
- Placer les commentaires multilignes entre les symboles /* et */

Exemple :

```
DECLARE
...
    v_sal NUMBER (9,2);
BEGIN
    /* Compute the annual salary based on the
       monthly salary input from the user */
    v_sal := :g_monthly_sal * 12;
END;      -- This is the end of the block
```

Fonctions SQL en PL/SQL

- Disponibles dans les instructions procédurales :
 - fonctions monolignes numériques
 - fonctions monolignes de type caractère
 - fonctions de conversion de type de données
 - fonctions de date
 - fonctions d'horodatage
 - fonctions GREATEST et LEAST
 - fonctions diverses
 - Non disponibles dans les instructions procédurales :
 - fonction DECODE
 - fonctions de groupe
-  Identique en SQL

Fonctions SQL en PL/SQL : exemples

- Créer l'adresse postale d'une société

```
v_mailing_address := v_name || CHR(10) ||  
                    v_address || CHR(10) || v_state ||  
                    CHR(10) || v_zip;
```

- Convertir le nom d'un employé en minuscules

```
v_ename          := LOWER(v_ename) ;
```


Conversion de type de données

- Convertir des données en types de données comparables
- Mélanger les types de données peut provoquer des erreurs ou nuire aux performances
- Fonctions de conversion :
 - TO_CHAR
 - TO_DATE
 - TO_NUMBER

```
DECLARE
    v_date DATE := TO_DATE('12-JAN-2001', 'DD-MON-YYYY');
BEGIN
    . . .
```

Conversion de type de données

L'instruction suivante génère une erreur de compilation si la variable `v_date` est déclarée en tant que type de données `DATE` .

```
v_date := 'January 13, 2001' ;
```

Conversion de type de données

Pour corriger cette erreur, il faut utiliser la fonction de conversion TO_DATE .

```
v_date := TO_DATE ('January 13, 2001',  
                    'Month DD, YYYY');
```

Blocs imbriqués et portée des variables

- Les blocs PL/SQL peuvent être imbriqués partout où une instruction exécutable est permise
- Un bloc imbriqué devient une instruction
- Une section de traitement des exceptions peut contenir des blocs imbriqués
- La portée d'un identificateur correspond à la région d'un programme (bloc, sous-programme ou package) à partir de laquelle vous pouvez référencer l'identificateur

Blocs imbriqués et portée des variables

Exemple :

```
...  
  x  BINARY_INTEGER;  
BEGIN  
  ...  
  DECLARE  
    y  NUMBER;  
  BEGIN  
    y := x;  
  END;  
  ...  
END;
```

The diagram illustrates the scope of variables `x` and `y` in the provided code. A large rectangle, labeled "Portée de x" in red, encompasses the entire code block from the declaration of `x` to the final `END;`. A smaller rectangle, labeled "Portée de y" in red, is nested within the first, encompassing only the inner `BEGIN` block where `y` is declared and used. This visualizes that `x` is visible throughout the entire scope, while `y` is only visible within its own nested block.

Portée de l'identificateur

Un identificateur est visible dans les régions à partir desquelles vous pouvez le référencer sans devoir le qualifier :

- **un bloc peut effectuer une recherche dans le bloc englobant**
- **un bloc ne peut pas effectuer de recherche dans les blocs qu'il englobe**

Qualifier un identificateur

- Le qualificatif peut correspondre à l'étiquette d'un bloc englobant
- Qualifier un identificateur en utilisant l'étiquette du bloc en tant que préfixe

```
<<outer>>
  DECLARE
    birthdate DATE;
  BEGIN
    DECLARE
      birthdate DATE;
    BEGIN
      ...
      outer.birthdate :=
        TO_DATE('03-AUG-1976',
                'DD-MON-YYYY');
    END;
  ...
END;
```

Déterminer la portée d'une variable

Exercice

<<outer>>

DECLARE

 v_sal NUMBER(7,2) := 60000;

 v_comm NUMBER(7,2) := v_sal * 0.20;

 v_message VARCHAR2(255) := ' eligible for commission';

BEGIN

 DECLARE

 v_sal NUMBER(7,2) := 50000;

 v_comm NUMBER(7,2) := 0;

 v_total_comp NUMBER(7,2) := v_sal + v_comm;

 BEGIN

 v_message := 'CLERK not' || v_message;

 outer.v_comm := v_sal * 0.30;

 END;

 v_message := 'SALESMAN' || v_message;

END;

1



2



Opérateurs en PL/SQL

- Opérateur logique
- Opérateur arithmétique
- Opérateur de concaténation
- Parenthèses permettant de contrôler l'ordre des opérations



**Identiques
en SQL**

- Opérateur exponentiel (**)

Opérateurs en PL/SQL

Exemples :

- **Incrémenter le compteur d'une boucle.**

```
v_count      := v_count + 1;
```

- **Définir la valeur d'un indicateur booléen.**

```
v_equal      := (v_n1 = v_n2) ;
```

- **Vérifier si un numéro d'employé contient une valeur.**

```
v_valid      := (v_empno IS NOT NULL) ;
```

Remarques relatives à la programmation

Faciliter la maintenance du code en :

- **commentant le code**
- **développant une convention d'utilisation des majuscules et des minuscules**
- **développant des conventions d'appellation pour les identificateurs et les autres objets**
- **réalisant des indentations pour améliorer la clarté**

Indenter le code

Pour plus de clarté, indenter chaque niveau du code

Exemple :

```
BEGIN
  IF x=0 THEN
    y:=1;
  END IF;
END;
```

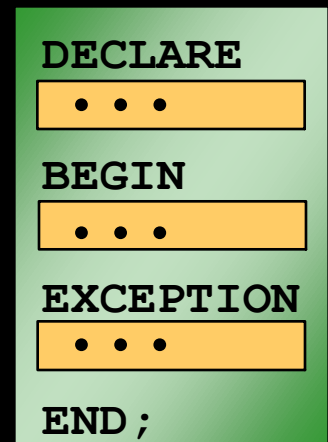
```
DECLARE
  v_deptno          NUMBER(4);
  v_location_id     NUMBER(4);
BEGIN
  SELECT  department_id,
          location_id
  INTO    v_deptno,
          v_location_id
  FROM    departments
  WHERE   department_name
          = 'Sales';

  ...
END;
/
```

Synthèse

Ce chapitre vous a appris les règles suivantes :

- **syntaxe et remarques relatives aux blocs PL/SQL**
- **utilisation correcte des identificateurs**
- **structure des blocs PL/SQL : imbrication des blocs et règles de portée**
- **programmation en PL/SQL :**
 - **fonctions**
 - **fonctions de conversion de type de données**
 - **opérateurs**
 - **conventions et remarques**



Présentation de l'exercice 2

Dans cet exercice, vous allez :

- **revoir les règles de portée et d'imbrication**
- **développer et tester des blocs PL/SQL**