Chapitre 3 : Langage procédural pour SQL (PL/SQL)

Responsable de module : Marwa HARZI

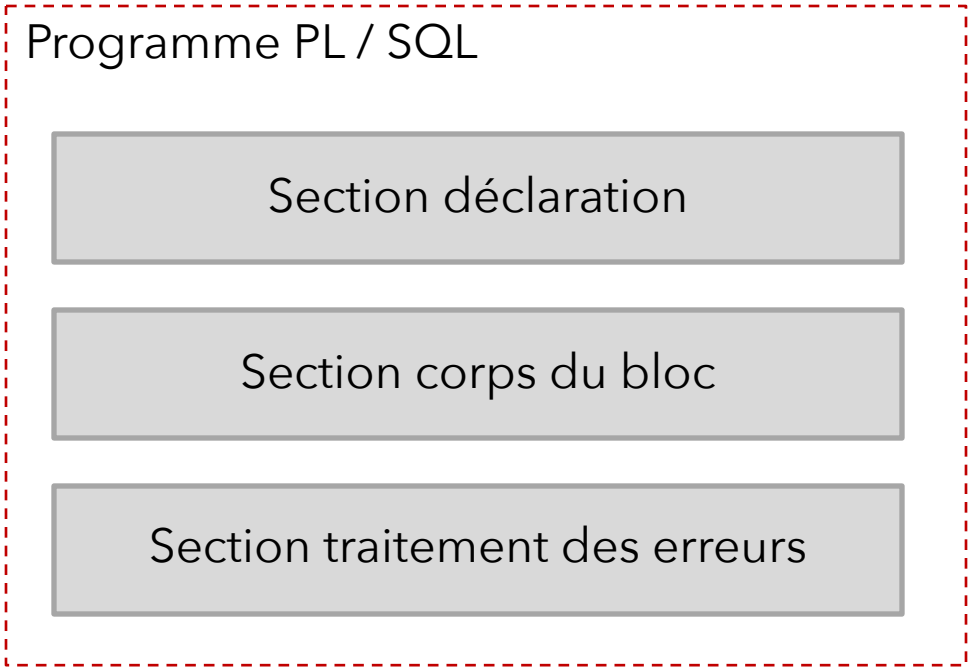
Répartition:

- Groupe LSI1 ---> Wejden Abdallah (wejden.abdallah@intervenants.efrei.net)
- Groupe LSI2 ---> Marwa HARZI (marwa.harzi@efrei.fr)

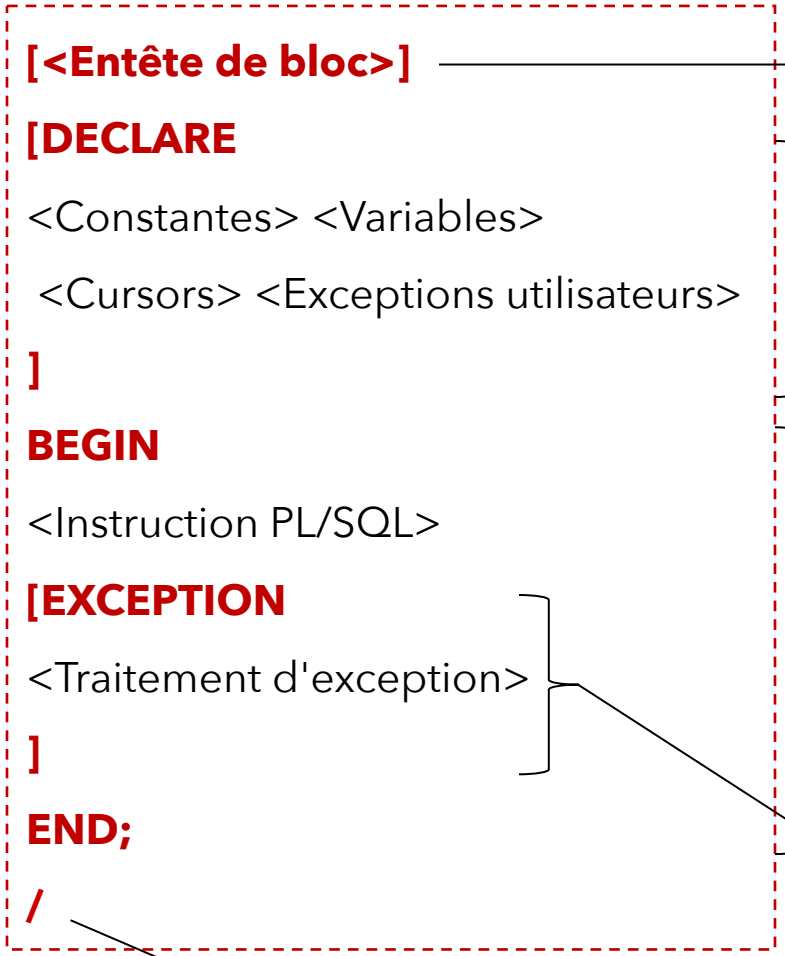
Définition :

Un programme PL/SQL est un ensemble de un ou plusieurs blocs.

Chaque bloc comporte trois sections :



Structure des blocs PL/SQL :



Entête de bloc: fonction, procédure ou package. Sinon, bloc anonyme

- Contient la description des structures et des variables utilisées dans le bloc
- Section facultative
- Commence par le mot clé **DECLARE**

- Contient les instructions du programme et éventuellement, à la fin, la section de traitement des erreurs
- Obligatoire
- Introduite par le mot clé **BEGIN**
- Se termine par le mot clé **END**

- Facultative
- Introduite par le mot clé **EXCEPTION**

A ajouter obligatoirement dans l'exécution d'un script

Exemples de PL/SQL :

Exemple n°1 :

```
SET SERVEROUTPUT ON
DECLARE
Mot VARCHAR2(10);
BEGIN
Mot := 'Salut';
DBMS_OUTPUT.PUT_LINE(Mot);
END;
/


Salut
```

Active l'affichage des messages sous SQLPLUS (mode sortie console).

dbms_output.putline permet d'écrire sur la console SQLPlus.

C'est la demande d'exécution du programme tapée.

Déclarations de variables :

1. Variable des types usuels :

```
DECLARE
        dateEmbauche date;  /*null */
        nom varchar2(80) := 'Benali';
        trouve boolean;   /* null */
        incrSalaire constant number(3,2) := 1.5;  /* constante */
. . .
BEGIN . . . END;
```

** initialisation implicite avec null, null est valeur par défaut, si pas d'initialisation (sauf si NOT NULL spécifié).

Déclarations de variables par référence aux données de la base :

2. Variable de type colonne :

```
DECLARE
        Salmax employee.salary%TYPE;
```

- *Salmax* est une variable PLSQL de même type que la colonne *salary* de la table *employee*.
- Utile pour garantir la compatibilité des affectations (e.g. dans la clause INTO).

3. Variable de type ligne :

```
DECLARE
        Recemployee employee%ROWTYPE;
```

- Recemployee est une structure de tuple de même type que le schéma de la relation *employee*.
- Usage avec SELECT * ...

Lecture des variables :

- En PL/SQL, comme dans tout langage de programmation, il est possible de lire une valeur:

```
ACCEPT nom_variable PROMPT 'entrez une valeur : '
```

- L'instruction suivante pour l'affectation de la valeur a la variable:

```
nom_variable = & nom_variable;
```

- On peut affecter a d'autres variables la valeur saisie:

```
nom_variable2 = & nom_variable;
```

Structures de contrôle : structure alternative et structure répétitives

1. Structure alternative :

IF *condition* THEN
instructions;
END IF;

IF *condition* THEN
instructions;
ELSE
instructions;
END IF;

IF *condition* THEN
instructions;
ELSIF *condition* THEN
instructions;
ELSE
instructions;
END IF;

Exemple:

```
DECLARE
    part_info participant%ROWTYPE;
BEGIN
    SELECT * INTO part_info FROM participant WHERE Nomparticipant='Jimmy';
    IF part_info.age>=50 THEN
        DBMS_OUTPUT.PUT_LINE ('Le participant '|| part_info.nomparticipant || ' a plus de 50 ans');
    ELSE DBMS_OUTPUT.PUT_LINE ('Le participant ' || part_info.nomparticipant|| ' a moins de 50 ans');
    END IF ;
END ;
```

Structures de contrôle : structure alternative et structure répétitives

2. Structure répétitives :

- Boucle WHILE :

```
WHILE <condition> LOOP
    <séquence d'instructions>;
END LOOP ;
```

Exemple :

```
DECLARE
   i number;
BEGIN
   i :=0;
   WHILE i<8 LOOP
       dbms_output.put_line(i);
        i:= i+1;
    END LOOP ;
END;
```

Structures de contrôle : structure alternative et structure répétitives

- Boucle FOR :

```
FOR <index> IN [reverse] <lower bound> … <upper bound> LOOP
        <séquence d'instructions> ;
END LOOP ;
```

Exemple :

```
BEGIN
  FOR i IN 4..7 LOOP
    dbms_output.put_line(i);
  END LOOP ;
END;
```

Structures de contrôle : structure alternative et structure répétitives

- Boucle LOOP :

```
LOOP
        <séquence d'instructions> ;
END LOOP ;
```

---> L'instruction LOOP exécute les instructions dans son corps et renvoie le contrôle au début de la boucle.

Exemple :

```
x := 0;
Loop
   x := x + 1; y := y - x;
   Exit When x > y;
End Loop;
```

- Généralement, le corps de la boucle contient au moins une instruction EXIT ou EXIT WHEN pour terminer la boucle. Sinon, la boucle devient une boucle infinie.

Procédures et fonctions

Syntaxe :

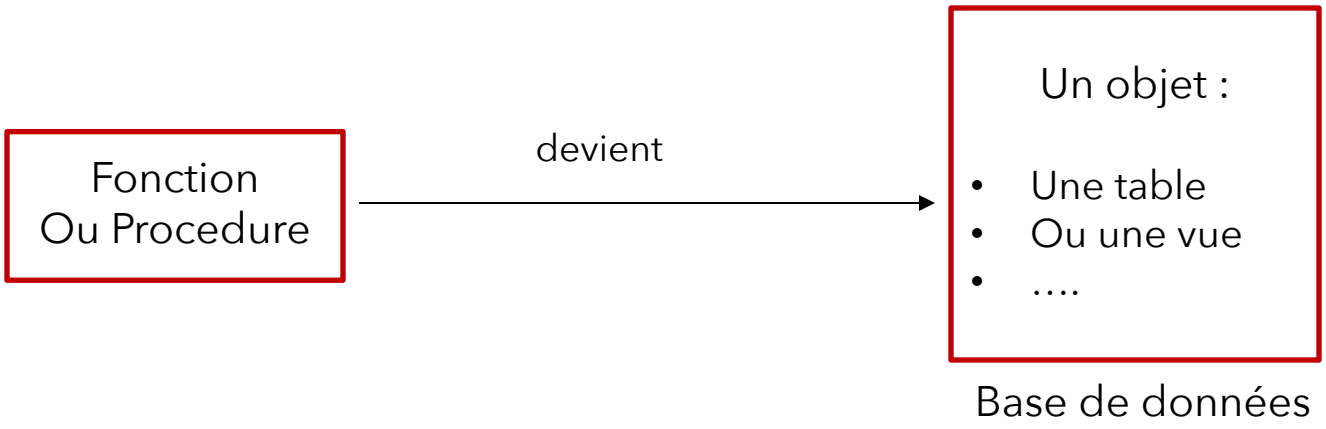
```
CREATE OR REPLACE PROCEDURE <nom>[ (liste de parametres) ] AS
        <zone de declaration de variables>
BEGIN
        <corps de la procedure>
        EXCEPTION
          <traitement des exceptions>
END;
```

- Pour les fonctions, le principe est le même, l'en-tête devient :

```
CREATE OR REPLACE FUNCTION <nom>[(parametres)]
RETURN <type du résultat>
```

---> L'instruction *RETURN <expression >* spécifie le résultat est renvoyé.

Procédures et fonctions



- L'appel d'une fonction ou d'un procédure peut se faire depuis n'importe quel bloc PL/SQL.

- Il y a deux façons de passer les parametres dans une procédure :
  1. *IN* (lecture seule),
  2. *OUT* (écriture seule).

*** Le mode IN est réservé aux parametres qui ne doivent pas être modifies par la , procédure. Le mode OUT, pour les parametres transmis en résultat.

Procédures et fonctions : Exemple

```
CREATE TABLE Tab (val1 INTEGER, val2 INTEGER)

CREATE OR REPLACE PROCEDURE essai(x IN NUMBER, z OUT NUMBER, y IN
NUMBER :=7) AS
BEGIN
        z:=x*y;
        INSERT INTO Tab VALUES(x,z);
END;
```

- Cette fonction est appelée dans le bloc suivant :

```
DECLARE
        a NUMBER;
        b NUMBER;
BEGIN
        a:=5;
        essai(2,b,a);
        dbms_output.put_line('essai(2,b,a) = ' ||b) ;
        essai(2,b);
        dbms_output.put_line('essai(2,b) = ' ||b) ;
END ;
```

Vérifiez l'insertion avec :

```
SELECT * FROM Tab;
```