



# Programmer avec PL/SQL

École Supérieure de la Statistique et de l'Analyse de  
l'Information (ESSAI)

Fatma CHAKER KHARRAT

1Support de cours PL/SQLF.CHAKER KHARRAT



★ **Avertissement** : cette partie du cours n'est qu'un survol du langage PL/SQL, utile pour écrire des procédures stockées simples

★ Elle laisse de côté de nombreuses fonctionnalités du langage

2Support de cours PL/SQLF.CHAKER KHARRAT



## 1. Introduction

---

1.1 Qu'est ce que PL/SQL

1.2 Fonctionnalités de PL/SQL

3Support de cours PL/SQLF.CHAKER KHARRAT



### 1.1 Qu'est ce que PL/SQL

---

**SQL** : Est un langage **ensembliste** et non procédural. Les traitements complexes sont parfois difficiles à écrire si on ne peut utiliser des variables et les structures de programmation comme les boucles et les alternatives

 **PL/SQL** : Est un langage **procédural**, qui intègre des ordres SQL de gestion de la base de données

4Support de cours PL/SQLF.CHAKER KHARRAT



### 1.2 Fonctionnalités de PL/SQL

---

★ PL/SQL est un langage algorithmique complet.  
PL/SQL ne comporte pas d'instructions du LDD (ALTER, CREATE, RENAME) ni les instructions de contrôle comme GRANT et REVOKE.

★ PL/SQL est un langage au même titre que SQL.

★ Tout comme SQL, PL/SQL peut être utilisé au sein des outils de la famille Oracle comme : Sql\*Plus, Sql\*Forms, Sql\*Pro, ....

5Support de cours PL/SQLF.CHAKER KHARRAT



### 1.2 Fonctionnalités de PL/SQL

---

★ Instructions SQL intégrées dans PL/SQL :

- La partie LID (select)
- La partie LMD (insert, update, delete)
- La partie gestion de transactions (commit, rollback, savepoint, ...)
- Les fonctions (to\_char, to\_date, upper, substr, round, ...)

★ + Une partie procédurale (IF, WHILE, ...)

★ Instructions spécifiques à PL/SQL :

- Définition de variables
- Traitements conditionnels
- Traitements répétitifs
- Traitement des curseurs
- Traitement des erreurs

6Support de cours PL/SQLF.CHAKER KHARRAT

## 2. Le Bloc PL/SQL

### 2.1 Structure d'un bloc PL/SQL

### 2.2 Exemple

7

Support de cours PL/SQL

F.CHAKER KHARRAT

## 2.1 Structure d'un bloc PL/SQL

### DECLARE

-- Déclaration de variables, constantes, exceptions

### BEGIN

-- Les instructions à exécuter (commandes exécutables, instructions SQL et PL/SQL, Possibilité de blocs fils (imbrication de blocs))

### EXCEPTION

-- Traitement des exceptions (gestion des erreurs)

Les blocs, comme les instructions se terminent par un « ; »

Seuls BEGIN et END sont obligatoires

### END ;

8

Support de cours PL/SQL

F.CHAKER KHARRAT

## 2.2 Exemple

Fichier de commandes : controle\_stock :

```
DECLARE
  qte_stock NUMBER(5)
BEGIN
  SELECT quantite INTO qte_stock
  FROM stock
  WHERE produit = :commande.produit;
  -- Contrôler et mettre à jour le stock
  IF qte_stock > 0
  THEN UPDATE stock
  SET quantite = quantite - 1
  WHERE produit = :commande.produit;
  INSERT INTO vente
  VALUES (:commande.produit || 'vendu',
          SYSDATE);
  ELSE
  INSERT INTO commande
  VALUES (:commande.produit || 'demande',
          SYSDATE);
  END IF;
  COMMIT;
END;
```

9

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3. Les Variables

### 3.1 Types de variables

### 3.2 Déclaration de variables

### 3.3 Initialisation des variables

### 3.4 Visibilité des variables

10

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3. Les variables

- ★ Identificateurs Oracle :
  - 30 caractères au plus
  - commence par une lettre
  - peut contenir lettres, chiffres, \_, \$ et #
- ★ Pas sensible à la casse
- ★ Portée habituelle des langages à blocs
- ★ Doivent être déclarées avant d'être utilisées

11

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.1 Types de variables

### PL/SQL Datatypes

#### Scalar Types

BINARY\_INTEGER  
DEC  
DECIMAL  
DOUBLE PRECISION  
FLOAT  
INT  
INTEGER  
NATURAL  
NUMBER  
NUMERIC  
POSITIVE  
REAL  
SMALLINT

CHAR  
CHARACTER  
LONG  
LONG RAW  
RAW  
ROWID  
STRING  
VARCHAR  
VARCHAR2

DATE

BOOLEAN

#### Composite Types

RECORD

TABLE

12

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.1 Types de variables

★ Les types habituels correspondants aux types SQL2 ou Oracle : **integer, varchar,...**

★ Types composites adaptés à la récupération des colonnes et lignes des tables SQL :  
**%TYPE, %ROWTYPE**

★ Type référence : **REF**

13

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.2 Déclaration de variables

Les variables locales se définissent dans la partie **DECLARE** d'un bloc PL/SQL.  
identificateur [CONSTANT] type [:= valeur];

★ **VARIABLES PL/SQL de type oracle :**

Exemple :

```
DECLARE
  nom          CHAR(15);
  numero       NUMBER;
  date_jour    DATE;
  salaire      NUMBER(7,2);
BEGIN
  .....
END;
```



Déclarations multiples interdites  
~~ij := valeur;~~

14

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.2 Déclaration de variables

★ **VARIABLES PL/SQL de type booléen :**

Exemple :

```
DECLARE
  reponse BOOLEAN := true;
BEGIN
  .....
END;
```

15

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.2 Déclaration de variables

★ **VARIABLES faisant référence au dictionnaire de données :**

☒ Variables reprenant le même type qu'une colonne dans la table :

Syntaxe :

Nom\_variable table.colonne%TYPE ;

Exemple :

```
DECLARE
  nom emp.ename%TYPE ;
BEGIN
  .....
END;
```

16

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.2 Déclaration de variables

☒ Variables reprenant la même structure qu'une ligne d'une table :

Syntaxe :

nom\_variable table%ROWTYPE ;

Exemple :

```
DECLARE
  enreg emp%ROWTYPE ;
BEGIN
  .....
END;
```



Chaque variable de la structure Enreg a le même nom et le même type que la colonne associée



17

Support de cours PL/SQL

F.CHAKER KHARRAT

## 3.2 Déclaration de variables

☒ Variables de même type qu'une variable précédemment définie :

Syntaxe :

nom\_variable2 nom\_variable1%TYPE ;

Exemple :

```
DECLARE
  commi NUMBER(7,2) ;
  salaire commi%TYPE ;
BEGIN
  .....
END;
```

18

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.3 Initialisation des variables

★ L'initialisation d'une variable peut se faire par :

- ✓ L'opérateur `:=` dans les sections **DECLARE**, **BEGIN** et **EXCEPTION**
- ✓ L'ordre **SELECT ... INTO ...** dans la section **BEGIN**
- ✓ Le traitement d'un **CURSEUR** dans la section **BEGIN**

★ Une variable est **visible** dans le bloc où elle a été déclarée et dans les blocs imbriqués si elle n'a pas été redéfinie.

19

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.3 Initialisation des variables

#### ① L'opérateur `:=`

```
DECLARE
    nom          CHAR(10)  := 'Ben Amor' ;
    salaire      NUMBER(7,2) := 1500 ;
    reponse      BOOLEAN   := TRUE ;
```

```
BEGIN
    ....
END;
```



Figurer l'affectation de valeur à une variable avec la clause **CONSTANT**.

```
DECLARE
    pi          CONSTANT NUMBER(7,2) := 3.14 ;
```

```
BEGIN
```

```
END ;
```

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.3 Initialisation des variables



Interdire les valeurs non renseignées avec la clause **NOT NULL**

```
DECLARE
    i          NUMBER NOT NULL := 1000 ;
BEGIN
    ....
END ;
```

21

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.3 Initialisation des variables

#### ② L'ordre **SELECT**

Syntaxe :

```
SELECT col1, col2 INTO var1, var2 FROM table [WHERE condition] ;
```



La clause **INTO** est **OBLIGATOIRE**.

Le **SELECT** doit obligatoirement ramener une ligne et une seule sinon erreur.

Pour traiter un ordre **SELECT** qui permet de ramener plusieurs lignes, on utilise un **curseur**.

22

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.3 Initialisation des variables

Exemple :

```
DECLARE
    nom_emp      CHAR(15) ;
    salaire      emp.sal%TYPE ;
    commission   emp.comm%TYPE ;
    nom_depart   CHAR(15) ;
BEGIN
    SELECT ename, sal, comm, dname
    INTO nom_emp, salaire, commission, nom_depart
    FROM emp, dept
    WHERE ename = 'Hammami' AND emp.deptno=dept.deptno ;
    ....
END ;
```

23

Support de cours PL/SQL

F.CHAKER KHARRAT

### 3.4 Visibilité des variables

```
DECLARE
    compte NUMBER(5) ;
    credit_max NUMBER(9,2) ;
BEGIN
    ....
    DECLARE
        compte CHAR(20) ;
        balance1 NUMBER(9,2) ;
    BEGIN
        ....
    END ;
    ....
    DECLARE
        balance2 NUMBER(9,2) ;
    BEGIN
        ....
    END ;
    ....
END ;
```

compte (NUMBER)  
credit\_max

compte (CHAR)  
balance1  
credit\_max

compte (NUMBER)  
balance2  
credit\_max

compte (NUMBER)  
credit\_max

24

Support de cours PL/SQL

F.CHAKER KHARRAT

## 4. Traitements Conditionnels

### 4.1 Définition et syntaxe

### 4.2 Exemple

25

Support de cours PL/SQL

F.CHAKER KHARRAT

## 4.1 Définition et syntaxe

```
IF condition THEN
  instructions;
END IF;
```

```
IF condition THEN
  instructions 1;
ELSE
  instructions 2;
END IF;
```

```
IF condition 1 THEN
  instructions 1;
ELSEIF condition 2 THEN
  instructions 2;
ELSEIF ...
...
ELSE
  instructions N;
END IF;
```

26

Support de cours PL/SQL

F.CHAKER KHARRAT

## 4.2 Exemple

```
DECLARE
  emploi CHAR(10)
  nom CHAR(15) := 'Hammami';
  mes CHAR(30);
BEGIN
  SELECT job INTO emploi FROM emp WHERE ename=nom;
  IF emploi IS NULL THEN mes := nom || 'n'a pas d'emploi';
  ELSIF emploi = 'Commercial'
  THEN UPDATE emp SET com = 1000 WHERE ename = nom;
  mes := nom || 'commission modifiée';
  ELSE UPDATE emp SET com = 0 WHERE ename = nom;
  mes := nom || 'pas de commission';
  END IF;
  INSERT INTO resultat VALUES (mes);
  COMMIT;
END;
```

27

Support de cours PL/SQL

F.CHAKER KHARRAT

## 5. Traitements répétitifs

### 5.1 La boucle de base (LOOP)

### 5.2 La boucle FOR

### 5.3 La boucle WHILE

28

Support de cours PL/SQL

F.CHAKER KHARRAT

## 5.1 La boucle de base

```
LOOP
  instructions;
EXIT [WHEN condition];
instructions;
END LOOP;
```

**Exemple :** Insérer les 10 premiers chiffres dans la table resultat.

```
DECLARE
  nbre NUMBER := 1;
BEGIN
  LOOP
  INSERT INTO resultat VALUES (nbre);
  nbre := nbre + 1;
  EXIT WHEN nbre > 10;
  END LOOP;
END;
```

29

Support de cours PL/SQL

F.CHAKER KHARRAT

## 5.2 La boucle FOR

```
FOR compteur IN [REVERSE] inf.. Sup LOOP
  instructions;
END LOOP;
```

compteur varie de Sup à inf avec un pas de -1

**Exemple :** Calcul de factorielle 9.

```
DECLARE
  fact NUMBER := 1;
BEGIN
  FOR i IN 1..9
  LOOP
    fact := fact * i;
  END LOOP;
  INSERT INTO resultat
  VALUES (fact, 'FACTORIELLE9');
END;
```

30

Support de cours PL/SQL

F.CHAKER KHARRAT

## 5.3 La boucle WHILE

Condition est une condition d'expressions au moyen des opérateurs <, >, =, !=, AND, OR, LIKE, ...

```
WHILE condition LOOP
instructions;
END LOOP;
```

**Exemple :** Reste de la division de 7324 par 9.

```
DECLARE
reste NUMBER := 7324 ;
BEGIN
WHILE reste >= 9
LOOP
reste := reste - 9 ;
END LOOP ;
INSERT INTO resultat
VALUES (reste,'reste division de 7324 par 9 ');
END;
```

31

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6. Les Curseurs

- 6.1 Définition
- 6.2 Les types de curseurs
- 6.3 Les étapes d'utilisation d'un curseur explicite
- 6.4 Les attributs d'un curseur
- 6.5 La boucle FOR pour un curseur

32

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.1 Définition

- Pour traiter une commande SQL, PL/SQL ouvre une zone de contexte pour exécuter la commande et stocker les informations.
- Le curseur permet de nommer cette zone de contexte, d'accéder aux informations et éventuellement de contrôler le traitement.
- Cette zone de contexte est une mémoire de taille fixe, utilisée par le noyau pour analyser et interpréter tout ordre SQL.
- Les statuts d'exécution de l'ordre se trouvent dans le curseur.

33

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.2 Les types de curseurs

#### *Le curseur explicite*

Il est créé et géré par l'utilisateur pour traiter un ordre Select qui ramène plusieurs lignes. Le traitement du select se fera ligne par ligne.

#### *Le curseur implicite*

Il est généré et géré par le noyau pour les autres commandes SQL.

34

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3 Les curseurs explicites

- \* Pour traiter les select qui renvoient plusieurs lignes
- \* Ils doivent être déclarés
- \* Le code doit les utiliser explicitement avec les ordres **OPEN**, **FETCH** et **CLOSE**
- \* Le plus souvent on les utilise dans une boucle dont on sort quand l'attribut **NOTFOUND** du curseur est vrai

35

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3 Les curseurs explicites

L'utilisation d'un curseur pour traiter un ordre Select ramenant plusieurs lignes, nécessite 4 étapes :

1. Déclaration du curseur
2. Ouverture du curseur
3. Traitement des lignes
4. Fermeture du curseur.

36

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.1 Déclaration d'un curseur explicite

#### Déclaration

Tout curseur explicite utilisé dans un bloc PL/SQL, doit être déclaré dans la section **DECLARE** du bloc en donnant :

- Son nom
- L'ordre select associé

#### Syntaxe

```
CURSOR nom_curseur IS ordre_select;
```

37

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.1 Déclaration d'un curseur explicite

#### Exemple :

```
DECLARE
    CURSOR dept_10 IS
    Select ename,Sal FROM emp
    WHERE deptno = 10
    ORDER BY sal;
BEGIN
    ... ;
    ... ;
END;
```

38

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.2 Ouverture d'un curseur explicite

#### Ouverture

Après avoir déclaré le curseur, il faut l'ouvrir pour faire exécuter l'ordre **SELECT**.

L'ouverture du curseur se fait dans la section **BEGIN** du bloc

#### Syntaxe

```
OPEN nom_curseur;
```

39

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.2 Ouverture d'un curseur explicite

#### Exemple :

```
DECLARE
    CURSOR dept_10 IS
    Select ename,sal FROM emp
    WHERE deptno = 10
    ORDER BY sal;
BEGIN
    ... ;
    OPEN dept_10;
    ... ;
END;
```

40

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.3 Traitement des lignes

#### Traitement des lignes

- Après l'exécution du **SELECT**, les lignes ramenées sont traitées **une par une**.
- La valeur de chaque colonne du **SELECT** doit être stockée dans une variable réceptrice

#### Syntaxe

```
Fetch nom_curseur INTO liste_variables;
```



Le Fetch ramène **une seule ligne** à la fois.

Pour traiter **n lignes**, il faut prévoir une **boucle**

41

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.3 Traitement des lignes

#### Exemple :

```
DECLARE
    CURSOR dept_10 IS
    Select ename,sal FROM emp
    WHERE deptno = 10
    ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    OPEN dept_10;
    LOOP
    FETCH dept_10 INTO nom, salaire;
    IF salaire >2500 THEN
        INSERT INTO resultat
        VALUES (nom,salaire);
    END IF;
    EXIT WHEN salaire = 5000;
    END LOOP;
END;
```

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.4 Fermeture du curseur

#### Fermeture

Après le traitement des lignes, on ferme le curseur pour libérer la place mémoire

#### Syntaxe

```
CLOSE nom_curseur;
```

43

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.3.4 Fermeture du curseur

#### Exemple :

```
DECLARE
    CURSOR dept_10 IS Select ename,sal FROM emp
    WHERE deptno = 10
    ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    OPEN dept_10;
    LOOP
    FETCH dept_10 INTO nom, salaire;
    IF salaire >2500 THEN INSERT INTO resultat VALUES (nom,salaire);
    END IF;
    EXIT WHEN salaire = 5000;
    END LOOP;
    CLOSE dept_10;
END;
```

44

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.4 Les attributs d'un curseur

Les attributs d'un curseur (implicite ou explicite) sont des indicateurs sur l'état d'un curseur

- \* **%FOUND** → Dernière ligne traitée
- \* **%NOTFOUND** → Dernière ligne traitée
- \* **%ISOPEN** → Ouverture d'un curseur
- \* **%ROWCOUNT** → nombre de lignes traitées

45

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.4 Les attributs d'un curseur

#### Attribut **%FOUND**

- \* **Type**: booléen
- \* **Syntaxe**:

- Curseur implicite : SQL%FOUND
- Curseur explicite : nom\_curseur%FOUND

Si sa valeur est vrai donc le dernier FETCH a ramené une ligne

46

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.4 Les attributs d'un curseur

#### Attribut **%FOUND** (Exemple)

```
DECLARE
    CURSOR dept_10 IS Select ename,sal FROM emp
    WHERE deptno = 10 ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    OPEN dept_10;
    FETCH dept_10 INTO nom, salaire;
    WHILE dept_10%FOUND
    LOOP
    IF salaire >2500 THEN
    INSERT INTO resultat VALUES (nom,salaire);
    END IF;
    FETCH dept_10 INTO nom, salaire;
    END LOOP;
    CLOSE dept_10;
END;
```

47

Support de cours PL/SQL

F.CHAKER KHARRAT

### 6.4 Les attributs d'un curseur

#### Attribut **%NOTFOUND**

- \* **Type**: booléen
- \* **Syntaxe**:

- Curseur implicite : SQL%NOTFOUND
- Curseur explicite : nom\_curseur%NOTFOUND

Si sa valeur est vrai donc le dernier FETCH n'a pas ramené une ligne

48

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %NOTFOUND (Exemple)

```
DECLARE
    CURSOR dept_10 IS Select ename,sal FROM emp
    WHERE deptno = 10 ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    OPEN dept_10;
    LOOP
    FETCH dept_10 INTO nom, salaire;
    EXIT WHEN DEPT_10%NOTFOUND;
    IF salaire >2500 THEN INSERT INTO resultat VALUES (nom,salaire);
    END IF;
    END LOOP;
    CLOSE dept_10;
END;
```

49

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ISOPEN

- \* **Type:** booléen
- \* **Syntaxe:**
  - **Curseur implicite :** SQL%ISOPEN  
toujours à **FALSE** car Oracle referme les curseurs après utilisation
  - **Curseur explicite :** nom\_curseur%ISOPEN  
Si sa valeur est vrai donc le curseur est ouvert

50

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ISOPEN (Exemple)

```
DECLARE
    CURSOR dept_10 IS
    Select ename,sal FROM emp WHERE deptno = 10 ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    IF NOT (dept_10%ISOPEN) THEN OPEN dept_10;
    END IF;
    LOOP
    FETCH dept_10 INTO nom, salaire;
    EXIT WHEN DEPT_10%NOTFOUND;
    IF salaire >2500 THEN
    INSERT INTO resultat VALUES (nom,salaire);
    END IF;
    END LOOP;
    CLOSE dept_10;
END;
```

51

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ROWCOUNT

- \* **Type:** numérique
- \* **Syntaxe:**
  - **Curseur explicite :** nom\_curseur%ROWCOUNT.  
Traduit la n<sup>ième</sup> ligne ramenée par le **FETCH**

52

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ROWCOUNT (Exemple)

```
DECLARE
    CURSOR dept_10 IS
    Select ename,sal FROM emp WHERE deptno = 10 ORDER BY sal;
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
BEGIN
    OPEN dept_10;
    LOOP
    FETCH dept_10 INTO nom, salaire;
    EXIT WHEN DEPT_10%NOTFOUND OR DEPT_10%ROWCOUNT>15 ;
    IF salaire >2500 THEN INSERT INTO resultat VALUES (nom,salaire);
    END IF;
    END LOOP;
    CLOSE dept_10;
END;
```

53

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ROWTYPE

Cet attribut permet la déclaration implicite d'une structure dont les éléments sont d'un type identique aux colonnes ramenées par le curseur.

#### **Syntaxe :**

- Dans la partie déclarative du bloc.  
**CURSOR nomcurseur IS ordre\_select;  
nomrecord nomcurseur%Rowtype;**
- Les éléments de la structure sont identifiés par :  
**nomrecord.nomcolonne**
- La structure est renseignée par le Fetch :  
**FETCH nomcurseur INTO nomrecord;**

54

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.4 Les attributs d'un curseur

### Attribut %ROWTYPE (Exemple)

```
DECLARE
    CURSOR c1 IS Select ename,sal+NVL(comm,0) saltot FROM emp
    WHERE deptno = 10 ORDER BY sal;
    c1_rec c1%ROWTYPE;
BEGIN
    OPEN c1;
    LOOP
    FETCH c1 INTO c1_rec;
    EXIT WHEN c1%NOTFOUND
    IF c1_rec.saltot >2500 THEN INSERT INTO resultat
    VALUES (c1_rec.ename, c1_rec.saltot);
    END IF;
    END LOOP;
    CLOSE c1;
END;
```

55

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.5 Boucle FOR pour un curseur

★ Elle simplifie la programmation car elle évite d'utiliser explicitement les instructions OPEN, FETCH et CLOSE

★ Elle déclare implicitement une variable de type « row » associée au curseur.

```
DECLARE
    CURSOR nom_curseur IS ordre_select;
BEGIN
    FOR nom_rec IN nom_curseur LOOP
        /*-----Traitements-----*/
    END LOOP;
END;
```

56

Support de cours PL/SQL

F.CHAKER KHARRAT

## 6.5 Boucle FOR pour un curseur

### Exemple :

```
DECLARE
    CURSOR dept_10 IS
    Select ename,sal FROM emp WHERE deptno = 10
    ORDER BY sal;
BEGIN
    FOR nom_rec IN dept_10;
    LOOP
    IF salaire >2500 THEN
        INSERT INTO resultat VALUES (nom_rec.ename, nom_rec.sal);
    END IF;
    END LOOP;
END;
```

Variable de type  
dept\_10%ROWTYPE

57

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7. Gestion des exceptions

### 7.1 Introduction

### 7.2 Les exceptions internes

### 7.3 Les exceptions utilisateurs

58

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.1 Introduction

- Le mécanisme de gestion d'erreurs dans PL/SQL est appelé **gestionnaire des exceptions**.
- Il permet au développeur de planifier sa gestion et d'abandonner ou de continuer le traitement en présence d'une erreur.
- Il faut affecter un traitement approprié aux erreurs apparues dans un bloc PL/SQL.

59

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.1 Introduction

On distingue **2 types d'erreurs ou d'exceptions** :

- **Erreur interne** : dans ce cas la main est rendue directement au système environnant.
- Anomalie **déterminée par l'utilisateur**.

### La solution :

- Donner un nom à l'erreur (si elle n'est pas déjà prédéfinie),
- Définir les anomalies utilisateurs, leur associer un nom,
- Définir le traitement à effectuer.

60

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.2 Les exceptions internes

- Une erreur **interne** est produite quand un bloc PL/SQL viole une règle d'Oracle ou dépasse une limite dépendant du système d'exploitation.
- Les erreurs Oracle générées par le noyau sont numérotées, or le gestionnaire des exceptions de PL/SQL, ne sait que gérer des erreurs nommées.
- Pour cela PL/SQL a redéfini quelques erreurs Oracle comme des exceptions. Ainsi, pour gérer d'autres erreurs Oracle, l'utilisateur doit utiliser le gestionnaire OTHERS ou EXCEPTION\_INIT pour nommer ces erreurs.

61

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.2 Les exceptions internes

Les exceptions fournies par Oracle sont regroupées dans ce tableau

Nom d'exception	Valeur SqlCode	Erreur Oracle
CURSOR_ALREADY_OPEN	-6511	ORA-06511
DUP_VAL_ON_INDEX	-1	ORA-00001
INVALID_CURSOR	-1001	ORA-01001
INVALID_NUMBER	-1722	ORA-01722
LOGIN_DENIED	-1017	ORA-01717
NO_DATA_FOUND	-1403	ORA-01413
NOT_LOGGED_ON	-1012	ORA-01012
PROGRAM_ERROR	-6501	ORA-06501
STORAGE_ERROR	-6500	ORA-06500
TIMEOUT_ON_RESOURCE	-51	ORA-00051
TOO_MANY_ROWS	-1422	ORA-01422
TRANSACTION_BACKED_OUT	-61	ORA-00061
VALUE_ERROR	-6502	ORA-06502
ZERO_DIVIDE	-1476	ORA-01476

62

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.2 Les exceptions internes

Exemple:

```
DECLARE
    wsal emp.sal%type;
BEGIN
    select sal into wsal from emp;
EXCEPTION

    WHEN TOO_MANY_ROWS then ... ;
    -- gérer erreur trop de lignes
    WHEN NO_DATA_FOUND then ... ;
    -- gérer erreur pas de ligne
    WHEN OTHERS then ... ;
    -- gérer toutes les autres erreurs
END;
```

63

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.3 Les exceptions utilisateurs

- PL/SQL permet à l'utilisateur de définir **ses propres exceptions**.
- La gestion des anomalies utilisateur peut se faire dans un bloc PL/SQL en effectuant les opérations suivantes :

- **Nommer l'anomalie** (type exception) dans la partie DECLARE du bloc.

**DECLARE**

**Nom\_ano EXCEPTION;**

- **Déterminer l'erreur** et passer la main au traitement approprié par la commande

**RAISE**

**BEGIN**

...

**IF (condition\_anomalie) THEN RAISE Nom\_ano ;**

- **Effectuer le traitement** défini dans la partie EXCEPTION du Bloc.

**EXCEPTION**

**WHEN (Nom\_ano) THEN (traitement);**

64

Support de cours PL/SQL

F.CHAKER KHARRAT

## 7.3 Les exceptions utilisateurs

Exemple :

```
DECLARE
    salaire numeric(8,2);
    salaire_trop_bas EXCEPTION;
BEGIN
    SELECT sal INTO salaire FROM emp WHERE matr=50;
    IF salaire < 300 THEN RAISE salaire_trop_bas;
    END IF;
    -- suite du bloc
EXCEPTION
    WHEN salaire_trop_bas THEN ... ;
    dbms_output.put_line('message');
END;
```

Permet l'affichage d'un message sur écran

65

Support de cours PL/SQL

F.CHAKER KHARRAT