# PL/SQL : Les Fonctions et Procédures Stockées

# Les Procédures Stockées

## ➢ DÉFINITION

Une **procédure stockée** en PL/SQL est un sous-programme qui :

- Est stocké dans la base de données sous un format compilé.
- Effectue une tâche ou un ensemble de traitements spécifiques.
- Peut recevoir des paramètres pour personnaliser son comportement.
- Ne retourne pas de valeur directement (contrairement à une fonction).
- Elle est appelée pour exécuter des instructions répétitives ou des traitements spécifiques, ce qui permet d'améliorer la réutilisabilité et la performance dans les systèmes de gestion de bases de données.

# Les Procédures Stockées

➢ **Syntaxe :**

```
Procedure nom_procedure [(liste_paramètres)] IS | AS
    [liste_variables_constantes]
BEGIN
    liste_instructions;
END [nom_procedure];
```

1. **Procedure** : Déclaration du mot-clé pour créer une procédure.
2. **nom_procedure** : Le nom de la procédure.
3. **liste_paramètres** (facultatif) : Les paramètres à passer à la procédure (avec leurs modes et leurs types de données).
4. **IS | AS** : Section pour déclarer des variables locales ou constantes utilisées dans le corps de la procédure.
5. **BEGIN ... END** : Le bloc principal qui contient les instructions de la procédure.

# Les Procédures Stockées

➢ **Exemple 1: Procedures Stockées Sans Paramétres**

```
CREATE  PROCEDURE augsalaire
IS
BEGIN
   UPDATE salaries SET salaire=salaire*1.2 WHERE numero_sal=1;
   DBMS_OUTPUT.PUT_LINE('Salaire augmenté');
END augsalaire;
BEGIN
augsalaire;
END;
```

# Les Procédures Stockées

➢ Exemple 1: Procedures Stockées Avec Paramétres

```
CREATE   PROCEDURE  augmenter_salaire(num_sal  IN int)  IS
BEGIN
     UPDATE salaries
     SET  salaire = salaire *1.1
     WHERE  numero_sal=num_sal;
    DBMS_OUTPUT.PUT_LINE('Salaire augmenté de 10%');
End augmenter_salaire;
BEGIN
    augmenter_salaire(2);
END
```

# Les Procédures Stockées

Les **modes de paramètres** définissent comment les valeurs sont transmises et manipulées dans une procédure ou une fonction. Les trois modes principaux sont **IN**, **OUT**, et **IN OUT**.

**Mode IN** Utilisé pour **passer une valeur en entrée** à la procédure ou la fonction.

- Le paramètre est **en lecture seule** à l'intérieur du bloc PL/SQL : sa valeur ne peut pas être modifiée.
- Par défaut, tous les paramètres sont en mode IN si aucun mode n'est spécifié.

```
CREATE OR REPLACE PROCEDURE afficher_nom(emp_id IN NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('ID de l''employé : ' || emp_id);
END afficher_nom;

-- Appel
BEGIN
    afficher_nom(101);
END;
```

# Les Procédures Stockées

**Mode OUT :** Utilisé pour **retourner une valeur** depuis la procédure vers l'appelant.

- Le paramètre est **en écriture seule** dans la procédure. Sa valeur est initialisée uniquement par le code de la procédure.
- À la fin de l'exécution, la valeur est accessible à l'appelant.

```
CREATE OR REPLACE PROCEDURE obtenir_salaire(emp_id IN NUMBER, sal OUT NUMBER) IS
BEGIN
    SELECT salaire INTO sal FROM employes WHERE id = emp_id;
END obtenir_salaire;
-- Appel
DECLARE
    salaire NUMBER;
BEGIN
    obtenir_salaire(101, salaire);
    DBMS_OUTPUT.PUT_LINE('Salaire : ' || salaire);
END;
```

# Les Procédures Stockées

**Mode IN OUT** Utilisé pour **passer une valeur d'entrée** à la procédure, qui peut ensuite être **modifiée** et **retournée** à l'appelant. Le paramètre agit à la fois comme une **entrée** (il transmet une valeur initiale) et une **sortie** (il retourne une valeur modifiée). Nécessite que la variable passée soit une variable, car elle doit pouvoir être modifiée.

```
CREATE OR REPLACE PROCEDURE ajuster_salaire(emp_id IN NUMBER, montant IN OUT NUMBER)
IS
BEGIN
          montant := montant * 1.1; -- Augmentation de 10%
          UPDATE employes
          SET salaire = montant
          WHERE id = emp_id;
END ajuster_salaire;

-- Appel
DECLARE       nouveau_salaire NUMBER := 1000;
BEGIN
    ajuster_salaire(101, nouveau_salaire);
    DBMS_OUTPUT.PUT_LINE('Nouveau salaire : ' || nouveau_salaire);   END;
```

# Les Fonctions Stockées

## ➢ DÉFINITION

- Une fonction stockée est un sous-programme PL/SQL :Stockée dans la base de données sous un format compilé.
- Qui retourne une **valeur unique** (via l'instruction RETURN).
- Peut être utilisée dans des expressions, des requêtes SQL ou des blocs PL/SQL.
- Peut recevoir des paramètres pour personnaliser son exécution.

**Différence clé avec une procédure :**

- Une procédure n'a pas de valeur de retour directe, tandis qu'une fonction retourne toujours une valeur via RETURN.

# Les Fonctions Stockées

- **Syntaxe**

```
CREATE [OR REPLACE] FUNCTION nom_fonction
( param1 [IN | OUT | IN OUT] type_donnees [DEFAULT valeur],
  param2 [IN | OUT | IN OUT] type_donnees [DEFAULT valeur])
RETURN type_de_retour IS | AS
    -- Déclarations locales (variables, constantes)
BEGIN
    -- Instructions
    RETURN valeur_retour; -- Valeur retournée par la fonction
END nom_fonction;
```

# Les Fonctions Stockées

- Syntaxe

```
CREATE OR REPLACE FUNCTION somme RETURN NUMBER IS
total number:=0;
BEGIN
        SELECT SUM(salaire) INTO total from SALARIES ;
        RETURN total;
END somme;

DECLARE
        som number;
BEGIN
        som:=somme;
        DBMS_OUTPUT.PUT_LINE(som);
END;
```

# Les Fonctions Stockées

- Exemple

```
CREATE OR REPLACE FUNCTION  service_salaire(num_service IN  INTEGER)
RETURN NUMBER IS
   Total_salaire  number ;
BEGIN
   SELECT SUM(salaire) INTO Total_salaire FROM  salaries.
   WHERE numero_serv=num_service;
   RETURN  Total_salaire;
END  service_salaire;
BEGIN
   DBMS_OUTPUT.PUT_LINE(service_salaire(10));
END;
```

# Les Fonctions Stockées

➢ **APPEL D'UNE FONCTION**

```
BEGIN
Nom_variable:=fonction(arguments,….);
END;
```

- **Dans un bloc PL/SQL**

```
DECLARE
    total NUMBER;
BEGIN
    total := salaire_total_service(10);
    DBMS_OUTPUT.PUT_LINE('Salaire total
: ' || total);
END;
```

- **Dans une requête SQL**

```
SELECT
salaire_total_service(10)
AS salaire_total
FROM dual;
```

# Les Fonctions Stockées

- **Gestion des exceptions :** Les procédures et les fonctions peuvent inclure des blocs pour gérer les erreurs :

```
CREATE OR REPLACE FUNCTION salaire_moyen RETURN NUMBER IS
    moyenne NUMBER;
BEGIN
    SELECT AVG(salaire) INTO moyenne FROM employes;
    RETURN moyenne;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0; -- Valeur par défaut en cas d'erreur
END salaire_moyen;
```

# Les Fonctions Stockées

- **Pour supprimer une procédure : DROP PROCEDURE nomProcedure;**
- **Pour compiler une procédure : ALTER PROCEDURE nomProcedure;**
- **Pour supprimer une fonction : DROP FUNCTION nomFunction;**
- **Pour compiler une fonction: ALTER FUNCTION nomfonction;**