SGBD - 2e

PL-SQL - Chapitre 7 - Les procédures et les fonctions

Daniel Schreurs

14 février 2022

Haute École de Province de Liège

Table des matières du chapitre i

Table des matières du chapitre ii

CREATE PROCEDURE

CREATE PROCEDURE

Création d'une procédure

```
1 CREATE PROCEDURE NomDeLaProcedure [(,.,)] IS
2 -- déclaration des variables locales ou types ou
      curseurs
3 BEGIN
4     ...
5     EXCEPTION
6     ...
7 END;
```

CREATE PROCEDURE

Chaque procédure ou fonction peut comprendre des paramètres.
Pour chaque paramètre, on doit spécifier :

- Son nom ;
- Son mode d'accessibilité (`IN`, `OUT` ou `IN OUT`) ;
- Son type (pas de précision ni de longueur) ;
- Éventuellement sa valeur par défaut.

CREATE PROCEDURE

Afficher l'employé

```
1  CREATE OR REPLACE PROCEDURE Afficher(NumSecu IN Employes
       .NumSecu%TYPE)
2  AS
3      UnEmploye Employes%ROWTYPE;
4  BEGIN
5      SELECT * INTO UnEmploye FROM Employes WHERE NumSecu
           = Afficher.NumSecu;
6      DBMS_OUTPUT.PUT_LINE('Nom : ' || UnEmploye.Nom);
7  EXCEPTION
8      WHEN NO_DATA_FOUND THEN
9          DBMS_OUTPUT.PUT_LINE('Aucun employé trouvé');
10     WHEN OTHERS THEN
11         DBMS_OUTPUT.PUT_LINE('ERREUR : ' || SQLCODE ||
               SQLERRM);
12 END Afficher;
```

CREATE PROCEDURE

Important

Une procédure ou une fonction peut également être déclarée localement dans une autre.

Compilation

Compilation

Compilation sous SQLPLUS

```
SQL> START ProcAfficher.sql    -- nom du fichier
Procédure créée.

--Ou

SQL> @ ProcAfficher.sql
Procédure créée.
```

Le code est stocké dans le dictionnaire de données. [1]

1. Les sources des objets (procédures, fonctions, packages) sont mémorisés dans la table SOURCE$ (propriétaire SYS).

Compilation

Compilation sous SQLPLUS :

- Lors de la compilation d'un objet, le moteur PL/SQL génère les messages d'erreurs dans la table `ERROR$`
- Sous SQLPLUS, la commande SHOW ERRORS permet de visualiser les erreurs de compilation.

Exécuter une procédure

Exécuter une procédure

Appeler une procédure

```
1 SET SERVEROUTPUT ON
2 BEGIN
3     Afficher('11111111');-- ici
4 EXCEPTION
5     WHEN OTHERS THEN
6         DBMS_OUTPUT.PUT_LINE('ERREUR : ' || SQLCODE ||
              SQLERRM);
7 END;
```

Création d’une fonction

Création d'une fonction

Sortir d'une fonction

```
1 CREATE FUNCTION NomDeLaFonction [(,.,)]
2   RETURN return_type IS
3 [déclaration des variables locales ou types ou curseurs]
4 BEGIN
5   RETURN ... ; -- sortie
6 EXCEPTION
7     ...
8   RETURN ... ; -- sortie
9 END
```

Création d'une fonction i

Recherche

```
1  CREATE OR REPLACE FUNCTION Rechercher(NumSecu IN
       Employes.NumSecu%TYPE)
2      RETURN Employes%ROWTYPE
3  AS
4      UnEmploye Employes%ROWTYPE;
5  BEGIN
6      SELECT * INTO UnEmploye FROM Employes WHERE NumSecu
           = Rechercher.NumSecu;
7      RETURN UnEmploye;
8  EXCEPTION
9      WHEN NO_DATA_FOUND THEN
10         DBMS_OUTPUT.PUT_LINE('Aucun employé trouvé');
11         RETURN NULL;
12     WHEN OTHERS THEN
```

Création d'une fonction ii

```
13          DBMS_OUTPUT.PUT_LINE('ERREUR : ' || SQLCODE ||
                SQLERRM);
14          RETURN NULL;
15  END Rechercher;
```

Exécuter une fonction

Exécuter une fonction

Recherche

```
SET SERVEROUTPUT ON
DECLARE
    UnEmploye Employes%ROWTYPE;
BEGIN
    UnEmploye := Rechercher('121212');
    DBMS_OUTPUT.PUT_LINE('Nom : ' || UnEmploye.Nom);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('ERREUR : ' || SQLCODE ||
            SQLERRM);
END;
```

Utiliser les paramètres avec *NOCOPY*

Utiliser les paramètres avec *NOCOPY*

- Par défaut, les paramètres `OUT` et `IN OUT` sont passés par valeur.[2]
- Pendant l'exécution de celui-ci, des variables temporaires sont créées pour contenir les données des paramètres de type `OUT`.
- Si le sous-programme termine normalement son exécution, les valeurs sont alors recopiées dans les paramètres.
- **Ces copies ralentissent fortement les performances et encombrent la mémoire.**
- Pour éviter ces inconvénients : `NOCOPY`.

2. Les valeurs sont donc copiées avant l'exécution du sous-programme.

RAISE_APPLICATION_ERROR

RAISE_APPLICATION_ERROR

Lancer une exception

```
1 BEGIN
2     RAISE_APPLICATION_ERROR(code_erreur, message);
3 END;
```

- Définie dans `DBMS_STANDARD`
- Plage de codes d'erreur de -20000 à -20999

RAISE_APPLICATION_ERROR i

Fonction retournant un code d'erreur personnalisé

```
1  CREATE OR REPLACE FUNCTION Rechercher(NumSecu IN
       Employes.NumSecu%TYPE)
2      RETURN Employes%ROWTYPE
3  AS
4      UnEmploye Employes%ROWTYPE;
5  BEGIN
6      SELECT *
7      INTO UnEmploye
8      FROM Employes
9      WHERE NumSecu = Rechercher.NumSecu;
10     RETURN UnEmploye;
11 EXCEPTION
12     WHEN NO_DATA_FOUND THEN
```

RAISE_APPLICATION_ERROR ii

```
13          RAISE_APPLICATION_ERROR(-20001, 'Aucun employé
                trouvé');
14      WHEN OTHERS THEN
15          RAISE;
16  END Rechercher;
```

RAISE_APPLICATION_ERROR

Appel de cette nouvelle fonction

```
1 DECLARE
2   UnEmploye Employes%ROWTYPE;
3 BEGIN
4   UnEmploye := Rechercher ('111111');
5   DBMS_OUTPUT.PUT_LINE ('Nom : ' || UnEmploye.Nom);
6 EXCEPTION
7   WHEN OTHERS THEN -- On peut tester le code de l'
        exception SQLCODE
8     DBMS_OUTPUT.PUT_LINE ('ERREUR : '|| SQLCODE || ' '
          || SQLERRM);
9 END;
```

Les notations nommées

Les notations nommées

Les notations nommées sont nécessaires lors de l'appel :

- Pour utiliser les valeurs par défaut des paramètres
- Pour spécifier les paramètres dans n'importe quel ordre

Les notations nommées

L'ordre des paramètres n'a plus d'importance

```
1  CREATE OR REPLACE PROCEDURE Search_Client(
2      p_NomClient IN,
3      p_AdresseClient IN)
4  BEGIN
5      null;
6  END;
7
8  -- utilisation
9  begin
10 Search_Client (p_NomClient=>'DELMAL',
11       p_AdresseClient => 'HUY');
12 end;
```

Les paramètres par défaut

Les paramètres par défaut

Lorsque les paramètres sont spécifiés dans le mode IN il est possible de leur affecter des valeurs par défaut

```
1  CREATE OR REPLACE PROCEDURE Select_Client
2    (Nom IN CHAR, CodePostal IN NUMBER := 4500) IS
3  BEGIN
4    null;
5  END;
6
7  -- utilisation
8  begin
9  Select_Client ('DELMAL'); -- appel valide
10 Select_Client ();--appel invalide
11 end;
```