TCH054 – SÉANCE 10
PROCÉDURES ET FONCTIONS STOCKÉES: INTRO À PL/SQL

Anis Boubaker, Ph.D.
Maître d'enseignement
École de Technologie Supérieure

PLAN DE LA SÉANCE

- Extensions procédurales:
  - Qu'est-ce c'est?
  - Procédures stockées
  - Fonctions stockée
  - Différences?
- Introduction au langage PL/SQL
  - Bloc de code procédural
  - Variables
  - Structures conditionnelles
  - Structures itératives: boucles for, while

3 LES EXTENSIONS PROCÉDURALES

ÉTS 4

SQL PAS SUFFISANT...

- Dans le langage SQL il n'y a pas de:
  - Structures conditionnelles:
    - Ex.: Pour un événement donné de la table Evenements, afficher la liste des employés **si** c'est un évènement qui récompense les employés ou la liste des clients **si** c'est un événement qui récompense les clients.
  - Structures itératives:
    - Ex.: Pour chacun des clients ayant reçu un produit X dans les 6 derniers mois, ajouter un appel de service à son vendeur régionnal.
  - Défaut d'impédance:
    - Différences au niveau des structures de données avec les langages de programmation
    - Différences de pouvoir expressif
    - Pas Turing-Complet (absence de récursivité)

ÉTS 5

EXTENSIONS PROCÉDURALES

- Les éditeurs de SGBD proposent des langages de programmation permettant de combler les lacunes évoquées:
  - Oracle: PL/SQL
  - Microsoft SQL Server: Transact-SQL
  - PostgreSQL: PL/pgSQL, PL/Tcl, PL/Perl, and PL/Python
  - MySQL : MySQL

ÉTS 6

PROCÉDURES / FONCTIONS STOCKÉES

- Sous-programmes procéduraux compilés et stockés dans la base de données

- Avantages:
  - Pas de déploiement nécessaire
  - Évite de communications réseau inutiles et coûteuses
  - Adaptés aux traitements de données
- Inconvénients:
  - Langage et librairies rudimentaires
  - Syntaxe archaïque (ex.: Le PL/SQL dérive du langage ADA)
    - Problèmes de maintenabilité du code
  - Spécifique au SGBD utilisé (non portable)

ÉTS 7

7 INTRODUCTION À PL/SQL

ÉTS 8

BLOC PL/SQL

- Tout code PL/SQL doit se trouver dans un **bloc PL/SQL**

Bloc PL/SQL

```
[ DECLARE
        déclaration;
        [déclaration;] …]
BEGIN
        instructions
[EXCEPTION
        énoncéException
        [énoncéException]…]
END;
/
```

ÉTS 9

BLOC PL/SQL - EXEMPLE

```
DECLARE
    la_quantite      NUMBER(10);
BEGIN
    SELECT quantite
    INTO la_quantite
    FROM Produit
    WHERE code_produit='23E9';

    IF la_quantite>10 THEN
        DBMS_OUTPUT.PUT_LINE('L''article #23E9 est en stock');
    ELSIF la_quantite>0 THEN
        DBMS_OUTPUT.PUT_LINE('L''article #23E9 est bientôt en
                              rupture de stock');
    ELSE
        DBMS_OUTPUT.PUT_LINE('L''article #23E9 est en
                              rupture de stock');
    END IF;
END;
/
```

PROCÉDURE PL/SQL

ÉTS 10

```
CREATE OR REPLACE PROCEDURE p_Etat_Stock
    (produit  Produit.code_produit%TYPE,
     seuil    NUMBER) IS
    -- Déclarations de variables
    qte_stock      NUMBER(10); --la quantité en stocke de produit
BEGIN
    --Interrogation de la base de données
    SELECT quantite
    INTO qte_stock
    FROM Produit
    WHERE code_produit = produit ;
    --Affichage de l'état du stock
    IF qte_stock>seuil THEN
        DBMS_OUTPUT.PUT_LINE('L''article ' || produit ||' est en
                              stock');
    ELSIF qte_stock>0 THEN
        DBMS_OUTPUT.PUT_LINE('L''article ' || produit ||' est
                              bientôt en rupture de stock');
    ELSE
        DBMS_OUTPUT.PUT_LINE('L''article ' || produit ||' est en
                              rupture de stock');
    END IF;
END;
```

ÉTS 11

PROCÉDURE PL/SQL

Note: Le mot clé DECLARE a disparu!

Procédure stockée

```
CREATE OR REPLACE PROCEDURE  nomProcédure
 [ (déclarationParam [,déclarationParam…]) ]
{ IS | AS }
    [déclarationVar;
    [déclarationVar;] …]
BEGIN
        instructions
[EXCEPTION
        énoncéException
        [énoncéException]…]
END;
/
```

Note: AS et IS sont synonymes…

ÉTS 12

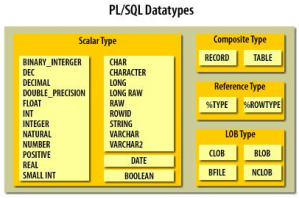
LES VARIABLES - DÉCLARATION

- Les variables doivent être déclarées.

nom_variable TYPE_DONNEE [:= val_initiale];

- Types de données

Source: RelationalDBDesign.com



ÉTS 13

LES VARIABLES - DÉCLARATION

- Déclarer une variable du même type qu'une colonne de la base de données:

```
nom_variable  TABLE.colonne%TYPE [:=val_initiale];
```

Exemple:

```
prix      Produit.cout%TYPE := 100;
```

- Déclarer une variable de type enregistrement d'une table avec ROWTYPE:

```
nom_variable  Nom_Table%ROWTYPE;
```

ÉTS 14

LES CONSTANTES

- La valeur d'une constante ne peut changer .
- Sa valeur doit être définie lors de la déclaration

```
nom_constante CONSTANT    TYPE_DONNEE  := val_initiale;
```

- Exemple:

```
taux     CONSTANT    NUMBER(3)  := 25;
```

ÉTS 15

LES VARIABLES - AFFECTATION

- **Utiliser l'opérateur :=**

```
une_var_numerique  := 10;
une_var_caractere  := 'Bonjour'
une_var_enregistrement.champs  := 10;
```

- **Affecter le résultat d'une requête avec SELECT… INTO**

```
nom       Produit.desc_produit%TYPE ;
prix      Produit.cout%TYPE ;

BEGIN
    SELECT desc_produit, cout
    INTO nom, prix
    FROM Produit
    WHERE code_produit = '05W30';
```

ÉTS 16

LES STRUCTURES CONDITIONNELLES

Structure conditionnelle

```
IF condition THEN
        instructions
[ELSIF condition THEN
        instructions ]…
[ELSE
        instructions]
END IF;
```

- Exemple:

```
IF val IN (2,5,7,11) THEN
        DBMS_OUTPUT.PUT_LINE('Nombre premier!');
ELSE
        DBMS_OUTPUT.PUT_LINE('Pas premier!');
END IF;
```

ÉTS 17

LES BOUCLES

- Plusieurs approches pour déclarer une structure itérative:
  - La boucle FOR
  - La boucle WHILE
  - La boucle LOOP
  - Les curseurs (CURSOR) – Vus la séance prochaine

ÉTS 18

BOUCLE FOR…

Boucle FOR

```
FOR indice IN [REVERSE] valeur_initiale .. valeur_finale LOOP
        instructions
END LOOP;
```

- Exemple:

```
FOR i IN 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE( i);
END LOOP;
```

ÉTS 19

BOUCLE WHILE...

Boucle WHILE

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

- **Exemple:**

```
i := 10;
WHILE i>0 LOOP
    DBMS_OUTPUT.PUT_LINE( i);
    i:=i-1;
END LOOP;
```

ÉTS 20

BOUCLE LOOP

Boucle WHILE

```
LOOP
        instructions
END LOOP;
```

- Pour sortir de la boucle utiliser:
  - EXIT: Sortie immédiate de la boucle
  - EXIT WHEN: Sortie lorsqu'une condition est vraie.
- **Exemple:**

```
i := 10;
LOOP
   DBMS_OUTPUT.PUT_LINE( i);
   i:=i-1;
   EXIT WHEN i=0;
END LOOP;
```

ÉTS 21

LES FONCTIONS

- Une fonction est un sous-programme qui retourne une valeur.
- Une fonction ne peut pas avoir d'effet de bord: interdit de modifier des données de la base.

Fonction

```
CREATE OR REPLACE FUNCTION  nomFonction
 [ (déclarationParam [,déclarationParam …]) ]
RETURN typeRetour
{ IS | AS }
    [déclarationVar;
     [déclarationVar;] …]
BEGIN
        instructions
[EXCEPTION
        énoncéException
        [énoncéException]…]
END;
/
```

ÉTS 22

LES FONCTIONS

```
SQL> CREATE FUNCTION fQuantitéEnStock
  2  (unNoArticle  Article.noArticle%TYPE)
  3  RETURN Article.quantitéEnStock%TYPE IS
  4
  5    uneQuantitéEnStock  Article.quantitéEnStock%TYPE;
  6  BEGIN
  7   SELECT  quantitéEnStock
  8   INTO   uneQuantitéEnStock
  9   FROM   Article
 10   WHERE   noArticle = unNoArticle;
 11   RETURN uneQuantitéEnStock;
 12  END fQuantitéEnStock;
 13
 14  /

Function created.
```

Source: Godin, 2003

ÉTS 23

PROCEDURE VS FONCTION

- Une fonction retourne une valeur, une procédure ne retourne (normalement pas de valeurs)
- Une fonction ne peut pas appeler une procédure. Une procédure peut appeler d'autres procédures/fonctions
- Une fonction peut-être utilisée dans une requête, pas une procédure.
- Une fonction ne peut modifier l'état du système (pas d'effet de bord), tandis que c'est permis pour une procédure.
  Par exemple, une fonction:
  - Ne peut pas modifier les données (INSERT, UPDATE, DELETE)
  - Ne peut pas agir sur les transactions (COMMIT, ROLLBACK, SAVEPOINT)

ÉTS 24

PROCHAINE SÉANCE

- Itérer sur le résultat d'une requête: les curseurs
- Passage de paramètres par valeur/référence
- Gestion des exceptions
- Normes de qualité