9 Procédures Stockées Paquetages

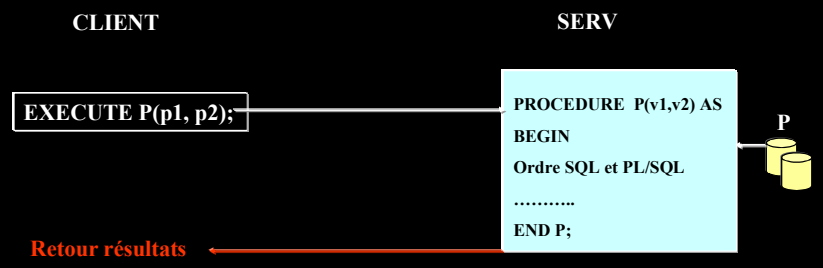
ORACLE

Procédures Stockées : Principe (1)

- Programme (PL/SQL) stocké dans la base
- Le programme client exécute ce programme en lui passant des paramètres (par valeur)
- Si le code est bon, le SGBD conserve le programme source (USER_SOURCE) et le programme compilé

ORACLE

Procédures Stockées : Principe (2)



ORACLE

Optimisation des procédures

- Recompilation automatique d'une procédure si un objet est modifé
- Recompilation manuelle possible

```
ALTER PROCEDURE <nom_procédure> COMPILE;
```

ORACLE

Avantages des procédures stockées

- **Vitesse** : programme compilé et optimisé
  - Une requête SQL normale est interprétée et optimisée à chaque exécution
- **Performance** : moins de transfert réseau
  - Plus de transfert de bloc de programme
  - Une procédure pour plusieurs utilisateurs
- **Abstraction** : augmentation du niveau d'abstraction des développeurs Client

Déclaration d'une procédure stockée

```
CREATE [OR REPLACE] PROCEDURE <nom_procédure>
[(variable1 type1, ..., variablen typen [OUT])] AS
...
-- déclarations des variables et
-- curseurs utilisées dans le corps de la procédure
BEGIN
....
-- instructions SQL ou PL/SQL
EXCEPTION
....
END;
/
```


Exemple 1 de procédure stockée inscription d'un étudiant

```
CREATE PROCEDURE inscription (pnom etudiant.nom%TYPE,
                              ... ,pdip diplome.idDip%TYPE)
AS
 CURSOR uv_ins IS SELECT c.iduv AS uv FROM composition c
                  WHERE c.idDip=pdip;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Début inscription de '||pnom);
  INSERT INTO etudiant VALUES(seqEtu.NEXTVAL,pnom,…,pdip);
  FOR uv_l IN uv_ins LOOP
        INSERT INTO inscrire VALUES(seqEtu.CURRVAL,uv_l.uv);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Transaction réussie');
  COMMIT;
EXCEPTION
....
END;
/
```


Exemple 1 : appel de la procédure

- A partir de sqlplus

```
ACCEPT vnom PROMPT 'Entrer le nom : '
……
EXECUTE inscription('&vnom',……, '&vdip');
```

- A partir de PL/SQL

```
inscription('&vnom',……, '&vdip');
```

Exemple 2 : avec retour de valeurs suppression d'un étudiant

```
CREATE PROCEDURE suppression (pidEtu NUMBER
                              retour OUT NUMBER) AS
 inscriptions EXCEPTION;
 PRAGMA EXCEPTION_INIT(inscriptions,-2292);
 vnom etudiant.nom%TYPE;
BEGIN
 SELECT nom INTO vnom FROM etudiant WHERE idEtu=pidEtu;
 DELETE FROM etudiant WHERE idEtu=pidEtu;
 DBMS_OUTPUT.PUT_LINE('Etudiant '||vnom||' supprimé');
 COMMIT;
 retour:=0;
```

../..

Exemple 2 : avec retour de valeurs suppression d'un étudiant (suite)

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE(
                'Etudiant' || TO_CHAR(pidEtu) || 'inconnu);
    retour:=1;
  WHEN inscriptions THEN
    DBMS_OUTPUT.PUT_LINE('Encore des inscriptions');
    retour:=2;
    ..........
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
    retour:=9;
END;
/
```


Exemple 2 : appel avec retour

```
VARIABLE ret NUMBER
ACCEPT vnom PROMPT 'Entrer le nom : '
..........
EXECUTE inscription('&vnom',....., '&vdip',:ret);
PRINT ret
```


Les Fonctions stockées

- Comme une procédure mais qui ne retourne qu'un seul résultat
- Même structure d'ensemble qu'une procédure
- Utilisation du mot clé RETURN pour retourner le résultat
- Appel possible à partir de :
  - Une requête SQL normale
  - Un programme PL/SQL
  - Une procédure stockée ou une autre fonction stockée
  - Un programme externe

Déclaration d'une fonction stockée

```
CREATE [OR REPLACE] FUNCTION nom_fonction
[(paramètre1 type1, ..... , paramètren typen)]
RETURN type_résultat IS
-- déclarations de variables,curseurs et exceptions
BEGIN
 -- instructions PL et SQL

 RETURN(variable);
END;
/
```

1 ou plusieurs RETURN

Exemple 1 de fonction stockée

```
CREATE OR REPLACE FUNCTION moy_points_marques
                                (eqj joueur.ideq%TYPE)
RETURN NUMBER IS
      moyenne_points_marques NUMBER(4,2);
BEGIN
      SELECT AVG(totalpoints) INTO moyenne_points_marques
      FROM joueur WHERE ideq=eqj;

      RETURN(moyenne_points_marques);
END;
/
```


Utilisation d'une fonction

- A partir d'une requête SQL

```
SELECT moy_points_marques('e1') FROM dual;
```

```
SELECT nomjoueur FROM joueur WHERE
totalpoints > moy_points_marques('e1');
```

- A partir d'une procédure ou fonction

```
BEGIN
.....
IF moy_points_marques(equipe) > 20 THEN .....

END;
```


Exemple 2 de fonction stockée

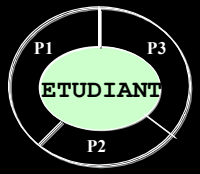
```
CREATE OR REPLACE FUNCTION bon_client
                (pidclient NUMBER, pchiffre NUMBER)
RETURN BOOLEAN IS
    total_chiffre NUMBER;
BEGIN
    SELECT SUM(qte*prix_unit) INTO total_chiffre
    FROM commande WHERE idclient=pidclient;

    IF total_chiffre > pchiffre THEN
       RETURN(TRUE);
    ELSE RETURN(FALSE);
    END IF;
END;
```

```
BEGIN
.....
IF bon_client(client,10000) THEN .....
.....
```

Les Paquetages

- Ensemble de programmes ayant un lien logique entre eux
- Exemple : package étudiant qui peut regrouper tous les programmes écrits sur les étudiants
- Début de l'approche objet avec les méthodes associées à une classe (MEMBER en Objet-Relationnel)



Structure d'un paquetage

- Partie 'visible' ou spécification
  - Interface accessible au programme appelant
  - Ne contient que les déclarations des procédures ou fonctions publiques
  - Variables globales et session
  - Curseurs globaux
- Partie body
  - Corps des procédures ou des fonctions citées dans la partie spécification
  - Nouvelles procédures ou fonctions privées accessibles uniquement par des procédures ou fonctions du paquetage

Déclaration d'un paquetage partie spécification

```
-- Partie Spécification

CREATE [OR REPLACE] PACKAGE nom_package AS
    Procedure Procédure1(liste des paramètres);
    ...........
    Function Fonction1(liste des paramètres);
    ...........
    Variable_globale1 type1;
    ...........
    CURSOR Curseur_global1 IS ...........
    ...........
END nom_package;
/
```


Déclaration d'un paquetage partie body

```
-- Partie body
CREATE [OR REPLACE] PACKAGE BODY nom_package AS
   Procedure Procédure1(liste des paramètres)IS
      ...........
      BEGIN
      ...........
   END Procédure1;
   Function Fonction1(liste des paramètres)
   RETURN type IS
      ...........
      BEGIN
      ...........
      RETURN(.....);
   END Fonction1;
END nom_package;
/
```

Exemple : package ‘étudiant’ (1)

```
CREATE PACKAGE etudiant AS
   -- Procédure publique inscription
   PROCEDURE inscription (pnom etudiant.nom%TYPE,
                          ... ,pdip diplome.idDip%TYPE);
   -- Procédure publique suppression
   PROCEDURE suppression(pidetu NUMBER);
END nom_package;
/
CREATE PACKAGE BODY etudiant AS
    PROCEDURE inscription (pnom etudiant.nom%TYPE,
                           ... ,pdip diplome.idDip%TYPE) IS
    CURSOR uv_ins IS SELECT c.iduv AS uv FROM
          composition c WHERE c.idDip=pdip;
BEGIN
```


Exemple : package ‘étudiant’ (2)

```
  INSERT INTO etudiant VALUES(seqEtu.NEXTVAL,pnom,…,pdip);
  FOR uv_l IN uv_ins LOOP
    INSERT INTO inscrire VALUES(seqEtu.CURRVAL,uv_l.uv);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Transaction réussie');
  COMMIT;
EXCEPTION   ....
END inscription;
-- fonction privée inscrit_uv
FUNCTION inscrit_uv(pidetu NUMBER) RETURN BOOLEAN IS
  nbre_ins NUMBER;
  BEGIN
    SELECT COUNT(*) INTO nbre_ins FROM inscrire WHERE
    Idetu=pidetu;
    IF nbre_ins>0 THEN RETURN(TRUE) ELSE RETURN(FALSE)
    END IF;
END inscrit_uv;
```


Exemple : package ‘étudiant’ (3)

```
 PROCEDURE suppression (pidetu NUMBER) AS
 BEGIN
   IF inscrit_uv(pidetu) THEN
    DBMS_OUTPUT.PUT_LINE('Cet étudiant est inscrit à des UV');
    DBMS_OUTPUT.PUT_LINE('Impossible de le supprimer');
   ELSE
    DELETE FROM etudiant WHERE idetu=pidetu;
    DBMS_OUTPUT.PUT_LINE('Etudiant supprimé');
    COMMIT;
   END IF;
 END suppression;
END etudiant;
/
```


Appel d’un programme d’un package

- A partir des SQL

```
ACCEPT vnom PROMPT 'Entrer le nom : '
……
EXECUTE etudiant.inscription('&vnom',……, '&vdip');
```

- A partir d’un autre package

```
etudiant.inscription(nom,……, dip);
```

- Uniquement les programmes PUBLICS