Cours Bases de données 2ème année IUT

Cours 3 : PL/SQL : ou comment faire plus avec ORACLE
1ère partie

Anne Vilnat
http://www.limsi.fr/Individu/anne/cours

Plan

Pourquoi PL/SQL?

L'utilisation de PL/SQL est indiquée :

- Pour effectuer des traitements de données qui impliquent des transactions complexes (plusieurs requêtes et manipulations de données liées).
- Pour effectuer un contrôle d'intégrité des données (dans des triggers par exemple).
- Pour stocker dans la base de données les opérations fréquentes. (Packages et procédures stockées.)
- Pour minimiser le temps d'interaction entre la base de données et la portion interface.
- Pour minimiser l'impact des mises à jour sur une application avec un grand nombre de postes clients.

Structure d'un programme

Un programme ou une procédure PL/SQL est constitué d'un ou plusieurs blocs.

Description d'un bloc

3 sections :

- Déclaration des structures et des variables utilisées dans le bloc (facultative)
- Corps qui contient les instructions (obligatoire)
- Traitement des erreurs : pour gérer les erreurs. (facultative)

Syntaxe d'un bloc

Syntaxe d'un bloc anonyme

DECLARE
– Partie déclaration
....
BEGIN
– Partie code
....
EXCEPTION
– Partie gestion des exceptions levées dans le code.
....
END;

Exemple d'un bloc : la déclaration

On veut rechercher combien de réalisateurs ont joué dans leurs propres films, et l'afficher *proprement*

```
DECLARE
    nbRealAct NUMBER(5);
    singulierException EXCEPTION;
```

Exemple d'un bloc : le corps

```
DECLARE
    nbRealAct NUMBER(5);
    singulierException EXCEPTION;
BEGIN
  SELECT COUNT(distinct A.numIndividu) INTO nbRealAct
    FROM Film F, Acteur A
   WHERE A.numIndividu = realisateur
      AND F.numFilm=A.numFilm;
  IF nbRealAct = 1 THEN RAISE singulierException;
  END IF;
  DBMS_OUTPUT.PUT_LINE(nbRealAct||' réalisateurs ont
joué dans leur film');
```

Exemple d'un bloc

```
DECLARE
    nbRealAct NUMBER(5);
    singulierException EXCEPTION;
BEGIN
    SELECT COUNT(distinct A.numIndividu) INTO nbRealAct
        FROM Film F, Acteur A
        WHERE A.numIndividu = realisateur
          AND F.numFilm=A.numFilm;
    IF nbRealAct = 1 THEN RAISE singulierException; END IF;
    DBMS_OUTPUT.PUT_LINE(nbRealAct||' réalisateurs ont joué dans
leur film');
 EXCEPTION
     WHEN singulierException THEN
     DBMS_OUTPUT.PUT_LINE('Un seul réalisateur a joué
dans son film');
END;
```

Les Variables

Déclaration d'une variable

nom variable [CONSTANT] type [[NOT NULL] := expression]

CONSTANT : le bloc ne changera pas la valeur
expression : pour initialiser la variable
NOT NULL impose une initialisation
type : le type de la variable
simple : nombre, chaîne,...
composé : tableau, enregistrement,,...
faisant référence à une autre type

Les types de variables (1)

Types simples

- numériques : NUMBER[(e,d)], BINARY_INTEGER, BINARY_FLOAT
- alphanumériques : CHAR, VARCHAR2, LONG, ...
- date : DATE, TIMESTAMP (heures),...
- booléen : BOOLEAN
- une adresse d'une ligne dans une table : ROWID

Les types de variables (1)

Types simples

- numériques : NUMBER[(e,d)], BINARY_INTEGER, BINARY_FLOAT
- alphanumériques : CHAR, VARCHAR2, LONG, ...
- date : DATE, TIMESTAMP (heures),...
- booléen : BOOLEAN
- une adresse d'une ligne dans une table : ROWID

Exemple

```
nbActeurs NUMBER(5);
nomActeur VARCHAR2(20);
```

Les types de variables (2)

Types dérivés

- référence à une colonne ou à un type existant

  nomVariable nomTable.nomColonne%TYPE

  nomVariable nomVariableRef%TYPE

- référence à une ligne d'une table ou d'un curseur

  nomVariable nomTable%ROWTYPE

  nomVariable nomCurseur%ROWTYPE

Les types de variables (2)

Types dérivés

- référence à une colonne ou à un type existant
  nomVariable nomTable.nomColonne%TYPE
  nomVariable nomVariableRef%TYPE
- référence à une ligne d'une table ou d'un curseur
  nomVariable nomTable%ROWTYPE
  nomVariable nomCurseur%ROWTYPE

Exemple

Quel est le format de nomIndividu dans la table Individu?
Quels sont les formats des attributs de la table Film?

nomActeur Individu.NomIndividu%TYPE;
film Film%ROWTYPE;

Les types de variables (3)

Types définis

- sous-type d'un type existant, avec SUBTYPE
- type complexe : TABLE, RECORD, VARRAY, avec TYPE

Les types de variables (3)

Types définis

- sous-type d'un type existant, avec SUBTYPE
- type complexe : TABLE, RECORD, VARRAY, avec TYPE

Exemples

- les sous-types :
  SUBTYPE chaineCourte IS VARCHAR2(20);
  SUBTYPE nomPersonne IS Individu.NomIndividu%TYPE;
  SUBTYPE film IS Film%ROWTYPE;
- les types complexes :
  TYPE tableNombres IS TABLE OF NUMBER;
  TYPE tableFilms IS TABLE OF film;

Les instructions : l'affectation

Affectation

- simple : nomVariable := valeur
- par un SELECT ... INTO quand le SELECT ne retourne qu'une ligne

Syntaxe

```
SELECT listeAttributs INTO listeVariables
        FROM...
SELECT listeAttributs INTO nomStructure
        FROM...
Exemple :
SELECT * INTO film
        FROM Film
        WHERE NumFilm=1;
```

Les instructions : la condition

Syntaxe

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

Les instructions : la boucle

Syntaxe

```
LOOP
        instructions;
        IF condition THEN EXIT;
        instructions;
END LOOP;
```

ou :

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

Les instructions : la boucle FOR

Syntaxe

FOR variableBoucle IN [REVERSE] borneInf .. borneSup
LOOP
    instructions;
END LOOP;

avec ;

- variableBoucle : variable locale à la boucle, non déclarée
- borneInf et borneSup : variables déclarées et initialisées ou constantes
- pas de boucle fixe à 1, éventuellement en décrement si REVERSE

Les instructions : la boucle WHILE

Syntaxe

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

Les instructions sont répétées tant que la condition est vraie...

Les curseurs: définition

Quand une instruction `SELECT` peut renvoyer plusieurs lignes, il faut utiliser un *curseur*.
Un curseur est une structure qui contient les informations suivantes :

- le texte source de l'instruction SQL,
- le texte *compilé* de l'instruction SQL,
- un tampon qui contiendra une ligne du résultat,
- le statut du curseur (`cursor status`),
- des informations de travail,
- des informations de contrôle.

Les curseurs : usage

Déclaration

CURSOR nomCurseur IS <texte requete>;

Ouverture

OPEN nomCurseur ;

Utilisation

FETCH nomCurseur INTO <liste variable> | <nomStructure> ;

Fermeture

CLOSE nomCurseur ;

Les curseurs : exemple

On veut afficher les titres des comédies.
La déclaration :

```
DECLARE
monNumero ens2004.film.numfilm%type;
monTitre ens2004.film.Titre%type;
Cursor monCurseur IS
        SELECT f.numfilm, titre
        FROM ens2004.film f, ens2004.genrefilm g
        WHERE f.numfilm=g.numfilm
        AND codegenre='CO';
```

Les curseurs : exemple suite

```
DECLARE
monNumero ens2004.film.numfilm%type;
monTitre ens2004.film.Titre%type;
Cursor monCurseur IS
        SELECT f.numfilm, titre
        FROM ens2004.film f, ens2004.genrefilm g
        WHERE f.numfilm=g.numfilm
        AND codegenre='CO';
BEGIN
OPEN monCurseur ;
LOOP
        FETCH monCurseur INTO monNumero, monTitre;
        EXIT WHEN monCurseur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(' Son Numéro : ' ||
        monNumero||' SonTitre : ' ||monTitre) ;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Voici le nombre total de comédies : '
        || monCurseur%rowCount);
CLOSE MonCurseur;
END;
```

Les curseurs : exemple suite, autre formulation

```
DECLARE
cpt NUMBER:=0;
Cursor monCurseur IS
        SELECT f.numfilm, titre
        FROM ens2004.film f, ens2004.genrefilm g
        WHERE f.numfilm=g.numfilm
        AND codegenre='CO';
BEGIN
FOR ligneCurseur IN monCurseur ;
LOOP
        DBMS_OUTPUT.PUT_LINE(' Son Numéro : ' ||
         ligneCurseur.numFilm || '
        cpt:=cpt+1;
        SonTitre : ' || ligneCurseur.titre) ;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Voici le total des comédies : '||cpt)
END;
```