# Cours Bases de données 2ème année IUT

## Cours 4 : PL/SQL : ou comment faire plus avec ORACLE 2ème partie

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

# Plan

# Rappel : 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)**

# Rappel : 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

```
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;
```

# Définition

## Définition d'une exception

Un exception est une erreur ou un avertissement, prédéfini par Oracle ou défini par le programmeur. 4 catégories d'exceptions :

- nommées prédéfinies par Oracle : Oracle les déclenche, l'utilisateur peut les récupérer (WHEN nomException)
- nommées définies par l'utilisateur : l'utilisateur les déclare, les lève dans un programme PL/SQL, et peut les récupérer (WHEN nom...)
- anonymes prédéfinies par Oracle : Oracle les déclenche sans les nommer, l'utilisateur peut quand même les récupérer (WHEN OTHERS ...)
- anonymes définies par l'utilisateur : idem, mais c'est l'utilisateur qui les lève avec RAISE APPLICATION_ERROR

# Exceptions nommées prédéfinies

Beaucoup d'erreurs prédéfinies par Oracle, dont :

- CURSOR_ALREADY_OPEN : tentative d'ouvrir un curseur déjà ouvert
- LOGON_DENIED : mauvais login/password lors de la connexion
- ROWTYPE_MISMATCH : types de paramètre incompatibles
- TOO_MANY_ROWS : trop de lignes renvoyées par un SELECT... INTO
- ...

# Exceptions nommées définies par l'utilisateur

```
DECLARE
    nomException EXCEPTION;
BEGIN
    ...
    RAISE nomException;
    ...
EXCEPTION
     WHEN nomException THEN traitement1
     [WHEN OTHERS THEN traitementN]
END;
```

# Exceptions nommées définies par l'utilisateur : exemple

```
DECLARE
   cpt NUMBER := 0;     monException EXCEPTION;
BEGIN
   ...
   IF cpt < 0 THEN
     RAISE monException;
   END IF;
   ...
EXCEPTION
    WHEN monException THEN
      DBMS_OUTPUT.PUT_LINE('cpt ne doit pas être négatif');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Je ne connais pas cette erreur');
END;
```

# Exceptions prédéfinies anonymes (1)

```
EXCEPTION
...
WHEN OTHERS THEN
    DECLARE – Bloc des autres erreurs
     codeErreur NUMBER := SQLCODE;
    BEGIN
     IF codeErreur = -02291 THEN
      – Contrainte d'intégrité
      RAISE_APPLICATION_ERROR (-20001, 'Client inexistant');
     ELSIF...
     ...
     END IF;
   END;– Bloc des autres erreurs
END;– Programme
```

# Exceptions prédéfinies anonymes(2)

```
DECLARE
   ...
   monException EXCEPTION
   PRAGMA EXCEPTION_INIT (monException, -2400)
BEGIN
    ...
EXCEPTION
   WHEN monException THEN...
END;
```

# Propagation d'une exception

## Où va une exception?

Une exception est déclenchée :
→ PL/SQL cherche un gestionnaire dans la partie Exception du bloc courant (WHEN adéquat)
→ Si pas de gestionnaire : on cherche dans le bloc englobant...
→ Si aucun : PL/SQL envoie un message d'exception non gérée à l'application appelante

# Fonctions et procédures stockées

## Pourquoi?

- Pour enregistrer des programmes dans le noyau d'Oracle
- Comme une table ou une vue, elles peuvent être utilisées par d'autres utilisateurs, s'ils ont les droits voulus.
- Stockées sous forme de pseudo-code : pas de nouvelle compilation $\rightarrow$ efficace

# Déclaration procédure stockée

## Syntaxe

```
CREATE [OR REPLACE] PROCEDURE nom_procedure
    [(liste paramètres formels)]
    AS | IS
    [partie déclaration]
    BEGIN
      ...
    [EXCEPTION
      ...]
    END [nom_procedure];
```

partie déclaration: similaire à celle d'un bloc PL/SQL

# Déclaration procédure stockée : paramètres

## Syntaxe

nom paramètre [IN |
OUT [NOCOPY]|
IN OUT [NOCOPY]] type_paramètre
[ := | DEFAULT expression ]

type_paramètre : un type PL/SQL
IN : paramètre en entrée, non modifié par la procédure
OUT : paramètre en sortie, peut être modifié par la procédure, transmis au programme appelant
IN OUT : à la fois en entrée et en sortie
par défaut : IN
NOCOPY : pour passer des références et non des valeurs (mais le compilateur décide!)

# Exemple de procédure stockée

On cherche les réalisateurs qui ont joué dans un certain nombre de leur film...

```
CREATE PROCEDURE realActeursProc (nbFilms NUMBER) IS
    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 > nbFilms THEN
       DBMS_OUTPUT.PUT_LINE(nbRealAct||' réalisateurs ont joué
            dans plus de '||nbFilms||'de leurs films');
       ELSE DBMS_OUTPUT.PUT_LINE('Aucun réalisateur n'a joué
            dans plus de '||nbFilms||'de ses films');
    END IF;
END;
```

# Déclaration fonction stockée

## Syntaxe

```
CREATE [OR REPLACE] FUNCTION nom_fonction
    [(liste paramètres formels)]
    RETURN typeRetour     AS | IS
    [partie déclaration]
    BEGIN
      ...
      RETURN valeurRetout
      ...
    [EXCEPTION ...]
    END [nom_fonction];
```

partie déclaration: similaire à celle d'un bloc PL/SQL

typeRetour : le type PL/SQL de valeurRetour retournée par la fonction.

liste de paramètres : idem procédures, mais IN préférable dans les fonctions!!!

# Exemple de fonction stockée

On cherche toujours les réalisateurs qui ont joué dans plus de nbFilms de leurs films...

```
CREATE FONCTION nbRealActeurFonc (nbFilms NUMBER)
RETURN NUMBER IS
    nbRealAct NUMBER(5) := 0 ;
BEGIN
    SELECT COUNT(distinct A.numIndividu) INTO nbRealAct
        FROM Film F, Acteur A
        WHERE A.numIndividu = realisateur
          AND F.numFilm=A.numFilm;
    RETURN nbRealAct;
END;
```

# Appel de procédures et de fonctions stockées

## Appel à une procédure dans un programme PL/SQL

nom_procedure [(liste de paramètres effectifs)];

## Appel à realActeursProc

nbFilms:=20;
...
realActeursProc (nbFilms);

## Appel à une fonction dans un programme PL/SQL

nomVariable := nom_fonction [(liste de paramètres effectifs)];

## Appel à nbRealActeurFonc

nbFilms:=20;
...
nbGdActeursReals := nbRealActeurFonc (nbFilms);

# Récursivité

## Exemple de factorielle!

```
CREATE FUNCTION facto (N IN NUMBER(3))
    RETURN NUMBER IS
  BEGIN
    IF (N<=1) THEN
      RETURN 1;
    ELSE
      RETURN N * facto(N - 1);
    END IF;
  END;
```

La récursivité peut aussi être croisée.

# Compilation et suppression

## Compilation

Pour enregistrer une procédure ou une fonction stockée, il faut

- avoir le droit CREATE PROCEDURE
- donner l'ordre de création à Oracle.

Si pas d'erreur : $\rightarrow$ stockée sous forme compilée
Sinon : message d'erreur + erreurs stockées dans des vues générales (USER_ERRORS, ALL_ERRORS, DBA_ERRORS)
$\rightarrow$ les consulter pour corriger (REPLACE PROCEDURE...)

## Consultation

SELECT * FROM USER_ERRORS [WHERE NAME=nom];

## Suppression

DROP {PROCEDURE | FUNCTION} nom_procedure_ou_fonction

# Exemple

On cherche les acteurs ayant joué dans plus de N films... Et on veut leurs noms!

```
CREATE PROCEDURE nomsGdsActeurs (nbFilms NUMBER)
  Cursor lesActeurs IS
        SELECT nomIndividu
        FROM Individu
       WHERE numIndividu IN
            (SELECT numActeur
            FROM acteur
            GROUP BY numIndividu
            HAVING Count(numFilm)>nbFilms);
```

...→

# Exemple

```
...←
BEGIN
  DBMS_OUTPUT.PUT_LINE('Voici les acteurs ayant
          joué dans plus de '||nbFilms||' films : ')
  FOR ligneCurseur IN lesActeurs ;
  LOOP
       DBMS_OUTPUT.PUT_LINE(ligneCurseur.nomIndividu)
  END LOOP;
       DBMS_OUTPUT.PUT_LINE('Voici le nombre d'acteurs
          ayant joué dans plus de '||nbFilms||' films : '
          || lesActeurs%rowCount);
END;
```

# Exemple (suite alternative)

```
...←
sonNom nomIndividu.Individu%type;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Voici les acteurs ayant
          joué dans plus de '||nbFilms||' films : ')
  OPEN lesActeurs ;
  LOOP
       FETCH lesActeurs INTO sonNom;
       EXIT WHEN lesActeurs%NOTFOUND;
       DBMS_OUTPUT.PUT_LINE(sonNom)
  END LOOP;
       DBMS_OUTPUT.PUT_LINE('Voici le nombre d'acteurs
          ayant joué dans plus de '||nbFilms||' films : '
          || lesActeurs%rowCount);
END;
```