

Ecole Nationale des sciences  
appliquées Al-Hoceima  
Université Abdelmalek Essaadi

*Administration et optimisation  
des bases de données*

*ID1-S2*

*2021/2022*

*Mohamed CHERRADI*

# PL/SQL (Suite)

# Plan

- Introduction
- Bloc PL/SQL
- Déclaration des variables
- Structure de contrôle
- Curseurs
- **Les exceptions**
- Les fonctions et procédures
- Les packages
- Les triggers

## Les exceptions

# Introduction

- PL/SQL offre au développeur un mécanisme de gestion des exceptions. Il permet de préciser ***la logique du traitement des erreurs*** survenues dans l'un de ses blocs.
- Il s'agit donc d'un point clé dans **l'efficacité du langage** qui permettra de protéger l'intégrité du système.
- La gestion des exceptions c'est un mécanisme qui permet de gérer les erreurs qui survient durant l'exécution du programme
- Une exception est une condition d'erreur lors de l'exécution du programme.
- PL/SQL détecte de telles conditions à l'aide du bloc EXCEPTION

## Types des exceptions

- Il existe deux types d'exceptions:
  - ✓ **Exceptions internes**: générées par le moteur du système (connexion non établie, table inexistante, privilèges insuffisants, mémoire saturée, espace disque insuffisant, ...).
  - ✓ **Exception externes**: générées par le programmeur (division par zéro, ...)
- Liste des exceptions prédéfinies par Oracle  
[https://docs.oracle.com/cd/B10500\\_01/appdev.920/a96624/07\\_errs.htm](https://docs.oracle.com/cd/B10500_01/appdev.920/a96624/07_errs.htm)

## Types des exceptions

- Exceptions définies par ORACLE:
  - ✓ Nommées par Oracle
  - ✓ Exemples : **NO\_DATA\_FOUND**, **TOO\_MANY\_ROWS**, ...
  - ✓ Se déclenchent automatiquement
  - ✓ Nécessitent de prévoir la prise en compte de l'erreur dans la section **EXCEPTION**
  
- Exceptions définies par l'utilisateur :
  - ✓ Nommés par le programmeur
  - ✓ Sont déclenchées par une instruction du programme soit automatiquement (**PRAGMA**) ou manuellement par : **Raise**
  - ✓ Nécessitent de prévoir la prise en compte de l'erreur dans la section **EXCEPTION**.
  
- **NB:** Chaque exception Oracle décrit par: **SQLCODE** et **SQLERRM**

## Types des exceptions

- NB: Quand une exception est déclenchée, on peut identifier le code et le message de l'erreur, en utilisant les deux fonctions **SQLCODE** et **SQLERRM**.
  - ✓ **SQLCODE** : retourne le code de l'erreur
  - ✓ **SQLERRM** : retourne le message associé à l'erreur

SQLCODE	SQLERRM
0	No exception encountered
1	User_defined exception
+100	No data found
Nombre négatif	D'autres erreurs du serveur Oracle



## Exemple des exceptions prédefinies

- **ZERO\_DIVIDE** : tentative de division par zéro.
- **TOO\_MANY\_ROWS** : la commande **SELECT INTO** retourne plus d'une ligne.
- **NO\_DATA\_FOUND** : déclenchée si la commande **SELECT INTO** ne retourne aucune ligne ou si l'on fait référence à un enregistrement ,non initialisé, d'un tableau PL/SQL.
- **LOGIN\_DENIED** : connexion à la base est échouée, car le nom utilisateur ou le mot de passe est invalide.
- **CURSOR\_ALREADY\_OPEN** : tentative d'ouverture d'un curseur déjà ouvert.
- **INVALID\_NUMBER** : échec de la conversion d'une chaîne de caractères en numérique.
- **INVALID\_CURSOR** : opération incorrecte sur un curseur, comme par exemple la fermeture d'un curseur qui n'a pas été ouvert.
- **TIMEOUT\_ON\_RESOURCE** : dépassement du temps dans l'attente de libération des ressources (lié aux paramètres de la base).
- **Etc.**

## Exemple des exceptions prédefinies

Nom d'exception	Erreur ORACLE	SQLCODE
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
STORAGE_ERROR	ORA-06500	-6500
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

## Syntaxe (Les exceptions prédefinies)

**DECLARE**

*v\_sal emp.sal%type;*

**BEGIN**

*SELECT sal INTO v\_sal from emp;*

**EXCEPTION**

*WHEN TOO\_MANY\_ROWS then ... ;*

*-- gérer erreur trop de lignes*


*WHEN NO\_DATA\_FOUND then ... ;*

*-- gérer erreur pas de ligne*

*WHEN OTHERS then ... ;*

*-- gérer toutes les autres erreurs*

**END ;**



*dbms\_output.put\_line ('Code d'erreur : ' || SQLCODE) ;*

*dbms\_output.put\_line ('Message d'erreur : ' || SQLERRM) ;*

## Syntaxe (Les exceptions non-prédefinies)

```
DECLARE
    exception_name EXCEPTION;
BEGIN
    IF condition THEN
        RAISE exception_name;
    END IF;
EXCEPTION
    WHEN exception_name THEN
        -- statement;
END;
```

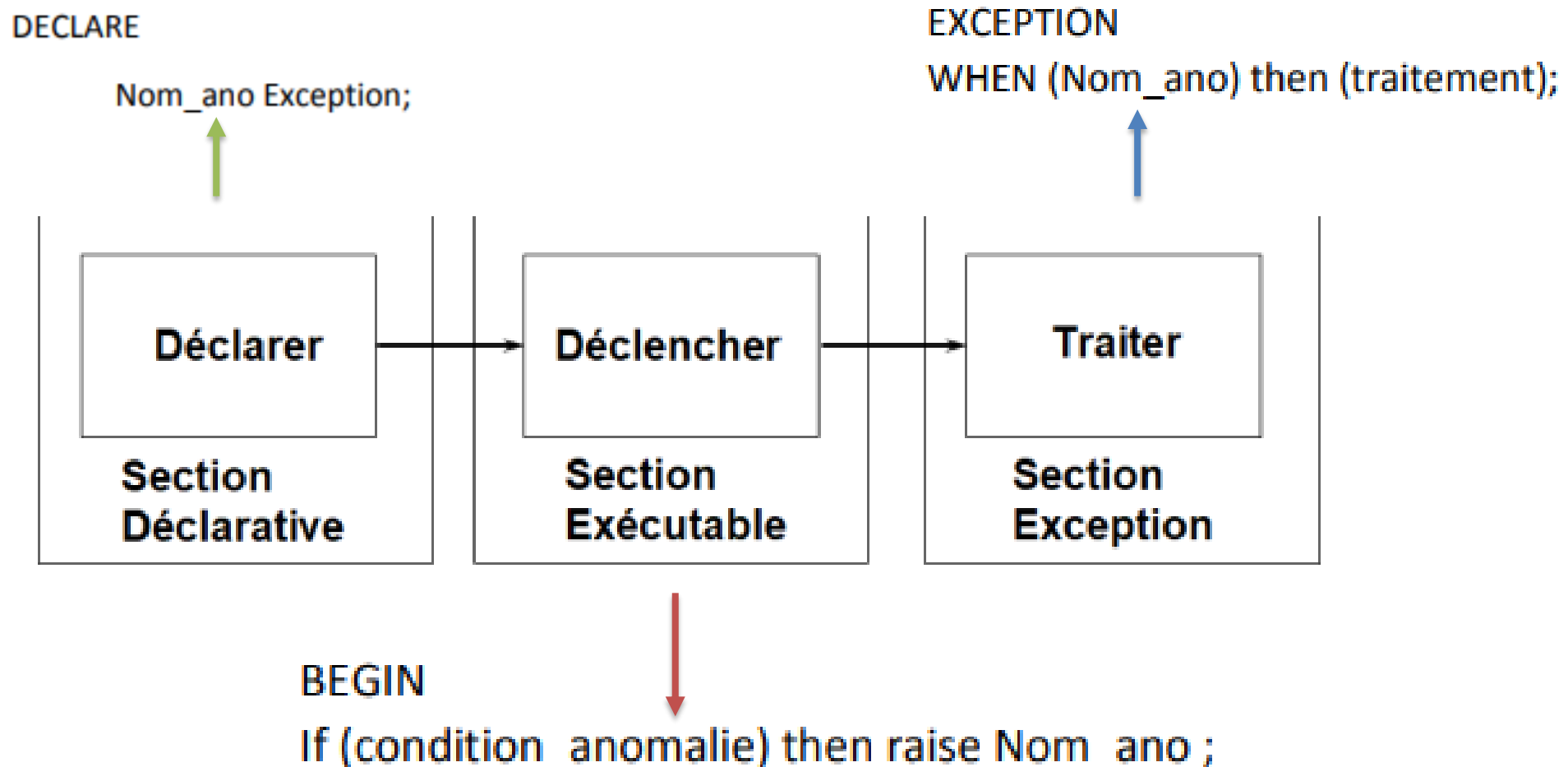
```
DECLARE
    c_id customers.id%type := &cc_id;
    c_name customerS.Name%type;
    c_addr customers.address%type;
    -- user defined exception
    ex_invalid_id EXCEPTION;
BEGIN
    IF c_id <= 0 THEN
        RAISE ex_invalid_id;
    ELSE
        SELECT name, address INTO c_name, c_addr
        FROM customers
        WHERE id = c_id;
        DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
        DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
    END IF;

    EXCEPTION
        WHEN ex_invalid_id THEN
            dbms_output.put_line('ID must be greater than zero!');
        WHEN no_data_found THEN
            dbms_output.put_line('No such customer!');
        WHEN others THEN
            dbms_output.put_line('Error!');
END;
```

## Syntaxe (Les exceptions non-prédefinies)

```
DECLARE
    ...
    Nom_ano EXCEPTION;
BEGIN
    instructions ;
    IF (condition_anomalie) THEN RAISE Nom_ano;
    ...
EXCEPTION
    WHEN Nom_ano THEN (traitement);
END ;
```

## Syntaxe (Les exceptions non-prédefinies)



## Syntaxe (Les exceptions non-prédefinies)

- Déclenchement de l'exception s'effectuer de deux manières:
  - ✓ Soit associer à cette erreur un code ORACLE, elle sera levée automatiquement
    - Syntaxe: **PRAGMA EXCEPTION\_INIT** (Nom\_exception, Code\_erreur)
  - ✓ Soit lever manuellement l'exception
    - Syntaxe: **RAISE** Nom\_exception

## Fonctions d'interception des exceptions

- Lorsqu'une exception est interceptée par la clause **WHEN OTHERS**, vous pouvez utiliser un ensemble de fonctions standard pour identifier l'erreur.

```
DECLARE
v_error_code    NUMBER;
v_error_message VARCHAR2(255);
BEGIN
    ...

    EXCEPTION
    ...
    WHEN OTHERS THEN
        ROLLBACK;
        v_error_code := SQLCODE;
        v_error_message := SQLERRM;
        insert into erreur values(v_error_code, v_error_message);
END;
```



# Fonctionnement des exceptions

- Lorsqu'une exception est détectée:
  1. Arrêt de l'exécution du bloc
  2. Branchement sur la section exception
  3. Parcours des clauses WHEN jusqu'au bon choix
  4. Exécution des instructions associées
  5. Une fois le traitement de l'erreur est terminé, c'est le bloc suivant qui est effectué.

## Exercice

- Ecrire un bloc PL/SQL qui:
  - ✓ Lit les deux entiers A et B
  - ✓ Calcule et affiche la division de A par B
  - ✓ Exécuter le bloc pour B=0. que remarque-t-on?
  - ✓ Modifier le bloc pour traiter l'exception levée en affichant un message approprié.

## Exercice

- On cherche à trouver le nom de l'employé qui porte le nom de « cherradi ». Sauf que si cet employé n'existe pas, ce qui retourne une erreur du type **no\_data\_found**.
- **TAF**: Rajouter un bloc EXCEPTION qui traitera l'erreur, en affichant un message qui dira que cet employé n'existe pas.

## Procédures et fonctions