

# TRAVAUX PRATIQUES : PL/SQL 4

## Gestion des Exceptions

### Tableaux Récapitulatifs des Commandes

#### 1) Les exceptions :

##### a) Exceptions prédéfinies :

Syntaxe :

la section dédiée  
au traitement des  
exceptions.

```
BEGIN
    Instructions de traitement du bloc PL/SQL
    ....
    EXCEPTION
    WHEN nom_exception THEN traitement de l'exception;
    ... etc
END;
```

Tableau récapitulatif des exceptions prédéfinies par Oracle :

Exception	Description
<b>NO_DATA_FOUND</b>	Déclenchée lorsqu'une requête <b>SELECT ... INTO</b> ne trouve pas de lignes correspondantes.
<b>TOO_MANY_ROWS</b>	Déclenchée lorsqu'une requête <b>SELECT .... INTO</b> trouve plus d'une ligne correspondante.
<b>ZERO_DIVIDE</b>	Déclenchée lors d'une tentative de division par zéro.
<b>DUP_VAL_ON_INDEX</b>	Déclenchée lorsqu'une tentative d'insertion ou de mise à jour viole une contrainte d'unicité d'index.
<b>VALUE_ERROR</b>	Déclenchée lorsqu'une opération de conversion de type, telle que l'assignation d'une chaîne de caractères à une variable numérique, échoue.
<b>CURSOR_ALREADY_OPEN</b>	Déclenchée lorsqu'une tentative est faite pour ouvrir un curseur qui est déjà ouvert.

## b) Exceptions définies par l'utilisateur :

### Syntaxe :

```
DECLARE
    nom_erreur EXCEPTION;
    ...
BEGIN
    ...
    IF condition THEN
        RAISE nom_erreur;    -- On déclenche l'erreur
    ...
EXCEPTION
    WHEN nom_erreur THEN traitement de l'erreur;
    ... etc
END;
```

Comme les curseurs, on peut exploiter Les variables de condition suivantes :

VARIABLES DE CONDITION	DESCRIPTION
SQL%FOUND	Variable booléenne qui vaut TRUE si la dernière instruction LMD exécutée a affecté au moins une ligne, FALSE dans le cas contraire
SQL%NOTFOUND	Variable booléenne qui vaut TRUE si la dernière instruction LMD exécutée n'a affecté aucune ligne, FALSE dans le cas contraire
SQL%ROWCOUNT	Variable numérique qui contient le nombre de lignes affectées par la dernière instruction LMD exécutée

## Manipulations :

### 1- Exceptions Prédéfinies :

- 1) Écrivez un bloc PL/SQL qui sélectionne le nom et prénom de l'employé ID "0" et qui gère l'exception "ligne inexistante" en utilisant l'exception : **NO\_DATA\_FOUND**.
- 2) Écrivez un bloc PL/SQL qui demande à l'utilisateur de saisir un ID de l'employé au clavier puis il affiche le nom et le salaire de cet employé, ou qui affiche un message d'erreur si aucun employé ne possède cet identifiant. Utiliser le nom Oracle de l'erreur (exception) : **NO\_DATA\_FOUND**.
- 3) Reprendre la même question que (2), mais, cette fois-ci, au d'utiliser directement le nom Oracle de l'erreur **NO\_DATA\_FOUND**, on utilise le SQLCODE Oracle de l'erreur **NO\_DATA\_FOUND** qui est +100.
- 4) Écrivez un bloc PL/SQL qui effectue une division de deux nombres et gère l'exception **ZERO\_DIVIDE**. Si une division par zéro est tentée, affichez un message d'erreur approprié.
  - **Indications :**
    - ❖ Déclarez deux variables numériques, par exemple **v\_num1** et **v\_num2**.
    - ❖ On demande à l'utilisateur de saisir la valeur de deux variables au clavier.
    - ❖ Effectuez une division **v\_num1 / v\_num2** et stockez le résultat dans une autre variable.
    - ❖ Utilisez **BEGIN ... EXCEPTION ... END;** pour gérer l'exception **ZERO\_DIVIDE**.
- 5) Écrivez un bloc PL/SQL qui tente d'insérer une ligne dans la table **JOBS** où une contrainte d'unicité existe (un identifiant unique dans la table **JOBS**, tel que l'exemple ci-dessous. Gérez l'exception **DUP\_VAL\_ON\_INDEX** pour informer l'utilisateur qu'une duplication a été tentée.

JOB_ID	ST_MAN
Job_title	Stock Manger
Min_salary	6000
Max_salary	9000

- **Indications :**
  - ❖ Utilisez l'instruction **INSERT** pour ajouter une ligne dans la table.
  - ❖ Gérez l'exception **DUP\_VAL\_ON\_INDEX** dans le bloc **EXCEPTION** en affichant un message.

## 2- Exceptions définies par l'utilisateur :

- 5) Écrivez un bloc PL/SQL pour vérifier si un employé spécifié existe dans la table **employees**. Si l'employé n'existe pas, déclenchez une exception personnalisée.
- **Indications :**
    - ❖ Demandez à l'utilisateur de saisir l'identifiant d'un employé.
    - ❖ Utilisez une requête **SELECT COUNT(\*) INTO** pour vérifier l'existence de l'employé.
    - ❖ Déclarez une exception personnalisée **e\_employe\_inexistant**.
    - ❖ Utilisez **RAISE e\_employe\_inexistant** si l'employé n'existe pas.
    - ❖ Gérez l'exception pour informer l'utilisateur que l'employé spécifié est inexistant.
- 6) Écrivez un bloc PL/SQL qui demande à l'utilisateur de saisir un identifiant d'employé et un nouveau salaire, puis met à jour le salaire de l'employé correspondant dans la table **employees**. Si l'employé n'existe pas, le bloc doit afficher un message d'erreur indiquant que l'employé n'existe pas, ainsi que le numéro et le nom de l'erreur.

### Indication :

- ❖ Déclarez des variables **v\_employee\_id**, **v\_salaire**, et **v\_error\_message** (de type Exception). Utilisez la clause **:=** pour initialiser **v\_employee\_id** et **v\_salaire** avec **des valeurs fournies par l'utilisateur**.
  - ❖ Utilisez la commande **UPDATE** pour mettre à jour le salaire de l'employé avec la nouvelle valeur **nv\_salaire**, en utilisant la condition **WHERE employee\_id = v\_employee\_id** pour identifier l'employé à mettre à jour.
  - ❖ dans une clause **IF**, utilisez la condition **SQL%NOTFOUND** pour vérifier si aucune ligne n'a été mise à jour:
    - Dans le cas d'une mise à jour, affichez un message de confirmation indiquant que l'employé a reçu un nouveau salaire et indiquant le nouveau salaire.
    - Dans le cas où il n'y a pas eu de mise à jour, lancez une exception **v\_error\_message** avec la commande **RAISE**.
  - ❖ Dans la section **EXCEPTION**, gérez l'exception **v\_error\_message** en affichant un message indiquant que l'employé n'existe pas et en affichant les informations d'erreur **SQLCODE** et **SQLERRM**.
- 7) Écrivez un bloc PL/SQL pour vérifier si un employé appartient à un département spécifique, par exemple, le département **'IT'**. Si non, déclenchez une exception personnalisée.
- **Indications :**
    - ❖ Demandez l'identifiant de l'employé et le nom du département.
    - ❖ Vérifiez si l'employé appartient au département spécifié en utilisant les tables **employees** et **departments**.
    - ❖ Déclarez une exception personnalisée **e\_non\_appartient\_au\_departement**.

- ❖ Utilisez **RAISE e\_non\_appartient\_au\_departement** si l'employé n'appartient pas au département spécifié.
- ❖ Gérez l'exception pour signaler la non-appartenance.
- ❖ Essayez les valeurs d'employee\_ID suivants: 100, 103, 105, 1000.

**8)** Ecrire un programme PL/SQL qui saisit un nouvel employé dans la table Employees. Gérer les erreurs suivantes :

- Le ID de l'employé existe déjà
- Le nom de l'employé n'est pas vide.
- Le ID du POSTE (job\_ID) n'existe pas

Les champs à insérer sont: EMPLOYEE\_ID , LAST\_NAME, EMAIL , HIRE\_DATE, JOB\_ID

**9)** Demander à l'utilisateur d'insérer un salaire au clavier.

Puis, si ce salaire (saisi par l'utilisateur) est inférieur à 2000, déclencher l'exception pré-définie

'INVALID\_NUMBER'. Puis, afficher le message d'erreur "Le salaire doit être supérieur ou égal à 2000."

Sinon, mettre à jour le salaire de l'employee\_ID 100 à un salaire saisi par l'utilisateur.

- Remarque: ne créez pas votre propre exception. Utilisez plutôt l'erreur Oracle 'INVALID\_NUMBER'.