Les Exceptions

Une exception est une erreur qui survient lors de l'exécution d'un programme.

En **PL/SQL**, les exceptions sont utilisées pour gérer les **erreurs** ou **situations exceptionnelles** qui surviennent pendant l'exécution d'un programme. Elles permettent de capturer ces erreurs et de définir une action à effectuer (comme afficher un message, annuler une transaction, etc.).

On utilise les exceptions pour :

1. **Gérer les erreurs automatiquement :** Par exemple, lorsqu'une requête ne retourne aucune ligne ou dépasse une limite.
2. **Améliorer la robustesse du code :** Éviter que le programme ne plante en cas d'erreur imprévue.
3. **Fournir des messages explicites :** Informer l'utilisateur ou le développeur de la nature de l'erreur.

Il existe 3 type d'exception :

1. Exceptions prédéfinies **2.** Exceptions définies par l'utilisateurs **3.** Exceptions non gérées

Les Exceptions

➢ Structure d'un bloc avec exceptions

Un bloc PL/SQL peut avoir une section **EXCEPTION** où les erreurs sont capturées et gérées.

```
BEGIN
   -- Code principal
EXCEPTION
   WHEN <exception_prédéfinie> THEN
      -- Action à effectuer en cas d'erreur
   WHEN OTHERS THEN
      -- Action pour toute autre erreur
END;
```

Les Exceptions

➢ **Les Exceptions prédéfinies :** les erreurs courantes que **PL/SQL gère automatiquement**.

- **NO_DATA_FOUND:** Se produit lorsqu'un SELECT INTO ne retourne **aucune ligne**.
- **TOO_MANY_ROWS:** Se produit lorsqu'un SELECT INTO retourne **plus d'une ligne**, alors qu'une seule était attendue.
- **ZERO_DIVIDE:** Se produit lorsqu'une division par zéro est tentée.
- **INVALID_CURSOR:** Se produit lorsqu'une opération illégale est effectuée sur un curseur.
- **VALUE_ERROR:** Se produit lorsqu'une valeur dépasse la limite ou le type de données attendu.
- **ACCESS_INTO_NULL:** Se produit lorsqu'une tentative d'accès est faite sur un attribut d'un objet ou d'un record **non initialisé**.
- **OTHERS:** Capture toute erreur non spécifiquement gérée par une exception prédéfinie.

Les Exceptions

➢ Les Exceptions prédéfinies : Exemple

```
BEGIN
   DECLARE
     v_nom VARCHAR2(50);
   BEGIN
     SELECT nom INTO v_nom
     FROM employes
     WHERE poste = 'Développeur';
   EXCEPTION
     WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur : Trop de résultats retournés.');
   END;
END;
/
```

Les Exceptions

➢ Les Exceptions Définies par l'utilisateurs :

Les **exceptions définies par l'utilisateur** permettent de gérer des **situations spécifiques** au programme, qui ne sont pas couvertes par les exceptions prédéfinies. On peut les déclarer, les lever manuellement et les traiter dans la section **EXCEPTION**.

Étapes pour gérer des exceptions définies par l'utilisateur

1. **Déclaration de l'exception :** Déclarez une exception dans la section **DECLARE**.
2. **Levée de l'exception (RAISE) :** Déclenchez l'exception dans le code lorsque la condition est remplie.
3. **Gestion de l'exception (WHEN) :**Traitez l'exception dans la section **EXCEPTION**.

Les Exceptions

➢ **Les Exceptions Définies par l'utilisateurs : Syntaxe**

```
DECLARE
   mon_exception EXCEPTION; -- Déclaration de l'exception
BEGIN
   -- Code principal
   IF <condition> THEN
      RAISE mon_exception; -- Levée de l'exception
   END IF;
EXCEPTION
   WHEN mon_exception THEN
      -- Traitement de l'exception
      DBMS_OUTPUT.PUT_LINE('Erreur définie par l\'utilisateur
détectée.');
END;
/
```

Les Exceptions

➢ **Les Exceptions Définies par l'utilisateurs : Exemple**

Détection d'un dépassement de budget: Si un budget dépasse une limite fixée lever une exception.

```
DECLARE
   budget_depasse EXCEPTION; -- Déclaration de l'exception
   v_budget NUMBER := 350000;
BEGIN
   IF v_budget > 300000 THEN
      RAISE budget_depasse; -- Levée de l'exception
   END IF;
EXCEPTION
   WHEN budget_depasse THEN
      DBMS_OUTPUT.PUT_LINE('Erreur:Le budget dépasse la limite ');
END;
/
```

Les Exceptions

➢ Les Exceptions Définies par l'utilisateurs : Exemple

```
DECLARE
   trop_peu_employes EXCEPTION; -- Déclaration de l'exception
   v_nb_employes NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_nb_employes
   FROM employes
   WHERE id_projet = 1;

   IF v_nb_employes < 3 THEN
      RAISE trop_peu_employes; -- Levée de l'exception
   END IF;
EXCEPTION
   WHEN trop_peu_employes THEN
      DBMS_OUTPUT.PUT_LINE('Erreur : Nombre d\'employés insuffisant pour ce
projet.');
END;
/
```

Les Exceptions

➢ **Les Exceptions Définies par l'utilisateurs : Exemple Exception avec des paramètres dynamique**

```
DECLARE
   projet_suspendu EXCEPTION; -- Déclaration de l'exception
   v_statut VARCHAR2(20);
BEGIN
   SELECT statut INTO v_statut   FROM projets    WHERE id_projet = 3;

   IF v_statut = 'Suspendu' THEN
      RAISE projet_suspendu; -- Levée de l'exception
   END IF;
EXCEPTION
   WHEN projet_suspendu THEN
      DBMS_OUTPUT.PUT_LINE('Erreur : Ce projet est suspendu.');
END;
/
```

Les Exceptions

➢ SQLCODE

Retourne le **code numérique** de l'erreur qui vient de se produire. il permet d'identifier une erreur grâce à son code.

Exemple de valeur :

- 100 : Correspond à l'erreur NO_DATA_FOUND.
- 1403 : Autre code d'erreur lié aux données.

➢ SQLERRM

Retourne le **message d'erreur** associé au dernier code d'erreur généré.il Fournir une explication lisible pour une erreur.

Exemple de valeur : Si une erreur NO_DATA_FOUND survient, SQLERRM retournera : *"ORA-01403: no data found"*

Les Exceptions

➢ SQLCODE , SQLERRM Exemple

```
BEGIN
   -- Une requête qui provoque une erreur
   SELECT nom INTO v_nom
   FROM employes
   WHERE id_employe = 999; -- ID inexistant
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Erreur détectée :');
      DBMS_OUTPUT.PUT_LINE('Code : ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('Message : ' || SQLERRM);
END;
/
```

Erreur détectée :
Code : 100
Message : ORA-01403: no data found