

# Systèmes de Gestion de Bases de Données - 2<sup>e</sup>

## PL-SQL - Chapitre 4 - Les exceptions

---

Daniel Schreurs

30 novembre 2021

Haute École de Province de Liège

# Table des matières du chapitre i

1. Utilisation des exceptions
2. Catégories d'exceptions
3. Petit guide...

# Utilisation des exceptions

---

# Table des matières de la section : Utilisation des exceptions i

## 1. Utilisation des exceptions

### 1.1 Déclaration des exceptions

### 1.2 Le gestionnaire d'exceptions

### 1.3 La clause WHEN OTHERS

## 2. Catégories d'exceptions

## 3. Petit guide...

# Utilisation des exceptions : Déclaration des exceptions

- Dans la section **DECLARE**
- **Nom\_de\_l\_exception** **EXCEPTION;**
- Lorsqu'une erreur survient, elle est lancée dans le *gestionnaire d'exception*.
- À nous d'en faire quelque chose.

# Utilisation des exceptions : Le gestionnaire d'exceptions

- Est chargé d'intercepter les différentes erreurs
- Peut être déclaré dans n'importe quel bloc PL/SQL
- Commence par le mot clé **EXCEPTION**

Une exception est lancée :

- Explicitement par un RAISE
- Implicitement par Oracle (exceptions prédéfinies)

# Utilisation des exceptions : La clause WHEN OTHERS

- La clause **WHEN OTHERS** dans un gestionnaire d'exceptions permet d'intercepter n'importe quel type d'exceptions non géré par ailleurs.
- La définition de cette clause devrait devenir systématique.

# Utilisation des exceptions : La clause WHEN OTHERS

## La clause WHEN OTHERS

```
1 BEGIN
2     -- Du code
3 EXCEPTION
4     WHEN OTHERS THEN
5         DBMS_OUTPUT.PUT_LINE
6             ('Code Erreur : ' || SQLCODE || 'Message : ' || SQLERRM);
7 END;
```

---



## Catégories d'exceptions

---

# Table des matières de la section : Catégories d'exceptions i

## 1. Utilisation des exceptions

## 2. Catégories d'exceptions

2.1 Exceptions définies "non redirigées"

2.2 Exceptions prédéfinies : NO\_DATA\_FOUND

2.3 Exceptions prédéfinies : TOO\_MANY\_ROWS

2.4 Exceptions prédéfinies : DUP\_VAL\_ON\_INDEX

2.5 Exceptions prédéfinies : INVALID\_NUMBER

2.6 Exceptions prédéfinies : VALUE\_ERROR

2.7 Exceptions définies "redirigées"

2.8 Exceptions définies "redirigées" : intégrité référentielle

# Table des matières de la section : Catégories d'exceptions ii

2.9 Exceptions définies "redirigées" : Ressource Busy

2.10 Exceptions définies "redirigées" : Ressource Busy

3. Petit guide...

# Catégories d'exceptions

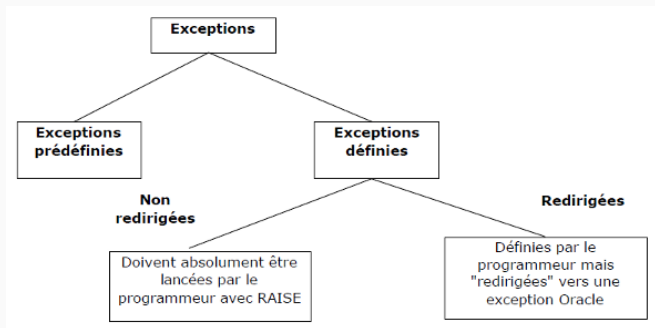


FIGURE 1 – Catégories d'exceptions

# Catégories d'exceptions : Exceptions définies "non redirigées"

- Doivent être déclarées
- Doivent être lancées par le programmeur avec la commande RAISE
- Ne sont pas associées à une erreur prédéfinie par Oracle

## Important

Ce type d'exception est utile pour gérer une exception spécifique liée à la logique de programmation du bloc PL/SQL.

# Catégories d'exceptions : Exceptions définies "non redirigées"

## Exceptions définies "non redirigées"

```
1 CREATE SEQUENCE SequenceEmp START WITH 1000;
2 DECLARE
3     Ename Emp.Ename%TYPE;
4     ExcEnameNULL EXCEPTION;
5 BEGIN
6     IF Ename IS NULL THEN RAISE ExcEnameNULL; END IF;
7     INSERT INTO Emp (Empno, Ename, HireDate) VALUES (SequenceEmp.
8         NEXTVAL, Ename, CURRENT_DATE);
9     COMMIT;
10 EXCEPTION
11     WHEN ExcEnameNULL THEN
12         DBMS_OUTPUT.PUT_LINE('Nom de l''employé est inconnu');
13     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
14 END;
15 -- Nom de l'employé est inconnu
16 -- Procédure PL/SQL terminée avec succès.
```

## Catégories d'exceptions : Exceptions définies "non redirigées"

Exception	Erreur ORACLE	Valeur du SQLCODE
ACCESS_INTO_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
<b>DUP_VAL_ON_INDEX</b>	<b>ORA-00001</b>	<b>-1</b>
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
<b>NO_DATA_FOUND</b>	<b>ORA-01403</b>	<b>+100</b>
NOT_LOGGED_ON	ORA-01012	-1012

## Catégories d'exceptions : Exceptions définies "non redirigées"

Exception	Erreur ORACLE	Valeur du SQLCODE
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
<b>TOO_MANY_ROWS</b>	<b>ORA-01422</b>	<b>-1422</b>
<b>VALUE_ERROR</b>	<b>ORA-06502</b>	<b>-6502</b>
ZERO_DIVIDE	ORA-01476	-1476



# Catégories d'exceptions : Exceptions prédéfinies : NO\_DATA\_FOUND

Le nr de l'employé n'existe pas

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3     -- il n'existe pas d'employé avec un nr 1111
4 BEGIN
5     SELECT * INTO UnEmploye FROM emp WHERE Empno = 1111;
6     DBMS_OUTPUT.PUT_LINE(UnEmploye.Ename);
7 EXCEPTION
8     WHEN NO_DATA_FOUND THEN
9         DBMS_OUTPUT.PUT_LINE
10             ('Le nr de l''employé n''existe pas');
11     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
12 END;
```

# Catégories d'exceptions : Exceptions prédéfinies : TOO\_MANY\_ROWS

## Plus d'un employé SCOTT

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3     -- il existe 2 employés Scott
4 BEGIN
5     SELECT * INTO UnEmploye FROM emp WHERE Ename = 'SCOTT';
6     DBMS_OUTPUT.PUT_LINE(UnEmploye.Job);
7 EXCEPTION
8     WHEN NO_DATA_FOUND THEN
9         DBMS_OUTPUT.PUT_LINE('Le nom n''existe pas');
10    WHEN TOO_MANY_ROWS THEN
11        DBMS_OUTPUT.PUT_LINE('Plus d''un employé SCOTT');
12    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
13 END;
```

---

# Catégories d'exceptions : Exceptions prédéfinies : DUP\_VAL\_ON\_INDEX

## Erreur de données

```
1 -- Il existe un employé avec un numéro 1234
2 BEGIN
3     INSERT INTO Emp (Empno, Ename, HireDate)
4     VALUES (1234, 'Dupont', CURRENT_DATE);
5     COMMIT;
6 EXCEPTION
7     WHEN DUP_VAL_ON_INDEX THEN
8         DBMS_OUTPUT.PUT_LINE
9             ('Numéro d'employé existe déjà !');
10    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
11 END;
```

---

# Catégories d'exceptions : Exceptions prédéfinies : INVALID\_NUMBER

## Erreur de données

```
1 -- Il existe un employé avec un numéro 1234
2 BEGIN
3     INSERT INTO Emp (Empno, Ename, HireDate)
4     VALUES (1234, 'Dupont', CURRENT_DATE);
5     COMMIT;
6 EXCEPTION
7     WHEN DUP_VAL_ON_INDEX THEN
8         DBMS_OUTPUT.PUT_LINE
9             ('Numéro d'employé existe déjà !');
10    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
11 END;
```

Chaine réceptrice trop petite!

```
1 DECLARE
2     Mes1 VARCHAR2(10);
3 BEGIN
4     Mes1 := SQLERRM;
5 EXCEPTION
6     WHEN VALUE_ERROR THEN
7         DBMS_OUTPUT.PUT_LINE
8             ('Chaine réceptrice trop petite !');
9     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
10 END;
```

---

## Catégories d'exceptions : Exceptions définies "redirigées"

- Une exception définie par le programmeur peut être associée au code d'erreur d'une exception définie par Oracle
- Oracle se charge de lancer ce type d'exception : pas besoin de lancer un **RAISE**
- Une exception redirigée se définit en utilisant le pragma **EXCEPTION\_INIT**

# Catégories d'exceptions : Exceptions définies "redirigées"

Techniques des exceptions définies "redirigées" :

- Déclarer l'exception : **NameExc** **EXCEPTION**
- Associer le nom de l'exception à un numéro d'erreur Oracle  
**PRAGMA EXCEPTION\_INIT (NameExc, Oracle\_error);**
- Utiliser l'exception dans la section des gestions d'exceptions.

## Important

Il ne faut pas lancer explicitement l'exception !

# Catégories d'exceptions : Exceptions définies "redirigées" : intégrité référentielle

Contrainte d'intégrité sur Mgr est violée

```
1 DECLARE
2     ExcCleEtrangere EXCEPTION;
3     PRAGMA EXCEPTION_INIT (ExcCleEtrangere,-2291);
4 BEGIN
5     -- le numéro d'employé 1000 n'existe pas
6     INSERT INTO Emp (Empno, Ename, HireDate, Mgr) VALUES (SequenceEmp
7         .NEXTVAL, USER, CURRENT_DATE, 1000);
8     COMMIT;
9 EXCEPTION
10    WHEN ExcCleEtrangere THEN DBMS_OUTPUT.PUT_LINE
11        ('Contrainte d'intégrité sur Mgr est violée');
12    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
13 END;
14 -- Contrainte d'intégrité sur Mgr est violée
```



# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

La ressource est occupée

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3     ExcRessourceBusy EXCEPTION;
4     PRAGMA EXCEPTION_INIT (ExcRessourceBusy,-54);
5 -- autre session a exécuté la requête SELECT * FROM emp FOR UPDATE;
6 BEGIN
7     SELECT *
8     INTO UnEmploye
9     FROM emp
10    WHERE Empno = 7788
11          FOR UPDATE NOWAIT;
12    DBMS_OUTPUT.PUT_LINE(UnEmploye.Ename);
13 EXCEPTION
14    WHEN ExcRessourceBusy THEN
15        DBMS_OUTPUT.PUT_LINE('La ressource est occupée');
16    WHEN NO_DATA_FOUND THEN
17        DBMS_OUTPUT.PUT_LINE('Le nr de 'l'employé 'n'existe pas');
18    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
19 END;
```

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

## Important

Les exceptions remontent de bloc en bloc jusqu'au code appelant si elles ne sont pas capturées.

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

## Une erreur non interceptée

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3     -- le numéro d'employé 1111 n'existe pas
4 BEGIN
5     SELECT *
6     INTO UnEmploye
7     FROM emp
8     WHERE Empno = 1111;
9     DBMS_OUTPUT.PUT_LINE(UnEmploye.Ename);
10 END;
11
12 -- ERREUR à la ligne 1 :
13 -- ORA-01403: Aucune donnée trouvée
14 -- ORA-06512: à ligne 5
```

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

## Une erreur interceptée

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3
4 BEGIN
5     SELECT * INTO UnEmploye FROM emp WHERE Empno = 1111;
6     DBMS_OUTPUT.PUT_LINE(UnEmploye.Ename);
7 EXCEPTION
8     WHEN NO_DATA_FOUND THEN
9         DBMS_OUTPUT.PUT_LINE
10             ('Le nr de l'employé n'existe pas');
11     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
12 END;
13
14 -- Le nr de l'employé 'n'existe pas
```

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

Une erreur non interceptée dans un bloc imbriqué

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3
4 BEGIN
5     DBMS_OUTPUT.PUT_LINE('Avant le bloc imbriqué');
6     BEGIN
7         SELECT * INTO UnEmploye FROM emp WHERE Empno = 1111;
8     END;
9     DBMS_OUTPUT.PUT_LINE('Après le bloc imbriqué');
10 EXCEPTION
11     WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE
12         ('Le nr de l''employé n''existe pas');
13     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
14 END;
```

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

## Une erreur interceptée dans un bloc imbriqué

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3 BEGIN
4     BEGIN
5         SELECT * INTO UnEmploye FROM emp WHERE Empno = 1111;
6         DBMS_OUTPUT.PUT_LINE('Trouvé ' || UnEmploye.Ename);
7     EXCEPTION
8         WHEN NO_DATA_FOUND THEN
9             DBMS_OUTPUT.PUT_LINE('Pas de numéro 1111');
10    END;
11    DBMS_OUTPUT.PUT_LINE('Après le bloc imbriqué');
12 EXCEPTION
13     WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE
14         ('Le nr de l'employé n'existe pas');
15     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
16 END;
```

# Catégories d'exceptions : Exceptions définies "redirigées" : Resource Busy

## L'instruction NULL

```
1 DECLARE
2     UnEmploye Emp%ROWTYPE;
3 BEGIN
4     DBMS_OUTPUT.PUT_LINE('Avant le bloc imbriqué');
5     BEGIN
6         SELECT * INTO UnEmploye FROM emp WHERE Empno = 1000;
7         DBMS_OUTPUT.PUT_LINE('Trouvé' || 'UnEmploye.Ename');
8     EXCEPTION
9         WHEN NO_DATA_FOUND THEN NULL;
10    END;
11    DBMS_OUTPUT.PUT_LINE('Après le bloc imbriqué');
12 EXCEPTION
13     WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE
14         ('Le nr de l''employé n''existe pas');
15     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
16 END;
```

Petit guide...

---



- Utiliser systématiquement la déclaration d'un gestionnaire d'exception dans tout bloc PL/SQL où une erreur est susceptible d'intervenir.
- Tout gestionnaire d'exceptions minimal doit comporter la clause WHEN OTHERS.

Les erreurs surviennent généralement :

- Pendant les opérations arithmétiques;
- Les opérations sur la base de données (**SELECT**, **INSERT**, **UPDATE**, **DELETE**);
- Ainsi que les conversions de données et/ou manipulations de chaînes de caractères.

- Définir des exceptions si le code écrit reçoit des encodages de données erronées ou ne respectant pas un format ou un type prédéfini. Exemple : les paramètres **NULL** ou les sélections qui ne ramènent aucune donnée Utiliser les attributs **%TYPE** et **%ROWTYPE** pour rendre le code indépendant des données accédées et des modifications, ajouts ou suppressions de colonnes.
- Pour chaque exception interceptée, se poser la question de savoir si les opérations effectuées dans la base de données doivent être confirmées ou annulées (**COMMIT** ou **ROLLBACK?**)

### Important

Les exceptions peuvent faire partie intégrante de la logique de programmation.

Dans le code d'une exception, un traitement particulier peut être effectué.

## Traitement dans le gestionnaire d'exceptions

```
1 BEGIN
2     INSERT INTO Clients
3         (refClient, nomClient, adresseClient, localite)
4     VALUES (1, 'Gramme', 'Rue de la dynamo', 'Rome');
5     COMMIT;
6 EXCEPTION
7     WHEN DUP_VAL_ON_INDEX THEN
8         UPDATE Clients
9             SET nomClient      = 'Gramme',
10                adresseClient = 'Rue de la dynamo',
11                localite      = 'Rome'
12         WHERE refClient = 1;
13     COMMIT;
14     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE
15         ('Code erreur : ' || SQLCODE || ', message : ' || SQLERRM);
16     ROLLBACK;
17 END;
```

---