# Les exceptions

Les exceptions sont les erreurs générées par le programme. Ces exceptions peuvent être prédéfinies, personnalisées, définies et déclenchées par l’utilisateur ou anonymes.

En Pl/SQL, les exceptions sont traitées dans une section séparée du code. Cette section commence par la clause « EXCEPTION ». Quand une erreur est détectée, le traitement en cours est arrêté et le branchement est effectué à la section EXCEPTION. Si aucune section EXCEPTION n’est prévue alors l’erreur provoque la sortie du programme.

Quand le traitement de l’exception a été effectué dans la section exception, il n’y a pas de retour au programme possible (même avec GOTO). La solution consiste à imbriquer les blocs.

Exceptions prédéfinies : Code connu voir tableau des exceptions prédéfinies.

Exceptions personnalisées : Le DEV déclare une exception, la déclenche avec RAISE puis la traite.

Exceptions avec un code non reconnu dans le tableau des exceptions prédéfinies : Le DEV déclare une exception, puis utilise PRAGMA avec la fonction EXCEPTION\_INIT pour associer code et exception.

Exception avec RAISE\_APPLICATION\_ERROR : pour définir un code libre avec une exception (sortie du programme).

*Exemple: Traitement des exceptions par bloc imbriqué.*

DECLARE

nom varchar2(3);

BEGIN

nom:='MARTIN';

dbms\_output.put\_line(nom);

EXCEPTION

WHEN others THEN

dbms\_output.put\_line('Aie');

END;

/

Aie

Dans ce cas, après traitement de l’exception (affichage de « Aie »), nous sortons du programme.

DECLARE

nom varchar2(3);

BEGIN

BEGIN

nom:='MARTIN';

dbms\_output.put\_line(nom);

EXCEPTION

WHEN others THEN

dbms\_output.put\_line('Aie');

END;

dbms\_output.put\_line('FIN');

END;

/

Aie

FIN

Dans ce deuxième cas, après le traitement de l’exception, nous sortons du bloc imbriqué puis l’instruction suivante du bloc principal est réalisée (affichage de « Aie » puis «FIN »).

Les exceptions sont traitées suite à un test pour déterminer le type d’erreur. Si le type d’erreur n’est pas connu, l’expression OTHERS récupère l’erreur. Grâce aux variables SQLCODE pour le code et SQLERRM pour le message nous pourrons déterminer de quelle erreur il s’agissait.

*Exemple: Suite du traitement précédent mais avec récupération du code et du message.*

DECLARE

nom varchar2(3);

BEGIN

nom:='MARTIN';

dbms\_output.put\_line(nom);

EXCEPTION

WHEN others THEN

dbms\_output.put\_line(SQLCODE||' - '||SQLERRM);

END;

/

-6502 - ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: tampon de chaîne de caractères trop petit

## 

## Exceptions prédéfinies

PL/SQL permet de récupérer le code d’une erreur par la variable SQLCODE ou par le message d’erreur d’Oracle à la sortie du programme. Muni de ce code et du tableau ci-dessous, nous pourrons traiter l’exception avec le nom de l’exception associé.

|  |  |  |  |
| --- | --- | --- | --- |
| **EXCEPTION** | **Oracle** | **SQLCODE** |  |
| ACCESS\_INTO\_NULL | 06530 06530 | -6530 -6530 | A program attempts to assign values to the attributes of an uninitialized object Un programme tente d'attribuer des valeurs aux attributs d'un objet non-initialisées |
| CASE\_NOT\_FOUND CASE\_NOT\_FOUND | 06592 06592 | -6592 -6592 | None of the choices in the WHEN clauses of a CASE statement is selected, and there is no ELSE clause. Aucun des choix d’une clause CASE n’est sélectionné, et il n'y a pas de clause ELSE. |
| COLLECTION\_IS\_NULL COLLECTION\_IS\_NULL | 06531 06531 | -6531 -6531 | A program attempts to apply collection methods other than EXISTS to an uninitialized nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray. Tentative d’affectation de valeur à un tableau non initialisé. |
| CURSOR\_ALREADY\_OPEN CURSOR\_ALREADY\_OPEN | 06511 06511 | -6511 -6511 | A program attempts to open an already open cursor. Un programme tente d'ouvrir un curseur déjà ouvert. A cursor must be closed before it can be reopened. Un curseur doit être fermé avant de pouvoir être rouverte. A cursor FOR loop automatically opens the cursor to which it refers, so your program cannot open that cursor inside the loop. Une boucle FOR ouvre automatiquement le curseur sur lequel elle porte. |
| DUP\_VAL\_ON\_INDEX DUP\_VAL\_ON\_INDEX | 00001 00001 | -1 -1 | A program attempts to store duplicate values in a column that is constrained by a unique index. Un programme tente de stocker les valeurs dupliquées dans une colonne qui est limitée par un index unique. |
| INVALID\_CURSOR INVALID\_CURSOR | 01001 01001 | -1001 -1001 | A program attempts a cursor operation that is not allowed, such as closing an unopened cursor. Un programme tente une opération sue un curseur qui n'est pas autorisé, comme la fermeture d'un curseur non ouvert. |
| INVALID\_NUMBER INVALID\_NUMBER | 01722 01722 | -1722 -1722 | n a SQL statement, the conversion of a character string into a number fails because the string does not represent a valid number. La transformation d'une chaîne de caractères en un nombre échoue. |
| LOGIN\_DENIED LOGIN\_DENIED | 01017 01017 | -1017 -1017 | A program attempts to log on to Oracle Database with an invalid username or password. Un programme tente de se connecter à la base de données Oracle avec un nom d'utilisateur ou mot de passe incorrect. |
| NO\_DATA\_FOUND NO\_DATA\_FOUND | 01403 01403 | +100 +100 | A SELECT INTO statement returns no rows, or your program references a deleted element in a nested table or an uninitialized element in an index-by table. A SELECT INTO ne retourne aucune ligne ou votre programme référence un élément supprimé dans un tableau imbriqués non-initialisées ou d'un élément d'un indice. |
| NOT\_LOGGED\_ON NOT\_LOGGED\_ON | 01012 01012 | -1012 -1012 | A program issues a database call without being connected to Oracle Database. Un programme appelle une base de données sans y être relié. |
| PROGRAM\_ERROR PROGRAM\_ERROR | 06501 06501 | -6501 -6501 | PL/SQL has an internal problem. PL / SQL a un problème interne. |
| ROWTYPE\_MISMATCH ROWTYPE\_MISMATCH | 06504 06504 | -6504 -6504 | The host cursor variable and PL/SQL cursor variable involved in an assignment have incompatible return types. Incompatibilité entre les variables hôte et les variables PL/SQL.. |
| SELF\_IS\_NULL SELF\_IS\_NULL | 30625 30625 | -30625 -30625 | A program attempts to invoke a MEMBER method, but the instance of the object type was not initialized. Un programme tente d'invoquer un Membre d’une instance sans quelle soit initialisée. |
| STORAGE\_ERROR STORAGE\_ERROR | 06500 06500 | -6500 -6500 | PL/SQL ran out of memory or memory was corrupted. PL / SQL se trouve à court de mémoire ou la mémoire était corrompu. |
| SUBSCRIPT\_BEYOND\_COUNT SUBSCRIPT\_BEYOND\_COUNT | 06533 06533 | -6533 -6533 | A program references a nested table or varray element using an index number larger than the number of elements in the collection. Un programme référence un nested table ou un élément de varray à l'aide d'un indice plus grand que le nombre d'éléments de la collection. |
| SUBSCRIPT\_OUTSIDE\_LIMIT SUBSCRIPT\_OUTSIDE\_LIMIT | 06532 06532 | -6532 -6532 | A program references a nested table or varray element using an index number (-1 for example) that is outside the legal range. Un programme référence un nested table ou un élément de varray à l'aide d'un numéro d'index (-1 par exemple) qui est en dehors de la portée légale. |
| SYS\_INVALID\_ROWID SYS\_INVALID\_ROWID | 01410 01410 | -1410 -1410 | The conversion of a character string into a universal rowid fails because the character string does not represent a valid rowid. La conversion d'une chaîne de caractères en un identifiant universel échoue parce que la chaîne de caractères ne représente pas un identifiant valide. |
| TIMEOUT\_ON\_RESOURCE TIMEOUT\_ON\_RESOURCE | 00051 00051 | -51 -51 | A time out occurs while Oracle Database is waiting for a resource. Un temps mort intervient alors que la base de données Oracle est en attente d'une ressource. |
| TOO\_MANY\_ROWS TOO\_MANY\_ROWS | 01422 01422 | -1422 -1422 | A SELECT INTO statement returns more than one row. A SELECT INTO retourne plus d'une ligne. |
| VALUE\_ERROR VALUE\_ERROR | 06502 06502 | -6502 -6502 | An arithmetic, conversion, truncation, or size-constraint error occurs. Provoqué par une erreur de type arithmétique, de conversion, de troncature, de contrainte ou de taille. |
| ZERO\_DIVIDE ZERO\_DIVIDE | 01476 01476 | -1476 -1476 | A program attempts to divide a number by zero. Un programme tente de diviser un nombre par zéro. |

Pour notre exemple, nous allons récupérer le code de l’erreur du programme précédent qui est -6502. Dans le tableau ce code correspond à l’exception VALUE\_ERROR. Il suffit d’ajouter le traitement correspondant dans la section EXCEPTION.

DECLARE

nom varchar2(3);

BEGIN

nom:='MARTIN';

dbms\_output.put\_line(nom);

EXCEPTION

WHEN **VALUE\_ERROR** THEN

dbms\_output.put\_line('Valeur trop grande');

WHEN others THEN

dbms\_output.put\_line(SQLCODE||' - '||SQLERRM);

END;

/

Valeur trop grande

*Exemple: Quel est le salaire de l’employée MAF (cet employée n’existe pas)?*

DECLARE

V\_salaire emp.sal%TYPE;

BEGIN

SELECT sal INTO V\_salaire FROM emp WHERE ename = 'MAF';

dbms\_output.put\_line(V\_salaire);

EXCEPTION

WHEN others THEN

dbms\_output.put\_line(SQLCODE||' - '||SQLERRM);

END;

/

100 - ORA-01403: aucune donnée trouvée

Le SQLCODE récupéré est 100 et correspond à l’exception NO\_DATA\_FOUND.

DECLARE

V\_salaire emp.sal%TYPE;

BEGIN

SELECT sal INTO V\_salaire FROM emp WHERE ename = 'MAF';

dbms\_output.put\_line(V\_salaire);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line(‘L’’employé n''existe pas');

WHEN others THEN

dbms\_output.put\_line(SQLCODE||' - '||SQLERRM);

END;

/

L’employé n'existe pas

## Exceptions personnalisées

Des erreurs peuvent être créées par le programmeur et sui sont spécifiques à l’application. Ces exceptions seront également traitées dans la section exception mais devront être déclarées et déclenchées par le développeur.

Ces exceptions personnalisées sont déclarées dans la section déclaration au même titre qu’une variable par exemple. Elles sont ensuite déclenchées dans la section exécution par la clause RAISE en fonction du programme.

*Exemple: Si le nombre d’employé est inférieur à 15 alors embauche.*

DECLARE

NB number;

**insuffisant EXCEPTION**;

BEGIN

SELECT count(\*) INTO NB FROM emp;

IF NB < 15 THEN

**RAISE insuffisant**;

END IF;

dbms\_output.put\_line(NB);

EXCEPTION

**WHEN insuffisant THEN**

dbms\_output.put\_line('Embauche');

END;

/

Embauche

## Exceptions non prédéfinies

Quand on récupère un code erreur non associé à une exception prédéfinie, nous pouvons lui définir un nom et le traitement pourra ensuite s’effectuer exactement comme une exception prédéfinie.

Pour associer un nom à une erreur interne, nous utiliserons dans la section déclaration la clause PRAGMA avec la fonction EXCEPTION\_INIT. L’exception devra être déclarée au préalable.

**Syntaxe :**

PRAGMA EXCEPTION\_INIT(nom\_exception,code\_exception) ;

Nom\_excpetion est le nom de l’exception déclarée et associée à l’erreur.

Code\_exception est le code de l’erreur que l’on récupère.

Dans notre exemple, nous allons créer une table avec une colonne de type varchar2(5). L’insertion d’une valeur trop grande pour la taille de la colonne déclenchera une exception qui n’est pas prédéfinie. Le code de cette erreur est -12899.

CREATE TABLE T1 (nom varchar2(5));

BEGIN

INSERT INTO T1 VALUES ('MARTIN');

END;

/

ORA-12899: valeur trop grande pour la colonne "JP"."T1"."NOM" (réelle : 6, maximum : 5)

DECLARE

TROP EXCEPTION;

PRAGMA EXCEPTION\_INIT(trop,-12899);

BEGIN

INSERT INTO T1 VALUES ('MARTIN');

EXCEPTION

WHEN trop THEN

dbms\_output.put\_line('Valeur trop grande');

END;

/

Valeur trop grande

## Exceptions anonymes

L’utilisation de ce type d’exception intervient quand nous voulons déclencher une exception spécifique à notre programme et que cette exception soit traitée comme une erreur interne. Cette exception provoque la sortie du programme.

L’exception est traitée par le serveur et non par le client contrairement au cas précédent.

Le traitement utilise la procédure RAISE\_APPLICATION\_ERROR.

**Syntaxe :**

RAISE\_APPLICATION\_ERROR(code\_erreur IN NUMBER, message\_erreur IN varchar2) ;

Code\_erreur est le code choisit par le programmeur au-delà de -20000.

Message\_erreur est le message associé à ce code.

DECLARE

TROP EXCEPTION;

NB number;

BEGIN

SELECT COUNT(\*) INTO NB FROM emp;

IF NB > 10 THEN

**RAISE\_APPLICATION\_ERROR(-20001,'Trop d''employés');**

END IF;

END;

/

ORA-20001: Trop d'employés

OU

DECLARE

**TROP EXCEPTION;**

NB number;

BEGIN

SELECT COUNT(\*) INTO NB FROM salaries;

IF NB > 10 THEN

**RAISE trop;**

END IF;

EXCEPTION

**WHEN trop THEN RAISE\_APPLICATION\_ERROR(-20001,'Trop d''employés');**

END;

/

ORA-20001: Trop d'employés

Il est possible que la même exception se produise dans le même bloc. Utilisons des blocs imbriqués :

DECLARE

nom\_emp emp.ename%TYPE;

dep\_emp emp.deptno%TYPE;

BEGIN

BEGIN

SELECT ename INTO nom\_emp from emp WHERE comm=0;

dbms\_output.put\_line(nom\_emp);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Employe non trouve');

end;

begin

SELECT deptno INTO dep\_emp FROM emp WHERE deptno=63;

dbms\_output.put\_line(dep\_emp);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Département non trouve');

end;

END;

/

TURNER

Département non trouve