

Traiter les exceptions

Objectifs

A la fin de ce chapitre, vous pourrez :

- **définir des exceptions PL/SQL**
- **reconnaître les exceptions non traitées**
- **lister et utiliser les différents types de traitement des exceptions PL/SQL**
- **intercepter les erreurs non prédéfinies**
- **décrire l'effet d'une propagation des exceptions dans des blocs imbriqués**
- **personnaliser les messages d'erreurs PL/SQL**

Traiter des exceptions en PL/SQL

- Une exception est un identificateur PL/SQL détecté pendant la phase d'exécution.
- Comment est-elle déclenchée ?
 - A la suite d'une erreur Oracle.
 - Explicitement, par le programme.
- Comment la traiter ?
 - En l'interceptant à l'aide d'un gestionnaire.
 - En la propageant à l'environnement appelant.

Traiter les exceptions

Intercepter l'exception

Exception
déclenchée

Exception
interceptée

DECLARE

BEGIN

EXCEPTION

END ;

Propager l'exception

Exception
déclenchée

Exception
non
interceptée

DECLARE

BEGIN

EXCEPTION

END ;

Exception propagée
à l'environnement
appelant

Types d'exception

- Exception prédéfinie du serveur Oracle
 - Exception non prédéfinie du serveur Oracle
- } **Exception déclenchée implicitement**
-
- Exception définie par l'utilisateur
- Exception déclenchée explicitement**

Intercepter les exceptions

Syntaxe :

EXCEPTION

WHEN *exception1* [OR *exception2* . . .] THEN

statement1;

statement2;

. . .

[WHEN *exception3* [OR *exception4* . . .] THEN

statement1;

statement2;

. . .]

[WHEN OTHERS THEN

statement1;

statement2;

. . .]

Règles d'interception des exceptions

- Le mot-clé `EXCEPTION` débute la section de traitement des exceptions.
- Plusieurs gestionnaires d'exceptions sont permis.
- Un seul gestionnaire d'exceptions est exécuté avant de sortir du bloc.
- `WHEN OTHERS` est la dernière clause.

Intercepter les erreurs prédéfinies du serveur Oracle

- Utiliser le nom standard à l'intérieur du sous-programme de traitement des exceptions.
- Exemples d'exceptions prédéfinies :
 - NO_DATA_FOUND
 - TOO_MANY_ROWS
 - INVALID_CURSOR
 - ZERO_DIVIDE
 - DUP_VAL_ON_INDEX

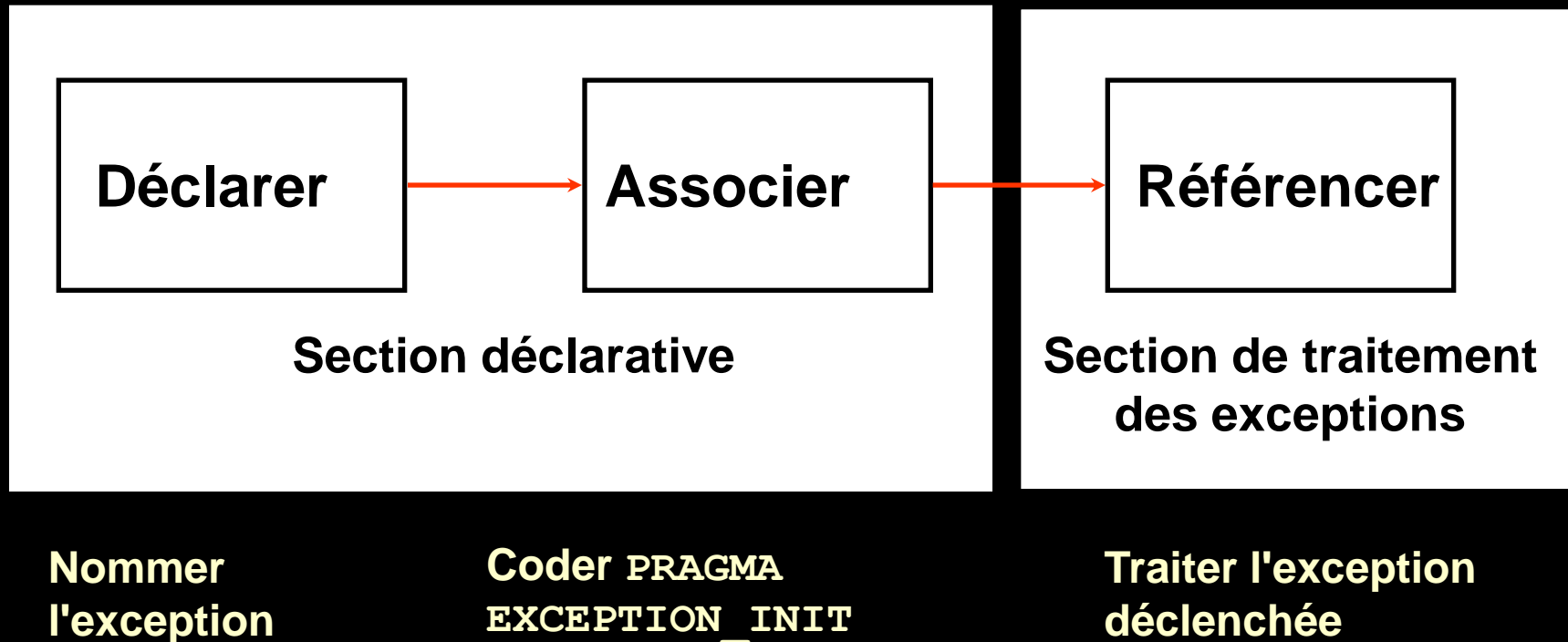
Exceptions prédéfinies

Syntaxe :

```
BEGIN
. . .
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    statement1;
    statement2;

  WHEN TOO_MANY_ROWS THEN
    statement1;
  WHEN OTHERS THEN
    statement1;
    statement2;
    statement3;
END;
```

Intercepter les erreurs non prédéfinies du serveur Oracle



Erreur non prédéfinie

Intercepter une violation de contrainte d'intégrité
(code d'erreur du serveur Oracle –2292).

```
DEFINE p_deptno = 10
DECLARE
  e_emps_remaining EXCEPTION;
  PRAGMA EXCEPTION_INIT
    (e_emps_remaining, -2292);
BEGIN
  DELETE FROM departments
  WHERE department_id = &p_deptno;
  COMMIT;
EXCEPTION
  WHEN e_emps_remaining THEN
    DBMS_OUTPUT.PUT_LINE ('Cannot remove dept ' ||
      TO_CHAR(&p_deptno) || '. Employees exist. ');
END;
```

1

2

3

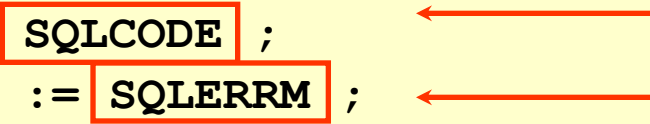
Fonctions d'interception des exceptions

- **SQLCODE** : renvoie la valeur numérique du code d'erreur
- **SQLERRM** : renvoie le message associé au code d'erreur

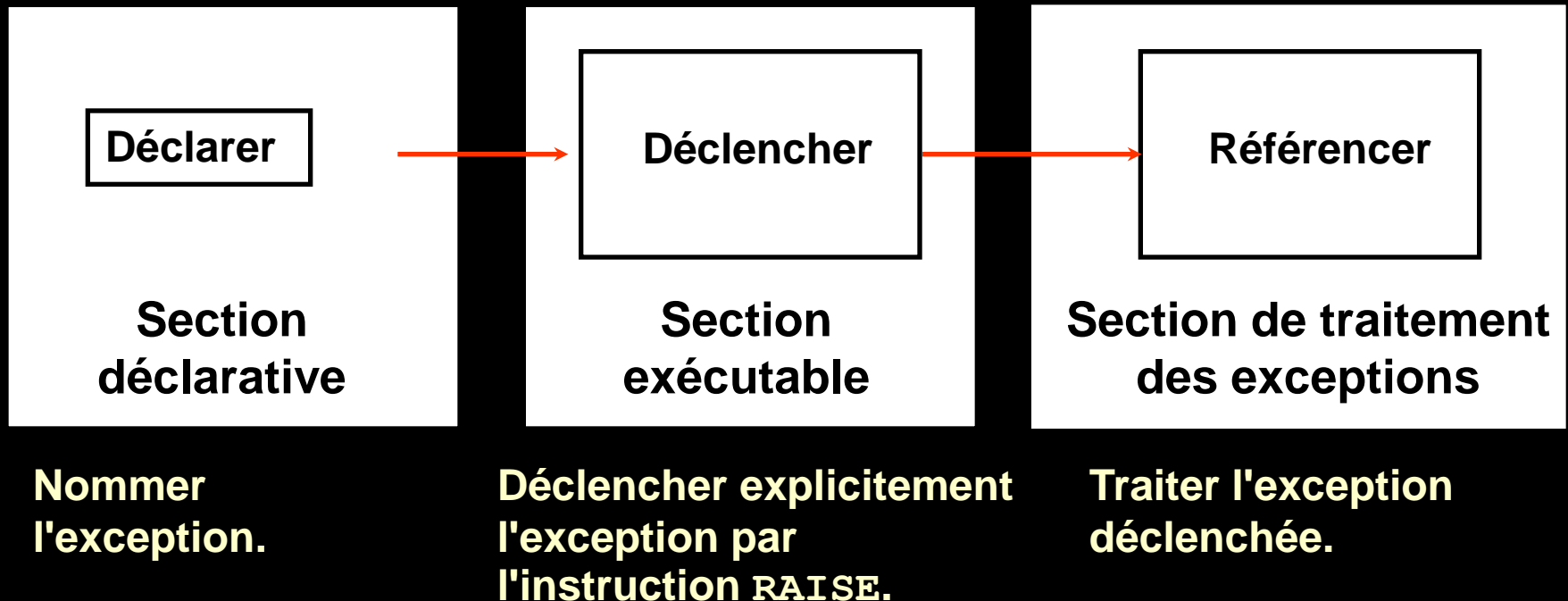
Fonctions d'interception des exceptions

Exemple :

```
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 errors
            VALUES (v_error_code, v_error_message) ;
END;
```



Intercepter les exceptions définies par l'utilisateur



Exceptions définies par l'utilisateur

Exemple :

```
DEFINE p_department_desc = 'Information Technology '  
DEFINE P_department_number = 300
```

```
DECLARE
```

```
e_invalid_department EXCEPTION;
```

```
BEGIN
```

```
UPDATE      departments  
SET          department_name = '&p_department_desc'  
WHERE        department_id = &p_department_number;  
IF SQL%NOTFOUND THEN
```

```
    RAISE e_invalid_department;
```

```
END IF;
```

```
COMMIT;
```

```
EXCEPTION
```

```
    WHEN e_invalid_department THEN
```

```
        DBMS_OUTPUT.PUT_LINE('No such department id.');
```

```
END;
```

1

2

3

Environnements appelants

iSQL*Plus	Affiche le code d'erreur et le message à l'écran
Procedure Builder	Affiche le code d'erreur et le message à l'écran
Oracle Developer Forms	Accède au code et au message d'erreur d'un déclencheur (trigger) en utilisant les fonctions de package ERROR_CODE et ERROR_TEXT.
Application précompilée	Accède au code d'erreur via la structure de données SQLCA
Bloc PL/SQL englobant	Intercepte les exceptions dans le sous-programme de traitement des exceptions du bloc englobant

Propager des exceptions

**Des sous-blocs
peuvent traiter une
exception ou la
transmettre à un bloc
englobant.**

```
DECLARE
    . . .
    e_no_rows      exception;
    e_integrity     exception;
    PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
    FOR c_record IN emp_cursor LOOP
        BEGIN
            SELECT ...
            UPDATE ...
            IF SQL%NOTFOUND THEN
                RAISE e_no_rows;
            END IF;
        END;
    END LOOP;
EXCEPTION
    WHEN e_integrity THEN ...
    WHEN e_no_rows THEN ...
END;
```

Procédure RAISE_APPLICATION_ERROR

Syntaxe :

```
raise_application_error (error_number,  
                           message[, {TRUE | FALSE}]);
```

- La procédure permet de délivrer des messages d'erreur définis par l'utilisateur à partir de sous-programmes stockés.
- Elle permet de signaler les erreurs à l'application et d'éviter le renvoi d'exceptions non traitées.

Procédure RAISE_APPLICATION_ERROR

- Elle peut être utilisée à deux endroits :
 - section exécutable
 - section de traitement des exceptions
- Elle renvoie à l'utilisateur les conditions de l'erreur de manière cohérente par rapport aux autres erreurs du serveur Oracle

RAISE_APPLICATION_ERROR

Section exécutable :

```
BEGIN
...
  DELETE FROM employees
    WHERE  manager_id = v_mgr;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20202,
      'This is not a valid manager');
  END IF;
...
```

Section de traitement des exceptions :

```
...
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR (-20201,
      'Manager is not a valid employee.');
```

END;

Synthèse

- **Types d'exception :**
 - erreur prédéfinie du serveur Oracle
 - erreur non prédéfinie du serveur Oracle
 - erreur définie par l'utilisateur
- **Intercepter une exception**
- **Traiter une exception :**
 - intercepter l'exception dans un bloc PL/SQL.
 - propager l'exception.

Présentation de l'exercice 8

Dans cet exercice, vous allez :

- **traiter des exceptions nommées**
- **créer et appeler des exceptions définies par l'utilisateur**