

Advance Database Management System

Lecture 20:

Exception Handling

Learning Objectives

2

To know about:

- PL/SQL Exception Handling

Handling Exceptions with PL/SQL

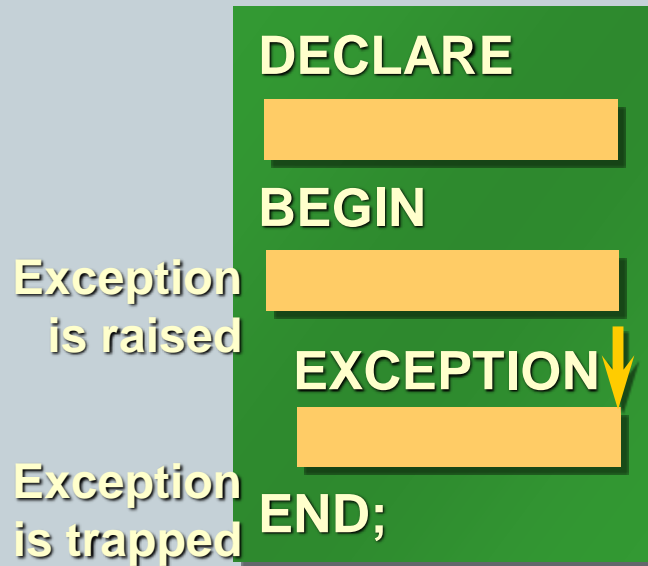
3

- What is an exception?
 - ✦ Identifier in PL/SQL that is raised during execution
- How is it raised?
 - ✦ An Oracle error occurs.
 - ✦ You raise it explicitly.
- How do you handle it?
 - ✦ Trap it with a handler.
 - ✦ Propagate it to the calling environment.

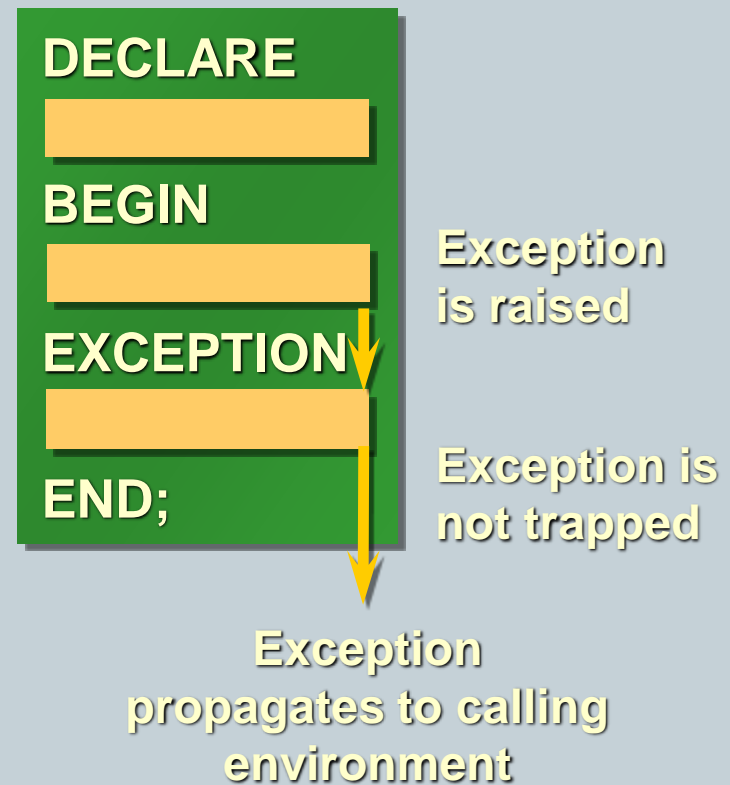
Handling Exceptions

4

Trap the exception



Propagate the exception



Exception Types

5

- Predefined Oracle Server
- Non-predefined Oracle Server
- User-defined

} Implicitly
raised
Explicitly raised

Trapping Exceptions

6

- Syntax

EXCEPTION

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

```
    statement1;
```

```
    statement2;
```

```
    . . .
```

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

```
    statement1;
```

```
    statement2;
```

```
    . . .]
```

```
[WHEN OTHERS THEN
```

```
    statement1;
```

```
    statement2;
```

```
    . . .]
```

Trapping Exceptions Guidelines

7

- WHEN OTHERS is the last clause.
- EXCEPTION keyword starts exception-handling section.
- Several exception handlers are allowed.
- Only one handler is processed before leaving the block.

Trapping Predefined Oracle Server Errors

8

- Reference the standard name in the exception-handling routine.
- Sample predefined exceptions:
 - ✦ NO_DATA_FOUND
 - ✦ TOO_MANY_ROWS
 - ✦ INVALID_CURSOR
 - ✦ ZERO_DIVIDE
 - ✦ DUP_VAL_ON_INDEX

Predefined Exception

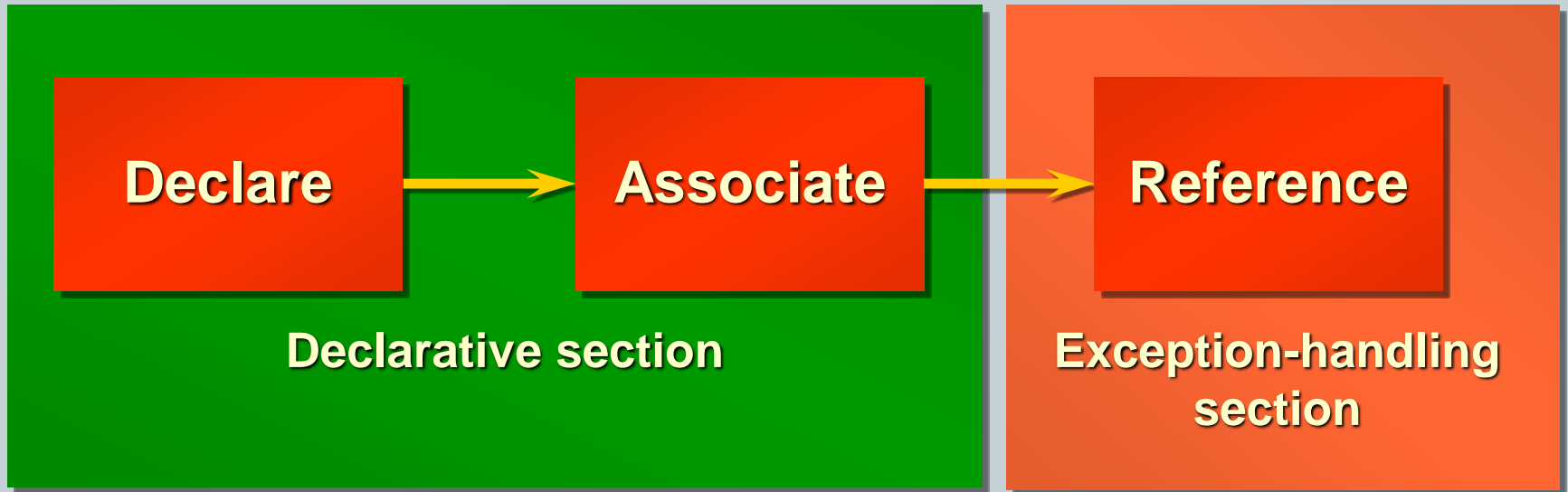
9

- Syntax

```
BEGIN  SELECT ... COMMIT;  
EXCEPTION  
  WHEN NO_DATA_FOUND THEN  
    statement1;  
    statement2;  
  WHEN TOO_MANY_ROWS THEN  
    statement1;  
  WHEN OTHERS THEN  
    statement1;  
    statement2;  
    statement3;  
END;
```

Trapping Non-Predefined Oracle Server Errors

10



- Name the exception
- Code the PRAGMA EXCEPTION_INIT
- Handle the raised exception

Non-Predefined Error

11

- Trap for Oracle Server error number –2292, an integrity constraint violation.

```
DECLARE
    e_emp_remaining EXCEPTION;
    PRAGMA EXCEPTION_INIT (
        e_emp_remaining, -2292);
    v_deptno    dept.deptno%TYPE := &p_deptno;
BEGIN
    DELETE FROM dept
    WHERE      deptno = v_deptno;
    COMMIT;
EXCEPTION
    WHEN e_emp_remaining THEN
        DBMS_OUTPUT.PUT_LINE ('Cannot remove dept ' ||
            TO_CHAR(v_deptno) || '. Employees exist. ');
END;
```

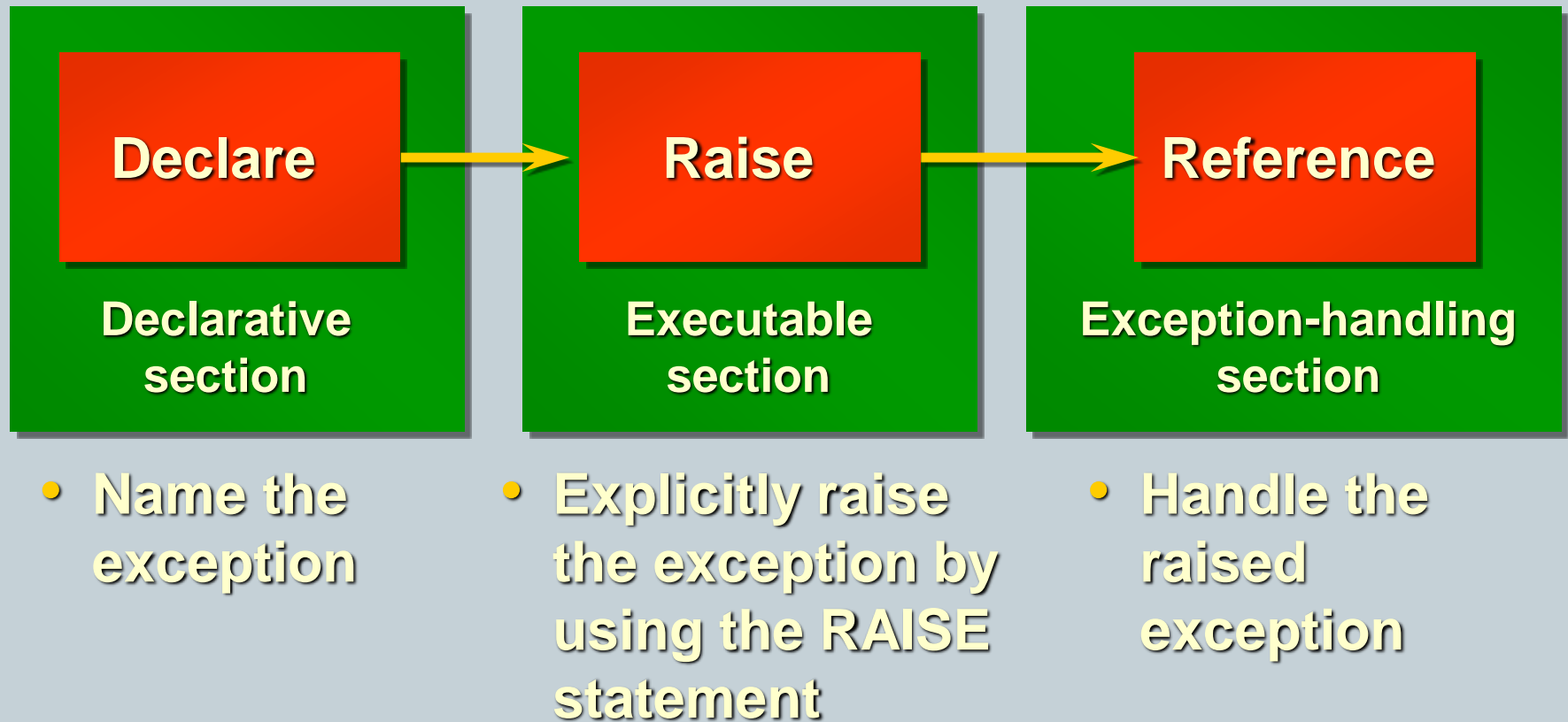
1

2

3

Trapping User-Defined Exceptions

12



User-Defined Exception

13

Example

```
DECLARE
  e_invalid_product EXCEPTION;
BEGIN
  UPDATE      product
  SET         descrip = '&product_description'
  WHERE       prodid = &product_number;
  IF SQL%NOTFOUND THEN
    RAISE e_invalid_product;
  END IF;
  COMMIT;
EXCEPTION
  WHEN e_invalid_product THEN
    DBMS_OUTPUT.PUT_LINE('Invalid product number. ');
END;
```

1

2

3

Functions for Trapping Exceptions

14

- **SQLCODE**

Returns the numeric value for the error code

- **SQLERRM**


Returns the message associated with the error number

Functions for Trapping Exceptions

15

- Example

```
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;
```



Calling Environments

16

SQL*Plus	Displays error number and message to screen
Procedure Builder	Displays error number and message to screen
Oracle Developer Forms	Accesses error number and message in a trigger by means of the ERROR_CODE and ERROR_TEXT packaged functions
Precompiler application	Accesses exception number through the SQLCA data structure
An enclosing PL/SQL block	Traps exception in exception-handling routine of enclosing block

Propagating Exceptions

17

Subblocks can handle an exception or pass the exception to the enclosing block.

```
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;
        EXCEPTION
            WHEN e_integrity THEN ...
            WHEN e_no_rows THEN ...
        END;
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN . . .
    WHEN TOO_MANY_ROWS THEN . . .
END;
```

RAISE_APPLICATION_ERROR Procedure

18

- Syntax

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

- A procedure that lets you issue user-defined error messages from stored subprograms
- Called only from an executing stored subprogram

RAISE_APPLICATION_ERROR Procedure

19

- Used in two different places:
 - ✦ Executable section
 - ✦ Exception section
- Returns error conditions to the user in a manner consistent with other Oracle Server errors

THANK YOU