eBOX BY AMPHISOFT

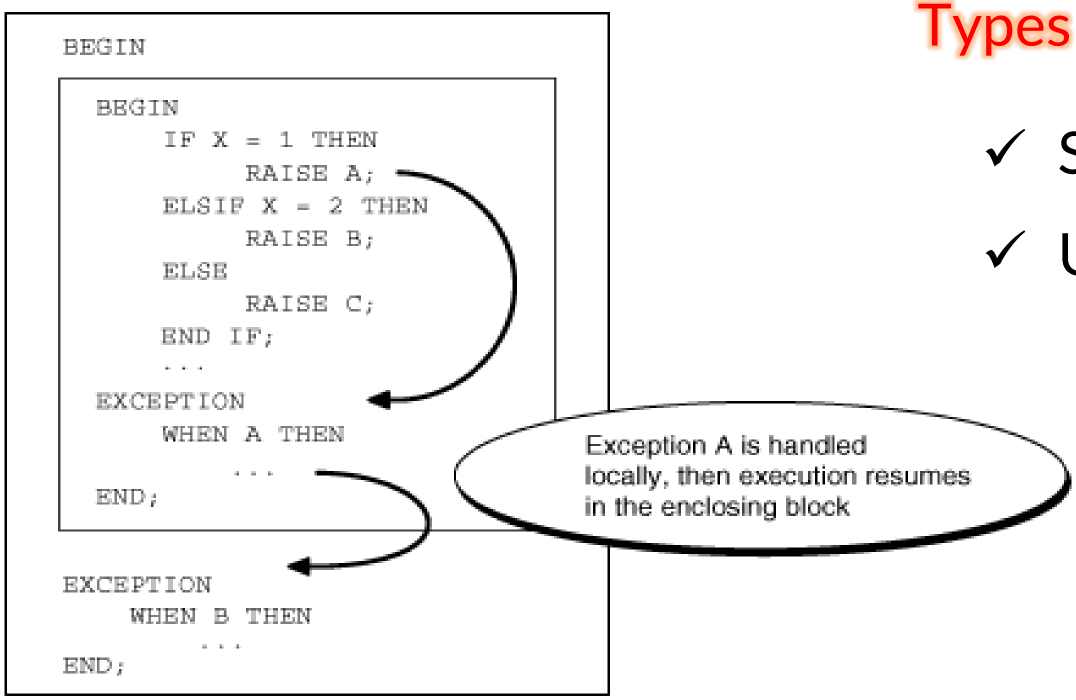
PL/SQL Exception

In this session, you will learn:

- Introduction to PL/SQL Exception
- An example for System-defined Exceptions
- How to define an User-defined exception
- How to raise an exception

Introduction to PL/SQL Exception

- An exception is defined as a special condition that changes the program execution flow.
- PL/SQL catches and handles exceptions by using exception handler architecture.



```
BEGIN
  BEGIN
      IF X = 1 THEN
           RAISE A;
      ELSIF X = 2 THEN
           RAISE B;
      ELSE
           RAISE C;
      END IF;
      ...
  EXCEPTION
      WHEN A THEN
           ...
  END;
EXCEPTION
    WHEN B THEN
         ...
END;
```

Exception A is handled locally, then execution resumes in the enclosing block

Types

- ✓ System-defined Exceptions
- ✓ User-defined Exceptions

System-Defined Exception – Example

Example 2

Create a function named 'findPdtType' that will accept the Pdt_Id as input.
Based on this input, the function must return the product type of varchar type.

Function name : findPdtType
Input Parameter : Pdt_Id of int type

Design rules:
1)If the Pdt_Id passed as input, matches with the Product_Id in the Product table, then it returns the product type of the given Pdt_Id.
2)If the Pdt_Id passed as input, does not match with the Product_Id in the Product table, then it throws 'no_data_found' exception and displays it with the text as 'No such Product

Exception	Oracle Error	SQL Code	Description
NO_DATA_FOUND	01403	+100	It is raised when a select into statement returns no rows.
ROWTYPE_MISMATCH	06504	-6504	It is raised when a cursor fetches value in a variable having incompatible data type.
TOO_MANY_ROWS	01422	-1422	It is raised when a SELECT INTO statement returns more than one row.
VALUE_ERROR	06502	-6502	It is raised when an arithmetic, conversion, truncation, or size-constraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt is made to divide a number by zero.

System-Defined Exception – Example

Example 2

```
CREATE OR REPLACE FUNCTION findPdtType (Pdt_Id IN INT) RETURN VARCHAR
IS
    type_name varchar(30) ;
BEGIN
    SELECT Pdt_Type INTO type_name FROM Product WHERE Product_Id = Pdt_Id;
    RETURN type_name;
EXCEPTION
   WHEN no_data_found THEN
   type_name  := 'No such Product';
   RETURN type_name;
end;
/
```

```
SET SERVEROUTPUT ON
DECLARE
       name varchar(30);
BEGIN
       name := findPdtType(300);
       dbms_output.put_line(name);
END;
/
```

How to define an User-Defined Exception

Syntax

```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
EXCEPTION
   <exception handling goes here >
   WHEN exception1 THEN
       exception1-handling-statements
   WHEN exception2  THEN
      exception2-handling-statements
   ........
   WHEN others THEN
      exception3-handling-statements
END;
```

User-Defined Exception – Example

Example

```
set serveroutput on
DECLARE
  p_id Product.Product_Id%type := -100;
  ptype  Product.Pdt_Type%type;
  ex_invalid_id  EXCEPTION;
BEGIN
  IF p_id <= 0 THEN
    RAISE ex_invalid_id;
  ELSE
    SELECT Pdt_Type INTO  ptype FROM Product WHERE Product_Id = p_id;
    DBMS_OUTPUT.PUT_LINE ('Product Type : '||  ptype);
  END IF;
EXCEPTION
  WHEN ex_invalid_id THEN
    dbms_output.put_line('ID must be greater than zero!');
  WHEN no_data_found THEN
    dbms_output.put_line('No such product!');
END;
/
```

→

```
ID must be greater then zero!

PL/SQL procedure successfully completed.
```

RAISE_APPLICATION_ERROR

Example

```
set serveroutput on
DECLARE
   balance integer := 24;
BEGIN
   IF balance <= 100 THEN
      RAISE_APPLICATION_ERROR(-20343, 'The balance is too low.‘);
   END IF;
END;
/
```

↓

```
ORA-20343: The balance is too low.
```

THANKS

eBOX
BY AMPHISOFT