7 1

Agustín Alejandro Mota Hinojosa

November 26, 2023

Contents

1 Vocabulary

1. Exception Handler Code that defines the recovery actions to be performed when execution-time errors occur.
2. Exception Occurs when an error is discovered during the execution of a program that disrupts the normal operation of the program.

2 Try It / Solve It

1. What happens when Oracle encounters a runtime problem while executing a PL/SQL block?

   Launches an Exception

2. What do you need to add to your PL/SQL block to address these problems?

   An exception handler

3. List three advantages of handling exceptions within a PL/SQL block.

   (a) Protects users from errors
   (b) Protects database from errors
   (c) Protects from software crashes

4. Run this PL/SQL code and then answer the questions that follow.

```
DECLARE
        v_jobid employees.job_id%TYPE;
BEGIN
      SELECT job_id INTO v_jobid FROM employees WHERE department_id = 80;
END;
```

1. What happens when you run the block? In your own words, explain what you can do to fix this problem.

   The server returns an error. You can fix this by changing the code so the `department_id` is not 80.

2. Modify the code to fix the problem. Use a TOO$_{\text{MANYROWS}}$ exception handler.

   ```
   DECLARE
       v_jobid employees.job_id%TYPE;
   BEGIN
       SELECT job_id INTO v_jobid FROM employees WHERE department_id = 80;

   EXCEPTION
       WHEN NO_DATA_FOUND THEN
           DBMS_OUTPUT.PUT_LINE('No data found for the given condition.');
       WHEN TOO_MANY_ROWS THEN
           DBMS_OUTPUT.PUT_LINE('Too many rows returned for the given condition.');
       WHEN OTHERS THEN
           DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLCODE || ' - ' || SQLERRM
   END;
   ```

3. Run your modified code. What happens this time?

   It prints a message when the exception is launched

4. Run the following PL/SQL block, which tries to insert a new row (with department$_{\text{id}}$ = 50) into the departments table. What happens and why?

   ```
   BEGIN
       INSERT INTO departments (department_id, department_name, manager_id, locatio
   EXCEPTION
   ```

```
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('An exception has occurred.');
    END;
```

This block tries to insert a new row into the departments table. If the insertion is successful, it will print 'The new department was inserted.' If an exception occurs, it will print 'An exception has occurred.' The outcome depends on factors like whether there is a department with the ID 50, constraints on the table, etc.

5. Enter the following code to create a copy of the employees table for this and the next question.

```
    CREATE TABLE emp_temp AS SELECT * FROM employees;
```

In the new emp_{temp} table, delete all but one of the employees in department 10.

```
SELECT * FROM emp_temp WHERE department_id = 10;
DELETE FROM emp_temp WHERE employee_id = ...; (repeat as necessary)
```

Enter the following PL/SQL block, which tries to SELECT all the employees in a specific department. Run it three times, using $\text{department}_{\text{ids}}$ 10, 20, and 30. What happens and why?

```
    DECLARE
        v_employee_id employees.employee_id%TYPE;
        v_last_name employees.last_name%TYPE;
    BEGIN
        SELECT employee_id, last_name INTO v_employee_id, v_last_name
        FROM employees
        WHERE department_id = 10; -- run with values 10, 20, and 30
        DBMS_OUTPUT.PUT_LINE('The SELECT was successful');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An exception has occurred');
    END;
```

This block tries to select an employee's ID and last name from the employees table for a specified department. If the select is successful, it prints 'The SELECT was successful.' If an exception occurs, it prints 'An exception has occurred.'

1. Modify your code from question 6 to add two more exception handlers to trap the possible exceptions individually. Use NO$_{\text{DATAFOUND}}$ and TOO$_{\text{MANYROWS}}$. Re-run the block three times, using 10, 20, and 30 as before. Observe the message displayed in each case. When finished, remember to delete the emp$_{\text{temp}}$ table. DROP TABLE emp$_{\text{temp}}$;

```
DECLARE
    v_employee_id employees.employee_id%TYPE;
    v_last_name employees.last_name%TYPE;
BEGIN
    SELECT employee_id, last_name INTO v_employee_id, v_last_name
    FROM employees
    WHERE department_id = 10; -- Run with values 10, 20, and 30
    DBMS_OUTPUT.PUT_LINE('The SELECT was successful');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for the given condition.');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Too many rows returned for the given condition.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception has occurred');
END;
```

2. List three guidelines for trapping exceptions.

   (a) Specific Exception Handling: Handle specific exceptions when possible (e.g., NO$_{\text{DATAFOUND}}$, TOO$_{\text{MANYROWS}}$) rather than using the generic OTHERS exception.

   (b) Logging and Messaging: Implement proper logging and messaging within the exception handlers to provide meaningful information about the error.

   (c) Graceful Degradation: Handle exceptions gracefully, ensuring that the application can degrade gracefully in the face of unexpected issues.

3. Enter and run the following PL/SQL block. Explain the output. Note: the WHEN OTHERS handler successfully handles any type of exception which occurs.

```
DECLARE
```

```
    v_number NUMBER(2);
BEGIN
    v_number := 9999;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception has occurred');
END;
```

This block sets v_{number} to 9999. Since there are no exceptions, it will not enter the exception block. The output will be nothing because there's no exception.

4. Modify the block in question 9 to omit the exception handler, then re-run the block. Explain the output.

   it will likely get a standard Oracle error message.

5. Enter and run the following code and explain the output.

```
    DECLARE
        v_number NUMBER(4);
    BEGIN
        v_number := 1234;

        DECLARE
            v_number NUMBER(4);
        BEGIN
            v_number := 5678;
            v_number := 'A character string';
        END;

    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An exception has occurred');
            DBMS_OUTPUT.PUT_LINE('The number is: ' || v_number);
    END;
```

This code has a nested PL/SQL block that declares v_{number} again. The inner block assigns a character string to v_{number}, which conflicts with the outer block's declaration of v_{number} as a NUMBER. This will result in a runtime error when the inner block is executed. The OTHERS exception handler catches the error and prints a message along with the value of $\mathrm{v}_{\mathrm{number}}$ at that point in the code.