Table of Contents

[CHAPTER 11: EXCEPTION HANDLING 2](#_Toc416718174)

[Theory 2](#_Toc416718175)

[AIM 6](#_Toc416718176)

[Lab Exercise 11: EXCEPTION HANDLING 7](#_Toc416718177)

[1. Predefined Exception 8](#_Toc416718178)

[2. Internally Defined Exception 23](#_Toc416718179)

[3. User-Defined Exception 33](#_Toc416718180)

[4. Raise and View Exception 36](#_Toc416718181)

[5. Exception Propagation 42](#_Toc416718182)

[6. Bulk Exception 54](#_Toc416718183)

[SUMMARY 57](#_Toc416718184)

[REFERENCES 58](#_Toc416718185)

[INDEX 59](#_Toc416718186)

# CHAPTER 11: EXCEPTION HANDLING

## Theory

PL/SQL handles exception in EXCEPTION part of the block. It is simple as that. However, it turns to a cumbersome work if you miss understand some concepts about exceptions structure and propagation. In this chapter, you will examine these issues.

Exception refers to runtime error. In comparison, compile-time error and warning are not classified as exception. You may show a compile-time error and warning using SQL\*PLUS command:

**SHOW ERRORS**

The basic structure of Oracle EXCEPTION is as shown below:

Name

**SQLCODE**

**SQLERRM**

Message

Code

1. Code: is a number that uniquely identify the exception.
2. Name: is a character string that represents a given name for the exception.
3. Message: is a character string that represents a description of the exception.

You can get the exception code using SQLCODE function and get its message using SQLERRM.

There are three exception categories:

1. **Internally Defined**: it is automatically defined and raised by Oracle database. Internally defined exception always has code and message but not name.
2. **Predefined**: it is the same as Internally Defined Exception but PL/SQL gives it name in STANDARD package.
3. **User-defined**: this is the exception that you define and raise.

You may give name for Oracle Internally Defined Exception and you may override the given name of Predefined Exception. The name is very important when you handle the exception. You can't handle a specific exception without a given name. However, you may handle it ambiguously under OTHERS section. The next figure shows the basic structure of the EXCEPTION part in PL/SQL block.

**EXCEPTION**  
 **WHEN** *exception1\_name* **THEN**   
 statement(s);  
 **WHEN** **OTHERS** **THEN**   
 statement(s);  
 **WHEN** ***exception2\_name* THEN**   
 statement(s);  
**END**;

**EXCEPTION**  
 **WHEN** *exception1\_name* **THEN**   
 statement(s);  
 **WHEN** *exception2\_name* **THEN**   
 statement(s);  
 **WHEN** **OTHERS** **THEN**   
 statement(s);  
**END**;

**EXCEPTION**  
 **WHEN** *exception1\_name* **THEN**   
 statement(s);  
 **WHEN** ***exception2\_name*** **THEN**   
 statement(s);  
 **WHEN** **OTHERS** **THEN**   
 statement(s);  
**END**;

After EXCEPTION keyword, you should decode for exception **name**. If the exception has no name, it will be trapped into OTHERS section, If OTHERS section exists. It is important to know that OTHERS section will trap any exception not listed above. Therefore, all exceptions listed under OTHERS section will never be reached.

Exception Propagation is an important topic you must know about. Exception Propagation refers to the process in which the exception from one PL/SQL block to another until being trapped or returns an unhandled exception error to the invoker or host environment. There are three parts of the block: Declaration, Executable, and Exception parts. If an exception is raised from the Executable part of the block, it will be checked first in its Exception part. If there is no handler for it in the block, it propagates to the next enclosing block. Again, if there is no handler for it in the next enclosing block, it propagates to the next, and so forth.

BEGIN  
  
...  
EXCEPTION  
 WHEN z THEN   
 ....  
END;

**There is no x handler in inner /outer block.**

BEGIN  
  
  
  
  
  
  
  
  
  
.....  
.....  
....  
EXCEPTION  
 WHEN z THEN  
END;

BEGIN  
...  
RAISE **x;**  
...  
EXCEPTION  
 **WHEN x THEN** .....  
END;

Please note: in the second example, the inner block handle the exception x. After executing statement under x handler, the execution moves to the first statement of the outer block. It does not continue executing the statements after "RAISE x;" statement.

BEGIN  
  
  
  
  
  
  
  
  
EXCEPTION  
 **WHEN x THEN** ......   
END;

BEGIN  
......  
**EXCEPTION** WHEN a THEN  
 **RAISE x;  
 WHEN x THEN** **....**END;

**DECLARE** .... **RAISE x**BEGIN  
......  
EXCEPTION  
 **WHEN x THEN** .....  
END;

BEGIN  
  
  
  
  
  
  
  
  
EXCEPTION  
 **WHEN x THEN** ......   
END;

In comparison, if the exception is raise in the Declaration part of the block, it propagates to the next enclosing block's Exception-Handling part, not to its own part. Similarly, if the exception is raise in the Exception-Handling part of the block, it propagates to the next enclosing block's Exception-Handling part, not to its own part. Please note: the second example shows that inner block raise exception "x" and there is an "x" exception handler in the same block. However, it does not handle the exception.

## AIM

The AIM of the following exercise is to demonstrate how to deal with different kinds of exceptions in PL/SQL block.

The steps involved will include:

* Predefined Exception
* Internally Defined Exception
* User-Defined Exception
* Raise and View Exception
* Exception Propagation
* Bulk Exception

In general, lab exercises are done in sequential order. Thus, it is assumed that you successfully completed the previous labs. However, not all previous labs are required. Please be sure to run the following lab before proceeding:

* Installing Oracle Database 12c.
* Bulk SQL and Bulk Binding

Estimated Completion Time:

35 minutes

# Lab Exercise 11: EXCEPTION HANDLING

|  |
| --- |
|  |

## Predefined Exception

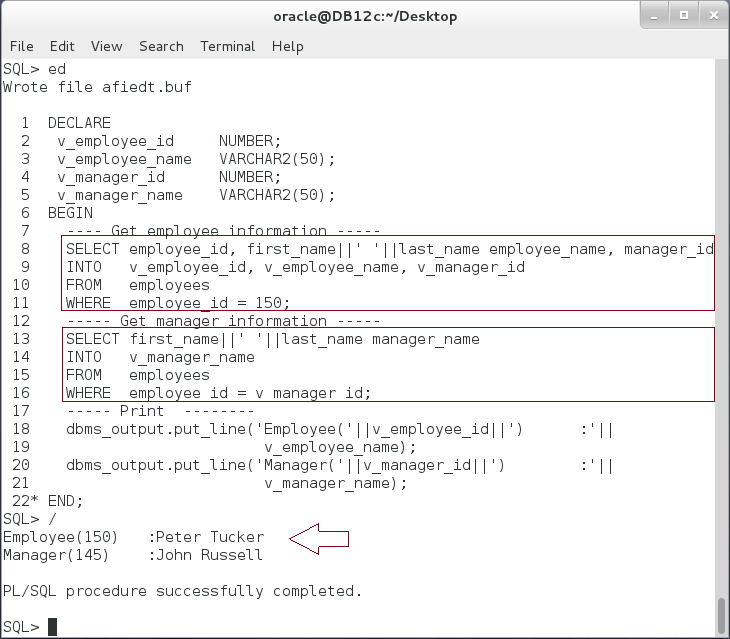
**Step 1:** Open the Terminal, open SQL\*Plus console and connect to hr schema.

|  |  |
| --- | --- |
| Command | Description |
| sqlplus | Open SQL\*Plus console. |
| hr/oracle | connect to **hr** schema. |

****

**Step 2:** Execute the following block:

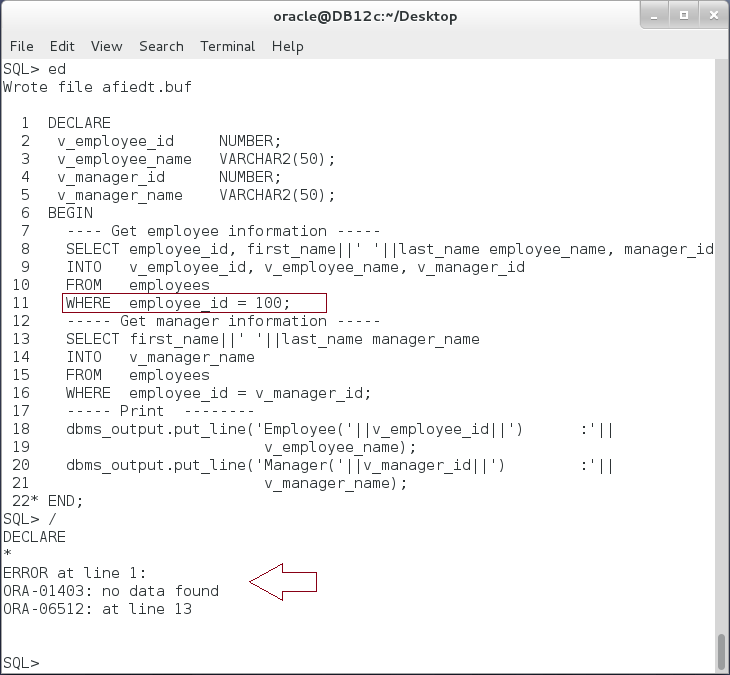
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| BEGIN |  |
| ---- Get employee information ----- |  |
| **SELECT** employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | **Select employee** |
| **INTO** v\_employee\_id, v\_employee\_name, v\_manager\_id |
| **FROM** employees |
| **WHERE** employee\_id = 150; |
| ----- Get manager information ----- |  |
| **SELECT** first\_name||' '||last\_name manager\_name | **Select manager** |
| **INTO** v\_manager\_name |
| **FROM** employees |
| **WHERE** employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| END; |  |
| / |  |

****

**Please note**: the previous PL/SQL block executed successfully.

**Step 3:** Modify the previous PL/SQL block as show below:

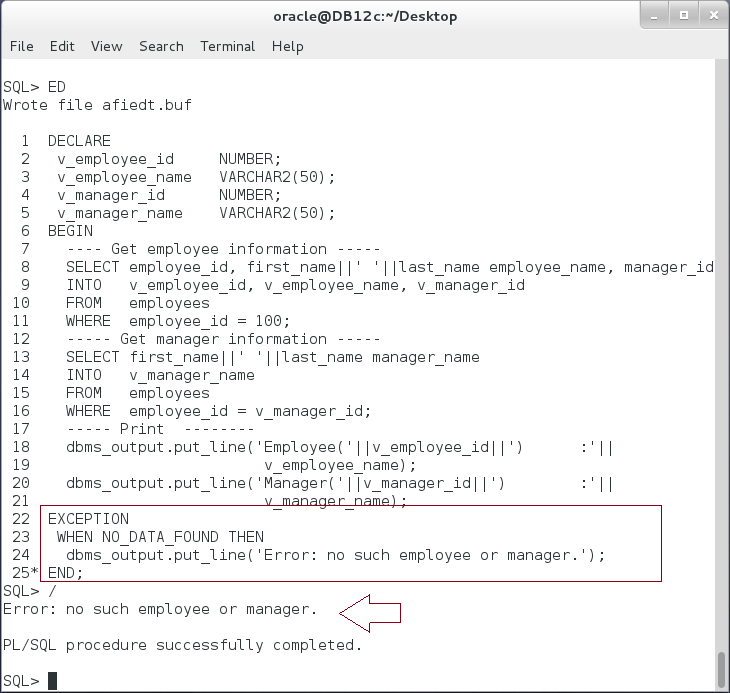
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| BEGIN |  |
| ---- Get employee information ----- |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | **Select another employee id:100** |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| **WHERE employee\_id = 100**; |
| ----- Get manager information ----- |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| END; |  |
| / |  |



**Please note:** you got a runtime error. The error is not in the syntax, it is in the data retrieved or used in executing the block. In our case, employee id was 150 and became 100.

**Step 4:** This is a predefined exception "NO\_DATA\_FOUND". Modify the previous block as shown below:

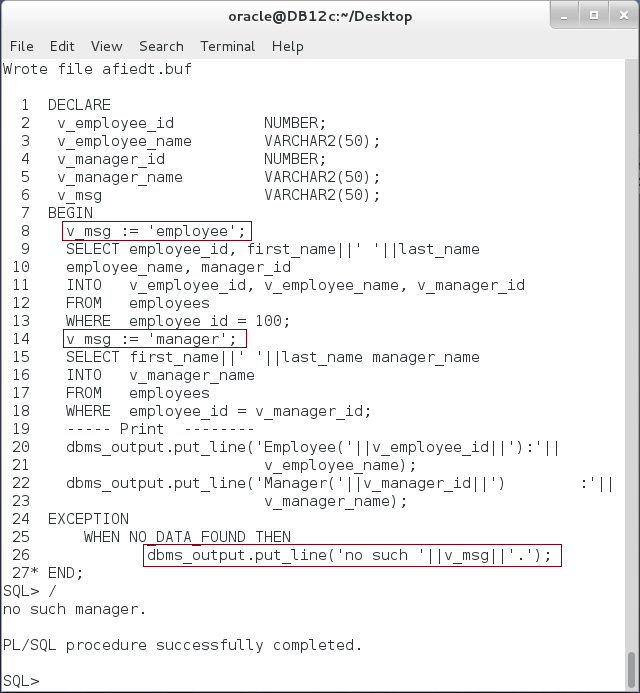
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| BEGIN |  |
| ---- Get employee information ----- |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | Select employee id:100 |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| WHERE employee\_id = 100; |
| ----- Get manager information ----- |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| **EXCEPTION** |  |
| **WHEN NO\_DATA\_FOUND THEN** |  |
| dbms\_output.put\_line(' no such employee or manager.'); |  |
| END; |  |
| / |  |

****

**Please note:** The exception handler catches the error raised from any query inside the body, no matter how many they are. In our case, there are two SELECT statements that can raise NO\_DATA\_FOUND. **Which statement is that?**

**Step 5:** To know which statement and to generate more precise error message, you may use a step technique. Modify the previous block as shown below:

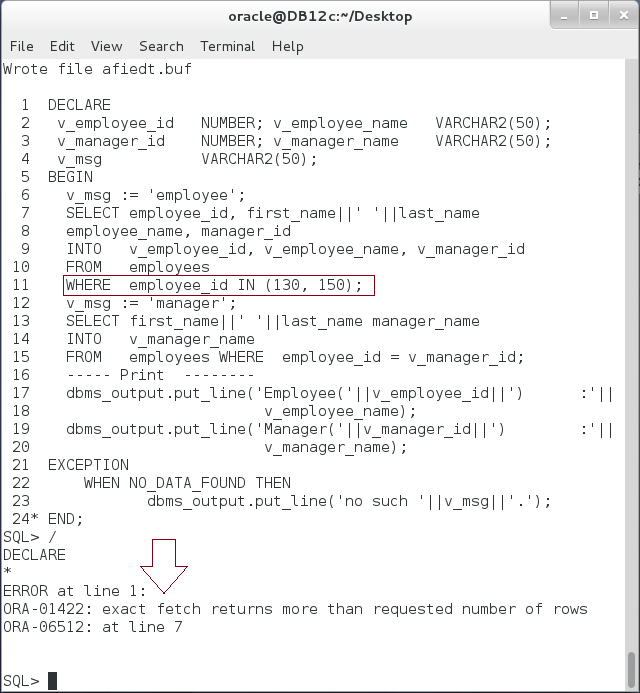
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| **v\_msg VARCHAR2(50);** |  |
| BEGIN |  |
| **v\_msg := 'employee';** |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | Select employee id:100 |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| WHERE employee\_id = 100; |
| **v\_msg := 'manager';** |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| EXCEPTION |  |
| WHEN NO\_DATA\_FOUND THEN |  |
| **dbms\_output.put\_line('no such '||v\_msg||'.');** |  |
| END; |  |
| / |  |



**Please note**: You should change the value of "**v\_msg**" before executing the statement. If the statement failed, then its value refers to the failed statement.

**Step 6:** There are many other predefined exceptions which you may regularly encounter. Change the previous block as shown below:

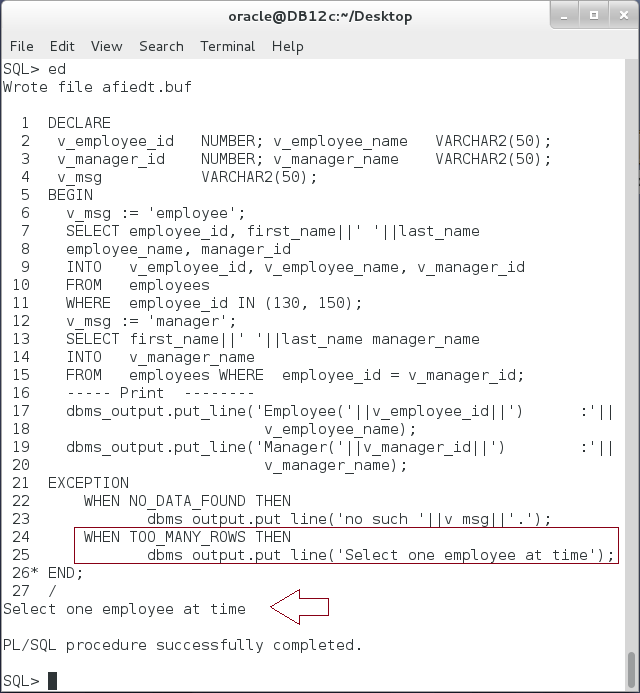
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| v\_msg VARCHAR2(50); |  |
| BEGIN |  |
| v\_msg := 'employee'; |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | **Select employee id: (150,130)** |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| **WHERE employee\_id IN (150,130);** |
| v\_msg := 'manager'; |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| EXCEPTION |  |
| WHEN NO\_DATA\_FOUND THEN |  |
| dbms\_output.put\_line('no such '||v\_msg||'.'); |  |
| END; |  |
| / |  |



**Please Note:** The first SELECT statement raised unhandled exception: Too many rows.

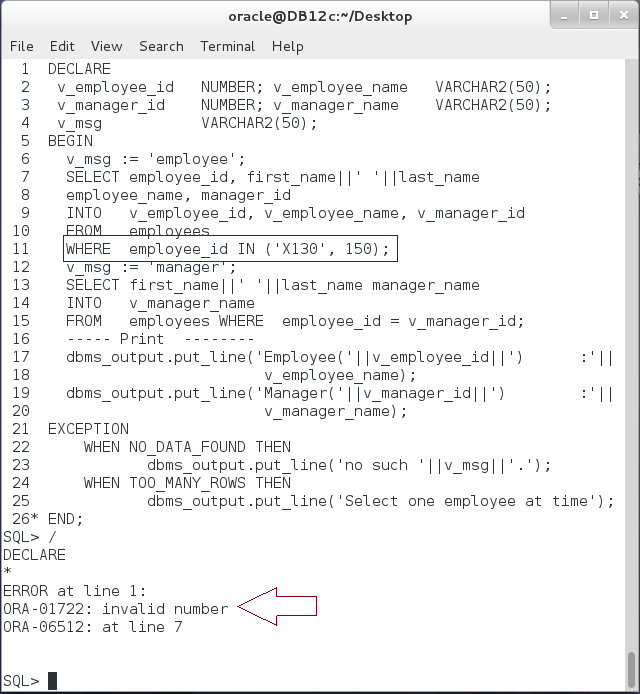
**Step 7:** Execute the following PL/SQL block:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| v\_msg VARCHAR2(50); |  |
| BEGIN |  |
| v\_msg := 'employee'; |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | **Select employee id: (150,130)** |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| **WHERE employee\_id IN (150,130);** |
| v\_msg := 'manager'; |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| EXCEPTION |  |
| WHEN NO\_DATA\_FOUND THEN |  |
| dbms\_output.put\_line('No such '||v\_msg||'.'); |  |
| **WHEN TOO\_MANY\_ROWS THEN** |  |
| **dbms\_output.put\_line('Select one employee at time');** |  |
| END; |  |
| / |  |

****

**Step 8:** Modify the block again as show below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_employee\_name VARCHAR2(50); |  |
| v\_manager\_id NUMBER; |  |
| v\_manager\_name VARCHAR2(50); |  |
| v\_msg VARCHAR2(50); |  |
| BEGIN |  |
| v\_msg := 'employee'; |  |
| SELECT employee\_id, first\_name||' '||last\_name  employee\_name, manager\_id | **Select employee id: ('X130','150')** |
| INTO v\_employee\_id, v\_employee\_name, v\_manager\_id |
| FROM employees |
| **WHERE employee\_id IN ('X130', '150');** |
| v\_msg := 'manager'; |  |
| SELECT first\_name||' '||last\_name manager\_name | Select manager |
| INTO v\_manager\_name |
| FROM employees |
| WHERE employee\_id = v\_manager\_id; |
| ----- Print -------- |  |
| dbms\_output.put\_line('Employee('||v\_employee\_id||'):'|| |  |
| v\_employee\_name); |  |
| dbms\_output.put\_line('Manager('||v\_manager\_id||') :'|| |  |
| v\_manager\_name); |  |
| EXCEPTION |  |
| WHEN NO\_DATA\_FOUND THEN |  |
| dbms\_output.put\_line('no such '||v\_msg||'.'); |  |
| WHEN TOO\_MANY\_ROWS THEN |  |
| dbms\_output.put\_line('Select one employee at time'); |  |
| END; |  |
| / |  |

****

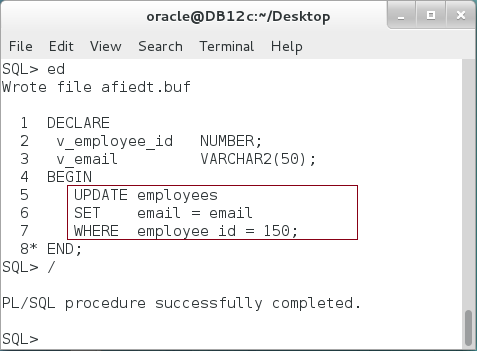
**Step 9:** Go to Oracle Documentation in the following link and find the predefined exception name to handle the previous error:

|  |
| --- |
| Link |
| https://docs.oracle.com/database/121/LNPLS/errors.htm#LNPLS00703 |

## Internally Defined Exception

**Step 1:** Execute the following PL/SQL block:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| BEGIN |  |
| UPDATE employees | **Update an employee record** |
| SET email = email |
| WHERE employee\_id = 150; |
| END; |  |
| / |  |



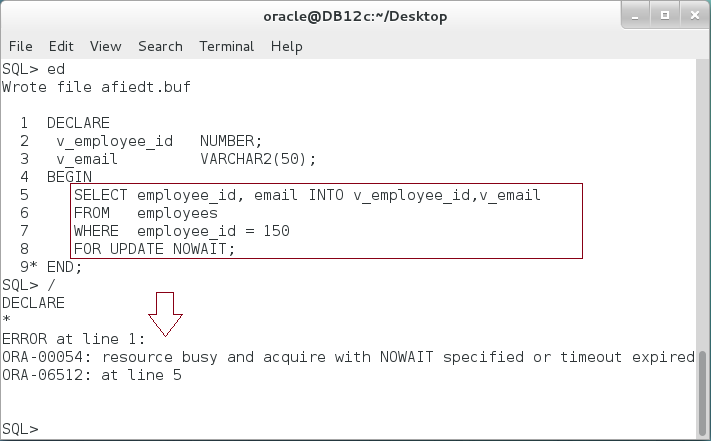
**Step 2:** Keep the previous session opened and open another session as show below:

|  |  |
| --- | --- |
| Command | Description |
| sqlplus | Open SQL\*Plus console. |
| hr/oracle | connect to **hr** schema. |

****

**Step 3:** Modify the previous block as show below:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| BEGIN |  |
| **SELECT** employee\_id, email **INTO** v\_employee\_id,v\_email | Select the same employee record with **FOR UPDATE NOWAIT** option. |
| **FROM** employees |
| **WHERE** employee\_id = **150** |
| **FOR UPDATE NOWAIT;** |
| END; |  |
| / |  |

****

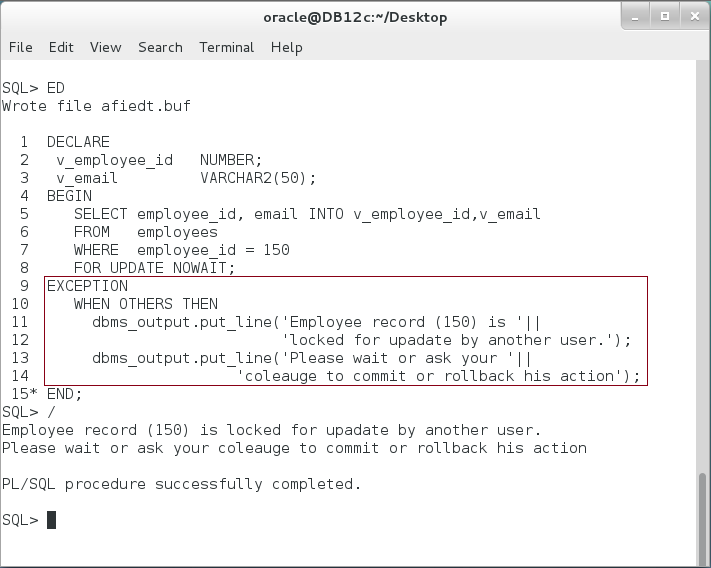
**Please note:** The block raised unhandled exception: resource busy.

**Step 4:** Try to find a predefined exception to handle this exception and write a more specific message to the user. Find all predefined exceptions in the following link:

|  |
| --- |
| Link |
| https://docs.oracle.com/database/121/LNPLS/errors.htm#LNPLS00703 |

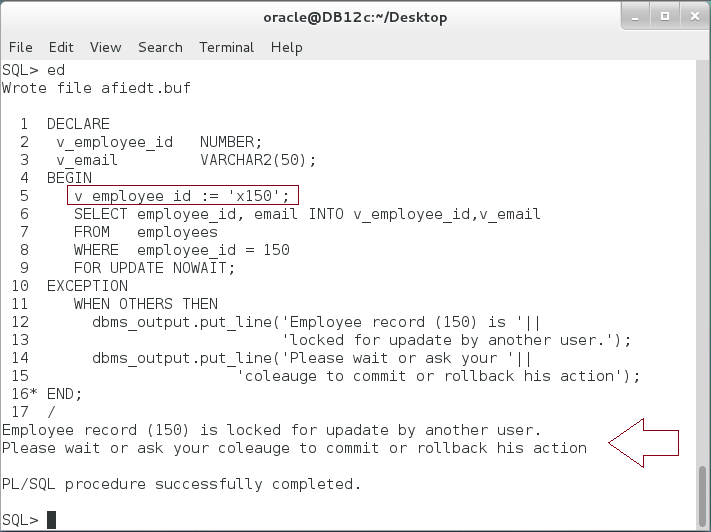
**Step 5:** There is no predefined exception for ORA-00054. Thus, it is an internally defined exception and has no name. You can't handle the exception by its name because it does not have one. However, you can handle it using "**OTHERS**" keyword as show below:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| BEGIN |  |
| SELECT employee\_id, email INTO v\_employee\_id,v\_email |  |
| FROM employees |  |
| WHERE employee\_id = 150 |  |
| FOR UPDATE NOWAIT; |  |
| **EXCEPTION** | Handle internally defined exception using **OTHERS** condition. |
| **WHEN OTHERS THEN** |
| dbms\_output.put\_line('Employee record (150) is '|| |
| 'locked for upadate by another user.'); |
| dbms\_output.put\_line('Please wait or ask your '|| |  |
| 'coleauge to commit or rollback his action'); |  |
| END; |  |
| / |  |



**Step 6:** Remember, OTHERS handles all exceptions appear on the Executable part of the block. Modify the block as shown below:

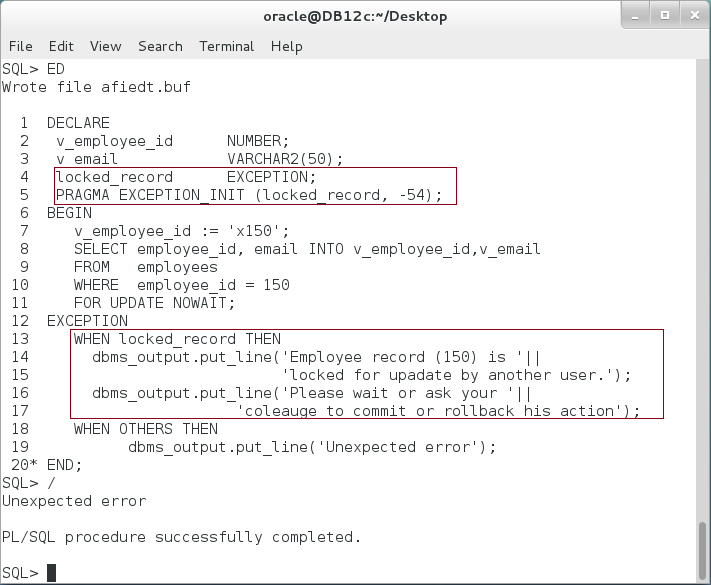
|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| BEGIN |  |
| **v\_employee\_id := 'x150';** | **Raise exception.** |
| SELECT employee\_id, email INTO v\_employee\_id,v\_email |  |
| FROM employees |  |
| WHERE employee\_id = 150 |  |
| FOR UPDATE NOWAIT; |  |
| EXCEPTION | Handle internally defined exception using OTHERS condition. |
| WHEN OTHERS THEN |
| dbms\_output.put\_line('Employee record (150) is '|| |
| 'locked for upadate by another user.'); |
| dbms\_output.put\_line('Please wait or ask your '|| |  |
| 'coleauge to commit or rollback his action'); |  |
| END; |  |
| / |  |

****

**Please note:** In the previous block, Invalid number exception is raised before resource busy exception. The message appearing to for the user is completely unrelated. For this reason, it is recommended not to assume that "OTHERS" condition handles specific exception only. The best practice here is to use exception name.

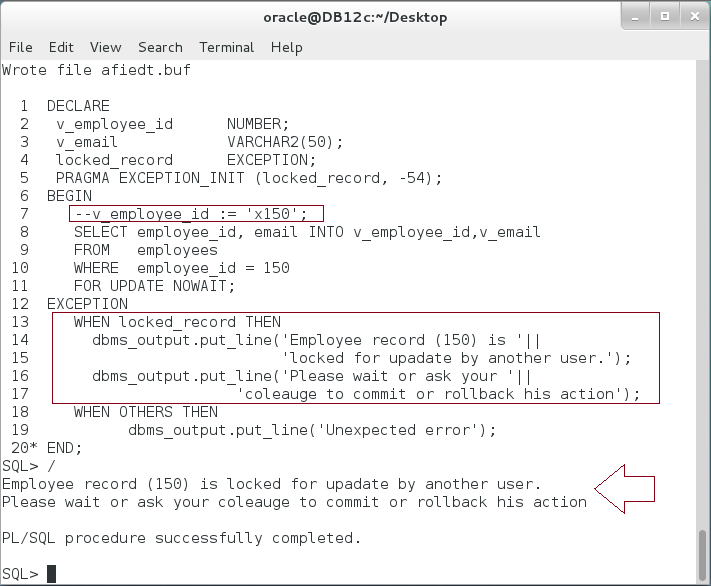
**Step 7:** Modify the previous block as show below:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| **locked\_record EXCEPTION;** |  |
| **PRAGMA EXCEPTION\_INIT (locked\_record, -54);** |  |
| BEGIN |  |
| v\_employee\_id := 'x150'; | Raise exception. |
| SELECT employee\_id, email INTO v\_employee\_id,v\_email |  |
| FROM employees |  |
| WHERE employee\_id = 150 |  |
| FOR UPDATE NOWAIT; |  |
| **EXCEPTION** | Handle internally defined exception using **Exception NAME**. |
| **WHEN locked\_record THEN** |
| dbms\_output.put\_line('Employee record (150) is '|| |
| 'locked for upadate by another user.'); |
| dbms\_output.put\_line('Please wait or ask your '|| |  |
| 'coleauge to commit or rollback his action'); |  |
| **WHEN OTHERS THEN** |  |
| dbms\_output.put\_line('Unexpected error'); |  |
| END; |  |
| / |  |

****

**Step 8:** Modify the previous block as shown:

|  |  |
| --- | --- |
| Line | Description |
| DECLARE |  |
| v\_employee\_id NUMBER; |  |
| v\_email VARCHAR2(50); |  |
| locked\_record EXCEPTION; |  |
| PRAGMA EXCEPTION\_INIT (locked\_record, -54); |  |
| BEGIN |  |
| **-- v\_employee\_id := 'x150';** | **Remove the exception** |
| SELECT employee\_id, email INTO v\_employee\_id,v\_email |  |
| FROM employees |  |
| WHERE employee\_id = 150 |  |
| FOR UPDATE NOWAIT; |  |
| **EXCEPTION** | The exception is now handled by name. |
| **WHEN locked\_record THEN** |
| dbms\_output.put\_line('Employee record (150) is '|| |
| 'locked for upadate by another user.'); |
| dbms\_output.put\_line('Please wait or ask your '|| |  |
| 'coleauge to commit or rollback his action'); |  |
| WHEN OTHERS THEN |  |
| dbms\_output.put\_line('Unexpected error'); |  |
| END; |  |
| / |  |

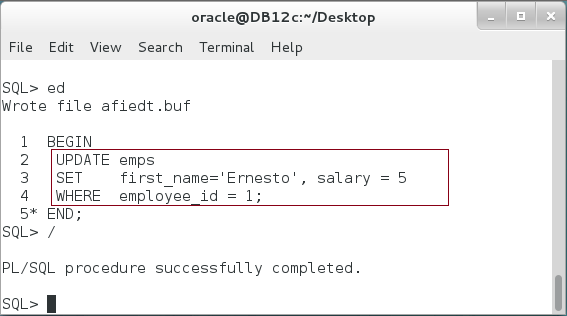
****

**Please note:** The error message is now more rational.

## User-Defined Exception

**Step 1:** Execute the following block:

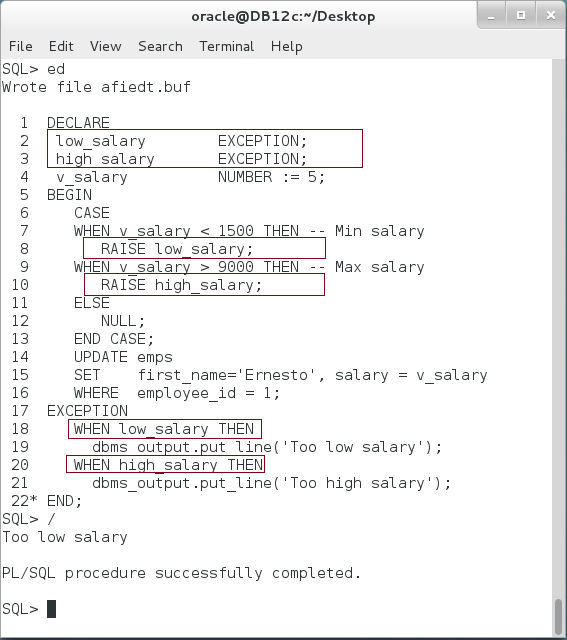
|  |  |
| --- | --- |
| Command | Description |
| BEGIN |  |
| **UPDATE** emps | Update employee record |
| **SET** first\_name='Ernesto', **salary = 5** |
| **WHERE** employee\_id = 1; |
| END; |  |
| / |  |

****

**Please note:** The previous block seems good, but it is **not** rational to assign $5 as a salary!!

**Step 2:** You may define your exception to prevent user entering un-logical data. Modify the previous block as shown below:

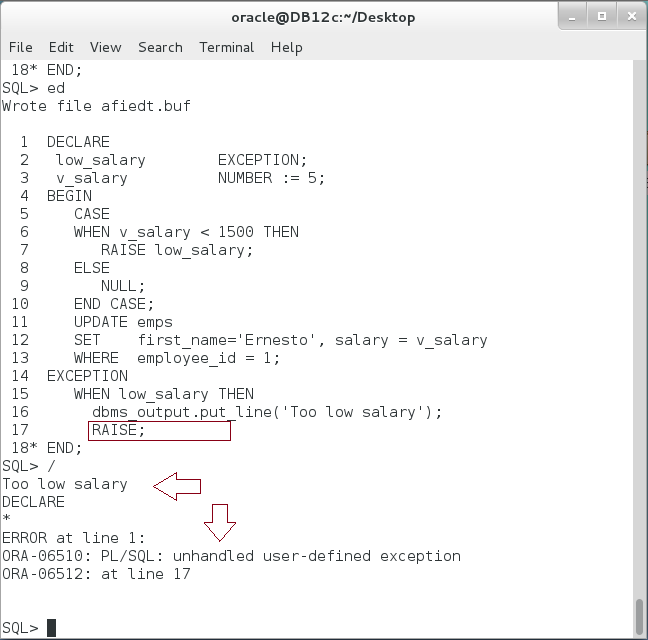
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| **low\_salary EXCEPTION;** | **Define** your exception |
| **high\_salary EXCEPTION;** |
| v\_salary NUMBER := 5; |
| BEGIN |  |
| CASE |  |
| WHEN v\_salary < 1500 THEN -- Min salary | **Raise** your exception |
| **RAISE low\_salary;** |
| WHEN v\_salary > 9000 THEN -- Max salary |
| **RAISE high\_salary;** |
| ELSE |  |
| NULL; |  |
| END CASE; |  |
| UPDATE emps |  |
| SET first\_name='Ernesto', salary = v\_salary |  |
| WHERE employee\_id = 1; |  |
| EXCEPTION |  |
| **WHEN low\_salary THEN** | **Handle** your exception |
| dbms\_output.put\_line('Too low salary'); |
| **WHEN high\_salary THEN** |
| dbms\_output.put\_line('Too high salary'); |
| END; |  |
| / |  |

****

## Raise and View Exception

**Step 1:** You can **re-raise the exception** in its handler. This is useful when you intend to prevent the invoker block to continue the subsequent statements or to let the invoker handle the exception again and add its feedback. Build on the previous block as shown below:

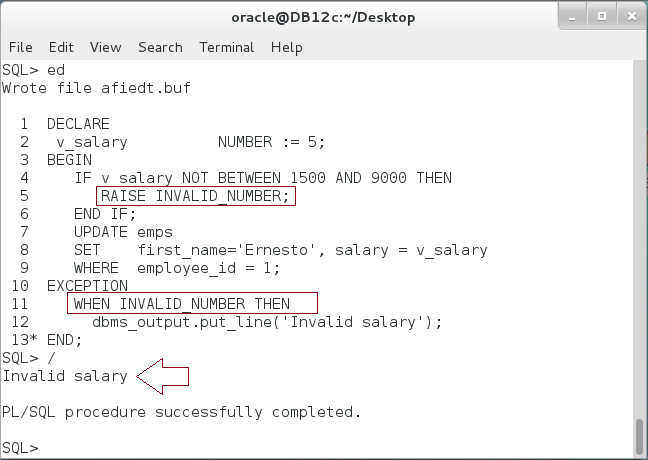
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| low\_salary EXCEPTION; | Define your exception |
| v\_salary NUMBER := 5; |
| BEGIN |
| CASE |  |
| WHEN v\_salary < 1500 THEN |  |
| RAISE low\_salary; | Raise your exception |
| ELSE |
| NULL; |
| END CASE; |
| UPDATE emps |  |
| SET first\_name='Ernesto', salary = v\_salary |  |
| WHERE employee\_id = 1; |  |
| EXCEPTION |  |
| **WHEN low\_salary THEN** | **Handle** your exception  **Re-raise** the exception |
| dbms\_output.put\_line('Too low salary'); |
| **RAISE;** |
| END; |
| / |



**Please note:** You don't need to mention the exception name in its handler. RAISE statement automatically raises the same exception to the invoker.

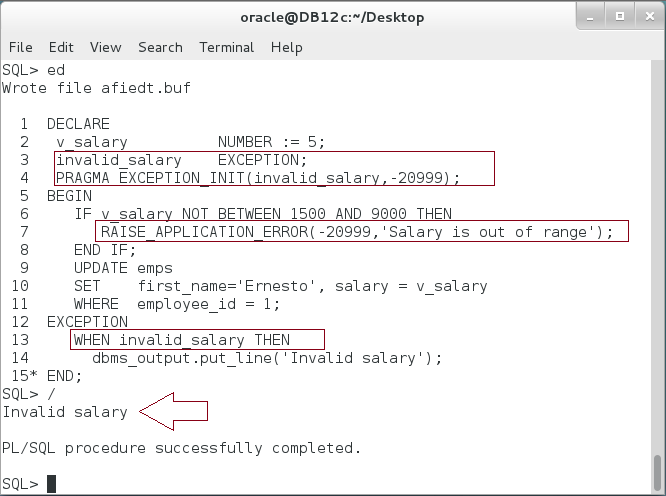
**Step 2:** You can raise a predefined exception instead of defining your own. Modify the previous block to raise "INVALID\_NUMBER" predefined exception as shown below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_salary NUMBER := 5; |  |
| BEGIN |  |
| IF v\_salary NOT BETWEEN 1500 AND 9000 THEN | **Raise** a predefined exception **INVALID\_NUMBER** |
| **RAISE INVALID\_NUMBER;** |
| END IF; |
| UPDATE emps |  |
| SET first\_name='Ernesto', salary = v\_salary |  |
| WHERE employee\_id = 1; |  |
| **EXCEPTION** | **Handle** the exception |
| **WHEN INVALID\_NUMBER THEN** |
| dbms\_output.put\_line('Invalid salary'); |
| END; |
| / |  |



**Step 3:** You can assign code and message to the User-Defined Exception (UDE) as shown in the following block:

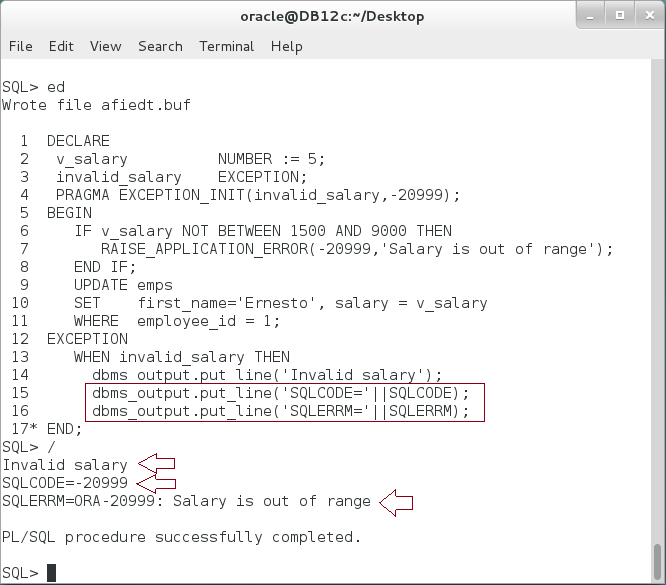
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_salary NUMBER := 5; | **Define** and assign **code** and **name** to **UDE** |
| **invalid\_salary EXCEPTION;** |
| **PRAGMA EXCEPTION\_INIT(invalid\_salary,-20999);** |
| BEGIN |  |
| IF v\_salary NOT BETWEEN 1500 AND 9000 THEN | **Raise** exception using **Raise\_Application\_Error** |
| **RAISE\_APPLICATION\_ERROR(-20999,'Salary is out of range');** |
| END IF; |
| UPDATE emps |
| SET first\_name='Ernesto', salary = v\_salary |  |
| WHERE employee\_id = 1; |  |
| **EXCEPTION** | **Handle** the exception by name. |
| **WHEN invalid\_salary THEN** |
| dbms\_output.put\_line('Invalid salary'); |
| END; |
| / |  |



**Please note:** The message assigned to the exception INVALID\_SALARY: "Salary is out of range." is not shown.

**Step 4:** To view the code and message assigned to the exception, use SQLCODE and SQLERRM respectively. Modify the previous block as shown below:

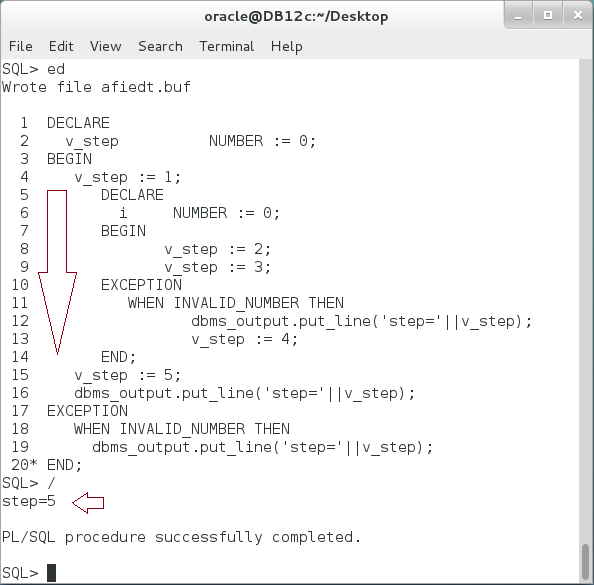
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_salary NUMBER := 5; |  |
| invalid\_salary EXCEPTION; |  |
| PRAGMA EXCEPTION\_INIT(invalid\_salary,-20999); |  |
| BEGIN |  |
| IF v\_salary NOT BETWEEN 1500 AND 9000 THEN |  |
| RAISE\_APPLICATION\_ERROR(-20999,'Salary is out of range'); |  |
| END IF; |  |
| UPDATE emps |  |
| SET first\_name='Ernesto', salary = v\_salary |  |
| WHERE employee\_id = 1; |  |
| EXCEPTION | View the error code and message of the exception using **SQLCODE** and **SQLERRM** |
| WHEN invalid\_salary THEN |
| dbms\_output.put\_line('Invalid salary'); |
| dbms\_output.put\_line('SQLCODE='||**SQLCODE**); |
| dbms\_output.put\_line('SQLERRM='||**SQLERRM**); |
| END; |
| / |  |



## Exception Propagation

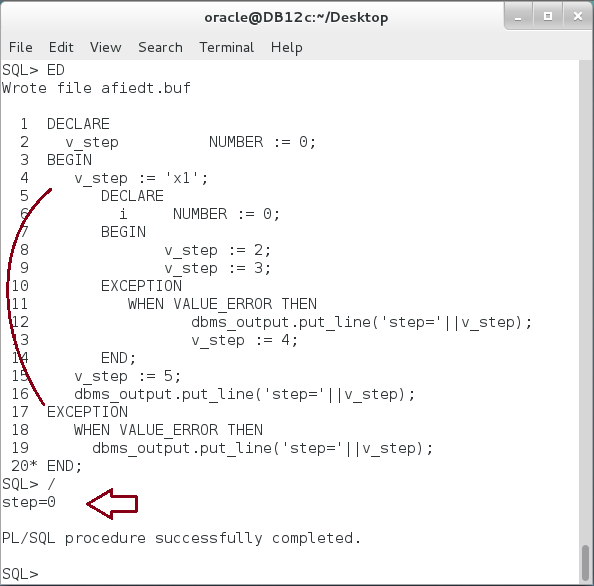
**Step 1:** Execute the following block:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 1;** |  |
| DECLARE |  |
| i NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 2;** |  |
| **v\_step := 3;** |  |
| EXCEPTION |  |
| WHEN VALUE\_ERROR THEN |  |
| dbms\_output.put\_line('step='||v\_step); | Execution reach **v\_step := 5**  Normal execution. |
| v\_step := 4; |
| END; |
| **v\_step := 5;** |
| dbms\_output.put\_line('step='||v\_step); |
| EXCEPTION |
| WHEN VALUE\_ERRORTHEN |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| END; |  |
| / |  |



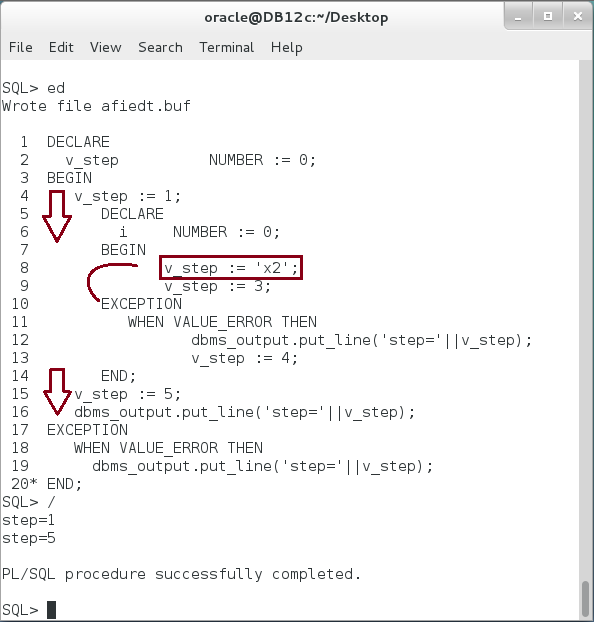
**Step 2:** Modify the block as shown below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 'x1';** | Raise exception here. |
| DECLARE |
| i NUMBER := 0; | All the remaining statements are not executed. |
| BEGIN |
| v\_step := 2; |
| v\_step := 3; |
| EXCEPTION |
| WHEN VALUE\_ERRORTHEN |
| dbms\_output.put\_line('step='||v\_step); |
| v\_step := 4; |
| END; |
| v\_step := 5; |
| dbms\_output.put\_line('step='||v\_step); |
| **EXCEPTION** | Exception handled here and v\_step value = 0 |
| **WHEN VALUE\_ERROR THEN** |
| dbms\_output.put\_line('step='||v\_step); |
| END; |
| / |  |



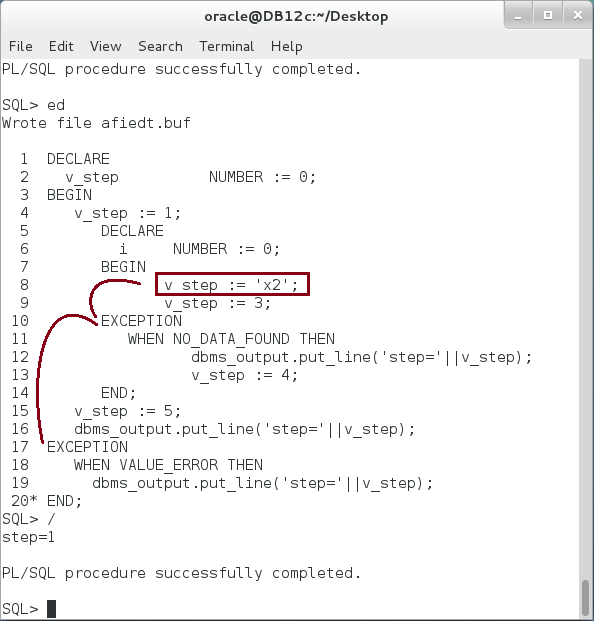
**Step 3:** Modify the block as shown below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| v\_step := 1; |  |
| DECLARE |  |
| i NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 'x2';** | **Raise Exception here** |
| v\_step := 3; | **Not executed.** |
| **EXCEPTION** |  |
| **WHEN VALUE\_ERROR THEN** |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| v\_step := 4; |  |
| END; |  |
| v\_step := 5; |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| EXCEPTION |  |
| WHEN VALUE\_ERROR THEN |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| END; |  |
| / |  |



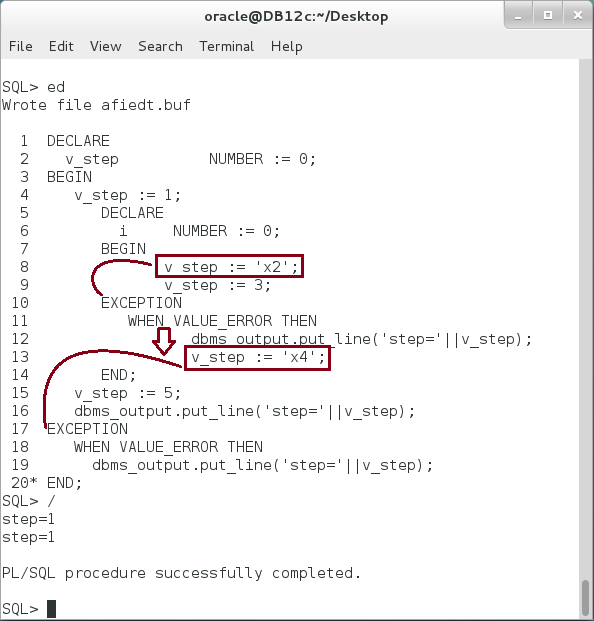
**Step 4:** Modify the block as shown below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| v\_step := 1; |  |
| DECLARE |  |
| i NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 'x2';** | **Raise exception.** |
| v\_step := 3; | Not executed. |
| EXCEPTION |  |
| **WHEN NO\_DATA\_FOUND THEN** | **There is no handler.** |
| dbms\_output.put\_line('step='||v\_step); |  |
| v\_step := 4; |  |
| END; |  |
| v\_step := 5; | Not executed. |
| dbms\_output.put\_line('step='||v\_step); |
| **EXCEPTION** | **Handle** The exception. |
| **WHEN VALUE\_ERROR THEN** |
| dbms\_output.put\_line('step='||v\_step); |  |
| END; |  |
| / |  |



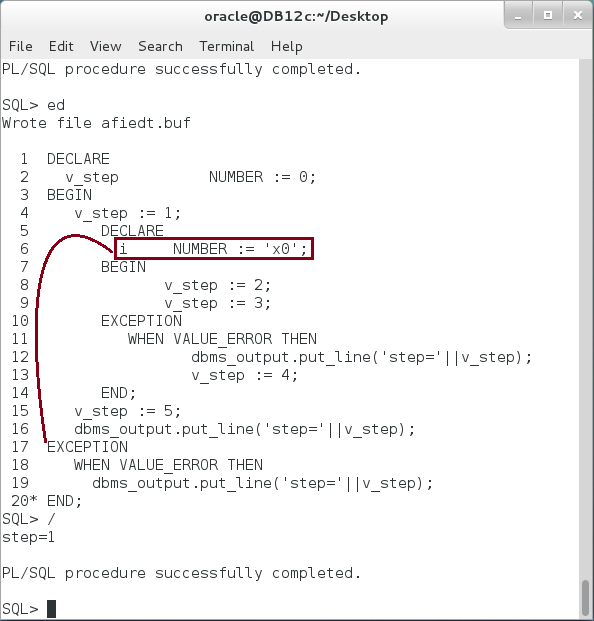
**Step 5:** Modify the block as shown below:

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| v\_step := 1; |  |
| DECLARE |  |
| i NUMBER := 0; |  |
| BEGIN |  |
| **v\_step := 'x2';** | **Raise exception.** |
| v\_step := 3; |  |
| EXCEPTION | **Handle** exception. |
| WHEN VALUE\_ERROR THEN |
| dbms\_output.put\_line('step='||v\_step); | **Raise** exception in **Exception-Handling** part. |
| **v\_step := 'x4';** |
| END; |  |
| v\_step := 5; |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| **EXCEPTION** | **Handle** the last exception. |
| **WHEN VALUE\_ERROR THEN** |
| dbms\_output.put\_line('step='||v\_step); |  |
| END; |  |
| / |  |



**Step 6:** Modify the block as shown below:

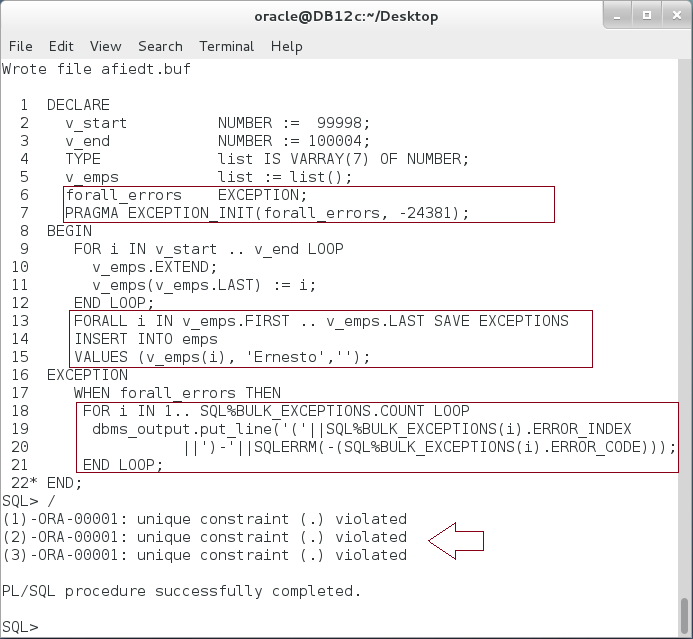
|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_step NUMBER := 0; |  |
| BEGIN |  |
| v\_step := 1; |  |
| DECLARE | Raise error in **Declaration Part** |
| **i NUMBER := 'x0';** |
| BEGIN |  |
| v\_step := 2; |  |
| v\_step := 3; |  |
| EXCEPTION |  |
| WHEN VALUE\_ERROR THEN |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| v\_step := 4; |  |
| END; |  |
| v\_step := 5; |  |
| dbms\_output.put\_line('step='||v\_step); |  |
| **EXCEPTION** | The **outer** block's Exception-Handling handles the exception. |
| **WHEN VALUE\_ERROR THEN** |
| dbms\_output.put\_line('step='||v\_step); |
| END; |  |
| / |  |



## Bulk Exception

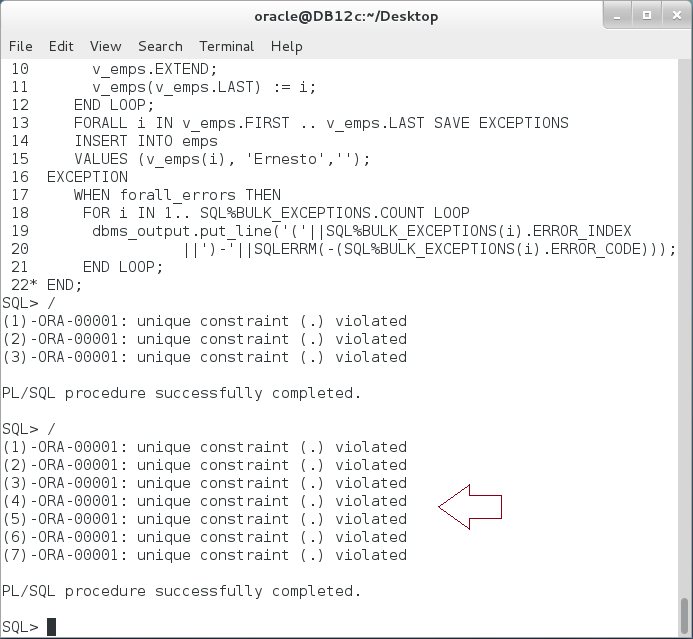
**Step 1:** To allow a FORALL statement to continue even if some of its DML statements fail, include the SAVE EXCEPTIONS clause. When a DML statement fails, PL/SQL does not raise an exception; instead, it saves information about the failure. After the FORALL statement completes, PL/SQL raises a single exception for the FORALL statement (ORA-24381). In the exception handler for ORA-24381, you can get information about each individual DML statement failure from the implicit cursor attribute SQL%BULK\_EXCEPTIONS.

|  |  |
| --- | --- |
| Command | Description |
| DECLARE |  |
| v\_start NUMBER := 99998; |  |
| v\_end NUMBER := 100004; |  |
| TYPE list IS VARRAY(7) OF NUMBER; |  |
| v\_emps list := list(); | **Define** exception  -24381 |
| **forall\_errors EXCEPTION;** |
| **PRAGMA EXCEPTION\_INIT(forall\_errors, -24381);** |
| BEGIN |  |
| FOR i IN v\_start .. v\_end LOOP |  |
| v\_emps.EXTEND; |  |
| v\_emps(v\_emps.LAST) := i; |  |
| END LOOP; |  |
| FORALL i IN v\_emps.FIRST .. v\_emps.LAST **SAVE EXCEPTIONS** | Add "**SAVE EXCEPTIONS**" keyword. |
| INSERT INTO emps |
| VALUES (v\_emps(i), 'Ernesto',''); |
| EXCEPTION |  |
| **WHEN forall\_errors THEN** |  |
| FOR i IN 1.. **SQL%BULK\_EXCEPTIONS**.COUNT LOOP | Use **BULK\_EXCEPTIONS** associative array |
| dbms\_output.put\_line('('||  **SQL%BULK\_EXCEPTIONS(i).ERROR\_INDEX**  ||')-'||*SQLERRM(*  **(SQL%BULK\_EXCEPTIONS(i).ERROR\_CODE)***)*); |
| END LOOP; |  |
| END; |  |
| / |  |



**Explain the output?**

**Step 2:** Re-execute the previous block. **Explain the output.**



# SUMMARY

Exception refers to the runtime errors. PL/SQL block handles the exceptions in Exception-Handling part. Since the exceptions are different, it is better to handle exceptions by name. Oracle has three different exception categories: (1) Predefined, (2) Internally Defined, and (3) User-Defined exception. Predefined exception is defined and raised automatically by PL/SQL. It also has name. Thus, you can handle it by its name in the PL/SQL block. Internally Defined exception is the same as Predefined exception but it has no name. So, you must give it name before you can handle it by name. User-Defined exception is defined, raised, and handled explicitly. Exception Propagation refers to when and where the exception navigate between blocks and invoker/host environment. You should be aware of the rules of propagation to avoid un-expected results. Bulk exception handler is very useful to reduce the effort of track the exceptions one by one.

After completing this lab exercise, you should be able to recognize different types of exceptions and handle them successfully.

# REFERENCES

* https://docs.oracle.com/database/121/LNPLS/errors.htm#LNPLS007
* https://docs.oracle.com/database/121/LNPLS/tuning.htm#LNPLS879

# INDEX

Exception Propagation 3, 6, 37, 49

Internally Defined Exception 2, 6, 21

Predefined Exception 2, 6, 8

User-Defined Exception 6, 29, 34