



INSTITUTO DE INVESTIGACIÓN FIIS-UNI
Facultad de Ingeniería Industrial y de Sistemas

TALLER DE PROGRAMACION CON ORACLE PL/SQL

Capítulo 07 Tratamiento de Errores

Contenido

- Tipos de Errores
- Excepciones
- Esquema General
- Excepciones Predefinidas
- Instrucción RAISE
- Excepciones de Usuario
- Generación de Mensajes de Error

Tipos de Errores

| Tipo de Error | Quién Informa | Cómo es tratado |
|----------------|---------------------------|---|
| De Compilación | Compilador PL/SQL | Interactivamente: el compilador informa de los errores y el programador debe corregirlos. |
| De Ejecución | Motor de Ejecución PL/SQL | Programáticamente: las excepciones son generadas e interceptadas por las rutinas de tratamiento de excepciones. |

Excepciones

Es el mecanismo de tratamiento de errores en tiempo de ejecución. Tenemos dos tipos de excepciones:

- Definidas por el usuario
- Predefinidas

Esquema General

Formato:

```
BEGIN

    < Bloque de instrucciones a controlar >

EXCEPTION
    WHEN nombre_excepción THEN
        secuencia_de_instrucciones;
    WHEN nombre_excepción THEN
        secuencia_de_instrucciones;
    [ WHEN OTHERS THEN
        secuencia_de_instrucciones; ]
END;
```

Excepciones Predefinidas

Oracle ha definido diversas excepciones que corresponden con los errores Oracle más comunes.

| Excepción | Descripción |
|---------------------|--|
| INVALID_CURSOR | Ocurre cuando se hace referencia a un cursor que esta cerrado. |
| CURSOR_ALREADY_OPEN | Ocurre cuando se trata de abrir un cursor que ya esta abierto. |
| NO_DATA_FOUND | Ocurre cuando una sentencia SELECT no retorna ninguna fila. |
| TOO_MANY_ROWS | Ocurre cuando una sentencia SELECT retorna mas de una fila. |
| VALUE_ERROR | Ocurre cuando hay conflicto de tipos de datos. |

Script 1

Desarrollar un procedimiento para consultar el salario de un empleado.

```
create or replace procedure FindEmp( Cod Emp.EmpNo%Type )
is
    Salario Emp.Sal%Type;
Begin
    Select Sal Into Salario
    From Emp
    Where EmpNo = Cod;
    DBMS_Output.Put_Line( 'Salario: ' || Salario );
Exception
    When No_Data_Found Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución:

```
SQL> exec FindEmp( 9999 );
Código no existe.

PL/SQL procedure successfully completed.
```

Instrucción RAISE

Permite generar una excepción.

Script 2

```
create or replace procedure UpdateSalEmp
(Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise No_Data_Found;
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When No_Data_Found Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución:

```
SQL> exec UpdateSalEmp( 9999, 5000 );
Código no existe.

PL/SQL procedure successfully completed.
```

Excepciones de Usuario

Script 3

Desarrollar una segunda versión del procedimiento UpdateSalEmp, pero con una excepción de usuario.

```
create or replace procedure UpdateSalEmp2
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
    Excepl Exception;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise Excepl;
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When Excepl Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución:

```
SQL> exec UpdateSalEmp2( 9999, 5000 );
Código no existe.

PL/SQL procedure successfully completed.
```

Generación de Mensajes de Error

Los mensajes de error que se generan con `RAISE_APPLICATION_ERROR` deben estar entre -20,999 y -20,000.

Script 4

Desarrollar una tercera versión del procedimiento `UpdateSalEmp`, pero esta vez genere un mensaje de error.

```
create or replace procedure UpdateSalEmp3
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise_Application_Error( -20000, 'No existe empleado.' );
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
End;
```

Ejecución:

```
SQL> exec UpdateSalEmp3( 9999, 5000 );
BEGIN UpdateSalEmp3( 9999, 5000 ); END;

*
ERROR at line 1:
ORA-20000: No existe empleado.
ORA-06512: at "SCOTT.UPDATESALEMP3", line 10
ORA-06512: at line 1
```

Script 5

Otra versión del mismo procedimiento.

```
create or replace procedure UpdateSalEmp4
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise_Application_Error( -20000, 'No existe empleado.' );
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When Others Then
        dbms_output.put_line ( 'Error Nro. ORA' || to_char(sqlcode) );
        dbms_output.put_line ( sqlerrm );
End;
```

Ejecución:

```
SQL> exec UpdateSalEmp4( 9999, 5000 );
Error Nro. ORA-20000
ORA-20000: No existe empleado.

PL/SQL procedure successfully completed.
```