Datenbank und Informationssysteme

DI. Hilbe Klaus, MBA

Exceptions

An exception handler is a piece of code which is executed when an exception is raised.

Exceptions are errors which might occur during execution of the code. These exceptions should be handled. Usually if the exception is not handled the code stops executing. Examples for exceptions are:

- User errors (eg. user entered a non valid entry, division by zero,…)
- System errors (eg. hard disk is full)
- Data errors (eg. inserting a null value where not allowed)

Exceptions – Syntax

```
BEGIN
  execution_section
EXCEPTION
  WHEN exceptionname1 [OR exceptionname2…] THEN
    exception_code1;
  [WHEN exceptionname3 [OR exceptionname4…] THEN
    exception_code2;   ]
  [WHEN OTHERS THEN
    exception_code3;   ]
END;
```

Example: Simple Exception handling

```
DECLARE
    v_last_name employees.last_name%TYPE := 'Grantx';
    v_salary    employees.salary%TYPE;
BEGIN
    SELECT salary
    INTO   v_salary
    FROM   employees
    WHERE  last_name = v_last_name;
END;
```

IF no row is returned following exception is raised by oracle:

```
Error report:
ORA-01403: no data found
ORA-06512: at line 5
01403. 00000 -  "no data found"
```

Example: How to handle Exceptions

```
DECLARE
    v_last_name employees.last_name%TYPE := 'Grantx';
    v_salary    employees.salary%TYPE;
BEGIN
    SELECT salary
    INTO   v_salary
    FROM   employees
    WHERE  last_name = v_last_name;
EXCEPTION
    WHEN no_data_found THEN
      dbms_output.Put_line('No employee found with
         Last Name ' || v_last_name);
END;
```

Exceptions

- For each block you can define **ONE** exception handler
- When a exception occurs the code execution starts immediately at the exception handler. Code after the line of the exception is skipped
- Only one exception can occur at a time
- "WHEN OTHERS …" is an optional exceptional section which handles any exceptions not addressed before in the exception part

Types of Exceptions and how to raise an Exception

- Types of Exceptions
  - Predefined by Oracle like NO_DATA_FOUND, TOO_MANY_ROWS, DUP_VAL_ON_INDEX
  - Non-predefined server errors. These errors have an error number (ORA-nnnnn) and an error message, but the name of the error message is not defined. In the declare section the developer can define a name for the error handling and associate with it the corresponding error number
  - User defined exception: an exception defined by the user
- There are two Methods how to raise an Exception
  - Automatically raised by the DB server, eg. the error NO_DATA_FOUND
  - Defined by the developer. The developer can raise an exception explicitly with the **RAISE** command, eg.
    …
    RAISE NO_DATA_FOUND;
    …

Defining a non predefined Error – Syntax

DECLARE

nonpredef_exceptionname **EXCEPTION;**
PRAGMA EXCEPTION_INIT
(*nonpredef_exceptionname*, *Oracle_exception_no***)**

BEGIN

…

EXCEPTION

…

WHEN *nonpredef_exceptionname* **THEN**

exception_code1;

…

END;

Example: Defining a non predefined Error Catching inserting null in non nullable field

```
DECLARE
    e_insert_null EXCEPTION; -- Declare exception
    -- associate exception with Oracle error number
    PRAGMA EXCEPTION_INIT (e_insert_null, -01400);


BEGIN
    INSERT INTO countries (country_id,country_name)
    VALUES (NULL, 'Austria');


EXCEPTION
    WHEN e_insert_null THEN -- handle exception
        dbms_output.Put_line('Can not insert
            null value in non nullable column');
END;
```

Defining a User defined Error – Syntax

```
DECLARE
    usrdef_exceptionname EXCEPTION;
BEGIN
    …
    RAISE usrdef_exceptionname;
    …
EXCEPTION
    …
    WHEN usrdef_exceptionname THEN
    exception_code1;
    …
END;
```

Example: Defining a User defined Error Catching no update done

```
DECLARE
  v_last_name employees.last_name%TYPE := 'Grantx';
  e_no_update EXCEPTION;
BEGIN
  UPDATE employees
  SET    salary = salary * 1.10 -- 10 % increase
  WHERE  last_name = v_last_name;
  IF SQL%NOTFOUND THEN
    RAISE e_no_update;
  END IF;
EXCEPTION
  WHEN e_no_update THEN
   dbms_output.Put_line('No Record was updated!');
END;
```

Retrieving Error Code and Error Message

Oracle provides two functions to return the error code and error message. These functions can not be used directly in SQL Statements:

- SQLERRM: returns the error messages
- SQLCODE: returns the numeric error code

Example: Retrieving Error Code and Error Message

```
DECLARE
    v_error_code    NUMBER;
    v_error_message VARCHAR2(1000);
    v_last_name     employees.last_name%TYPE := 'Taylor';
    v_salary        employees.salary%TYPE;
BEGIN
    SELECT salary INTO   v_salary
    FROM   employees
    WHERE  last_name = v_last_name; -- There are 2 'Taylor'
EXCEPTION
    WHEN OTHERS THEN
      v_error_code := SQLCODE;
      v_error_message := SQLERRM;
      INSERT INTO error_log
         (error_code, error_message, error_date)
      VALUES (v_error_code, v_error_message, SYSDATE());
END;
```

Raising Application Error

With the RAISE_APPLICATION_ERROR the developer can associate with an user defined error an error number and an error message. It can be used in the **execution** and **exception section**.

The Syntax is:

RAISE_APPLICATION_ERROR(*error_number, error_message*
[,{TRUE|FALSE}]**)**;

- Error_number: error number associate with the error, must be between -20000 and - 20999
- Error_message: user defined error message, maximum 2048 bytes long
- TRUE|FALSE: optional Boolean parameter
  - FALSE: the error replaces all previous errors, is the default value
  - TRUE: error is placed on the stack of the previous errors. The error is added to the errors which happened till now

Example: Raising Application Error

```
CREATE PROCEDURE Inc_salary
AS
  v_last_name employees.last_name%TYPE := 'Grantxx';
BEGIN
    UPDATE employees
    SET    salary = salary * 1.10 -- 10 % increase
    WHERE  last_name = v_last_name;
    IF SQL%NOTFOUND THEN
      Raise_application_error(-20201,
        'No employees found, no salary increased!');
    END IF;
END;
```

Example: Raising Application Error in the Exception Section

```
DECLARE
    e_no_update EXCEPTION;
    v_last_name employees.last_name%TYPE := 'Grantxx';
BEGIN
    UPDATE employees
    SET    salary = salary * 1.10 -- 10 % increase
    WHERE  last_name = v_last_name;
    IF SQL%NOTFOUND THEN
      RAISE e_no_update;
    END IF;
EXCEPTION
    WHEN e_no_update THEN
      Raise_application_error(-20201,
           'No employees found, no salary increased!');
END;
```

Example: Exceptions and Blocks

```
CREATE OR replace PROCEDURE Loc_exception
AS
  v_salary employees.salary%TYPE;
BEGIN
    /* local block - like TRY concept */
    BEGIN
        SELECT salary INTO v_salary FROM   employees;
    EXCEPTION
        WHEN OTHERS THEN
          dbms_output.Put_line('Local except. handler:'
              ||SQLCODE ||SQLERRM);
    END;
    dbms_output.Put_line('** Hello **');
EXCEPTION
  WHEN OTHERS THEN
          dbms_output.Put_line('Main except. handler:'
              || SQLCODE ||SQLERRM);
END loc_exception;
```

Tasks

- Ändern Sie das Gehalt des Mitarbeiters 129:
  - auf das Gehalt des Mitarbeiters mit dem Vornamen 'Peter'.
  - wenn 'Peter' nicht in der Datenbank gespeichert ist, dann ermitteln sie das Durchschnittsgehalt aller Mitarbeiter
  - wenn es mehr als einen 'Peter' gibt, dann ermitteln Sie das kleinste Einkommen aller 'Peter'
  Verwenden Sie Exceptions!
- Erstellen Sie eine Prozedur, der die Abteilungsnummer mitgegeben wird. Suchen Sie den Mitarbeiter dieser Abteilung mit dem höchsten Einkommen und machen Sie diesen zum Manager der Abteilung.
  Programmieren Sie sicher und verwenden Sie Exceptions.
  Geben Sie Beispiele an, wo die Verarbeitung funktioniert und wo es Probleme gibt.