Oracle PL/SQL

PL/SQL Exceptions

Agenda

01 PL/SQL Exception Handling

02 Built-in Exceptions

03 User-defined Exceptions

04 User-named Exceptions

05 WHEN OTHERS Clause

06 SQLCODE Function

07 SQLERRM Function

PL/SQL Exception Handling

PL/SQL Exception Handling

An error condition during a program execution is called an exception.

The mechanism for resolving such an exception is exception handling.

Exception

Exception Handling

PL/SQL Exception Handling

- PL/SQL provides a mechanism to handle such exceptions so that the normal flow of a program can be maintained.

Built-in Exceptions

Raised implicitly by the run-time system

User-defined Exceptions

Raised explicitly by RAISE statements

Built-in Exceptions

Built-in Exceptions

- Exceptions are predefined and raised automatically into Oracle engine when any error occurs during a program
- Each error has a unique number and message which is predefined

Built-in Exceptions

Exception	Error Code	Description
ACCESS_INTO_NULL	ORA-06530	Exception raised when we assign an uninitialized (NULL) object
CASE_NOT_FOUND	ORA-06592	Exception raised when any choice case, as well as an ELSE clause, is not found in the CASE statement
CURSOR_ALREADY_OPEN	ORA-06511	Exception raised when you open a cursor that is already opened
DUP_VAL_ON_INDEX	ORA-00001	Exception raised when you store duplicate value in a unique constraint column
INVALID_CURSOR	ORA-01001	Exception raised when you perform an operation on a cursor which is not really opened
INVALID_NUMBER	ORA-01722	Exception raised when your try to explicitly convert a string to a number fails
LOGIN_DENIED	ORA-01017	Exception raised when you login to Oracle with wrong username or password
NOT_LOGGED_ON	ORA-01012	Exception raised when your program tries to get data from the database, while you are not connected to Oracle
NO_DATA_FOUND	ORA-01403	Exception raised when the SELECT ... INTO statement doesn't fetch any row from a database table

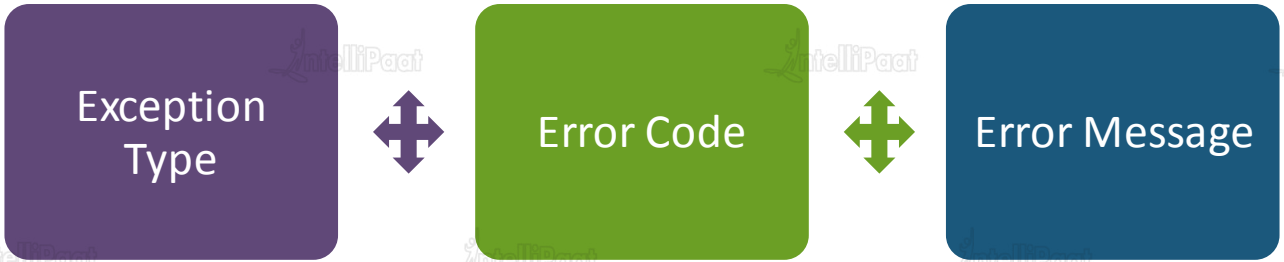
Built-in Exceptions

Exception	Error Code	Description
PROGRAM_ERROR	ORA-06501	Exception raised when your program is error-prone (internal error)
STORAGE_ERROR	ORA-06500	Exception raised when PL/SQL program runs out of memory, or in case memory is dumped/corrupted
SYS_INVALID_ROWID	ORA-01410	Exception raised when your try to explicitly convert a character string to a universal ROWID (UID) fails
TIMEOUT_ON_RESOURCE	ORA-00051	Exception raised when the database is locked, or Oracle is waiting for a resource
TOO_MANY_ROWS	ORA-01422	Exception raised when the SELECT ... INTO statement returns more than one row
VALUE_ERROR	ORA-06502	Exception raised when an arithmetic, conversion, or size-constraint error occurs
ZERO_DIVIDE	ORA-01476	Exception raised when your program tries to attempt division by zero

User-defined Exceptions

User-defined Exceptions

- Provides facility to define the custom or user-defined exceptions according to the need of a program
- Must be declared and then raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR
- PL/SQL exceptions consist of three parts:



How to Define an Exception?

Declare an Exception

You have to declare the user-defined exception name in the DECLARE block

user_define_exception_name EXCEPTION;

RAISE the Exception

Use the RAISE statement to raise the defined exception name and control transfer to an EXCEPTION block

RAISE user_define_exception_name;

Implement the Exception Condition

In the PL/SQL EXCEPTION block, add the WHEN condition to implement the user-defined action

WHEN user_define_exception_name THEN User defined statement (action) will be taken;

How to Define an Exception?

Syntax

```
DECLARE
   user_define_exception_name EXCEPTION;
BEGIN
   statement(s);
   IF condition THEN
      RAISE user_define_exception_name;
   END IF;
EXCEPTION
   WHEN user_define_exception_name THEN
      User defined statement (action) will be
taken;
END;
```

How to Define an Exception?

Example

```
SQL>edit user_exp
DECLARE
   myex EXCEPTION;
   i NUMBER;
BEGIN
   FOR i IN (SELECT * FROM enum) LOOP
      IF i.eno = 3 THEN
         RAISE myex;
      END IF;
   END LOOP;
EXCEPTION
   WHEN myex THEN
      dbms_output.put.line('Employee number already exist in enum
table.');
END;
/
```

How to Define an Exception?

Output

```
SQL>@user_exp
Employee number already exist in enum table.

PL/SQL procedure successfully operation.
```

User-named Exceptions

User-named Exceptions

- You can define your own error message and error number using the Pragma EXCEPTION_INIT or RAISE_APPLICATION_ERROR function.

Pragma EXCEPTION_INIT

- The pragma is a keyword directive to execute proceed at compile time
- The pragma EXCEPTION_INIT function takes two arguments:

```
PRAGMA EXCEPTION_INIT(exception_name, -error_number);
```

exception_name

A character string supporting

up to 2048 bytes

error_number

A negative integer ranging

from –20000 to –20999

Pragma EXCEPTION_INIT

Syntax

```
DECLARE
   user_define_exception_name EXCEPTION;
   PRAGMA EXCEPTION_INIT(user_define_exception_name,-error_number);
BEGIN
   statement(s);
   IF condition THEN
      RAISE user_define_exception_name;
   END IF;
EXCEPTION
   WHEN user_define_exception_name THEN
      User defined statement (action) will be taken;
END;
```

Pragma EXCEPTION_INIT

Example

```
SQL>edit user-named_exp
DECLARE
   myex EXCEPTION;
   PRAGMA EXCEPTION_INIT(myex,-20015);
   n NUMBER := &n;
BEGIN
   FOR i IN 1..n LOOP
      dbms_output.put.line(i);
      IF i=n THEN
         RAISE myex;
      END IF;
   END LOOP;
EXCEPTION
   WHEN myex THEN
      dbms_output.put.line('loop  finish');
END;
/
```

Pragma EXCEPTION_INIT

Output

```
SQL>@user-named_exp
n number  &n= 5
1
2
3
4
5
loop finish

PL/SQL procedure successfully operation.
```

RAISE_APPLICATION_ERROR

- It is used to assign an exception name and an exception error code.

```
raise_application_error(error_number, error_message);
```

RAISE_APPLICATION_ERROR

Example

```
SQL>edit user-named_exp
DECLARE
   myex EXCEPTION;
   n NUMBER := &n;
BEGIN
   FOR i IN 1..n LOOP
      dbms_output.put.line(i);
      IF i=n THEN
         RAISE myex;
      END IF;
   END LOOP;
EXCEPTION
   WHEN myex THEN
      RAISE_APPLICATION_ERROR(-20015, 'loop finish');
END;
/
```

RAISE_APPLICATION_ERROR

Result

```
SQL>@raise_app_error
n number  &n= 5
1
2
3
4
5
ORA-20015: loop finish

PL/SQL procedure successfully operation.
```

WHEN OTHERS Clause

WHEN OTHERS Clause

- The WHEN OTHERS clause is used to trap all remaining exceptions that have not been handled by your Named System Exceptions and Named Programmer-defined Exceptions

Syntax

```
CREATE [OR REPLACE] FUNCTION function_name
  [ (parameter [,parameter]) ]
  RETURN return_datatype
IS | AS
  [declaration_section]

BEGIN
  executable_section

EXCEPTION
  WHEN exception_name1 THEN
    [statements]

  WHEN exception_name2 THEN
    [statements]

  WHEN exception_name_n THEN
    [statements]

  WHEN OTHERS THEN
    [statements]

END [function_name];
```

WHEN OTHERS Clause

Example

```
CREATE OR REPLACE PROCEDURE add_new_order
  (order_id_in IN NUMBER, sales_in IN NUMBER)
IS
  no_sales EXCEPTION;

BEGIN
  IF sales_in = 0 THEN
    RAISE no_sales;

  ELSE
    INSERT INTO orders (order_id, total_sales)
    VALUES ( order_id_in, sales_in );
  END IF;

EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    raise_application_error (-20001,'You have tried to insert a duplicate order_id.');

  WHEN no_sales THEN
    raise_application_error (-20001,'You must have sales in order to submit the order.');

  WHEN OTHERS THEN
    raise_application_error (-20002,'An error has occurred inserting an order.');

END;
```

SQLCODE Function

SQLCODE Function

- The SQLCODE function returns the error number associated with the most recently raised error exception. This function should only be used within the exception handling section of your code.

Syntax

```
SQLCODE
```

SQLCODE Function

Example

```
EXCEPTION

  WHEN OTHERS THEN

    raise_application_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

END;
```

SQLCODE Function

Example

```
EXCEPTION

  WHEN OTHERS THEN

    err_code := SQLCODE;

    err_msg := SUBSTR(SQLERRM, 1, 200);



    INSERT INTO audit_table (error_number,  error_message)

    VALUES (err_code, err_msg);

END;
```

SQLERRM Function

SQLERRM Function

- The SQLERRM function returns the error message associated with the most recently raised error exception. This function should only be used within the Exception Handling section of your code.

Syntax

```
SQLERRM
```

SQLERRM Function

Example

```
EXCEPTION

  WHEN OTHERS THEN

    raise_application_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

END;
```

SQLERRM Function

Example

```
EXCEPTION

  WHEN OTHERS THEN

    err_code := SQLCODE;

    err_msg := SUBSTR(SQLERRM, 1, 200);



    INSERT INTO audit_table (error_number,  error_message)

    VALUES (err_code, err_msg);

END;
```

Quiz

Quiz 1

What will be the value of svar after the execution ?

A	Error
B	10
C	5
D	None of the above

Answer 1

What will be the value of svar after the execution ?

A	Error
B	10
C	5
D	None of the above

Quiz 2

Which of the following is not correct about an Exception?

A	Raised automatically / Explicitly in response to an ORACLE_ERROR
B	An exception will be raised when an error occurs in that block
C	Process terminates after completion of error sequence.
D	A Procedure or Sequence of statements may be processed.

Answer 2

Which of the following is not correct about an Exception?

A	Raised automatically / Explicitly in response to an ORACLE_ERROR
B	An exception will be raised when an error occurs in that block
C	Process terminates after completion of error sequence.
D	A Procedure or Sequence of statements may be processed.

Quiz 3

Which of the following is not correct about User_Defined Exceptions?

A	Must be declared
B	Must be raised explicitly
C	Raised automatically in response to an Oracle error
D	None of the above

Answer 3

Which of the following is not correct about User_Defined Exceptions?

A	Must be declared
B	Must be raised explicitly
C	Raised automatically in response to an Oracle error
D	None of the above

Quiz 4

Observe the syntax given below Code Snippet –

The optional [FOR EACH ROW] clause specifies

A	A table with index.
B	A table with primary key.
C	A row level trigger.
D	A table with a unique key.

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Answer 4

Observe the syntax given below Code Snippet –

The optional [FOR EACH ROW] clause specifies

A	A table with index.
B	A table with primary key.
C	A row level trigger.
D	A table with a unique key.

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Quiz 5

List the correct sequence of commands to process a set of records when using explicit cursors ?

A	INITIALIZE, GET, CLOSE
B	CURSOR, GET, FETCH, CLOSE
C	OPEN, FETCH, CLOSE
D	CURSOR, FETCH, CLOSE

Answer 5

List the correct sequence of commands to process a set of records when using explicit cursors ?

A	INITIALIZE, GET, CLOSE
B	CURSOR, GET, FETCH, CLOSE
C	OPEN, FETCH, CLOSE
D	CURSOR, FETCH, CLOSE

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor