PL/SQL Exceptions

PL/SQL exceptions are predefined and raised automatically into oracle engine when any error occur during a program.

Each and every error has defined a unique number and message. When warning/error occur in program it's called an exception to contains information about the error.



In PL/SQL built in exceptions or you make user define exception. Examples of built-in type (internally) defined exceptions division by zero, out of memory. Some common built-in exceptions havepredefined names such as ZERO\_DIVIDE and STORAGE\_ERROR.

Normally when exception is fire, execution stops and control transfers to the exception-handling part of your PL/SQL block. Internal exceptions are raised implicitly (automatically) by the run-time system. User-defined exceptions must be raised explicitly by RAISE statements, which are also raise predefined exceptions.

PL/SQL exceptions consist following three,

1. Exception Type
2. Error Code
3. Error Message

#### **Syntax**

DECLARE

declaration statement(s);

BEGIN

statement(s);

EXCEPTION

WHEN built-in\_exception\_name\_1 THEN

User defined statement (action) will be taken;

WHEN built-in\_exception\_name\_2 THEN

User defined statement (action) will be taken;

END;

#### **Example Code**

*builtin\_exp.sql*

SQL>edit buitin\_exp

DECLARE

temp enum%rowtype;

BEGIN

SELECT \* INTO temp FROM enum

WHERE eno=3;

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line("Table haven't data");

END;

/

#### **Example Result**

SQL>@buitin\_exp  
Table haven't data  
  
PL/SQL procedure successfully operation.

#### **PL/SQL built in exceptions**

Following are built in type exception,

|  |  |  |
| --- | --- | --- |
| Exception | Error Code | Description |
| ACCESS\_INTO\_NULL | ORA-06530 | Exception raised when assign uninitialized (NULL) object. |
| CASE\_NOT\_FOUND | ORA-06592 | Exception raised when no any choice case found in CASE statement as well as no ELSE clause in 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 unique constraint column. |
| INVALID\_CURSOR | ORA-01001 | Exception raised when you perform operation on cursor and cursor is not really opened. |
| INVALID\_NUMBER | ORA-01722 | Exception raised when you try to explicitly conversion from string to a number fail. |
| LOGIN\_DENIED | ORA-01017 | Exception raised when log in into oracle with wrong username or password. |
| NO\_DATA\_FOUND | ORA-01403 | Exception raised when SELECT ... INTO statement doesn't fetch any row from a database table. |
| NOT\_LOGGED\_ON | ORA-01012 | Exception raised when your program try to get data from database and actually user not connected to Oracle. |
| 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 may be memory is dumped/corrupted. |
| SYS\_INVALID\_ROWID | ORA-01410 | Exception raised when you try to explicitly conversion from string character string to a universal rowid (uid) fail. |
| TIMEOUT\_ON\_RESOURCE | ORA-00051 | Exception raised when database is locked or ORACLE is waiting for a resource. |
| TOO\_MANY\_ROWS | ORA-01422 | Exception raised when SELECT ... INTO statement returns more than one row. |
| VALUE\_ERROR | ORA-06502 | Exception raised when arithmetic, conversion, defined size constraint error occurs. |
| ZERO\_DIVIDE | ORA-01476 | Exception raised when you program try to attempt divide by zero number. |

### PL/SQL User Defined Exception

PL/SQL user defined exception to make your own exception. PL/SQL give you control to make your own exception base on oracle rules. User define exception must be declare yourself and RAISE statement to raise explicitly.

#### **How to Define Exception**

1. Declare exception

You must have to declare user define exception name in DECLARE block.

user\_define\_exception\_name EXCEPTION;

Exception and Variable both are same way declaring but exception use for store error condition not a storage item.

1. RAISE exception

RAISE statement to raised defined exception name and control transfer to a EXCEPTION block.

RAISE user\_define\_exception\_name;

1. Implement exception condition

In PL/SQL EXCEPTION block add WHEN condition to implement user define action.

WHEN user\_define\_exception\_name THEN

User defined statement (action) will be taken;

#### **Syntax**

Check this user defined 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;

#### **Example Code**

*user\_exp.sql*

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;

/

#### **Example Result**

SQL>**@user\_exp**  
Employee number already exist in enum table.  
  
PL/SQL procedure successfully operation.

### User Named Exception PRAGMA EXCEPTION\_INIT, RAISE\_APPLICATION\_ERROR

PL/SQL user named exception. you can define your own error message and error number using Pragma EXCEPTION\_INIT or RAISE\_APPLICATION\_ERROR function.

### PL/SQL pragma EXCEPTION\_INIT

pragma EXCEPTION\_INIT : Pragma is a keyword directive to execute proceed at compile time. pragma EXCEPTION\_INIT function take this two argument,

1. exception\_name
2. error\_number

You can define pragrma EXCEPTION\_INIT in DECLARE BLOCK on your program.

PRAGMA EXCEPTION\_INIT(exception\_name, -error\_number);

*exception\_name* and *error\_number*define on yourself, where exception\_name is character string up to 2048 bytes suppot and error\_number is a negative integer range from -20000 to -20999.

#### **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;

#### **Example Code**

*user-named\_exp.sql*

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;

/

#### **Example Result**

SQL>**@user-named\_exp**  
n number &n= 5  
1  
2  
3  
4  
5  
loop finish  
  
PL/SQL procedure successfully operation.

### PL/SQL RAISE\_APPLICATION\_ERROR

In PL/SQL RAISE\_APPLICATION\_ERROR function use to assign exception name and exception error code. Define RAISE\_APPLICATION\_ERROR function syntax,

raise\_application\_error(error\_number, error\_message);

#### **Example Code**

*raise\_app\_error.sql*

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;

/

#### **Example Result**

SQL>**@raise\_app\_error**  
n number &n= 5  
1  
2  
3  
4  
5  
ORA-20015: loop finish  
  
PL/SQL procedure successfully operation.

When RAISE\_APPLICATION\_ERROR execute it's return error message and error code looking same as oracle built-in error.