

Learn How To Handle Exceptions In PL/SQL

Published on Oct 14, 2019 4.4K Views



Sanjana Nayan

If you are a programmer, you might be familiar with the concept of exception handling is an integral part of any [programming language](#). As errors are inevitable and even the smartest of us can make mistakes while writing code, we must be acquainted with how to handle them. In this article, we will be learning particularly about the exception handling in PL/SQL.

Below are the topics covered in this article :

- [What is an Exception?](#)
- [Syntax of Exception Handling](#)
- [Types of Exceptions](#)
 - [System defined](#)
 - [Named system exceptions](#)
 - [Unnamed system exceptions](#)
 - [User-defined](#)
 - [Steps to declare User-defined functions](#)
 - [Examples of User-defined functions](#)



What is an Exception?

Any abnormal condition or event that interrupts the normal flow of our program instructions at run time or in simple words an exception is an error.

Syntax of Exception Handling in PL/SQL

[illegible]

Here, we can list down as many exceptions as we want to handle. The default exception will be handled using 'WHEN others THEN'

Example of Exception Handling in PL/SQL

The below program displays the name and address of a student whose ID is given. Since there is no student with ID value 8 in our database, the program raises the run-time exception `NO_DATA_FOUND`, which is captured in the `EXCEPTION` block.

```

1 DECLARE
2     s_id studentS.id%type := 8;
3     s_name studentS.Name%type;
4     s_loc studentS.loc%type;
5 BEGIN
6     SELECT name, loation INTO s_name, s_loc
7     FROM students
8     WHERE id = s_id;
9     DBMS_OUTPUT.PUT_LINE ('Name: ' || s_name);
10    DBMS_OUTPUT.PUT_LINE ('Location: ' || s_loc);
11 EXCEPTION
12     WHEN no_data_found THEN
13         dbms_output.put_line('No such student!');
14     WHEN others THEN
15         dbms_output.put_line('Oops, Error!');
16 END;

```



Here, we can list down as many exceptions as we want to handle. The default exception will be handled using **'WHEN others THEN'**

Types of Exceptions in PL/SQL

- System defined
- User defied

Next in this article on exception handling in [PL/SQL](#), let us discuss about both of these types in detail.

System defined

Defined and maintained implicitly by the Oracle server, these exceptions are mainly defined in the Oracle Standard Package. Whenever an exception occurs inside the program, Oracle server matches and identifies the appropriate exception from the available set of exceptions available in the oracle standard package. Basically, these exceptions are predefined in [PL/SQL](#) which gets raised *WHEN particular database rule is violated*.

The **System-defined exceptions** are further divided into two categories:

- Named system exceptions
- Unnamed system exceptions

Named system Exceptions

The named PL/SQL exceptions are *named in the standard package of PL/SQL*, hence the developer does not need to define the PL/SQL exceptions in their code. PL/SQL provides many pre-defined named exceptions, which are executed when any database rule is violated by a program. The following table lists a few of the important pre-defined exceptions –

Exception	Oracle Error	SQLCODE	Description
ACCESS_INTO_NULL	06530	-6530	It is raised when a null object is automatically assigned a value.
CASE_NOT_FOUND	06592	-6592	It is raised when none of the choices in the WHEN clause of a CASE statement is selected, and there is no ELSE clause.
COLLECTION_IS_NULL	06531	-6531	It is raised when a program attempts to apply collection methods other than EXISTS to an uninitialized nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray.
DUP_VAL_ON_INDEX	00001	-1	It is raised when duplicate values are attempted to be stored in a column with a unique index.
INVALID_CURSOR	01001	-1001	It is raised when attempts are made to make a cursor operation that is not allowed, such as closing an unopened cursor.
INVALID_NUMBER	01722	-1722	It is raised when the conversion of a character string into a number fails because the string does not represent a valid number.
LOGIN_DENIED	01017	-1017	It is raised when a program attempts to log on to the database with an invalid username or password.
NO_DATA_FOUND	01403	+100	It is raised when a SELECT INTO statement returns no rows.
NOT_LOGGED_ON	01012	-1012	It is raised when a database call is issued without being connected to the database.
PROGRAM_ERROR	06501	-6501	It is raised when PL/SQL has an internal problem.
ROWTYPE_MISMATCH	06504	-6504	It is raised when a cursor fetches value in a variable having incompatible data type.
SELF_IS_NULL	30625	-30625	It is raised when a member method is invoked, but the instance of the object type was not initialized.
STORAGE_ERROR	06500	-6500	It is raised when PL/SQL ran out of memory or memory was corrupted.
TOO_MANY_ROWS	01422	-1422	It is raised when a SELECT INTO statement returns more than one row.
VALUE_ERROR	06502	-6502	It is raised when an arithmetic, conversion, truncation, or size constraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt is made to divide a number by zero.

Example




```

5      INSERT INTO student (student_id, student_name ,
6      VALUES ( student_id_in, student_name_in );
7  EXCEPTION
8      WHEN DUP_VAL_ON_INDEX THEN
9      raise_application_error (-20001, 'Duplicate student_id');
10     WHEN OTHERS THEN
11     raise_application_error (-20002, 'An error occurred. ');
12 END;
```

Moving on in this article on exception handling in PL/SQL, let us understand what are unnamed system exceptions.

Unnamed System Exceptions

The system exceptions for which Oracle does not have a name are known as unnamed system exceptions. These exceptions do not occur frequently and are written with a code and an associated message.



MySQL DBA Certification Training

- Instructor-led Sessions
- Real-life Case Studies
- Assignments
- Lifetime Access

Explore Curriculum

There are basically two ways to handle unnamed system exceptions:

1. Using the WHEN OTHERS exception handler
2. Associating the exception code to a name and using it as a named exception.

Some steps followed for unnamed system exceptions are:

- Raise them implicitly.
- In case they are not handled in 'WHEN Others' then, they have to be handled explicitly.
- To handle the exception explicitly, they can be declared using Pragma EXCEPTION_INIT and handled by referencing the user-defined exception name in the exception section.

An example of handling unnamed exceptions using Pragma EXCEPTION_INIT is provided later in the article. Moving on in this article on exception handling in PL/SQL, let us understand the User-defined exceptions.

User-defined

Like all other programming languages, Oracle also allows you to declare and implement your own exceptions. Unlike System defined exceptions, these exceptions are raised explicitly in the PL/SQL block.

Steps to declare User-defined exceptions in the Oracle database

We can define User-defined exceptions in Oracle database in the following 3 ways:

- Using Variable of EXCEPTION type

Here, we can declare a User-defined exception by declaring a variable of EXCEPTION [datatype](#) in our code and raise it explicitly in our program using RAISE statement.

- Using PRAGMA EXCEPTION_INIT function

We can define a non-predefined error number with the variable of EXCEPTION datatype

- Using RAISE_APPLICATION_ERROR method

Using this method, we can declare a User-defined exception with our own customized error number and message.

Till now you might have got a rough idea on the ways in which we can raise User-defined exceptions in PL/SQL. We will learn about each of the above-mentioned methods with examples further in this article on exception handling in PL/SQL.

Next in this article, let us proceed with the demonstrations of User-defined exception handling.





SQL Essentials Training & Certification

Reviews

★★★★★ 5(6821)



MySQL DBA Certification Training

Reviews

★★★★★ 5(3757)



MongoDB Certification Training

Reviews

★★★★★ 4(15278)



Apache Cassandra Certification Training

Reviews

★★★★★ 5(12341)

Demonstration of User-defined Exceptions

Moving on in this article on Exception Handling in PL/SQL, let us understand how to use the variable of EXCEPTION type.

Using Variable of EXCEPTION type

The process of declaring user-defined exception is divided into three parts and these 3 parts are:

- Declare a variable exception datatype
- Raise the Exception
- Handle the Exception

Let's write a code to demonstrate the above steps in detail.

```
1 DECLARE
2     var_dividend NUMBER :=10;
3     var_divisor  NUMBER :=0
4     var_result  NUMBER;
5     ex-DivZero  EXCEPTION
```

In the above declaration block, we have four variables, among which the first three are normal number datatype variables and the fourth one which is ex_DivZero is the special exception datatype variable. The fourth one is our user-defined exception.

```
1 DECLARE
2     var_dividend NUMBER :=10;
3     var_divisor  NUMBER :=0
4     var_result  NUMBER;
5     ex-DivZero  EXCEPTION
```

The above execution part of this anonymous block, will come into action only when the divisor is 0. If the divisor is zero as it is in our case, the error will be raised and the control of the program will skip all the next steps and will look for matching exception handler. In the case where it finds any other, it will perform the action accordingly, otherwise it will either terminate the program or prompt us with an unhandled system defined error.

```
1 EXCEPTION WHEN ex_DivZero THEN
2     DBMS_OUTPUT.PUT_LINE(' ERROR, The divisor can't be zero');
```

This is the exception handler. As soon as the user enters divisor as 0, the above message string will be prompted.

Final Code:

```
1 DECLARE
2     var_dividend NUMBER :=10;
3     var_divisor  NUMBER :=0
4     var_result  NUMBER;
5     ex-DivZero  EXCEPTION
6 BEGIN
7     IF var_divisor =0 THEN
8         RAISE ex-DivZero;
9     END IF;
10    var_result := var_dividend/var_divisor;
11    DBMS_OUTPUT.PUT_LINE ('Result = ' || var_result);
12 BEGIN
13     IF var_divisor =0 THEN
14         RAISE ex-DivZero;
15     END IF;
16    var_result := var_dividend/var_divisor;
17    DBMS_OUTPUT.PUT_LINE ('Result = ' || var_result);
18 END;
```



In the PRAGMA EXCEPTION_INIT function, an exception name is associated with an Oracle error number. This name can be used in designing the exception handler for the error. For huge projects with many user defined errors, PRAGMA EXCEPTION_INIT is the most useful and suitable method.

Syntax:

```
1 | PRAGMA EXCEPTION_INIT(exception_name, -Oracle_error_number);
```

Example

```
1 | DECLARE
2 |     deadlock_detected EXCEPTION;
3 |     PRAGMA EXCEPTION_INIT(deadlock_detected, -60);
4 | BEGIN
5 |     NULL; -- Some operation that causes an ORA-00060 error
6 | EXCEPTION
7 |     WHEN deadlock_detected THEN
8 |         NULL; -- handle the error
9 | END;
```

The PRAGMA EXCEPTION_INIT tells the compiler to associate an exception name with an Oracle error number as mentioned earlier. It lets you refer to any internal exception by name and write a specific handler for it. When you see an error stack, or sequence of error messages, the one on top is the one that can be trapped and handled.

Moving on in this article on Exception Handling in PL/SQL, let us understand how to use the RAISE_APPLICATION_ERROR method.

Using RAISE_APPLICATION_ERROR method

It's a procedure that comes inbuilt with the oracle software. Using this procedure we can associate an error number with a custom error message. Combining both the error number and the custom error message, an error string can be composed which looks similar to those default error strings which are displayed by oracle when an error is encountered. RAISE_APPLICATION_ERROR procedure is found inside DBMS_STANDARD package



MySQL DBA Certification Training

[Weekday / Weekend Batches](#)

[See Batch Details](#)

Syntax

```
1 | raise_application_error (error_number, message [, {TRUE | FALSE}]);
```

Example

```
1 | /* A trigger trg_emp_detail_chk is created.*/
2 | CREATE OR REPLACE TRIGGER trg_emp_detail_chk
3 |
4 | /* The trigger timing is declared as BEFORE UPDATE on the EMPLOYEES table.*/
5 | Before UPDATE ON employees
6 |
7 | DECLARE
8 | permission_denied EXCEPTION;
9 | BEGIN
10 |
11 | /*Start of the IF condition checking whether the day of the system time is either Saturday or
12 | IF trim(TO_CHAR(sysdate,'Day')) IN ('Saturday', 'Sunday') THEN
13 | raise_application_error(-20000, 'You are not authorized to do any modification in the weekends
14 |
15 | /* The procedure raise_application_error is called with the first parameter value as -20000 an
16 | with a default text stating that the user is not authorized to do any modification in the week
17 | END IF;
18 | END;
```

With this we come to an end of this article on “Exception handling in PL/SQL”. I hope this topic is understood well and helped you. Try to write your own codes and incorporate the methods explained in this article.



and administer the MySQL Database. It includes hands-on learning on concepts like MySQL Workbench, MySQL Server, Data Modeling, MySQL Connector, Database Design, MySQL Command line, MySQL Functions etc. End of the training you will be able to create and administer your own MySQL Database and manage data.

Got a question for us? Please mention it in the comments section of this “Exception Handling in PL/SQL” article and we will get back to you as soon as possible.

Recommended videos for you



Introduction to MongoDB

Watch Now



Build Application With MongoDB

Watch Now

Recommended blogs for you



What are SQL Operators and how do they work?

Read Article



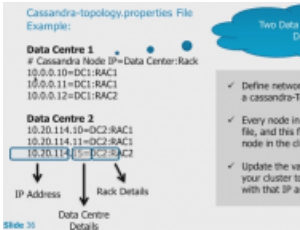
Foreign Key SQL : Everything You Need To Know About Foreign Key Operations

Read Article



SQL Functions: How to write a Function in SQL?

Read Article



Introduction to Snitch Cassandra

Read Article

⏪

Comments

0 Comments

Join the discussion

Enter your comment here...

Trending Courses in Databases



SQL Essentials Training & Certification

7k Enrolled Learners
Weekend/Weekday
Self Paced



MySQL DBA Certification Training

4k Enrolled Learners
Weekend
Live Class



MongoDB Certification Training

16k Enrolled Learners
Weekend
Live Class



Teradata Certification Training

3k Enrolled Learners
Weekend
Live Class





Browse Categories

- Artificial Intelligence
- BI and Visualization
- Big Data
- Blockchain
- Cloud Computing
- Cyber Security
- Data Science
- Data Warehousing and ETL
- DevOps
- Digital Marketing
- Enterprise
- Front End Web Development
- Mobile Development
- Operating Systems
- Programming & Frameworks
- Project Management and Methodologies
- Robotic Process Automation
- Software Testing
- Systems & Architecture



TRENDING CERTIFICATION COURSES

- [DevOps Certification Training](#)
- [AWS Architect Certification Training](#)
- [Big Data Hadoop Certification Training](#)
- [Tableau Training & Certification](#)
- [Python Certification Training for Data Science](#)
- [Selenium Certification Training](#)
- [PMP® Certification Exam Training](#)
- [Robotic Process Automation Training using UiPath](#)
- [Apache Spark and Scala Certification Training](#)
- [Microsoft Power BI Training](#)
- [Online Java Course and Training](#)
- [Python Certification Course](#)

COMPANY

- [About us](#)
- [News & Media](#)
- [Reviews](#)
- [Contact us](#)
- [Blog](#)
- [Community](#)
- [Sitemap](#)
- [Blog Sitemap](#)
- [Community Sitemap](#)
- [Webinars](#)

TRENDING MASTERS COURSES

- [Data Scientist Masters Program](#)
- [DevOps Engineer Masters Program](#)
- [Cloud Architect Masters Program](#)
- [Big Data Architect Masters Program](#)
- [Machine Learning Engineer Masters Program](#)
- [Full Stack Web Developer Masters Program](#)
- [Business Intelligence Masters Program](#)
- [Data Analyst Masters Program](#)
- [Test Automation Engineer Masters Program](#)
- [Post-Graduate Program in Artificial Intelligence & Machine Learning](#)
- [Post-Graduate Program in Big Data Engineering](#)

WORK WITH US

- [Careers](#)
- [Become an Instructor](#)
- [Become an Affiliate](#)
- [Become a Partner](#)
- [Hire from Edureka](#)

DOWNLOAD APP



CATEGORIES



CATEGORIES

- [Cloud Computing](#) | [DevOps](#) | [Big Data](#) | [Data Science](#) | [BI and Visualization](#) | [Programming & Frameworks](#) | [Software Testing](#) | [Project Management and Methodologies](#) | [Robotic Process Automation](#) | [Frontend Development](#) | [Data Warehousing and ETL](#) | [Artificial Intelligence](#) | [Blockchain](#) | [Databases](#) | [Cyber Security](#) | [Mobile Development](#) | [Operating Systems](#) | [Architecture & Design Patterns](#) | [Digital Marketing](#)

TRENDING BLOG ARTICLES



TRENDING BLOG ARTICLES

- [Selenium tutorial](#) | [Selenium interview questions](#) | [Java tutorial](#) | [What is HTML](#) | [Java interview questions](#) | [PHP tutorial](#) | [JavaScript interview questions](#) | [Spring tutorial](#) | [PHP interview questions](#) | [Inheritance in Java](#) | [Polymorphism in Java](#) | [Spring interview questions](#) | [Pointers in C](#) | [Linux commands](#) | [Android tutorial](#) | [JavaScript tutorial](#) | [jQuery tutorial](#) | [SQL interview questions](#) | [MySQL tutorial](#) | [Machine learning tutorial](#) | [Python tutorial](#) | [What is machine learning](#) | [Ethical hacking tutorial](#) | [SQL injection](#) | [AWS certification career opportunities](#) | [AWS tutorial](#) | [What Is cloud computing](#) | [What is blockchain](#) | [Hadoop tutorial](#) | [What is artificial intelligence](#) | [Node Tutorial](#) | [Collections in Java](#) | [Exception handling in java](#)



Become a Certified Professional →

© 2020 Brain4ce Education Solutions Pvt. Ltd. All rights Reserved. [Terms & Conditions](#)



[Legal & Privacy](#)

"PMP®", "PMI®", "PMI-ACP®" and "PMBOK®" are registered marks of the Project Management Institute, Inc. MongoDB®, Mongo and the leaf logo are the registered trademarks of MongoDB, Inc.

