**Exceptions**

**Introduction to Exceptions**

* An error condition is called an Exception
* When an error occurs, an exception is raised i.e. normal execution stops and control transfers to the exception handling part of the PL/SQL block or subprogram
* To handle raised exceptions, separate routines called exception handlers are written
* There are two types of exceptions
  + Pre-defined exceptions (Internal Exceptions)
  + User-defined exceptions
* You cannot declare an exception twice in the same block, but can declare the same exception in two different blocks
* Exceptions declared in a block are local to that block and global to all its sub-blocks
* Enclosing blocks cannot reference exceptions declared in a sub-block because blocks can only reference local or global exceptions

**Predefined Exceptions**

* Are implicitly raised whenever a PL/SQL block violates an Oracle rule or exceeds a system-dependent limit
* Every Oracle error has a number, but exceptions must be handled by name
* PL/SQL predefines some common Oracle errors as exceptions
* These predefined exceptions are declared globally by PL/SQL
* Some Pre-defined Exceptions
  + CURSOR\_ALREADY\_OPEN
  + NO\_DATA\_FOUND
  + TOO\_MANY\_ROWS
  + VALUE\_ERROR
  + ZERO\_DIVIDE
* More than one exception can be handled in a single exception handler by separating them with the keyword OR

EXCEPTION

WHEN NO\_DATA\_FOUND OR TOO\_MANY\_ROWS THEN

statements;

WHEN OTHERS THEN

statements;

END;

Examples of Exception handling –

1. **NO\_DATA\_FOUND error (Variable is not having any value.)**

declare

n emp.ename%type;

s emp.sal%type;

begin

select sal into s

from emp

where ename = '&n';

dbms\_output.put\_line('Salary is '|| s);

**Exception**

**When NO\_DATA\_FOUND then**

**dbms\_output.put\_line('No record');**

end;

1. **TOO\_MANY\_ROWS error (Variable is having more than one value)**

declare

s emp.sal%type;

begin

**select sal into s**

**from emp;**

dbms\_output.put\_line('The salary is '|| s );

Exception

When **TOO\_MANY\_ROWS** then

dbms\_output.put\_line('Variable can hold only one value at a time');

dbms\_output.put\_line('Please specify the name of person for getting the salary');

end;

**3) ZERO\_DIVIDE error (A number divided by zero)**

declare

x number;

y number;

z number;

begin

x := &x;

y := &y;

z := x/y;

dbms\_output.put\_line('The answer is ' || z);

Exception

When ZERO\_DIVIDE then

dbms\_output.put\_line('Cannot divide by zero!!!');

end;

1. **DUP\_VAL\_ON\_INDEX error (When a duplicate value is entered in a column having Unique constraint)**

declare

e emp.empno%type;

begin

e := &e;

insert into emp (empno )

values(e);

dbms\_output.put\_line('Successful');

Exception

When **DUP\_VAL\_ON\_INDEX** then

dbms\_output.put\_line('Value already exists');

end;

1. **VALUE\_ERROR (Error in conversion of string to number)**

declare

n number;

begin

n := '&n';

dbms\_output.put\_line(n);

Exception

When VALUE\_ERROR then

dbms\_output.put\_line('Please enter number only');

end;

1. **OTHERS (If no error handler works then at least OTHERS will work)**

declare

x number;

y number;

z number;

begin

x := &x;

y := &y;

z := x/y;

dbms\_output.put\_line('The answer is ' || z);

Exception

When OTHERS then

dbms\_output.put\_line('Some run time error has occurred');

dbms\_output.put\_line('Please execute the program again with proper values.');

rollback;

When too\_many\_rows then

dbms\_output.put\_line('More than one value');

When no\_data\_found then

dbms\_output.put\_line('No value');

end;

**Oracle Error Number for the Constraints violations:**

Primary Key and Unique is -00001

Foreign Key is -2291,

Not Null is -1400

Check is -2290

## SQLCODE AND SQLERRM

SQLCODE 🡪 Returns the numeric value for the error code.

SQLERRM 🡪 Returns the message associated with the error number.

**create table error\_log**

**(error\_number number,**

**error\_message varchar2(255)**

**);**

declare

s emp.sal%type;

v\_error\_code number;

v\_error\_message varchar2(255);

begin

**select sal into s from emp;**

exception

when others then

**v\_error\_code := SQLCODE;**

**v\_error\_message := SQLERRM;**

Insert into error\_log values(v\_error\_code, v\_error\_message);

commit;

end;

### Exception Handlers in nested block to continue after run time error comes

**declare**

**loan\_amt number;**

**no\_of\_months number;**

**installment\_rate number;**

**roi number;**

**tot\_amt number;**

**begin**

**loan\_amt := &loan\_amt;**

**no\_of\_months := &no\_of\_months;**

**begin**

**installment\_rate := loan\_amt / no\_of\_months;**

**exception**

**when zero\_divide then**

**no\_of\_months := 3;**

**installment\_rate := loan\_amt / no\_of\_months;**

**end;**

**/\* In any case the last 3 lines will get executed \*/**

**roi := installment\_rate \* 0.2; -- 20% roi**

**tot\_amt := roi + loan\_amt;**

**dbms\_output.put\_line('The total amount to be paid is '|| tot\_amt);**

**end;**

**User-defined Exceptions**

* User-defined exceptions need to be defined in the declarative part of a PL/SQL block, subprogram or database trigger
* Declared by naming the exception and defining it as datatype EXCEPTION
* Example

DECLARE

past\_due EXCEPTION;

zero\_error EXCEPTION;

* Like variables, user-defined exceptions must be given names
* Unlike variables, user-defined exceptions cannot be assigned values and cannot be used in SQL statements
* They need to be raised explicitly using the RAISE statement
* A block should RAISE an exception only when an error makes it impossible or impractical to finish processing
* RAISE statement for a given expression can be coded anywhere within the scope of that expression

IF mrec.ss\_fare <= 0 THEN

RAISE zero\_error;

END IF;

* An exception raised inside a handler immediately propagates to the enclosing block, which is searched to find a handler for the newly raised exception
* From there on, the exception propagates normally
* To re-raise an exception place a RAISE statement in its local handler

**Example of Exception variable using Raise key word**

declare

p number;

n number := 6;

si number;

r number := 10.5;

**EX Exception;**

Begin

p := &p;

if p < 100 then

**raise** **EX**;

else

si := (p \* n \* r) / 100;

dbms\_output.put\_line('The Simple Interest is '|| si);

end if;

---

---

---

Exception

When **EX** then

dbms\_output.put\_line('The principle amt should be greater than or equal to 100.');

End;

--------------------------------------------------------------------------------------

**RAISE\_application\_error**

This can be used to create user defined error message, which can be more descriptive than named exceptions.

Syntax - :

Raise\_application\_error(error number,error message);

where error number is any parameter between -20,000 and -20,999.Error message is text that is associated with this error. The message parameter must be less than 512 characters.

**Example of Raise\_application\_error**

declare

maths number;

Begin

maths := &maths;

if maths < 35 then

**raise\_application\_error(-20001,'Failed');**

else

dbms\_output.put\_line('Passed');

end if;

end;

**Whether code gets executed after the error??**

declare

maths number;

Begin

maths := &maths;

if maths < 35 then

raise\_application\_error(-20001,'Failed');

dbms\_output.put\_line('Line 1'); -- Code after error

else

dbms\_output.put\_line('Passed');

end if;

dbms\_output.put\_line('Line 2'); -- Code after error?????

end;

/

**What happens if that error is handled by the Exception block??**

declare

maths number;

Begin

maths := &maths;

if maths < 35 then

raise\_application\_error(-20001,'Failed');

else

dbms\_output.put\_line('Passed');

end if;

**Exception**

**when Others then**

**dbms\_output.put\_line('Error occurred');**

end;

/

What happens if the error number mentioned in the raise\_appilcation\_error is not within the limit of -20,000 and -20,999

declare

maths number;

Begin

maths := 23;

if maths < 35 then

raise\_application\_error(**-123**,'Failed');

dbms\_output.put\_line('Line 1'); -- Code after error

else

dbms\_output.put\_line('Passed');

end if;

dbms\_output.put\_line('Line 2'); -- Code after error

end;

/

**What if error comes in the Exception block??**

Declare

x number;

Begin

dbms\_output.put\_line('Line 1');

dbms\_output.put\_line(100 / 0);

dbms\_output.put\_line('Line 2');

Exception

**When Others Then**

**Begin**

**Select Sal into x from emp;**

**Exception**

**When Others Then**

**Null;**

**End;**

End;

/