EXCEPTION HANDLING IN PL/SQL

EXCEPTION: **RUNTIME ERRORS ARE CALLED AN EXCEPTION. IF AT ANY TIME AN ERROR OCCURS IN THE PL/SQL BLOCK AT THAT TIME PL/SQL BLOCK EXECUTION IS STOPPED AND ORACLE RETURNS AN ERROR MESSAGE.**

TO CONTINUE THE PROGRAM EXECUTION AND TO DISPLAY USER FRIENDLY MESSAGE EXCEPTION NEEDS TO BE HANDLE EXCEPTION INCLUDE EXCEPTION BLOCK IN PL/SQL.

EXCEPTIONS ARE CLASSIFIED INTO TWO TYPES. THOSE ARE

1) **SYSTEM/PRE-DEFINED EXCEPTION**
2) **USER DEFINED EXCEPTION**

SYNTAX:

```
DECLARE

< VARIABLES, CURSOR, USER DEFINE EXCEPTION>;

BEGIN

<STATEMENTS…………>;

EXCEPTION

WHEN <EXCEPTION NAME> THEN

<ERROR STATEMENTS……>;

END;
```

1) **SYSTEM/PRE-DEFINED EXCEPTION:**
   THESE ARE DEFINED BY ORACLE BY DEFAULT. WHENEVER RUNTIME ERROR IS OCCURRED IN PL/SQL THEN WE USE AN APPROPRIATE PRE-DEFINED EXCEPTION IN THE PROGRAM.

   SOME PRE-DEFINED EXCEPTIONS:
   i. NO_DATA_FOUND
   ii. TOO_MANY_ROWS
   iii. ZERO_DIVIDE
   iv. INVALID _CURSOR
   v. CURSOR_ALREADY_OPEN……ETC

<u>NO DATA FOUND:</u> WHENEVER PL/SQL BLOCK CARRY THE SELECT…..INTO CLAUSE AND ALSO IF REQUIRED DATA NOT AVAILABLE IN A TABLE THEN ORACLE SERVER RETURNS AN EXCEPTION.

EX: ORA-1403: NO DATA FOUND

TO HANDLE THIS EXCEPTION ORACLE PROVIDED "NO_DATA_FOUND" EXCEPTION.

<u>EX:</u>

```
DECLARE TENAME VARCHAR2(20); TSAL NUMBER (10);

BEGIN

SELECT ENAME, SAL INTO TENAME, TSAL FROM EMPLOYEE WHERE EID=&EID;

DBMS_OUTPUT.PUT_LINE(TENAME||','||TSAL);

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE ('RECORD IS NOT FOUND');

END;

/
```

<u>TOO MANY ROWS:</u> WHEN SELECT…. INTO CLAUSE TRY TO RETURN MORE THAN ONE VALUE OR ONE ROW THEN ORACLE SERVER RETURNS AN ERROR.

EX: ORA-1422: EXACT FETCH RETURNS MORE THAN REQUESTED NUMBER OF ROWS.

TO HANDLE FOR THIS ERROR ORACLE, PROVIDE "TOO_MANY_ROWS" EXCEPTION.

<u>EX:</u>

```
 DECLARE TSAL NUMBER (10);

 BEGIN

 SELECT SAL INTO TSAL FROM EMPLOYEE;

 DBMS_OUTPUT.PUT_LINE(TSAL);
 EXCEPTION

 WHEN TOO_MANY_ROWS THEN

 DBMS_OUTPUT.PUT_LINE ('FETCHING MORE THAN ONE');

 END;

/
```

<u>ZERO DIVIDE: -</u> IN ORACLE WHEN WE ARE TRIED TO PERFORM DIVISION WITH ZERO THEN ORACLE RETURN AN ERROR.

ORA-1476: DIVISOR IS EQUAL TO ZERO.

TO HANDLE FOR THIS ERROR ORACLE, PROVIDE “ZERO_DIVIDE” EXCEPTION

<u>EX:</u>

```
DECLARE X NUMBER (10); Y NUMBER (10); Z NUMBER (10);

BEGIN

X: =&X;

Y: =&Y;

Z: =X/Y;

DBMS_OUTPUT.PUT_LINE ('RESULT: -'||Z);

EXCEPTION

WHEN ZERO_DIVIDE THEN

DBMS_OUTPUT.PUT_LINE ('SECOND NUMBER SHOULD NOT BE ZERO');

END;

/
```

<u>INVALID CURSOR</u>: WHEN WE ARE NOT OPENING THE CURSOR BUT WE ARE TRY TO PERFORM OPERATIONS ON CURSOR THEN ORACLE RETURNS AN ERROR.

EX: ORA-1001: INVALID CURSOR

TO HANDLE THIS ERROR ORACLE, PROVIDE "INVALID_CURSOR" EXCEPTION.

EX:

```
DECLARE

CURSOR C1 IS SELECT * FROM EMPLOYEE;

TEID NUMBER (10); TENAME VARCHAR2(20); TSAL NUMBER (10);
TAGE NUMBER (10);

BEGIN

FETCH C1 INTO TEID, TENAME, TSAL, TAGE;

DBMS_OUTPUT.PUT_LINE (TEID||' '||TENAME||' '||TSAL||'
'||TAGE);

CLOSE C1;

EXCEPTION

WHEN INVALID_CURSOR THEN

DBMS_OUTPUT.PUT_LINE ('FIRST YOU MUST OPEN THE CURSOR');

END;

/
```

<u>CURSOR_ALREADY_OPEN</u>: BEFORE REOPENING THE CURSOR, WE MUST CLOSE THE CURSOR PROPERLY OTHERWISE ORACLE RETURNS AN ERROR I.E.

EX: ORA-6511: CURSOR_ALREADY_OPEN

TO HANDLE THIS ERROR ORACLE, PROVIDE 'CURSOR_ALREADY_OPEN' EXCEPTION.

EX:

```
DECLARE

CURSOR C1 IS SELECT * FROM EMPLOYEE;

TEID NUMBER (10); TENAME VARCHAR2(20); TSAL NUMBER (10);
TAGE NUMBER (10);

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO TEID, TENAME, TSAL, TAGE;

EXIT WHEN C1%NOTFOUND;

DBMS_OUTPUT.PUT_LINE (TEID||' '||TENAME||' '||TSAL||'
'||TAGE);

END LOOP;

OPEN C1;

EXCEPTION

WHEN CURSOR_ALREADY_OPEN THEN

DBMS_OUTPUT.PUT_LINE ('WE MUST CLOSE THE CURSOR BEFORE
REOPEN');

END;
```

SQLCODE & SQLERRM: PL/SQL PROVIDES FOLLOWING BUILT-IN PROPERTIES WHICH ARE USED IN ERROR HANDLING.

SQLCODE RETURNS ERROR CODE.

SQLERRM RETURNS ERROR MESSAGE.

EX:

```
DECLARE

X NUMBER (10);

Y NUMBER (20);

Z NUMBER (10);
```

```
BEGIN
X:=&X;
Y:=&Y;
Z:=X/Y;
DBMS_OUTPUT.PUT_LINE(Z);
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(SQLCODE);
DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

OUTPUT:

```
ENTER VALUE FOR X: 10
ENTER VALUE FOR Y: 2
5
ENTER VALUE FOR X: 10
ENTER VALUE FOR Y: 0
-1476---------ERROR CODE
ORA-01476: DIVISOR IS EQUAL TO ZERO------ERROR MESSAGE
```

USER DEFINE EXCEPTION:

- WHEN WE CREATE OUR OWN EXCEPTION NAME AND RAISE EXPLICITLY WHENEVER ISREQUIRED.THESE TYPE OF EXCEPTIONS ARE CALLED AS USER DEFINE EXCEPTIONS.

- GENERALLY, IF WE WANT TO RETURN MESSAGE AS PER CLIENT BUSSINESS RULES THEN WE MUST USE USER DEFINE EXCEPTIONS.

- TO CREATE A USER, DEFINE EXCEPTION NAME THEN WE FOLLOW THE FOLLOWING THREE STEPS ARE,

STEP1: DECLARE USER DEFINE EXCEPTION NAME:

SYNTAX:

```
<UD EXCEPTION NAME> EXCEPTION;
```

EX:

```
EX EXCEPTION;
```

STEP2: RAISE UD EXCEPTION:

SYNTAX:

```
RAISE <UD EXCEPTION NAME>;
```

EX:

```
RAISE EX;
```

STEP3: HANDLING UD EXCEPTION:

SYNTAX:

```
WHEN <UD EXCEPTION NAME> THEN
<STATEMENTS>;
END;
/
```

EX:

```
        WHEN EX THEN
        DBMS_OUTPUT.PUT_LINE ('UD MESSAGE');
        END;
        /
```

EX:

```
DECLARE
X INT;
Y INT;
Z INT;
EX EXCEPTION; ---------------(1)
BEGIN
X:=&X;
Y:=&Y;
IF Y=0 THEN
RAISE EX; -----------------(2)
ELSE
Z:=X/Y;
DBMS_OUTPUT.PUT_LINE(Z);
END IF;
EXCEPTION
WHEN EX THEN--------------(3)
DBMS_OUTPUT.PUT_LINE ('SECOND NUMBER NOT BE ZERO');
END;
/
```

RAISE APPLICATION ERROR (NUMBER, MESSAGE):

- IT IS A PRE-DEFINE METHOD WHICH IS USED TO DISPLAY A USER DEFINE EXCEPTION INFORMATION IN FORM OF ORACLE FORMAT.

- RAISE STATEMENT IS USED TO RAISE EXCEPTION AND ALSO HANDLING EXCEPTION WHERE AS RIASE_APPLICATION_ERROR () STATEMENT IS USED TO RAISE EXCEPTION BUT NOT HANDLING EXCEPTION.

- THIS METHOD IS HAVING TWO ARGUMENTS ARE NUMBER AND MESSAGE.

HERE,

NUMBER - NUMBER SHOULD BE -20001 TO -20999

MESSAGE - USER DEFINE EXCEPTION MESSAGE.

EX:

```
DECLARE

X INT;

Y INT;

Z INT;

EX EXCEPTION;

BEGIN

X:=&X;

Y:=&Y;

IF Y=0 THEN

RAISE EX;

ELSE

Z:=X/Y;

DBMS_OUTPUT.PUT_LINE(Z);

END IF;
```

```
EXCEPTION

WHEN EX THEN

RAISE_APPLICATION_ERROR(-20457,'SECOND NUMBER NOT BE
ZERO');

END;

/

ENTER VALUE FOR X: 10

ENTER VALUE FOR Y: 0

ERROR AT LINE 1:

ORA-20457: SECOND NUMBER NOT BE ZERO

ORA-06512: AT LINE 17
```

PRAGMA EXCEPTION_INIT (UNNAMED EXCEPTION):

- IN ORACLE IF WE WANT TO HANDLE OTHER THAN ORACLE PRE-DEFINE EXCEPTION NAME ERRORS THEN WE MUST USE "UNNAMED EXCEPTION" METHOD.IN THIS METHOD WE MUST CREATE A USER DEFINE EXCEPTION AND ASSOCIATE THIS EXCEPTION NAME ALONG WITH SOME ERROR NUMBER BY USING "PRAGMA EXCEPTION_INIT" METHOD.THIS METHOD IS HAVING TWO ARGUMENTS ARE,

SYNTAX:

```
PRAGMA EXCEPTION_INIT (<USER DEFINE EXCEPTION NAME>,
ERROR NUMBER)
```

EX:

```
DECLARE

X EXCEPTION;

PRAGMA EXCEPTION_INIT (X, -2291);

BEGIN
```

```
INSERT INTO EMP (EMPNO, ENAME, DEPTNO) VALUES
(1122,'SAI',50);

EXCEPTION

WHEN X THEN

DBMS_OUTPUT.PUT_LINE ('NOT ALLOWED INTO EMP TABLE
BECAUSE PARENT KEY IS NOT FOUND');

END;

/
```

NOTE: IN THE ABOVE PL/SQL PROGRAM TO HANDLE -2291 ERROR THEN USE THE EXCEPTION NAME IS "X".

EXCEPTION PROPAGATION:

- EXCEPTION BLOCK HANDLES EXCEPTION WHICH WAS RAISED IN BODY (EXECUTION BLOCK) BUT CANNOT HANDLE EXCEPTION WHICH WILL RAISE IN DECLARATION BLOCK.

EX:

```
DECLARE

X VARCHAR2(3):='PQRS';

BEGIN

DBMS_OUTPUT.PUT_LINE(X);

EXCEPTION

WHEN VALUE_ERROR THEN

DBMS_OUTPUT.PUT_LINE('INVALID STRING LENGTH');

END;

/

ERROR AT LINE 1:

ORA-06502: PL/SQL: NUMERIC OR VALUE ERROR: CHARACTER
STRING BUFFER TOO SMALL.
```

- TO OVERCOME THE ABOVE PROBLEM, WE NEED TO PREPARE NESTED PL/SQL BLOCK TO HANDLE EXCEPTION WHICH WAS RAISED

IN DECLARATION BLOCK THIS IS CALLED AS EXCEPTION PROPAGATION.

SOL:

```
BEGIN

DECLARE

X VARCHAR2(3):='PQRS';

BEGIN

DBMS_OUTPUT.PUT_LINE(X);

EXCEPTION

WHEN VALUE_ERROR THEN

DBMS_OUTPUT.PUT_LINE('INVALID STRING LENGTH');

END;

EXCEPTION

WHEN VALUE_ERROR THEN

DBMS_OUTPUT.PUT_LINE('STRING LENGTH IS GREATER THAN THE
SIZE OF VARIABLE X');

END;

/
```

OUTPUT:

```
STRING LENGTH IS GREATER THAN THE SIZE OF VARIABLE X.
```

<u>NOTE:</u>

- IN PL/SQL EXCEPTIONS ARE OCCURRED IN EXECUTION BLOCK, DECLARATION BLOCK.WHENEVER EXCEPTIONS ARE OCCURRED IN EXECUTION BLOCK THOSE EXCEPTIONS ARE HANDLED IN INNER BLOCK WHERE AS WHEN EXCEPTIONS ARE OCCURED IN DECLARATION BLOCK THOSE EXCEPTIONS ARE

HANDLED IN OUTER BLOCK ONLY.THIS MECHANISM IS CALLED AS "EXCEPTION PROPAGATION".