Examples of

PL/SQL Programs

```
# Oracle SQL*Plus
                          File Edit Search Options Help
                         SQL> --PL/SQL program to display the current date
type slash
                         SOL> DECLARE
then press
                                 TodaysDate DATE;
                           3 BEGIN
Enter to
                                 TodaysDate := SYSDATE;
execute the
                                 DBMS_OUTPUT.PUT_LINE('Today''s date is ');
                                 DBMS_OUTPUT.PUT_LINE(TodaysDate);
program
                              END:
your output
                         Today's date is
will be
                         28-NOV-01
different
                         PL/SQL procedure successfully completed.
```

Figure 4-6: PL/SQL program output

```
± Oracle SQL Plus
File Edit Search Options Help
SQL> --PL/SQL program to display the current date
SQL> DECLARE
        TodaysDate DATE;
     BEGIN
        TodaysDate := SYSDATE;
        DBMS_OUTPUT.PUT_LIME('Today''s date is ' || TO_CHAR(TodaysDate));
     END:
Today's date is 28-HOV-01
PL/SQL procedure successfully completed.
```

Figure 4-8: PL/SQL program with output concatenated on a single line

```
孝 Oracle SQL*Plus
                                                                             File Edit Search Options Help
SQL> DECLARE
        TodaysDate DATE;
  3
        Today VARCHAR2 (9);
        DayLength BINARY_INTEGER;
  5
     BEGIN
  6
        TodaysDate := SYSDATE;
        Today := TO_CHAR(SYSDATE, 'DAY');
  8
        Today := RTRIM(Today);
  9

    convert day display to lowercase

        Today := LOWER(Today);
 18
        DayLength := LENGTH(Today);
 11
        DBMS_OUTPUT.PUT_LINE('Today is ' || Today || ', ' || TO_CHAR(TodaysDa
 12
        DBMS_OUTPUT.PUT_LINE('The length of the word ' || Yoday ||
 <u>13</u>
14
        ' is ' || TO_CHAR(DayLength) || ' characters.');
 15
     END;
 16
Today is wednesday, 28-NOV-01
The length of the word wednesday is 9 characters.
PL/SQL procedure successfully completed.
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> --PL/SQL program to display the current day
SQL> DECLARE
        Today VARCHAR2(9);
     BEGIN
        Today := TO_CHAR(SYSDATE, 'DAY');
        Today := RTRIM(Today);
      --add IF/THEN statement to determine if current day is Friday
      IF Today !- 'FRIDAY' THEN
         DBMS_OUTPUT.PUT_LINE('Today is not Friday');
       END IF;
 10
     END;
 11
Today is not Friday
```



```
File Edit Search Options Help
SQL> --PL/SQL program to display the current day
SQL> DECLARE
        Today VARCHAR2(9);
     BEGIN
        Today :- TO CHAR(SYSDATE, 'DAY');
  5
        Today := RTRIM(Today);
  Ó
       --add IF/THEN statement to determine if current day is Friday
       IF Today - 'FRIDAY' THEN
         DBMS OUTPUT.PUT LINE('Today is Friday');
       ELSE
 18
         DBMS_OUTPUT.PUT_LINE('Today is not Friday');
 11
       END IF:
     END;
 12
 13
Today is not Friday
```



```
Help
File Edit Search Options
SQL> DECLARE
  2
       TermID BINARY INTEGER;
       TermDesc UARCHAR2(20);
    TermStatus VARCHAR2(20):
  5
     BEGIN
  6
       TermID := 1:
       TermDesc := 'Fall 2000';
       TermStatus := 'CLOSED';
      -- insert the records
 18
      INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 11
       TermID := TermID + 1:
      TermDesc := 'Spring 2001';
 12
 13
       INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 14
      TermID := TermID + 1;
 15
    TermDesc := 'Summer 2001':
 16
    INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 17
      COMMIT:
 18
    END:
 19
```

```
File Edit Search Options Help
SOL> DECLARE
 2
       LoopCount BINARY INTEGER:
  3
     BEGIN
       LoopCount := 0:
      LOGP
         LoopCount :- LoopCount + 1:
         IF LoopCount - 6 THEN
           EXIT:
         EMD IF:
 INSERT INTO count table UALUES(LoopCount):
      END LOOP:
 12 END:
 130
   PL/SOL procedure successfully completed.
ISOL> SELECT
    FROM count_table:
   COUNTER
```

```
♣ Oracle SQL Plus
```

```
File Edit Search Options Help
SQL> BEGIN
       FOR LoopCount 1H 1..5
       LOOP
         INSERT INTO count table VALUES(LoopCount):
       EMD LOOP:
  6 END;
PL/SQL procedure successfully completed.
SOL> SELECT
  2 FROH count_table;
   COUNTER
```

Exception Handling

```
🌲 Oracle SQL*Plus
 File Edit Search Options Help
SQL> DECLARE
        CurrentItemID BINARY_INTEGER;
        CurrentItemDesc VARCHAR2(30):
                                                 invalid
      BEGIN
                                                 item ID
        CurrentItemID := 890;
        SELECT ItemDesc
        INTO CurrentItemDesc
        FROM item
        WHERE itemid = CurrentItemID;
        DBMS_OUTPUT.PUT_LINE('The description of Item ID_;' || TO_CHAR(CurrentItemID) || ' is ' || CurrentItemDesc || '.');
 18
 11
 12
      END:
 13
DECLARE
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 6
```

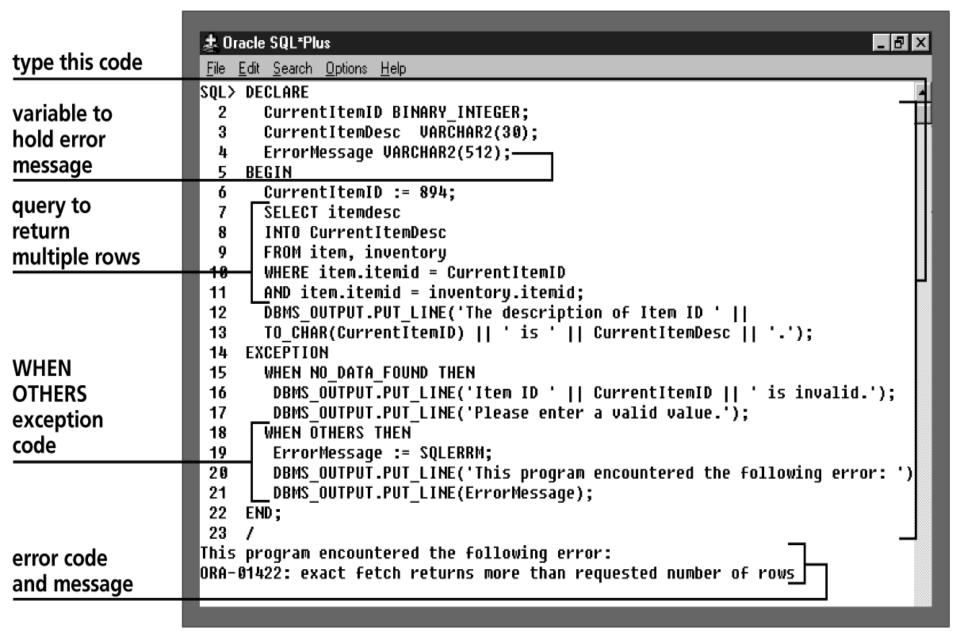


Figure 4-50: Displaying the Oracle error code and message in the WHEN OTHERS exception handler

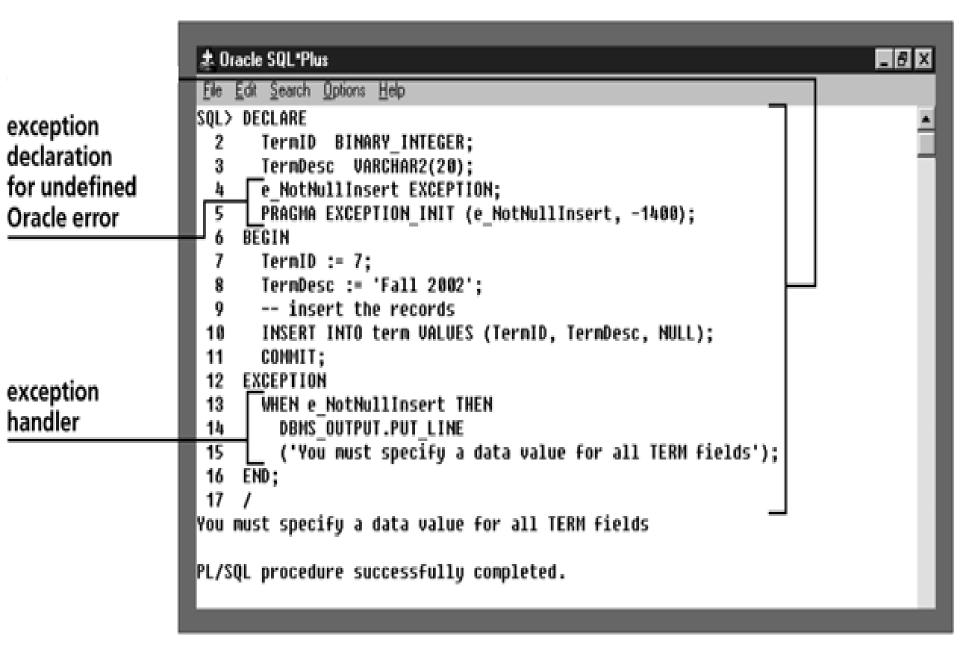


Figure 4-52: PL/SQL program with exception handler for undefined exception

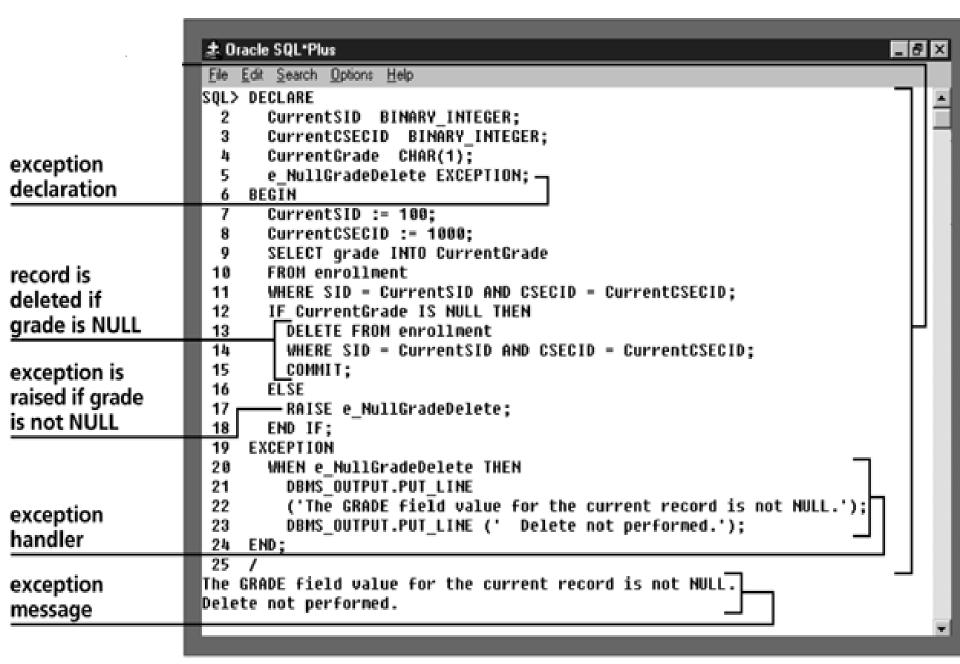


Figure 4-53: Creating and raising a user-defined exception

Examples of Cursors in PL/SQL

```
DECLARE
cursor payment_cursor is
 select cust_id, payment, total_due from payment_table;
cust_id payment_table.cust_id%TYPE;
payment payment_table.payment%TYPE;
total_due payment_table.total_due%TYPE;
BEGIN
 open payment_cursor;
 WHILE payment < total_due
   LOOP
      FETCH payment_cursor into cust_id, payment, total_due;
      EXIT WHEN payment_cursor%NOTFOUND;
      insert into underpay_table values (cust_id, 'STILL'
OWES');
   END LOOP;
  close payment_cursor;
```

You can use the FOR-LOOP in the previous block to implicitly fetch the current row of the cursor into the defined variables

Next we rewrite the previous example using FOR-LOOP

```
DECLARE
cursor payment_cursor is
 select cust_id, payment, total_due from payment_table;
cust_id payment_table.cust_id%TYPE;
payment payment_table.payment%TYPE;
total_due payment_table.total_due%TYPE;
Pay_rec payment_cursor%rowtype;
BEGIN
 open payment_cursor;
 FOR pay_rec IN payment_cursor
  LOOP
   IF pay_rec.payment < pay_rec.total_due THEN</pre>
     insert into underpay_table values (pay_rec.cust_id, 'STILL
OWES');
   END IF;
   END LOOP;
 close payment_cursor;
```

This example uses the FOR-LOOP to scroll the cursor. The FOR-LOOP is performing an implicit FETCH, which is omitted this time. Also, notice that the %NOTFOUND attribute has been omitted. This attribute is implied with the FOR-LOOP; therefore, this and the previous example yield the same basic results.

```
_ 8 ×
                     ♣ Oracle SQL*Plus
                     File Edit Search Options Help
                     SOL> DECLARE
                            CurrentBldqCode VARCHAR2(5);
                            CURSOR LocationCursor IS
                              SELECT bldg code, room, capacity
                              FROM location
                              WHERE bldq code = CurrentBldqCode;
                            LocationRow LocationCursor%ROWTYPE:
                         BEGIN
cursor
                            CurrentBldgCode := 'LIB';
                       9
%FOUND
                            OPEN LocationCursor:
                      10
attribute
                            FETCH LocationCursor
                     11
                     12
                            INTO LocationRow;
                            IF 'LocationCursor'&FOUND' THEN
                     13
                     14
                              DBMS OUTPUT.PUT LINE('The building code ' || CurrentBldgCode ||
                              ' returns at least one record.');
                      15
                     16
                            ELSE
                              DBMS OUTPUT.PUT LINE('The building code ' || CurrentBldqCode ||
                     17
                     18
                              ' does not return any records.');
                     19
                            END IF:
                     28
                            CLOSE LocationCursor:
                     21 END;
                     22
                     The building code LIB returns at least one record.
                    PL/SQL procedure successfully completed.
```

Figure 4-43: Using the %FOUND attribute to signal whether or not a cursor returns any records

Procedures and Functions in PL/SQL

Procedures

---- Procedure Call -----

Procedure Name (Actual Parameter List);

```
PROCEDURE UpdateInvValue IS
  CURSOR UpdateInvCursor IS
    SELECT * FROM inventory;
  InventoryRow UpdateInvCursor%ROWTYPE;
  InventoryValue NUMBER(11,2);
BEGIN
  FOR InventoryRow IN UpdateInvCursor LOOP
    InventoryValue := InventoryRow.curr price * InventoryRow.qoh;
    UPDATE inventory
    SET inv value := InventoryValue
    WHERE invid = InventoryRow.invid;
    COMMIT:
  END LOOP:
END;
```

Functions

```
Function < Function Name >
       ( <Parm1 Name> <Mode> <Data Type,
                                                      Formal
        <Parm2 Name> <Mode> <Data Type, ...
                                                    Parameters
       ) Return <Function Return Value Data Type> I
<Variable Declarations>
Begin
 Function Body; -- PL/SQL statements
 Return < Return Value>
End;
                 ---- Function Call -----
```

<Variable Name > := Function Name (Actual Parameter List);

```
FUNCTION Studentage (CurrentSID NUMBER)
  RETURN NUMBER
IS
              DATE:
  CurrentDate
 StudentDOB DATE;
 Currentage NUMBER;
BEGIN
  CurrentDate := SYSDATE:
  --retrieve SDOB for SID
  SELECT sdob
  INTO StudentDOB
  FROM student
  WHERE sid = CurrentSID;
 CurrentAge := TRUNC((CurrentDate - StudentDOB)/365);
  RETURN CurrentAge;
END:
```