File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/moreprfn.html

# More on Procedures and Functions

Remember, I am only INTRODUCING FUNCTIONS AND PROCEDURES in this course because we seem to have some extra time. You really need to do extensive reading on this topic because there are a lot of effective ways of approaching them and a lot of constraints that you need to make yourself aware of.

We are now going to look at a PL/SQL block of code that has both an internal procedure and an internal function. In the main block, I am looking at an input contribution amount. Depending on the amount, I set a code that will be passed to the function. The function uses the code to calculate the new contribution amount - my this is what you should give therefore what you did give - and passes it back to the main block which now calls the procedure to insert the new amounts in the new_donation table. In many ways this is adding extra work that would not be there if I had eliminate functions and procedures, but simple examples frequently have redundancy in them.

Looking at the code. The function is receiving the v_code and the v_amount that are passed and is storing them as f_code and f_contamt. Remember the pass for the function is accomplished with the statement: v_newcontamt := CalcNewDon(v_code, v_contamt); where CalcNewDon is the name of the function and the values in v_code and v_contamt are being passed to it. Note that v_newcontamt will received the result that is returned after the function has been executed. The statement: RETURN f_newcontamt; is the statement that sends back what is in f_newcontamt and it will be assigned to v_newcontamt.

There is one significant rule that we need to be aware of. When I described f_code and f_contamt, I gave them a type and not a length. The constraint of a length is not allowed in this definition area within a function. I did try using TYPE here, for example giving f_code a type of v_code and that worked. If you want more information on this, it is available in the PL/SQL book that was suggested for the course or you can look for notes up functions and parameters on the Internet.

Note that in the function, I passed the amount to it as well as the code. The processing would have worked if I had simply passed the code and used v_contamt instead of f_contamt in the processing. However, it is a good idea to pass the function the things that it needs and have it basically self-sufficient.

The procedure is the same as that in previous examples. The statement: AddDonProc (v_idno, v_driveno, v_newcontamt); sends the three pieces of data to the procedure where they are stored as p_idno, p_driveno and p_contamt. They are then used in the INSERT to write a new record/row to the new_donation table.

### SQL CODE:

```
SQL> edit call_adddonproc6
```

### PL/SQL CODE:

```
SET VERIFY OFF
DECLARE

  v_idno       new_donation.idno%TYPE :='&input_idno';
  v_driveno    new_donation.driveno%TYPE :='&input_driveno';
  v_contamt    new_donation.contamt%TYPE :=&input_contamt;
  v_newcontamt new_donation.contamt%TYPE;
  v_code       VARCHAR2(2);

  FUNCTION CalcNewDon
    (f_code    VARCHAR2, f_contamt NUMBER)
    RETURN NUMBER IS
      f_newcontamt  new_donation.contamt%TYPE;
    BEGIN
      IF f_code = 'EX' THEN
         f_newcontamt := f_contamt * 1.3;
      ELSE
         IF f_code = 'VG' THEN
            f_newcontamt := f_contamt * 1.2;
         ELSE
            IF f_code = 'OK' THEN
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search 9:37 AM 11/26/2019

Title: Nov 26-9:28 AM (1 of 12)

File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/moreprfn.html

```
SET VERIFY OFF
DECLARE

  v_idno       new_donation.idno%TYPE :='&input_idno';
  v_driveno    new_donation.driveno%TYPE :='&input_driveno';
  v_contamt    new_donation.contamt%TYPE :=&input_contamt;
  v_newcontamt new_donation.contamt%TYPE;
  v_code       VARCHAR2(2);

  FUNCTION CalcNewDon
    (f_code   VARCHAR2, f_contamt NUMBER)
    RETURN NUMBER IS
      f_newcontamt  new_donation.contamt%TYPE;
    BEGIN
      IF f_code = 'EX' THEN
         f_newcontamt := f_contamt * 1.3;
      ELSE
         IF f_code = 'VG' THEN
            f_newcontamt := f_contamt * 1.2;
         ELSE
            IF f_code = 'OK' THEN
               f_newcontamt := f_contamt * 1.1;
            ELSE
               f_newcontamt := f_contamt;
            END IF;
         END IF;
       END IF;
       RETURN f_newcontamt;
   END CalcNewDon;

  PROCEDURE AddDonProc
    (p_idno        new_donation.idno%TYPE,
     p_driveno     new_donation.driveno%TYPE,
     p_contamt     new_donation.contamt%TYPE) AS
  BEGIN
     INSERT INTO new_donation(idno, driveno,  contamt)
        VALUES(p_idno, p_driveno, p_contamt);
  END AddDonProc;

BEGIN
  IF v_contamt > 500 THEN
     v_code := 'EX';
     v_newcontamt := CalcNewDon(v_code, v_contamt);
  ELSE
     IF v_contamt > 250 THEN
        v_code := 'VG';
        v_newcontamt := CalcNewDon(v_code, v_contamt);
     ELSE
       IF v_contamt > 100 THEN
          v_code := 'OK';
          v_newcontamt := CalcNewDon(v_code, v_contamt);
       ELSE
          v_code := 'NG';
          v_newcontamt := CalcNewDon(v_code, v_contamt);
       END IF;
     END IF;
  END IF;
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search 9:39 AM 11/26/2019

Title: Nov 26-9:39 AM (2 of 12)

File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/moreprfn.html

```
     IF f_code = 'EX' THEN
        f_newcontamt := f_contamt * 1.3;
     ELSE
        IF f_code = 'VG' THEN
           f_newcontamt := f_contamt * 1.2;
        ELSE
           IF f_code = 'OK' THEN
              f_newcontamt := f_contamt * 1.1;
           ELSE
              f_newcontamt := f_contamt;
           END IF;
        END IF;
     END IF;
     RETURN f_newcontamt;
   END CalcNewDon;

  PROCEDURE AddDonProc
    (p_idno        new_donation.idno%TYPE,
     p_driveno     new_donation.driveno%TYPE,
     p_contamt     new_donation.contamt%TYPE) AS
  BEGIN
     INSERT INTO new_donation(idno, driveno,  contamt)
        VALUES(p_idno, p_driveno, p_contamt);
  END AddDonProc;

BEGIN
  IF v_contamt > 500 THEN
     v_code := 'EX';
     v_newcontamt := CalcNewDon(v_code, v_contamt);
  ELSE
     IF v_contamt > 250 THEN
        v_code := 'VG';
        v_newcontamt := CalcNewDon(v_code, v_contamt);
     ELSE
       IF v_contamt > 100 THEN
          v_code := 'OK';
          v_newcontamt := CalcNewDon(v_code, v_contamt);
       ELSE
          v_code := 'NG';
          v_newcontamt := CalcNewDon(v_code, v_contamt);
       END IF;
     END IF;
  END IF;

  AddDonProc (v_idno, v_driveno, v_newcontamt);
END;
/
SET VERIFY ON
```

### SQL CODE:

```
SQL> @ call_adddonproc6
Enter value for input_idno: 11111
Enter value for input_driveno: 100
Enter value for input_contamt: 700

PL/SQL procedure successfully completed.
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search 9:41 AM 11/26/2019

Title: Nov 26-9:41 AM (3 of 12)

File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/stillmoreprfn.html

### SQL CODE:

```
SQL> edit procfnc2
```

### PL/SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_fld1    testprfn2.fld1%TYPE;
  v_fld2    testprfn2.fld2%TYPE;
  v_fld3    testprfn2.fld3%TYPE;
  v_fld4    testprfn2.fld4%TYPE;
  v_fldtot  testprfn2.fld4%TYPE;

CURSOR fld_cursor IS
    SELECT * FROM testprfn2
       FOR UPDATE;

  PROCEDURE change_fld4
    (p_fldtot IN      testprfn2.fld1%TYPE,
     p_fld4   IN OUT testprfn2.fld4%TYPE) AS

  BEGIN
     IF p_fldtot > 30000 AND
       LENGTH (p_fld4) < 8 THEN
        p_fld4 := ('*'||p_fld4||'*');
     END IF;
  END change_fld4;

  FUNCTION calc_num
    (f_fld1 NUMBER, f_fld3 NUMBER)
    RETURN NUMBER IS
      f_fldtot   testprfn2.fld4%TYPE;
    BEGIN
      f_fldtot := f_fld3 - f_fld1;
      RETURN f_fldtot;
  END calc_num;

BEGIN
   OPEN fld_cursor;
   FETCH fld_cursor into v_fld1, v_fld2, v_fld3, v_fld4;
   WHILE fld_cursor%FOUND LOOP
      v_fldtot := calc_num(v_fld1, v_fld3);
      change_fld4(v_fldtot, v_fld4);
      UPDATE testprfn2
         SET fld4 = v_fld4
         WHERE CURRENT of fld_cursor;
      dbms_output.put_line(v_fld1||' '||v_fld4||' '||v_fldtot);
      FETCH fld_cursor into v_fld1, v_fld2, v_fld3, v_fld4;
   END LOOP;
   CLOSE fld_cursor;
END;
/
SET SERVEROUTPUT OFF
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search 9:44 AM 11/26/2019

Title: Nov 26-9:44 AM (4 of 12)

File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/stillmoreprfn.html

# Another example of internal procedures and functions

The first thing that I did for this example, was create a test table/file and make a copy of it. The tables are called testprfn and testprfn2.

### SQL CODE:

```
SQL> DESC testprfn;
 Name                            Null?    Type
 ------------------------------- -------- ----
 FLD1                                     NUMBER(3)
 FLD2                                     VARCHAR2(10)
 FLD3                                     NUMBER(5)
 FLD4                                     VARCHAR2(10)

SQL> SELECT * FROM testprfn;

     FLD1 FLD2            FLD3 FLD4
--------- ---------- --------- ----------
      123 Boston         12345 John
      234 Providence     23456 Ann
      345 Fall River     34567 Linda
      456 Boston         45678 Lawrence
      567 Fall River     56789 Susan
```

Then I wrote a program as a PL/SQL block with an embedded Procedure and an embedded Function. This program shows code but is not realistic in its achievement!

First the program declares a series of variables in the DECLARE that will be available throughout the code. They are variables to receive the data from the file and a variable to hold the result of a calculation. Since multiple records will be processed, I set up a cursor next. The cursor takes all of the records from the table testprfn2 and uses the FOR UPDATE clause since the table will be updated as a result of the code. Note that the procedures and functions are done after the variables and cursor.

The procedure is called change_fld4 and if you look at the logic, the function is done first. The procedure is passed p_fldtot which will only be used for input to the procedure and p_fld4 which will be changed and passed back. Notice again the way the fields being received are set up with ( ) around them and the word AS following them. In the procedure, I test to see if the p_fldtot which was passed is greater than 30000 and if the length of the p_fld4 that was passed is less than 8. If this is true, then p_fld4 is changed to have an * in front of and behind the name.

The function receives the contents of fld1 and fld3 from the record on the table (sent as v_fld1 and v_fld3 since this is where the fetch puts them). Within the function they are declared as f_fld1 and f_fld3 and as numbers without the length constraint that functions can not handle. The return field is also defined. A calculation is done that subtracts f_fld1 from f_fld3 and returns the defined return field using the RETURN syntax.

In the main block of the program, the cursor is opened and the initializing fetch is done, putting the fields from the record/row on the table in to v_fld1, v_fld2, v_fld3, and v_fld4. The loop is entered and will be executed until the fetch can no longer fine a record/row in the cursor to be processed. Then the calc_num function is executed using the statement: v_fldtot := calc_num(v_fld1, v_fld3); The result of the function execution is stored in v_fldtot. This is the field that is then passed change_fld4 along with v_fld4. The change is done within the procedure and the v_fld4 which was declared as IN OUT returns the adjusted name when appropriate. This is then used to update the current record/row from the cursor on the table. The dbms displays a record on the screen and then the fetch is done to fetch the next record/row from the cursor. After the loop has ended the cursor is closed and the block is ended.

### SQL CODE:

```
SQL> edit procfnc2
```

### PL/SQL CODE:

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search

9:44 AM 11/26/2019

Title: Nov 26-9:44 AM (5 of 12)

File Edit View History Bookmarks Tools Help

More on procedures and functions

www.pgrocer.net/Cis50/stillmoreprfn.html

```
DECLARE
  v_fld1    testprfn2.fld1%TYPE;
  v_fld2    testprfn2.fld2%TYPE;
  v_fld3    testprfn2.fld3%TYPE;
  v_fld4    testprfn2.fld4%TYPE;
  v_fldtot  testprfn2.fld4%TYPE;

CURSOR fld_cursor IS
    SELECT * FROM testprfn2
      FOR UPDATE;

  PROCEDURE change_fld4
    (p_fldtot IN     testprfn2.fld1%TYPE,
     p_fld4   IN OUT testprfn2.fld4%TYPE) AS

  BEGIN
     IF p_fldtot > 30000 AND
       LENGTH (p_fld4) < 8 THEN
        p_fld4 := ('*'||p_fld4||'*');
     END IF;
  END change_fld4;

  FUNCTION calc_num
    (f_fld1 NUMBER, f_fld3 NUMBER)
    RETURN NUMBER IS
      f_fldtot   testprfn2.fld4%TYPE;
    BEGIN
      f_fldtot := f_fld3 - f_fld1;
      RETURN f_fldtot;
  END calc_num;

BEGIN
   OPEN fld_cursor;
   FETCH fld_cursor into v_fld1, v_fld2, v_fld3, v_fld4;
   WHILE fld_cursor%FOUND LOOP
      v_fldtot := calc_num(v_fld1, v_fld3);
      change_fld4(v_fldtot, v_fld4);
      UPDATE testprfn2
         SET fld4 = v_fld4
         WHERE CURRENT of fld_cursor;
      dbms_output.put_line(v_fld1||' '||v_fld4||' '||v_fldtot);
      FETCH fld_cursor into v_fld1, v_fld2, v_fld3, v_fld4;
   END LOOP;
   CLOSE fld_cursor;
END;
/
SET SERVEROUTPUT OFF
```

**SQL CODE:**

```
SQL> edit procfnc2
SQL> @ procfnc2
123 John 12222
234 Ann 23222
345 *Linda* 34222
456 Lawrence 45222
567 *Susan* 56222
```

$$\begin{array}{r} 34567 \\ 345 \\ \hline 34222 \end{array} \qquad \begin{array}{r} 12345 \\ -\ 123 \\ \hline 12222 \end{array}$$

*Untitled - Notepad

File Edit Format View Help

```
SQL> SELECT * FROM testprfn;

     FLD1 FLD2             FLD3 FLD4
--------- ---------- --------- ----------
      123 Boston          12345 John
      234 Providence      23456 Ann
      345 Fall River      34567 Linda
      456 Boston          45678 Lawrence
      567 Fall River      56789 Susan
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? Refresh Firefox...

Type here to search

9:51 AM 11/26/2019

table I created, total_notes. Then I put in the command END trigger1;. Remember that I have to create the trigger before it will be used so the next SQL code is @ trigger1.

## SQL CODE:

```
SQL> edit trigger1
```

## PL/SQL CODE:

```
CREATE OR REPLACE TRIGGER trigger1
   AFTER UPDATE ON new_donation_one

DECLARE
   v_date     total_notes.changedate%TYPE :='27-JUL-99';
   v_totamt   total_notes.totamt%TYPE;

BEGIN
   SELECT SUM(contamt) INTO v_totamt
   FROM new_donation_one;
   INSERT INTO total_notes (changedate, totamt)
     VALUES(v_date, v_totamt);
END trigger1;
/
```

## SQL CODE:

```
SQL> @ trigger1
Input truncated to 1 characters

Trigger created.

SQL> edit use_trigger1
```

The use_trigger code does nothing special, it simply updates the contamt in the new_donation_one table for the record where the idno and the driveno are equal to the user input. There is no reference to the trigger. The trigger when it was created was associated with the updating of the new_donation_one table and since this code updates that table, it causes the trigger to be fired.

## PL/SQL CODE:

```
DECLARE
  v_idno     new_donation_one.idno%TYPE :='&input_idno';
  v_driveno  new_donation_one.driveno%TYPE :='&input_driveno';
  v_contamt  new_donation_one.contamt%TYPE :=&input_contamt;
BEGIN
  UPDATE new_donation_one
    SET contamt = v_contamt
    WHERE idno = v_idno AND driveno = v_driveno;
END;
/
```

The SQL below executes use_trigger1 which takes the user input and updates the table. The trigger is fired and a row/record is written in the total_notes table. The contents of both tables are shown below. Note that I executed use_trigger1, two times.

File Edit View History Bookmarks Tools Help

Introduction to Triggers

www.pgrocer.net/Cis50/triggers.html

```
old   2:   v_idno     new_donation_one.idno%TYPE :='&input_idno';
new   2:   v_idno     new_donation_one.idno%TYPE :='33333';
Enter value for input_driveno: 300
old   3:   v_driveno  new_donation_one.driveno%TYPE :='&input_driveno';
new   3:   v_driveno  new_donation_one.driveno%TYPE :='300';
Enter value for input_contamt: 30
old   4:   v_contamt  new_donation_one.contamt%TYPE :=&input_contamt;
new   4:   v_contamt  new_donation_one.contamt%TYPE :=30;

PL/SQL procedure successfully completed.

SQL> SELECT * FROM new_donation_one;

IDNO  DRI CONTDATE    CONTAMT
----- --- --------- ---------
11111 100 07-JAN-99        35
33333 300 10-MAR-99        30
22222 100 14-MAR-99        10
23456 300 14-JUN-99        10
12121 300 10-JUN-99        75

SQL> SELECT * FROM total_notes;

CHANGEDAT    TOTAMT
--------- ---------
27-JUL-99       160

SQL> @ use_trigger1
Input truncated to 1 characters
Enter value for input_idno: 23456
old   2:   v_idno     new_donation_one.idno%TYPE :='&input_idno';
new   2:   v_idno     new_donation_one.idno%TYPE :='23456';
Enter value for input_driveno: 300
old   3:   v_driveno  new_donation_one.driveno%TYPE :='&input_driveno';
new   3:   v_driveno  new_donation_one.driveno%TYPE :='300';
Enter value for input_contamt: 45
old   4:   v_contamt  new_donation_one.contamt%TYPE :=&input_contamt;
new   4:   v_contamt  new_donation_one.contamt%TYPE :=45;

PL/SQL procedure successfully completed.

SQL> SELECT * FROM new_donation_one;

IDNO  DRI CONTDATE    CONTAMT
----- --- --------- ---------
11111 100 07-JAN-99        35
33333 300 10-MAR-99        30
22222 100 14-MAR-99        10
23456 300 14-JUN-99        45
12121 300 10-JUN-99        75

SQL> SELECT * FROM total_notes;

CHANGEDAT    TOTAMT
--------- ---------
27-JUL-99       160
27-JUL-99       195
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search

10:04 AM 11/26/2019

Title: Nov 26-10:04 AM (8 of 12)

File Edit View History Bookmarks Tools Help

CIS150/50 Course Page

www.pgrocer.net/PFGCIS150F18.html

## *Oracle and SQL*
## *CIS150/50*

### *About This Course:*
- **Need help or support?**
- **Introduction to CIS150**
- **Syllabus**
- **Withdrawal Policy as of Fall 2017**

### *Site Resources:*
- **Notes and Handouts**
- **Presentations**
- **SmartBoard Presentations**
- **Audio**

### *Weekly Schedule:*
- **Weekly schedule - CIS150/50**

### *Using Oracle:*
- **Installation and login**

### *Links:*
- **Links & Tutorials**



| | |
|---|---|
| Week of December 2nd | chance to get caught up and improve your grade - PLEASE take advantage of it.<br>*Assignments:*<br>Write code to take in an id and determine if the check digit is correct - use the notes on check digits posted last week to see the logic.<br>Check back. |
| Week #13<br>Week of November 26th | I believe Lindsey is available Tues 2:30 to 6 and Thurs 3 to 5 in K101. Dave is available Mon and Wed from 4-7 in K116 and Sat 11 - 2 in K101. I am available Monday at 10 and Tuesday at 2 this week.<br>A group of alum are coming to talk about their careers on Tues at 11 and again at 12:30 in K101. Please try to come.<br>**Donor example with 1 cursor**<br>**More on internal procedures and functions**<br>**Still more on procedures and functions**<br>**Introduction to triggers**<br>**More on triggers**<br>**Notes on check digits** From last fall.<br>**More on Procedures - Internal Procedures**<br>**Separate speaker notes for More on Procedures - Internal Procedures**<br>**Introduction to Triggers**<br>**Separate speaker notes for Introduction to Triggers**<br>**More and Still More on Procedures and Functions**<br>**Separate speaker notes for More and Still More on Procedures and Functions**<br>**More on Triggers**<br>**Separate speaker notes for More on Triggers**<br>*Assignments:*<br>**Assignment on procedures, functions and triggers** |
| Week #12<br>Week of November 19th | Lindsey is available for help Tues 2:30 to 6 in K101. Dave will be available on Monday and Wednesday from 4 to 9 - check K116 and if he is not there, check with the help desk.<br>Example we will look at:<br>**type examples**<br>We will look at types and then move on to procedures and functions.<br>Here are notes on functions, procedures.<br>**Introduction to procedures**<br>**Introduction to functions**<br>**Internal procedures**<br>**Introduction to PL/SQL Procedures**<br>**Separate speaker notes for Introduction to PL/SQL Procedures**<br>**Introduction to PL/SQL Functions**<br>**Separate speaker notes for Introduction to PL/SQL Functions**<br>Catch up time! |
| Week #11<br>Week of November 12th | Lindsey is available for help Tues 2:30 to 6 and Thurs 3 to 5. Dave is available Wed from 4-9 (if anyone is around - let him know if you are coming late) in K116 and Thursday from 4 - 7 (room not set yet - ask at the help desk) and Friday from 3 to 5. I am available when I am not doing advisement so check K101 and K112 (my office).<br>Speaking of advisement - please sign up!<br>More cursors... Then exceptions.<br>**Introduction to exceptions**<br>**Exceptions in PL/SQL**<br>**Separate speaker notes for Exceptions in PL/SQL**<br>*Assignments:*<br>**Cursor/exception assignment** |

www.pgrocer.net/Cis51/mod11.html

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

Type here to search

10:05 AM 11/26/2019

Title: Nov 26-10:05 AM (9 of 12)

# MOD 11 Check Digit

A check digit is a number that is used to validate a series of numbers whose accuracy you want to insure. Frequently the last digit of a number string such as identification number is a check digit. Lets say the identification number starts out at 6 digits. A calculation is done using the six digits and a seventh digit is produced as a result of the calculation. This number is the check digit. There are many calculations that can be used - this example illustrates the logic of the MOD11 check digit.

**Steps to calculate the MOD11 check digit for a number such as an id #:**

- Assign weights to each digit of the id #. The weights in MOD11 are from 2 through a maximum of 10 beginning with the low order position in the field.
- Each digit in the id # is multiplied by its weight
- The results of the multiplication are added together
- This product is divided by the modulus number 11
- The remainder is subtracted from the modulus number 11 giving the check digit

Example: find the check digit for the number **036532**

| | | | | | |
|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 |
| x7 | x6 | x5 | x4 | x3 | x2 |
| 0 | 18 | 30 | 20 | 9 | 4 |

```
0 + 18 + 30 + 20 + 9 + 4 = 81
```

```
81/11 = 7 remainder 4
```

```
11 - 4 = 7
```

**7 is therefore the check digit.**

**PROBLEMS:** If the remainder from the division is 0 or 1, then the subtraction will yield a two digit number of either 10 or 11. This won't work, so if the check digit is 10, then X is frequently used as the check digit and if the check digit is 11 then 0 is used as the check digit. If X is used, then the field for the check digit has to be defined as character (PIC X) or there will be a numeric problem.

**Steps to verify if the check digit is included as part of the number:**

- The entire number is multiplied by the same weights that were used to calculate and the check digit itself is multiplied by 1.
- The results of the multiplication are added together.
- The sum is divided by 11 and if the remainder is 0, the number is correct.

**PROBLEM:** Note that if the check digit is X then 10 is used in the multiplication. Code for this occurrence must be included.

Example of verifying the number **0365327** where 7 is the calculated MOD11 check digit:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 | 7 |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 |
| 0 | 18 | 30 | 20 | 9 | 4 | 7 |

- Assign weights to each digit of the id #. The weights in MOD11 are from 2 through a maximum of 10 beginning with the low order position in the field.
- Each digit in the id # is multiplied by its weight
- The results of the multiplication are added together
- This product is divided by the modulus number 11
- The remainder is subtracted from the modulus number 11 giving the check digit

Example: find the check digit for the number **036532**

| | | | | | |
|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 |
| x7 | x6 | x5 | x4 | x3 | x2 |
| 0 | 18 | 30 | 20 | 9 | 4 |

```
0 + 18 + 30 + 20 + 9 + 4 = 81

81/11 = 7 remainder 4

11 - 4 = 7
```

**7 is therefore the check digit**.

**PROBLEMS:** If the remainder from the division is 0 or 1, then the subtraction will yield a two digit number of either 10 or 11. This won't work, so if the check digit is 10, then X is frequently used as the check digit and if the check digit is 11 then 0 is used as the check digit. If X is used, then the field for the check digit has to be defined as character (PIC X) or there will be a numeric problem.

**Steps to verify if the check digit is included as part of the number:**

- The entire number is multiplied by the same weights that were used to calculate and the check digit itself is multiplied by 1.
- The results of the multiplication are added together.
- The sum is divided by 11 and if the remainder is 0, the number is correct.

**PROBLEM:** Note that if the check digit is X then 10 is used in the multiplication. Code for this occurrence must be included.

Example of verifying the number **0365327** where 7 is the calculated MOD11 check digit:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 | 7 |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 |
| 0 | 18 | 30 | 20 | 9 | 4 | 7 |

```
0 + 18 + 30 + 20 + 9 + 4 + 7 = 88

88/11 is 8 remainder 0
```

Since the remainder from this calculation is 0, the check digit 7 is valid.

**Steps to calculate the MOD11 check digit for a number such as an id #:**

- Assign weights to each digit of the id #. The weights in MOD11 are from 2 through a maximum of 10 beginning with the low order position in the field.
- Each digit in the id # is multiplied by its weight
- The results of the multiplication are added together
- This product is divided by the modulus number 11
- The remainder is subtracted from the modulus number 11 giving the check digit

Example: find the check digit for the number **036532**

| | | | | | |
|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 |
| x7 | x6 | x5 | x4 | x3 | x2 |
| 0 | 18 | 30 | 20 | 9 | 4 |

```
0 + 18 + 30 + 20 + 9 + 4 = 81

81/11 = 7 remainder 4

11 - 4 = 7
```

**7 is therefore the check digit.**

**PROBLEMS:** If the remainder from the division is 0 or 1, then the subtraction will yield a two digit number of either 10 or 11. This won't work, so if the check digit is 10, then X is frequently used as the check digit and if the check digit is 11 then 0 is used as the check digit. If X is used, then the field for the check digit has to be defined as character (~~PIC X~~) or there will be a numeric problem.

**Steps to verify if the check digit is included as part of the number:**

- The entire number is multiplied by the same weights that were used to calculate and the check digit itself is multiplied by 1.
- The results of the multiplication are added together.
- The sum is divided by 11 and if the remainder is 0, the number is correct.

**PROBLEM:** Note that if the check digit is X then 10 is used in the multiplication. Code for this occurrence must be included.

Example of verifying the number **0365327** where 7 is the calculated MOD11 check digit:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 3 | 6 | 5 | 3 | 2 | 7 |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 |
| 0 | 18 | 30 | 20 | 9 | 4 | 7 |

```
0 + 18 + 30 + 20 + 9 + 4 + 7 = 88

88/11 is 8 remainder 0
```

Since the remainder from this calculation is 0, the check digit 7 is valid.