The COBOL Report Writer

Introduction

Aims

The aim of this unit is to provide you with a solid understanding of COBOL Report Writer. This unit introduces the report writer by -

1. Looking at a fairly complex report created using the report writer and examining what the Report Writer had to do to produce the report.
2. Looking at the Procedure Division of the program that produced the report.
3. Writing a simple version of the the program that produced the report.
4. Adding more features to the report program.
5. Examining the use of Declaratives in extending the functionality of the Report Writer.
6. Examining a full report program that uses Declaratives.

In the next unit we examine the syntax and semantics of the Report Writer clauses and verbs in detail.

Objectives

By the end of this unit you should -

1. Understand how the Report Writer works
2. Understand how and when to use the seven different types of report group.
3. Be able to control the placement of print items on the page and to control when the Report Writer goes to a new page.
4. Understand how to use the SUM clause to automatically accumulate totals.
5. Be able to set up control break items and understand how the relationship between them is controlled.
6. Understand how and when to use Declaratives.

Prerequisites

You should be familiar with the material covered in the unit;

- Data Declaration

- Iteration
- Selection
- Tables
- Edited Pictures
- The INSPECT verb
- The STRING verb
- The UNSTRING verb
- Sequential files

To top of page

The Report Writer in Action

Introduction

Producing printed reports is an important aspect of business programming. Unfortunately, report programs are often tedious to code. Report programs are long, and frequently consist of mere repetitions of the tasks and techniques used in other report programs. To simplify the task of writing report programs COBOL contains the Report Writer.

The COBOL Report Writer is very large. It includes many new DATA DIVISION entries, including a REPORT SECTION, and a number of new PROCEDURE DIVISION verbs.

An example report - with animated commentary.

The first page from an example report, produced by the Report Writer, is shown below. The report shows the sales made by Bible salespersons in all of the cities of Ireland.

In the attached animated commentary we will examine what the program has to do to produce the report.

The Procedure Division that produces the sales report

In a program that did not use the Report Writer, the PROCEDURE DIVISION required to produce the sales report shown above would occupy a hundred lines or so of code. When the REPORT WRITER is used, only the PROCEDURE DIVISION shown below is required.

All the work of printing the report is being done in just 10 statements.

```
PROCEDURE DIVISION.
Begin.
    OPEN INPUT SalesFile.
    OPEN OUTPUT PrintFile.
    READ SalesFile
         AT END SET EndOfFile TO TRUE
    END-READ.
    INITIATE SalesReport.
    PERFORM PrintSalaryReport
            UNTIL EndOfFile.
    TERMINATE SalesReport.
    CLOSE SalesFile, PrintFile.
    STOP RUN.


PrintSalaryReport.
    GENERATE DetailLine.
    READ SalesFile
          AT END SET EndOfFile TO TRUE
    END-READ.
```

So much work, so little Procedure Division code

How can so much work be done in so little PROCEDURE DIVISION code? How does the report writer know that page headings are required or that it is time to print the salesperson or city totals.

To achieve so much in so little PROCEDURE DIVISION code, the Report Writer uses a declarative approach to programming rather than the procedural one familiar to most programmers.

When the Report Writer is used, the programmer declares **what** to print when a page heading, page footing, salesman total, city total or final total is required and the Report Writer works out **when** to print these items.

In keeping with the adage that "there is no such thing as free lunch", the PROCEDURE DIVISION of a Report Writer program is short because most of the work is actually done in the (greatly expanded) DATA DIVISION.

How the Report Writer works

Report Groups

The Report Writer works by recognizing that many reports take (more or less) the same shape.

- There may be headings at the beginning of the report and footings at the end.
- There may be headings at the top of each page and footings at the bottom.
- Headings and Footings may need to be printed whenever there is a control break (i.e., when the value in a specified field changes, such as when the SalesPerson number changes in the example above).
- Detail lines must be printed.

The Report Writer calls these different report items - Report Groups. Reports are organized around report groups and the Report Writer recognizes seven types of report group (shown below). The indentation shows the importance/order of execution of the report groups.

REPORT HEADING or RH group
- printed once at the beginning of the report

PAGE HEADING of PH group
- printed at the top of each page

CONTROL HEADING or CH group
- printed at the beginning of each control break

DETAIL or DE group
- printed each time the GENERATE statement is executed

CONTROL FOOTING or CF group
- printed at the end of each control break

PAGE FOOTING or PF group
- printed at the bottom of each page

REPORT FOOTING or RF group
- printed once at the end of the report.

Report Groups are defined as records in the REPORT SECTION of the DATA DIVISION. Most groups are defined once for each report, but control groups are defined for each control break item. For instance, in the example program, control footings are defined on the SalesPersonNumber, the CityCode and FINAL. FINAL is a special control group that is invoked before the normal control groups (CONTROL HEADING FINAL) and after the normal control groups (CONTROL FOOTING FINAL).

An ordinary control group is defined on some control break data-item. The Report Writer monitors the contents of the data-item and when the value in the item changes a control break is automatically initiated and the CONTROL FOOTING group (if there is one) is printed.

Report writing made easy

Writing report programs is time consuming. It may be difficult to get the vertical and horizontal placement of printed items correct. Many lines of code may have to be written merely to move values to their corresponding items in the print line. A line count has to be kept to control when the report prints on a new page and a page count is often required.

The Report Writer makes all of this easy by;

- Allowing simple vertical and horizontal placement of printed items using the LINE IS and COLUMN IS phrases in the data declaration
- Automatically moving data values to output items
- Keeping a line count and automatically generating report and page headers and footers at the appropriate times
- Keeping a page count which can be referenced in the report declaration
- Recognizing control breaks and automatically generating the appropriate control headings and footings
- Automatically accumulating totals, sub-totals and final totals

Let's write a Report Writer program!

Introduction

Let's write a Report Writer program!

We'll start by writing a simpler version of the program that produces example report shown above and then we'll add to it to create a report program with even more features than the one shown.

Let's start by looking the data we have been giving to work with and at what is required in our simple report.

Report Specification

Bible salespersons work in each of the cities of Ireland. Each time a salesperson sells some bibles the value of that sale is recorded in a sales file along with the salesperson number and the city code.

The sales file has been validated and sorted on ascending SalesPersonNum within ascending CityCode.

We want to write a program that will produce a report that prints -

- a heading and a footing on each page
- for each sale record we want to print the the city name (obtained from a table), salesperson number and the value of the sale
- the total value of sales for each salesperson

A simple report

The first page produced by the simple report program we are going to write is shown below. In the section that follows we will examine the program that created the report.

```
           An example COBOL Report Program
     Bible Salesperson - Sales and Salary Report

 City      Salesperson     Sale
 Name       Number         Value
Dublin        1           $111.50
Dublin        1           $222.50
              Sales for salesperson 1     =      $334.00


Dublin        2         $1,111.50
Dublin        2         $1,222.50
Dublin        2         $1,333.50
              Sales for salesperson 2     =    $3,667.50


Dublin        3           $777.70
              Sales for salesperson 3     =      $777.70


Belfast       1           $111.50
Belfast       1           $222.50
              Sales for salesperson 1     =      $334.00


Belfast       2         $1,111.50
Belfast       2         $1,222.50
              Sales for salesperson 2     =    $2,334.00


Belfast       3           $111.10
Belfast       3           $111.10
              Sales for salesperson 3     =      $222.20


Belfast       4           $111.50
Belfast       4         $1,502.50
Belfast       4         $2,505.50
              Sales for salesperson 4     =    $4,119.50


Cork          1           $333.50











Programmer - Michael Coughlan              Page :  1
```

Code & Commentary

Let's add some features to our Report Program

Adding a city and a final total

Let's add city totals and a final total to the Report. The city total will be the sum of all the salesperson totals for the city and the final total will be the sum of all the city totals. A city total will be printed when the CityCode changes (i.e. when there is a control break on the CityCode) and the final total will be printed at the end of the report.

What changes do we have to make to our program to get it to print the city and final totals?

Printing the city total

If we want to print the city total we must set up a report group for it describing **what** we want printed. To describe **when** we want the group printed, we must specify that the group is a CONTROL FOOTING group and we must identify the CityCode as the control break data-item. Since the city control break is senior to the salesperson control break we must indicate this by placing CityCode before SalespersonNumber in the CONTROLS are phrase.

The total for the city is automatically accumulated by means of the SUM clause. Every time a salesperson total is printed the phrase - SUM SMS - adds the total to the city total. This is why the item for printing the salesperson total was given a name - so that we could refer to it in the SUM phrase of the city total.

The item that prints the city total has also been given a name. This is so that we can refer to it in the final total SUM phrase.

Printing a Final Total

To print the final total we must set up a CONTROL FOOTING FINAL group. Final is a special keyword that indicates the control break that occurs when the end of the report is reached. This is the most senior control break and we indicate this by placing the word FINAL ahead of all the other control break items in the CONTROLS ARE phrase.

The final total is automatically accumulated by means of the SUM CS clause. Every time a city total is printed the phrase SUM CS adds the city total to the final total. Note that the final total print item is not named because we never need to refer to it.

REPORT SECTION changes

```
REPORT SECTION.
RD  SalesReport
    CONTROLS ARE FINAL
                 CityCode
                 SalesPersonNum
    PAGE LIMIT IS 66
    HEADING 1
    FIRST DETAIL 6
    LAST DETAIL 42
    FOOTING 52.

01  TYPE IS PAGE HEADING.
    02 LINE 1.
       03 COLUMN 12     PIC X(32)
                        VALUE "An example COBOL Report Program".

    02 LINE 2.
       03 COLUMN 6      PIC X(17)
          VALUE "Bible Salesperson".
       03 COLUMN 23     PIC X(26)
          VALUE " - Sales and Salary Report".

    02 LINE 4.
       03 COLUMN 2      PIC X(4) VALUE "City".
       03 COLUMN 12     PIC X(11) VALUE "Salesperson".
       03 COLUMN 28     PIC X(4) VALUE "Sale".

    02 LINE 5.
       03 COLUMN 2      PIC X(4) VALUE "Name".
       03 COLUMN 13     PIC X(6) VALUE "Number".
       03 COLUMN 28     PIC X(5) VALUE "Value".


01  DetailLine TYPE IS DETAIL.
    02 LINE IS PLUS 1.
       03 COLUMN 1      PIC X(9)
                        SOURCE CityName(CityCode) GROUP INDICATE.
       03 COLUMN 15     PIC 9
                        SOURCE SalesPersonNum  GROUP INDICATE.
       03 COLUMN 25     PIC $$,$$$.99 SOURCE ValueOfSale.


01  SalesPersonGrp
    TYPE IS CONTROL FOOTING SalesPersonNum  NEXT GROUP PLUS 2.
    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(21) VALUE "Sales for salesperson".
       03 COLUMN 37     PIC 9 SOURCE SalesPersonNum.
       03 COLUMN 43     PIC X VALUE "=".
```

```
            03 SMS COLUMN 45 PIC $$$$$,$$$.99 SUM ValueOfSale.


01  CityGrp TYPE IS CONTROL FOOTING CityCode NEXT GROUP PLUS 2.
    02 LINE IS PLUS 2.
       03 COLUMN 15     PIC X(9) VALUE "Sales for".
       03 COLUMN 25     PIC X(9) SOURCE CityName(CityCode).
       03 COLUMN 43     PIC X VALUE "=".
       03 CS COLUMN 45  PIC $$$$$,$$$.99 SUM SMS.


01  TotalSalesGrp TYPE IS CONTROL FOOTING FINAL.
    02 LINE IS PLUS 4.
       03 COLUMN 15     PIC X(11)
                        VALUE "Total sales".
       03 COLUMN 43     PIC X VALUE "=".
       03 COLUMN 45     PIC $$$$$,$$$.99 SUM CS.


01  TYPE IS PAGE FOOTING.
    02 LINE IS 53.
       03 COLUMN 1      PIC X(29)
                        VALUE "Programmer - Michael Coughlan".
       03 COLUMN 45     PIC X(6) VALUE "Page :".
       03 COLUMN 52     PIC Z9 SOURCE PAGE-COUNTER.
```

Changes to the Procedure Division

What changes do we need to make to the Procedure Division code to print the city and final totals.

Why none at all!!!

The Procedure Division code remains exactly the same.

Report Writer Declaratives

Introduction

The Report Writer is a wonderful tool if the structure of the required report fits in to the way the Report Writer does things. But sometimes, the structure or requirements of the report are such that the standard Report Writer alone is an insufficient tool. In these cases, DECLARATIVES may be used to extend the functionality of the Report Writer.

The USE BEFORE REPORTING phrase allows code specified in the DECLARATIVES to be executed just before a report group is printed. The code in the DECLARATIVES extends the functionality of the Report Writer by performing tasks or calculations that the Report Writer can not do automatically or by selectively stopping the report group from being printed (using the SUPPRESS PRINTING command).

Calculating the salesperson's commission and salary

Suppose we want to extend our report so that as well as printing the salesperson total it also prints the salesperson's commission and salary. The commission will be a percentage of the saleperson's sales and the salary will be calculated as the commission plus a fixed amount obtained from a two dimension table.

Additionally to enhance our understanding of how the control break items are referenced we will show the number of the salesperson for which we have just calculated the totals, commission and salary and the salesperson number currently in the file buffer.

Calculating the commission requires more than just simple addition so the Report Writer cannot handle it automatically. To calculate the commission are DECLARATIVES are required. The USE BEFORE REPORTING Salespsn-Line phrase in the DECLARATIVES allows these items to be calculated when the Salespsn-Number control footer group is about to be printed.

In the example program, DECLARATIVES are used because the Report Writer cannot automatically calculate a salesperson's commission and salary. The USE BEFORE REPORTING SalesPersonGrp phrase in the DECLARATIVES allows these items to be calculated when the SalesPersonGrp control footer group is about to be printed.

Values of control break items

Have another look at the Report Section above. In particular look at the salesperson control footing group. Notice anything strange?

The salesperson number is being printed and the SOURCE clause is used to get its value from the SalesPersonNum in the sales record. But this is a control footing and that means that it is printed when there is a change of salesperson number. So the value in SalesPersonNum should be the number of the next salesperson and not the number of the salesperson whose totals have just been produced.

Yet the report produced has the correct salesperson number printed. How can this be?

In the control footing any reference to the control break item will be supplied with the previous value not the current value.

To help make the relationship between control break items and the Report Section, the Declaratives and the Procedure Division we will show the number of the salesperson for which we have just calculated the totals, commission and salary and the salesperson number currently in the file buffer. In the city footing we will show the city code for the city whose total sales have just been printed and the city code currently in the file buffer.

The full program and the report it produces

Changes from the simple version are shown in red.

```
      $ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  ReportExampleFull.
AUTHOR.  Michael Coughlan.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SalesFile ASSIGN TO "GBPAY.DAT"
           ORGANIZATION IS LINE SEQUENTIAL.
    SELECT PrintFile ASSIGN TO "SALESREPORT.LPT".


DATA DIVISION.
FILE SECTION.
FD  SalesFile.
01  SalesRecord.
    88 EndOfFile  VALUE HIGH-VALUES.
    02 CityCode         PIC 9.
    02 SalesPersonNum   PIC 9.
    02 ValueOfSale      PIC 9(4)V99.

FD  PrintFile
    REPORT IS SalesReport.
```

```
WORKING-STORAGE SECTION.
01  NameTable.
    02 TableValues.
       03 FILLER        PIC X(18) VALUE "Dublin   Belfast  ".
       03 FILLER        PIC X(18) VALUE "Cork     Galway   ".
       03 FILLER        PIC X(18) VALUE "Sligo    Waterford".
       03 FILLER        PIC X(9)  VALUE "Limerick".
    02 FILLER REDEFINES TableValues.
       03 CityName     PIC X(9) OCCURS 7 TIMES.

01  RateTable.
    02 TableValues.
       03 FILLER        PIC X(35)
                        VALUE "12300321004350056700123002340034500".
       03 FILLER        PIC X(35)
                        VALUE "12300543001230034200111001220013300".
       03 FILLER        PIC X(35)
                        VALUE "12000321001760018700133001440015500".
       03 FILLER        PIC X(35)
                        VALUE "32100123003210012000166001770018800".
       03 FILLER        PIC X(35)
                        VALUE "34500345004560054300111001220013200".
       03 FILLER        PIC X(35)
                        VALUE "19000180001780017900444003330022200".
       03 FILLER        PIC X(35)
                        VALUE "16700156001450014600222001110021200".
       03 FILLER        PIC X(35)
                        VALUE "12000132001230014300121003210043200".
       03 FILLER        PIC X(35)
                        VALUE "15400165001640017600111007770033300".

    02 FILLER REDEFINES TableValues.
       03 City OCCURS 7 TIMES.
          04 FixedRate  PIC 9(3)V99 OCCURS 9 TIMES.

01  MiscVariables.
    02 Commission       PIC 9(4)V99.
    02 Percentage       PIC V99 VALUE .05.
    02 Salary           PIC 9(6)V99.
    02 SalesPersonNow   PIC 9.
    02 CityNow          PIC 9.


REPORT SECTION.
RD  SalesReport
    CONTROLS ARE FINAL
                CityCode
                SalesPersonNum
    PAGE LIMIT IS 66
    HEADING 1
    FIRST DETAIL 6
    LAST DETAIL 42
    FOOTING 52.

01  TYPE IS PAGE HEADING.
    02 LINE 1.
       03 COLUMN 12     PIC X(32)
                        VALUE "An example COBOL Report Program".

    02 LINE 2.
       03 COLUMN 6      PIC X(17)
          VALUE "Bible Salesperson".
       03 COLUMN 23     PIC X(26)
          VALUE " - Sales and Salary Report".

    02 LINE 4.
       03 COLUMN 2      PIC X(4) VALUE "City".
       03 COLUMN 12     PIC X(11) VALUE "Salesperson".
       03 COLUMN 28     PIC X(4) VALUE "Sale".
    02 LINE 5.
       03 COLUMN 2      PIC X(4) VALUE "Name".
       03 COLUMN 13     PIC X(6) VALUE "Number".
       03 COLUMN 28     PIC X(5) VALUE "Value".


01  DetailLine TYPE IS DETAIL.
    02 LINE IS PLUS 1.
       03 COLUMN 1      PIC X(9)
                        SOURCE CityName(CityCode) GROUP INDICATE.
       03 COLUMN 15     PIC 9
                        SOURCE SalesPersonNum  GROUP INDICATE.
       03 COLUMN 25     PIC $$,$$$.99 SOURCE ValueOfSale.


01  SalesPersonGrp
    TYPE IS CONTROL FOOTING SalesPersonNum  NEXT GROUP PLUS 2.
    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(21) VALUE "Sales for salesperson".
       03 COLUMN 37     PIC 9 SOURCE SalesPersonNum.
       03 COLUMN 43     PIC X VALUE "=".
       03 SMS COLUMN 45 PIC $$$$$,$$$.99 SUM ValueOfSale.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(19) VALUE "Sales commission is".
       03 COLUMN 43     PIC X VALUE "=".
       03 COLUMN 45     PIC $$$$$,$$$.99 SOURCE Commission.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(22) VALUE "Salesperson salary is".
       03 COLUMN 43     PIC X VALUE "=".
       03 COLUMN 45     PIC $$$$$,$$$.99 SOURCE Salary.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(30)
                        VALUE "Current  salesperson number = ".
       03 COLUMN 45     PIC 9 SOURCE SalesPersonNow.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(30)
                        VALUE "Previous salesperson number = ".
       03 COLUMN 45     PIC 9 SOURCE SalesPersonNum.



01  CityGrp TYPE IS CONTROL FOOTING CityCode NEXT GROUP PLUS 2.
    02 LINE IS PLUS 2.
       03 COLUMN 15     PIC X(9) VALUE "Sales for".
       03 COLUMN 25     PIC X(9) SOURCE CityName(CityCode).
       03 COLUMN 43     PIC X VALUE "=".
       03 CS COLUMN 45  PIC $$$$$,$$$.99 SUM SMS.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(12)
                        VALUE "Current city".
       03 COLUMN 43     PIC X VALUE "=".
       03 COLUMN 45     PIC 9 SOURCE CityNow.

    02 LINE IS PLUS 1.
       03 COLUMN 15     PIC X(13)
                        VALUE "Previous city".
       03 COLUMN 43     PIC X VALUE "=".
       03 COLUMN 45     PIC 9   SOURCE CityCode.


01  TotalSalesGrp TYPE IS CONTROL FOOTING FINAL.
    02 LINE IS PLUS 4.
       03 COLUMN 15     PIC X(11)
                        VALUE "Total sales".
```

```
           03 COLUMN 43     PIC X VALUE "=".
           03 COLUMN 45     PIC $$$$$,$$$.99 SUM CS.


01  TYPE IS PAGE FOOTING.
    02 LINE IS 53.
       03 COLUMN 1      PIC X(29)
                        VALUE "Programmer - Michael Coughlan".
       03 COLUMN 45     PIC X(6) VALUE "Page :".
       03 COLUMN 52     PIC Z9 SOURCE PAGE-COUNTER.


PROCEDURE DIVISION.
DECLARATIVES.
Calc SECTION.
    USE BEFORE REPORTING SalesPersonGrp.
Calculate-Salary.
    MULTIPLY SMS BY Percentage
          GIVING Commission ROUNDED.
    ADD Commission, FixedRate(CityCode,SalesPersonNum)
          GIVING Salary.
END DECLARATIVES.

Main SECTION.
Begin.
    OPEN INPUT SalesFile.
    OPEN OUTPUT PrintFile.
    READ SalesFile
         AT END SET EndOfFile TO TRUE
    END-READ.
    INITIATE SalesReport.
    PERFORM PrintSalaryReport
            UNTIL EndOfFile.
    TERMINATE SalesReport.
    CLOSE SalesFile, PrintFile.
    STOP RUN.


PrintSalaryReport.
    MOVE CityCode TO CityNow.
    MOVE SalesPersonNum  TO SalesPersonNow.
    GENERATE DetailLine.
    READ SalesFile
          AT END SET EndOfFile TO TRUE
    END-READ.
```

```
           An example COBOL Report Program
     Bible Salesperson - Sales and Salary Report

 City      Salesperson     Sale
 Name       Number         Value
Dublin        1          $111.50
                         $222.50
              Sales for salesperson 1     =      $334.00
              Sales commission is         =       $16.70
              Salesperson salary is       =      $139.70
              Current  salesperson number = 2
              Previous salesperson number = 1


Dublin        2        $1,111.50
                       $1,222.50
                       $1,333.50
              Sales for salesperson 2     =    $3,667.50
              Sales commission is         =      $183.38
              Salesperson salary is       =      $504.38
              Current  salesperson number = 3
              Previous salesperson number = 2


Dublin        3           $777.70
              Sales for salesperson 3     =      $777.70
              Sales commission is         =       $38.89
              Salesperson salary is       =      $473.89
              Current  salesperson number = 1
              Previous salesperson number = 3

              Sales for Dublin            =    $4,779.20
              Current city                = 2
              Previous city               = 1


Belfast       1           $111.50
                          $222.50
              Sales for salesperson 1     =      $334.00
              Sales commission is         =       $16.70
              Salesperson salary is       =      $139.70
              Current  salesperson number = 2
              Previous salesperson number = 1









Programmer - Michael Coughlan               Page :  1
```

Summary of control break item reference

When a control break item is used in a control footing or in the DECLARATIVES before printing a control footing the value supplied by the Report Writer is the previous value. Referring to the same item in the Procedure Division yields the current value.

To top of page

Copyright Notice

These COBOL course materials are the copyright property of Michael Coughlan.

All rights reserved. No part of these course materials may be reproduced in any form or by any means - graphic, electronic, mechanical, photocopying, printing, recording, taping or stored in an information storage and retrieval system - without the written permission of the author.

(c) Michael Coughlan

Last updated : November 1998
e-mail : CSISwebeditor@ul.ie