

Finding a Needle in the Haystack: Efficient Filtering of Large Data Sets

Daniel R. Bretheim and Ryan K. Carr
William M. Mercer, Inc.

I. INTRODUCTION

Some data processing tasks require reading large external files (the "haystack") in order to find and extract records that satisfy very specific criteria (the "needle"). In some cases millions of records are read to find a few. This can be a costly undertaking, especially if it requires extensive sorting/matching.

This paper examines a technique to efficiently accomplish this task. Using a case study, we describe a health care application designed to screen insurance company claim files for patients with specific diagnoses. The objective is to select the records meeting the diagnosis criteria, plus all other records for each individual, even records occurring earlier in the file. We also contrast this technique with other more commonly used approaches, comparing resource requirements for CPU time.

II. PROCESSING REQUIREMENTS FOR A HEALTH CARE APPLICATION

We often face the task of extracting information for specific diagnostic conditions from multi-million record external (non-SAS) files. Once we have identified records having the diagnostic conditions of interest, we then have to find all other records for the individual, regardless of their diagnostic condition or location in the file. This task has many efficiency issues, such as minimizing passes through the external file and minimizing or even avoiding sorting.

The first twenty records in the external file might look like the sample displayed in Exhibit 1, with the columns containing the following information:

Column	Description
1	Person identifier code
2	Diagnosis code #1
3	Diagnosis code #2
4	Diagnosis code #3
5	Diagnosis code #4
6	Variable #1
7	Variable #2
8	Variable #3
9	Variable #4
10	Variable #5

Note: In this case study we are interested in finding people with diagnoses for brain and spinal cord cancer, where the diagnosis codes are in the range of 1900-1929 and 1943-1944. Zeros in the diagnosis code fields are simply filler, they have no clinical meaning.

EXHIBIT 1

Sample Data From External File

```
123AAA 1900 0000 0000 0000 11 22 33 44 55
123AAA 1900 0000 0000 0000 11 22 33 44 55
123AAA 9999 0000 0000 0000 11 22 33 44 55
234888 9999 0000 0000 0000 11 22 33 44 55
234888 1920 0000 0000 0000 11 22 33 44 55
234888 1929 0000 0000 0000 11 22 33 44 55
345CCC 1929 0000 0000 0000 11 22 33 44 55
456DDD 9999 0000 0000 0000 11 22 33 44 55
456DDD 9999 0000 0000 0000 11 22 33 44 55
567EEE 9999 0000 0000 0000 11 22 33 44 55
567EEE 1944 0000 0000 0000 11 22 33 44 55
567EEE 9999 0000 0000 0000 11 22 33 44 55
678FFF 9999 1900 0000 0000 11 22 33 44 55
789GGG 9999 9999 1900 0000 11 22 33 44 55
890HHH 9999 9999 9999 1900 11 22 33 44 55
901111 1900 1920 1929 1943 11 22 33 44 55
012JJJ 9999 9999 9999 9999 11 22 33 44 55
012JJJ 9999 9999 9999 9999 11 22 33 44 55
456DDD 1943 0000 0000 0000 11 22 33 44 55
456DDD 9999 0000 0000 0000 11 22 33 44 55
```

You can see from this example that the data contain several features.

- o Each person can have multiple records.
- o Each person can have multiple diagnoses per record.
- o The cancer diagnosis codes can occur in any of the four diagnosis code fields.
- o Records with cancer diagnoses can be preceeded or followed by other non-cancer records for the individual.
- o In addition to the diagnosis code fields, there are other fields of interest on the record that must be extracted for subsequent analysis.
- o Data for a single person can appear anywhere in the file.

There are many ways to develop code that can deal with these processing issues. However, given the significant amount of data volume, we focused on techniques that would increase processing efficiency.

III. CODE DEVELOPMENT

In the examples that follow, we will step through the code that we developed, with each step adding to the processing capabilities of the former.

Example 1 : Filtering the Values of a Single Variable

We consistently used the format technique for table lookup.

This approach is easy to code and maintain. It also provides the framework for more complex table lookup tasks, particularly where the table information is prone to change.

The FORMAT procedure that follows creates a format which defines the diagnosis code ranges for brain and spinal code cancer.

```
PROC FORMAT;
  VALUE $DX '1900' - '1929' = 'KEEP'
            '1943' - '1944' = 'KEEP'
            OTHER = 'DROP';
```

The DATA step below reads the four diagnosis code fields from the external file (see Exhibit 1) and outputs only the records containing a diagnosis code in the first diagnosis field (DIAG1) in the range specified by the format \$DX.

```
DATA FILTER;
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @8 DIAG1 $4.
        @13 DIAG2 $4.
        @18 DIAG3 $4.
        @23 DIAG4 $4.;
  IF PUT(DIAG1,$DX.) = 'KEEP';
```

While this DATA step certainly works, there is a simple modification that will greatly increase processing efficiency. By using the trailing @-sign to evaluate the first diagnosis code field (DIAG1) for the codes of interest, we will read the remaining fields only after determining that we want to output the record.

```
DATA FILTER;
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @8 DIAG1 $4. @;
  IF PUT(DIAG1,$DX.) = 'KEEP' THEN DO;
    INPUT @13 DIAG2 $4.
          @18 DIAG3 $4.
          @23 DIAG4 $4.;
    OUTPUT;
  END;
```

The resulting data set would contain the observations displayed in Exhibit 2.

EXHIBIT 2
Records With Cancer Diagnosis in DIAG1

OBS	DIAG1	DIAG2	DIAG3	DIAG4
1	1900	0000	0000	0000
2	1900	0000	0000	0000
3	1920	0000	0000	0000
4	1929	0000	0000	0000
5	1929	0000	0000	0000
6	1944	0000	0000	0000
7	1900	1920	1929	1943
8	1943	0000	0000	0000

Example 2 : Filtering the Values of Multiple Variables

The next DATA step adds more functionality by allowing us to examine all four diagnosis code fields for the cancer codes.

```
DATA FILTER;
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @8 DIAG1 $4.
        @13 DIAG2 $4.
        @18 DIAG3 $4.
```

```
        @23 DIAG4 $4. @;
  ARRAY D(4) $ DIAG1-DIAG4;
  DO I = 1 TO 4;
    IF PUT(D(I),$DX.) = 'KEEP' THEN DO;
      OUTPUT;
    END;
  END;
```

Unfortunately, this approach will output the record each time a cancer diagnosis code appears in any of the four diagnosis variables. The DATA step below addresses this problem and only outputs a cancer record once, regardless of the number of cancer diagnosis codes present.

```
DATA FILTER;
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @8 DIAG1 $4.
        @13 DIAG2 $4.
        @18 DIAG3 $4.
        @23 DIAG4 $4. @;
  ARRAY D(4) $ DIAG1-DIAG4;
  KEEP_IT = ' ';
  DO I = 1 TO 4 UNTIL(KEEP_IT = 'YES');
    IF PUT(D(I),$DX.) = 'KEEP' THEN KEEP_IT = 'YES';
  END;
  IF KEEP_IT = 'YES' THEN OUTPUT;
  DROP KEEP_IT I;
END;
```

The resulting data set would contain the observations displayed in Exhibit 3.

EXHIBIT 3
Records With Cancer Diagnosis in DIAG1-DIAG4

OBS	DIAG1	DIAG2	DIAG3	DIAG4
1	1900	0000	0000	0000
2	1900	0000	0000	0000
3	1920	0000	0000	0000
4	1929	0000	0000	0000
5	1929	0000	0000	0000
6	1944	0000	0000	0000
7	9999	1800	0000	0000
8	9999	9999	1900	0000
9	9999	9999	9999	1900
10	1900	1920	1929	1943
11	1943	0000	0000	0000

Example 3 : Creating a Lookup File For Use in Filtering the Primary File For All Observations Containing Any Value From the Lookup File

Having built a foundation for efficient filtering, we now move to the actual task we faced in the case study. In addition to identifying people with cancer diagnoses, we also needed to extract all their non-cancer records as well. These other records could occur any place in the file. Keeping our objectives in mind for minimizing sorting and passes through the external file, we built upon the format approach for table lookups as follows.

We started by creating a control data set that would be fed into the FORMAT procedure. The PROC FORMAT would produce a user-defined format where the format range and label would consist of individual patient identifier codes for the patients with cancer diagnoses. Essentially, this gave us the equivalent of a lookup file that could be used to filter the external file (primary file) for all observations containing any value from the lookup file. This approach would allow us to avoid the massive sorts required for a match-merge, and therefore save significant CPU

processing time.

The first DATA step builds the control data set that conforms to the control data set structure required by PROC FORMAT. For a complete discussion of control data sets see the article by Thomas Hahl in *Observations*, First Quarter 1992.

```
DATA IDS (KEEP=PERSONID FMTNAME START LABEL TYPE
HLO);
  INFILE 'C:\MWSUG\RAWIN';
  RETAIN FMTNAME 'IDS';
  LENGTH START LABEL $ 6;
  IF _N_ = 1 THEN DO;
    START = 'OTHER';
    LABEL = 'X';
    TYPE = 'C';
    HLO = '0';
  OUTPUT;
  END;
INPUT @1 PERSONID $6.
      @8 DIAG1   $4.
      @13 DIAG2  $4.
      @18 DIAG3  $4.
      @23 DIAG4  $4. @;
ARRAY D(4) $ DIAG1-DIAG4;
KEEP IT = ' ';
DO I = 1 TO 4 UNTIL(KEEP IT = 'YES');
  IF PUT(D(I),$DX.) = 'KEEP' THEN DO;
    KEEP IT = 'YES';
    START = PERSONID;
    LABEL = PERSONID;
    TYPE = 'C';
    HLO = ' ';
  END;
END;
IF KEEP IT = 'YES' THEN OUTPUT;
```

This DATA step searches each record in the external file for cancer diagnoses and creates a control data set that contains the person identifier code (PERSONID) for each cancer patient. The output is displayed in Exhibit 4.

EXHIBIT 4
Control Data Set

OBS	FMTNAME	START	LABEL	TYPE	HLO	PERSONID
1	IDS	OTHER	X	C	0	
2	IDS	123AAA	123AAA	C		123AAA
3	IDS	123AAA	123AAA	C		123AAA
4	IDS	234BBB	234BBB	C		234BBB
5	IDS	234BBB	234BBB	C		234BBB
6	IDS	345CCC	345CCC	C		345CCC
7	IDS	567EEE	567EEE	C		567EEE
8	IDS	678FFF	678FFF	C		678FFF
9	IDS	789GGG	789GGG	C		789GGG
10	IDS	890HHH	890HHH	C		890HHH
11	IDS	901111	901111	C		901111
12	IDS	456DDD	456DDD	C		456DDD

After sorting the dataset to eliminate duplicate values for PERSONID, the data set displayed in Exhibit 5 is read by PROC FORMAT to create the \$IDS format which is displayed in Exhibit 6.

```
PROC SORT DATA=IDS NODUPKEY OUT=SORTED;
  BY CLNID;
```

EXHIBIT 5
Input to FORMAT Procedure

OBS	FMTNAME	START	LABEL	TYPE	HLO
1	IDS	OTHER	X	C	0
2	IDS	123AAA	123AAA	C	
3	IDS	234BBB	234BBB	C	
4	IDS	345CCC	345CCC	C	
5	IDS	456DDD	456DDD	C	
6	IDS	567EEE	567EEE	C	
7	IDS	678FFF	678FFF	C	
8	IDS	789GGG	789GGG	C	
9	IDS	890HHH	890HHH	C	
10	IDS	901111	901111	C	

```
PROC FORMAT
  CNTLIN=SORTED (KEEP=FMTNAME START LABEL TYPE
  HLO);
```

A final DATA step is used to reread the external file in order to select all the records for each cancer patient by using the \$IDS format to filter for the person identifier codes (PERSONID) for the cancer patients. The additional variables for the analysis are also read at this time.

```
DATA FILTER;
  INFILE 'C:\MWSUG\RAWIN';

  INPUT @1 PERSONID $6. @;

  IF PUT(PERSONID,$IDS.) NE 'X' THEN DO;
    INPUT @ 8 DIAG1   $4.
          @13 DIAG2  $4.
          @18 DIAG3  $4.
          @23 DIAG4  $4.
          @28 VAR5    2.
          @31 VAR6    2.
          @34 VAR7    2.
          @37 VAR8    2.
          @40 VAR9    2.;
  OUTPUT;
  END;
```

The resulting dataset is displayed in Exhibit 7.

Two alternatives to this filtering approach were developed and tested. Both alternatives used the more traditional match-merge requiring two sorts. The first alternative read the entire record from the external file during the initial pass while the second alternative used an additional DATA step for that purpose. The code for both alternatives is contained in the Appendix.

IV. RESULTS

Using a test dataset of 100,000 observations, Example 3 and its alternatives were evaluated on a 66 MHz 486 machine with 16 MB of RAM. CPU utilization comparisons are displayed in Exhibit 8.

EXHIBIT 6 **Output From FORMAT Procedure**

FORMAT NAME: \$IDS			LENGTH: 6	NUMBER OF VALUES: 10
MIN LENGTH: 1	MAX LENGTH: 6	DEFAULT LENGTH: 6	FUZZ: STD	
START	END	LABEL (VER. 6.03 22APR94:16:12:29)		
123AAA	123AAA	123AAA		
234BBB	234BBB	234BBB		
345CCC	345CCC	345CCC		
456DDD	456DDD	456DDD		
567EEE	567EEE	567EEE		
678FFF	678FFF	678FFF		
789GGG	789GGG	789GGG		
890HHH	890HHH	890HHH		
901III	901III	901III		
OTHER	OTHER	x		

EXHIBIT 7 **Complete Extract**

OBS	CLMID	DIAG1	DIAG2	DIAG3	DIAG4	VAR5	VAR6	VAR7	VAR8	VAR9
1	123AAA	1900	0000	0000	0000	11	22	33	44	55
2	123AAA	1900	0000	0000	0000	11	22	33	44	55
3	123AAA	9999	0000	0000	0000	11	22	33	44	55
4	234BBB	9999	0000	0000	0000	11	22	33	44	55
5	234BBB	1920	0000	0000	0000	11	22	33	44	55
6	234BBB	1929	0000	0000	0000	11	22	33	44	55
7	345CCC	1929	0000	0000	0000	11	22	33	44	55
8	456DDD	9999	0000	0000	0000	11	22	33	44	55
9	456DDD	9999	0000	0000	0000	11	22	33	44	55
10	456DDD	1943	0000	0000	0000	11	22	33	44	55
11	456DDD	9999	0000	0000	0000	11	22	33	44	55
12	567EEE	9999	0000	0000	0000	11	22	33	44	55
13	567EEE	1944	0000	0000	0000	11	22	33	44	55
14	567EEE	9999	0000	0000	0000	11	22	33	44	55
15	678FFF	9999	1900	0000	0000	11	22	33	44	55
16	789GGG	9999	9999	1900	0000	11	22	33	44	55
17	890HHH	9999	9999	9999	1900	11	22	33	44	55
18	901III	1900	1920	1929	1943	11	22	33	44	55

EXHIBIT 8
CPU Utilization in Seconds

<u>Activity</u>	<u>Example 3</u>	<u>Alternative 1</u>	<u>Alternative 2</u>
FORMAT procedure	1	n/a	n/a
DATA step	105	194	103
SORT procedure	1	1	1
SORT procedure	n/a	120	87
FORMAT procedure	1	n/a	n/a
DATA step	39	37	141
DATA step	n/a	n/a	36
Total	147	352	368

V. SUMMARY

Filtering techniques that can avoid the overhead of the match-merge will achieve great savings in CPU utilization.

REFERENCES

Hahl, Thomas J., "Creating Formats with Control Data Sets Using the FORMAT Procedure", Observations", Vol.1 No.2, pp. 30-34, SAS Institute, Inc.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

APPENDIX

```

* ALTERNATIVE 1 TO EXAMPLE #3 ;
DATA CLAIMS IDS (KEEP=PERSONID);
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @1 PERSONID $6.
         @8 DIAG1    $4.
         @13 DIAG2   $4.
         @18 DIAG3   $4.
         @23 DIAG4   $4. @;
  ARRAY D(4) $ DIAG1-DIAG4;
  KEEP_IT = ' ';
  DO I = 1 TO 4 UNTIL(KEEP_IT = 'YES');
    IF PUT(D(I), $OX.) = 'KEEP' THEN DO;
      KEEP_IT = 'YES';
    END;
  END;
  IF KEEP_IT = 'YES' THEN OUTPUT IDS;
  INPUT @28 VAR5 2.
         @31 VAR6 2.
         @34 VAR7 2.
         @37 VAR8 2.
         @40 VAR9 2.;
  OUTPUT CLAIMS;

PROC SORT DATA=IDS NODUPKEEP OUT=SORTIDS;
  BY PERSONID;

PROC SORT DATA=CLAIMS;
  BY PERSONID;

DATA FILTER;
  MERGE CLAIMS SORTIDS (IN=KEEP);
  BY PERSONID;
  IF KEEP;

```

```

* ALTERNATIVE 2 TO EXAMPLE #3 ;
DATA IDS (KEEP=PERSONID);
  INFILE 'C:\MWSUG\RAWIN';
  INPUT @1 PERSONID $6.
         @8 DIAG1    $4.
         @13 DIAG2   $4.
         @18 DIAG3   $4.
         @23 DIAG4   $4. @;
  ARRAY D(4) $ DIAG1-DIAG4;
  KEEP_IT = ' ';
  DO I = 1 TO 4 UNTIL(KEEP_IT = 'YES');
    IF PUT(D(I), $OX.) = 'KEEP' THEN DO;
      KEEP_IT = 'YES';
    END;
  END;
  IF KEEP_IT = 'YES' THEN OUTPUT IDS;

PROC SORT DATA=IDS NODUPKEEP OUT=SORTIDS;
  BY PERSONID;

DATA CLAIMS;
  INFILE 'C:\MWSUG\RAWIN';

  INPUT @1 PERSONID $6.
         @8 DIAG1    $4.
         @13 DIAG2   $4.
         @18 DIAG3   $4.
         @23 DIAG4   $4.
         @28 VAR5    2.
         @31 VAR6    2.
         @34 VAR7    2.
         @37 VAR8    2.
         @40 VAR9    2.;

PROC SORT DATA=CLAIMS;
  BY PERSONID;

DATA FILTER;
  MERGE CLAIMS SORTIDS (IN=KEEP);
  BY PERSONID;
  IF KEEP;

```