

Indexed Table Look-up Against Multiple Data Sets

Doug McAllaster, LTC, HQ US Army, Pentagon

ABSTRACT

This paper shows how to use a single SAS® DATA step to perform an indexed table look-up against multiple data sets. The technique uses the KEY= data set option to directly read indexed data sets. The code shows how to use the automatic SAS variable _IORC_ to check whether the direct read found a matching observation. The code also shows how to avoid an important potential pitfall when the direct read does not find a matching observation.

GOAL

Our goal is to look-up information on subjects from data contained in legacy (historical) SAS data sets. We want to look-up variable values from the most recent legacy file where we can find the matching subject.

STRATEGY

Our strategy is to directly read indexed data sets using the KEY= data set option. We read the subject's key variable into the DATA step. We then attempt to find a matching observation in each of the legacy data sets in turn. If we find a match, we output the observation, which has the correct look-up variable values and go on to the next look-up subject. If we do not find a match in any of the legacy files, then we must set the look-up variables to missing. We facilitate this re-initialization (assigning variable values to missing for non-matches) by setting a data set, which contains missing values for all the look-up variables.

PROBLEM

We explain the technique as a solution to the following typical table look-up problem. Consider that we want to find the duration of time between customer transactions. We have a file of customer transactions from 1996. We identify each customer with a variable (KEY) that has unique values. This file also has another variable (DATE96) which is the transaction date from 1996 for that customer. Our problem is to find the most recent transaction date (DATE) and file name (FILE) from our historical legacy files.

DATA

The appendix shows the input data sets: "look-up problem," "conceptual layout of six legacy files: FILE95-FILE90," and "empty data set." It also shows the output data set: "look-up answer." For example, customer KEY=2 has a transaction on 960720 in 1996. We want to look-up his last transaction from previous years. For this customer, his last transaction is in FILE92 and occurred on 921209.

CODE

The appendix also has the SAS DATA step code to solve the problem. The SET sample statement reads the value of key we want to find. Each SET data9# KEY=key statement attempts to directly read (via the KEY= data set option) the matching observation from each data set, in turn. If the SET data9# KEY=key statement succeeds (finds a matching observation), then SAS assigns the automatic input-output return code variable to zero (_IORC_=0). We test for this condition and the RETURN statement

outputs the observation with the correct look-up values and returns to the top of the DATA step loop.

On the other hand, if all of the direct reads fail, then we have found no matching observation in any of the legacy data sets. Repeating, if all of the SET data9# KEY=key statements fail to find a matching observation, then `_IORC_ > 0` because the data step has found no match. This is where we must avoid a potential pitfall, as explained next.

Because the SET statement automatically retains variable values, we need to re-assign all variables to missing for those observations for which we find no matching observation from any of the legacy data sets. Otherwise, the non-matching observation would retain the look-up values of the previously found (output) observation, which would be incorrect values.

We could simply assign each variable to missing with separate assignment statements. Instead, we conveniently re-assign all the look-up variable values to missing with a single SET empty POINT= statement, as follows.

We previously created a data set (empty) which contains missing values for the look-up variables, as shown in the appendix. We initialize pointer to 1 via the RETAIN pointer 1 statement and we never change its value. Then, the SET empty POINT=pointer statement directly reads the one single observation from empty. Because empty has missing values for all the look-up variables, we now have correctly re-assigned the previously retained values of the look-up variables to missing via a single executable (SET) statement. Now, after all direct reads fail, the look-up variables correctly have missing values.

Finally, we note that this DATA step terminates when the SET sample statement encounters the

end of file (EOF) condition. Note that direct reads (SET with POINT= or KEY= data set options) do NOT set the EOF condition.

CONCLUSION

The DATA step provides a powerful yet simple and efficient method for looking up information from legacy data sets using the KEY= option. The SAS programmer needs to understand how to use the automatic SAS variable, `_IORC_`, to check the direct read and what to do when these direct reads fail.

ACKNOWLEDGEMENTS

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA

AUTHOR

Interested readers may contact me at MCALLDL@HQDA.ARMY.MIL to obtain code which generates sample data and which provides alternate but less efficient solutions to the problem.

conceptual layout of six legacy files: FILE95 - FILE90

KEY	FILE95	FILE94	FILE93	FILE92	FILE91	FILE90
1		940318				900511
2				921209		
3		940316			910826	
4	951118	941003	930209	920317		901028
5			930712	920306	910616	901020
6	950206			920310		900220
7	950313	940805		920202	910716	901010
8	951126	940922	930710	920419		901018
9	950419		930310	920422	910110	900821
10	950210	941206				
11		940221		920907		901125
12	950211				910127	901110
13	950215	940902	930610			
14			930414		910315	
15					910212	
16					910823	
18		940527		921117	910901	
19					910720	
20		940505	930401			
21	950919	940311			910216	
22					910224	900225
23	950723	940203	930318	921227	910911	900411
24	950714		931122		910104	901014
25						900509
26	950822		930913	920717		900122
27		940520	931125	920216		
28		940608	930522	920822		
29	950913	941122		921225		900205
30	950317	940618	930908			900418
31	951017		931011	921224	910723	900820
32	951110		930521			
33	951107				910620	
34	950102					
35		940619	931104		910413	900302
37		940316	930520		911006	
38	951208	940102			910513	900626
39	950119			921121		
40	950905	940117	931116	920317		

look-up problem

KEY	DATE96
2	960720
5	960619
16	960217
23	960507
32	960527
38	960821
41	961112
42	960619
43	960322

look-up answer

KEY	DATE96	FILE	DATE
2	960720	FILE92	921209
5	960619	FILE93	930712
16	960217	FILE91	910823
23	960507	FILE95	950723
32	960527	FILE95	951110
38	960821	FILE95	951208
41	961112		.
42	960619		.
43	960322		.

```
data empty;
  if 0 then set leg_file (drop=key);
  output;
  stop; run;
```

empty data set -- has 1 obs with missing values for look-up vars

OBS	FILE	DATE
1		.

```
data lookup;
  set data96 (keep=key date rename=(date=date96));
  set data95 key=key; if _iorc_=0 then return;
  set data94 key=key; if _iorc_=0 then return;
  set data93 key=key; if _iorc_=0 then return;
  set data92 key=key; if _iorc_=0 then return;
  set data91 key=key; if _iorc_=0 then return;
  set data90 key=key; if _iorc_=0 then return;
  retain pointer 1;
  set empty point=pointer; run;
```