Using Indexes for Direct Access to SAS Datasets

Judy Loren, ASG, Inc.

Introduction

SAS programmers have been able to take advantage of direct access to VSAM (KSDS) files for years. The ability to read one or more records based on the value of one or more variables (fields) has been very useful. Starting with release 6.07, you can use SAS datasets in a similar fashion. In this paper, I will show how to use this direct access capability and illustrate its utility in producing a complex report.

Disclaimers

Efficiency: In this paper I am interested primarily in showing the utility of direct access for program clarity. For more information about CPU performance issues, see Philip Weiss’s paper in the SUGI 18 Proceedings.

Operating System: This paper is based on SAS under MVS. Most of the information should apply to other operating systems.

Grammar: The correct plural of the word ‘index’, according to Webster’s, is either ‘indexes’ or ‘indices’. I use ‘indexes’ as more common.

What is Direct Access?

Direct access or keyed access (I will use the terms interchangeably in this paper) means choosing one observation from a SAS dataset based on the values of one or more variables. (This is not to be confused with access based on observation number.) For example, if you had a dataset containing the populations of the 50 states, one record per state, and you were processing a second file that contained the state variable, you could use direct access to find the population of the current state.

How?

To use keyed access to a SAS dataset you first have to create an index on the variable(s) you want to look up further information about. In the above example, you would first create an index on state. Then in another DATA step you could use a SET statement with KEY= at the point in the DATA step where you had a value in the indexed variable (STATE) and you wanted to look up the population.

This is a simple example. To illustrate the power that is available and some of the details of using KEY=, I want to define a more complex situation.

The Report

The Management Survey Report (not its real name) was based on a survey of the employees of a large corporation. The survey was intended to measure the employees’ feelings about their management. It asked employees to rate 50 statements about company management on a scale of 1 to 7 where 1 represented “Strongly Disagree” and 7 represented “Strongly Agree”. The survey looked like the example in Figure 1.

Management Survey 1994

Please circle the answer which corresponds most closely to your opinion.

<table>
<thead>
<tr>
<th></th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Somewhat Agree</th>
<th>Neither Agree nor Disagree</th>
<th>Somewhat Disagree</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
<th>Don’t Know</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. My work unit makes time to identify better ways to get the job done.</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2. In my work unit, people get rewarded for preventing problems.</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3. We communicate effectively in my work unit.</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4. In my work unit, employees treat each other with respect.</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 1.

288
Every employee in the company completed a survey that was coded to identify the supervisor of the department. SAS was used to aggregate the results for each supervisor, producing mean values for each statement on the survey across all that supervisor's employees.

The report got interesting with the addition of the following requirements:

1) On each supervisor's report, show the supervisor's score on each question for this year and last year. On the same line show the average score on that question for the supervisor's department (also called functional area) and for the corporation as a whole.

2) On the reports, group the questions into prescribed dimensions. Show on the first page the average score for each dimension, then on subsequent pages list the individual questions and the corresponding scores.

3) At the end of the report, produce a summary for each supervisor showing the three questions with the greatest percentage improvement since the previous year. In listing the questions, show all the same information shown previously in the report: this year's score, last year's score, the department average and the company score.

Figure 2 shows how each part of the report might look.

### Management Survey 1994 Results
For Supervisor: name, department

<table>
<thead>
<tr>
<th>Dimension Summary</th>
<th>Last Year</th>
<th>This Year</th>
<th>Change</th>
<th>Dept. Average</th>
<th>Company Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Master Change</td>
<td>5.6</td>
<td>5.8</td>
<td>0.2</td>
<td>5.2</td>
<td>5.0</td>
</tr>
<tr>
<td>2. Ethical and Compassionate</td>
<td>6.1</td>
<td>6.0</td>
<td>-0.1</td>
<td>5.8</td>
<td>4.5</td>
</tr>
<tr>
<td>3. Share and Listen</td>
<td>5.0</td>
<td>5.5</td>
<td>0.5</td>
<td>4.9</td>
<td>4.8</td>
</tr>
<tr>
<td>4. Efficient and Effective</td>
<td>4.2</td>
<td>4.0</td>
<td>-0.2</td>
<td>5.1</td>
<td>5.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Survey Statements</th>
<th>Last Year</th>
<th>This Year</th>
<th>Change</th>
<th>Dept. Average</th>
<th>Company Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension 1: Master Change</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. My work unit functions effectively in a constantly changing work environment.</td>
<td>5.2</td>
<td>5.4</td>
<td>0.2</td>
<td>5.0</td>
<td>4.9</td>
</tr>
<tr>
<td>15. In my work unit, taking initiative is valued and rewarded.</td>
<td>4.3</td>
<td>4.1</td>
<td>-0.2</td>
<td>4.0</td>
<td>4.2</td>
</tr>
<tr>
<td>22. In my work unit, we are sensitive to the impact and demands created by change.</td>
<td>4.9</td>
<td>4.9</td>
<td>0.0</td>
<td>5.0</td>
<td>4.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Most Improved</th>
<th>Last Year</th>
<th>This Year</th>
<th>Change</th>
<th>Dept. Average</th>
<th>Company Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>27. I feel valued for the work I do.</td>
<td>4.2</td>
<td>5.6</td>
<td>1.4</td>
<td>5.3</td>
<td>5.2</td>
</tr>
<tr>
<td>46. In my department, people listen well to one another.</td>
<td>4.9</td>
<td>6.0</td>
<td>1.1</td>
<td>6.0</td>
<td>5.0</td>
</tr>
<tr>
<td>33. My contributions are appropriately rewarded.</td>
<td>4.3</td>
<td>4.9</td>
<td>0.6</td>
<td>4.7</td>
<td>4.6</td>
</tr>
</tbody>
</table>

Figure 2.
The Setup

Calculating the question scores for each supervisor is a trivial exercise with PROC MEANS or PROC SUMMARY. Calculating the 10 dimension scores for each supervisor is equally trivial, given that you have a file matching question number to dimension number. Similarly, you can calculate company and department averages for each question. We will also assume away the problems in matching last year's data with this year's data, which is rarely trivial but not relevant to the topic at hand.

It is only slightly more complicated to get a dataset containing the 3 most improved questions for each supervisor. But here's where you want to make one adjustment allowing you to use KEY=. We will create a variable containing the rank of each question within the 3. For example, I used the numeric variable P with values of I, 2, or 3. Then when we want to retrieve the 3 questions in order, we can put the SET with the KEY = in a DO P=1 to 3 loop. The KEY will be a composite of the supervisor's code and P.

Creating Indexes

Indexes can be either simple (one variable) or composite (two or more variables). You can either create the index at the same time you create the dataset or use PROC DATASETS. As an example of a simple index, suppose we create a dataset with the corporate scores for the 50 questions, with one observation per question. To use the INDEX= dataset option on a DATA statement:

DATA CRPAVG94(INDEX=(QUESNO));

To add an index to an existing dataset:

PROC DATASETS LIBRARY=ddname; MODIFY CRPAVG94; INDEX CREATE QUESNO;

As an example of a composite index, suppose we create a dataset with all the supervisors' scores on each question, this year and last year, with 50 observations for each supervisor corresponding to the 50 questions. No index defined; will be used sequentially.

To add the composite index to an existing dataset:

PROC DATASETS LIBRARY=ddname; MODIFY SDIMS; INDEX CREATE CDIM=(CODE DIM); Whereas the name of a simple index is the same as the variable, the composite index is given a distinct name to be used after KEY=. When you refer to CDIM, the values of the variables CODE and DIM will be used to lookup a record in SDIMS.

The options UNIQUE (to specify that values of the key variables must be unique) and NOMISS (to leave missing values of the key variables out of the index) are available when creating indexes. See Technical Report P-242 for more information.

A Note about Creating Indexes

Unlike datasets, which you can create and overwrite, you cannot overwrite an index. If you try to create an index with the same name as an existing index, you will get an error that will cause SAS to set OBS=0. You can, if necessary, DROP an index via PROC DATASETS and re-create it.

The Datasets

Suppose now we have created the datasets we need for the report as follows:

1) CRPAVG94 has the company scores for each question, with one observation per question. Indexed on QUESNO.

2) CRPDIM94 has the company scores for each dimension, with one observation per dimension. Indexed on DIM.

3) TWOYRS has all the supervisors' scores on each question, this year and last year, with 50 observations for each supervisor corresponding to the 50 questions. No index defined; will be used sequentially.

4) SDIMS has all the supervisors' scores on each dimension, with 10 observations per supervisor corresponding to the 10 dimensions. Composite index CDIM defined on CODE and DIM.

5) PCT3 has the 3 most improved questions for each supervisor, one question per obs. Composite index CODEP defined on CODE and P where P is 1, 2, or 3.

The main body of the report, the 50 question results for each supervisor, will serve as the basic SAS dataset going into the DATA _NULL_ step. We will use KEY= in a number of different places. Figure 3 contains the skeleton code for the report. The numbered sections following the code explain the points noted in Figure 3.
DATA _NULL_;  
LENGTH DIMTITLE $ 30 QUESTEXT $ 142;  
/* See Note #6 in text */  
SET IN.TWOYS;  
BY CODE DIM QUESNO;  
FILE PRINT NOTITLES LINESIZE=80;  
/* See Note #1 in text */  
IF FIRST.CODE THEN LINK DIMPAGE;  
IF FIRST.DIM THEN DO;  
    LINK DIMLKUP;  
    /* See Note #3 in text */  
    LINK PAGEHEAD;  
    aID;  
    LINK DIMPAGE;  
    LINK DIMPRNT;  
    LINK SETSDIM;  
    LINK SETDDIM;  
    LINK SETCDIM;  
    /* print the dimension score for the supervisor, dept. and company; */  
    END;  
    DIM99;  
    /* See Note #4 in text */  
    LINK SETSDIM;  
    LINK SETDDIM;  
    LINK SETCDIM;  
    /* print overall totals; */  
    RETURN;  
SETDDIM:  
    CDDIM94=. ;  
    /* See Note #5 in text */  
    SET IN.SDIMs(KEEP=CDDIM94)  
        KEY=CDIM / UNIQUE;  
    /* See Note #7 in text */  
    SMD_IORC=_IORC_;  
    /* See Note #8 in text */  
    RETURN;  
SETCDIM:  
    CDIM94=. ;  
    SET IN.CRPDIM94(KEEP=CDIM94)  
        KEY=DIM / UNIQUE;  
    CDM_IORC=_IORC_;  
    RETURN;  
DIMPAGE:  
    LINK DIMLKUP;  
    LINK DIMPAGE;  
    LINK DIMPRNT;  
    LINK DEPTLKUP;  
    LINK CRPLKUP;  
    LINK QUESPRNT;  
    IF LAST.CODE THEN LINK SUMREPT;  
    RETURN;  
DIMLKUP:  
    DIMTITLE=' ';  
    SET IN.DIMs(KEEP=DIMTITLE)  
        KEY=DIM / UNIQUE;  
        /* See Note #2 in text */  
    DIM_IORC=_IORC_;  
    RETURN;  
DACPAGE:  
    DO DIM= 1 TO 10;  
    LINK DIMLKUP;  
    LINK DIMPAGE;  
    LINK SETSDIM;  
    LINK SETDDIM;  
    LINK SETCDIM;  
    /* print the dimension score for the supervisor, dept. and company; */  
    END;  
    DIM=99;  
    /* See Note #4 in text */  
    LINK SETSDIM;  
    LINK SETDDIM;  
    LINK SETCDIM;  
    /* print overall totals; */  
    RETURN;  
SUMREPT:  
    LINK SUMHEAD;  
    PUT @5 'Most Improved:' ;  
    IF AVG93 GT 0 THEN DO;  
        DO P=1 TO 3;  
            AVG93=. ;  
            AVG94=. ;  
            CRPAVG94=. ;  
            QUESNO=. ;  
            SET IN.PCT3(KEEP=AVG93 AVG94  
                QUESNO) KEY=CODEP / UNIQUE;  
            PCT_IORC=_IORC_;  
            LINK DQLKUP;  
            LINK DEPTLKUP;  
            LINK CRPLKUP;  
            LINK QUESPRNT;  
            aID;  
            aID;  
            RETURN;  
        SUMHEAD:  
            *code to print heading for each page;  
            RETURN;  
DEPTLKUP:  
    DAVG94=. ;  
    SET IN.DAVG94(KEEP=DAVG94)  
        KEY=DQUES / UNIQUE;  
    DEPT_IORC=_IORC_;  
    RETURN;  
CRPLKUP:  
    CRPAVG94=. ;  
    SET IN.CRPAVG94(KEEP=CRPAVG94)  
        KEY=QUESNO / UNIQUE;  
    CRP_IORC=_IORC_;  
    RETURN;  
DQLKUP:  
    QUESTEXT=' ';  
    SET IN.DIMQUES(KEEP=QUESTEXT)  
        KEY=QUESNO / UNIQUE;  
    DQ_IORC=_IORC_;  
    RETURN;  
Figure 3.
The Report Code

The numbered points below refer to notes in the code in Figure 3.

#1. We will use TWOYRS as the basis for the report. It is sorted by CODE DIM QUESNO. Every time we get to a new supervisor we want to print the page with the dimension scores first, then go on to the individual questions.

#2. The UNIQUE option on KEY= access (added in 6.08, not to be confused with the UNIQUE option on index creation) tells SAS to start every access at the top of the index. Without the UNIQUE option, SAS does skip sequential processing. This is important to know for two reasons. First, even if there are duplicate values of your key variables, you will get only one observation from any given lookup. It is rarely safe to leave the decision of which one to chance. Second, without the UNIQUE option, if you ever look for the same value of the key twice, it will try to find a second observation with the same key and it will not find the first. This will not cause a fatal error, but will cause problems for you (see When the Lookup Fails, below.)

If you know your lookup table should have only one observation for each key value, you will get only one observation from any given lookup. It is rarely safe to leave the decision of which one to chance. Second, without the UNIQUE option, if you ever look for the same value of the key twice, it will try to find a second observation with the same key and it will not find the first. This will not cause a fatal error, but will cause problems for you (see When the Lookup Fails, below.)

#3. When the flow of the program requires more than one SET with KEY= on the same dataset with the same index, you can put it in a subroutine. Having two different SET statements in the same DATA step for the same dataset, with or without KEY=, causes 2 separate sets of buffers and positions to be created for the same file. You may have occasions where this is desirable, but I find it easier to manage if I consolidate the lookups.

#4. I created the DIM files with PROC SUMMARY, which gave me records containing total scores over all dimensions as well. I set the value of DIM to 99 on these total records.

When the Lookup Fails: What Happens

#5. When you use direct access you need to be prepared for errors such as the situation in which no record can be found to match the key values you are looking for. The messages you get from SAS are cryptic, and they become ambiguous when you have more than one SET with a KEY= in a DATA step. First, you should know that when using the SET statement, failure to find a key value in a dataset does not cause a fatal error. (When using the MODIFY with KEY=, SAS will set observations to 0 and stop processing when it fails to find a key value.) You will get a dump of the program data vector at the bottom of the DATA step in the log (no message, just a dump), but it does not stop processing or give a bad return code. If a record cannot be found for the key values you seek, all the variables coming from the lookup dataset will retain their values from the previous successful access. This is rarely what you want to see. The solution is to insert statements setting the variables you want to read from the lookup dataset back to missing right before the SET with the KEY=, or to assign them to missing if the _IORC_ variable is not zero.

Special note for character variables:

#6. If you are bringing character variables in from a lookup dataset, take note of how SAS creates variables in the PDV. The syntax checker will create the variable the first time it encounters it. If you have your SET with KEY= in a subroutine and you have any reference to the variable you are reading above that subroutine, the syntax checker will create the variable as numeric by default. Then it will have a problem trying to read the character value. If you initialize the variable to missing before the SET with KEY=, as I recommend, the length of the variable will be defined by the number of spaces you use in your initialization. This will not create a problem for SAS but it might for you.

Use the LENGTH statement at the very top of the DATA step to declare any character variables you are planning to read via direct access.

Using KEEP= for Control

#7. To help document your code and to prevent unintended variable value overlays, it also helps to use the KEEP= dataset option in the SET with KEY= statement. Note that you do not have to KEEP the key variable(s).

When the Lookup Fails: How to Diagnose

#8. Whenever you use indexed access in SAS the _IORC_ variable contains a 0 if the lookup is successful, some other value if it is not. Since an unsuccessful lookup produces only a PDV dump and no message, a non-zero value of _IORC_ is often the only indication that the problem you're having is occurring in a SET with KEY=.
If you have more than one SET with KEY= in the DATA step, you have to check the value of _IORC_ after every one. Since _IORC_ is reset for every SET statement with KEY=, by the time it is dumped at the bottom of the DATA loop it may or may not contain the value from the statement that had a problem. There are a couple of solutions you might use here to help you figure out where the problem lies. One is to generate a message based on the value of the _IORC_ variable immediately after each SET with KEY=. As an example, let's use the subroutine to read the dataset containing the dimension scores for each supervisor:

```sas
SETSDIM:
  CODDIM94=.
  SET SDIMS(KEEP=CODDIM94)
    KEY=CDIM / UNIQUE;
  IF _IORC_ NE 0 THEN PUT 'PROBLEM WITH SETSDIM';
  RETURN;
```

[If you do it this way, make sure you reroute your message from FILE PRINT to FILE LOG and then reset it to FILE PRINT afterward.]

The code in Figure 3 shows another way to cope with the problem of re-using the _IORC_ variable in a DATA step with more than one SET with KEY=. Use different variable names to store the _IORC_ values from each SET with KEY=. Then when the variables are dumped at the bottom of the data loop, all the different _IORC_ values would be available for your inspection.

You may also want to store or print the value of the key at the time of the problem so you can ascertain whether 1) in fact a record does exist for that key and SAS is just not finding it; or 2) it is a legitimate no-find situation.

### The SYSRC Macro

The SYSRC macro exists to help you understand the non-zero values in _IORC_. It is very useful when you are using MODIFY, but has limited utility for SET with KEY=. I will describe it briefly here. Refer to P-222 for more information.

The SYSRC macro is used with one of several mnemonic arguments corresponding to the different types of errors that can occur in direct access. The numeric values returned by the macro match the various values of the _IORC_ variable. This macro is necessary because the specific values are system dependent and subject to change.

The only mnemonic relevant to SET with KEY= is _DSENOM meaning no matching observation was found in the lookup data set.

There are two ways to use the macro.

One way is to test it against _IORC_ to decide what message to PUT in the log. Example:

```sas
IF _IORC_=SYSRC(_DSENOM) THEN
  PUT 'SDIM Record not found for ' CODE= DIM=;
```

A second way is to dump the current numeric values of the different mnemonics for your operating system for your own reference and check the _IORC_ yourself. To do this, run the following type of statement for each mnemonic:

```sas
PUT %SYSRC(_DSENOM) , = _DSENOM';
```

Note that you still need to transfer the value of _IORC_ to a variable that differs for each SET with KEY= in the DATA step so that if the PDV is dumped it will be clear to you which statement had a problem. Example:

```sas
SET SDIMS(KEEP=DTMSCORE)
  KEY=CDIM / UNIQUE;
  SDM_IORC = _IORC_;
```

### Other Errors

One other error can occur that gives ambiguous information when you have more than one SET with KEY= in a DATA step. Suppose you have a complex report writing DATA step with multiple uses of SET with KEY= and you get the following error:

```
ERROR: DATA SET DOES NOT CONTAIN SELECTED KEYS.
```

This is a syntax check error meaning that the index itself does not exist, not that certain values of the keys can't be found. SAS will stop processing this step, etc. I do not know a good way to determine which data set lacks the appropriate index other than checking your records or using process of elimination. Sometimes you might have had to re-create one of the datasets and forgotten to re-create the index. Sometimes it's a typo. You can use PROC CONTENTS to list the current indexes for each dataset you are reading.

Another error is a little easier to find and correct:

```
"THE KEYS= OPTION IS INCONSISTENT WITH A DATA SET WITH WHERE EXPRESSION PROCESSING ACTIVE."
```
You cannot use WHERE with a SET with KEY=. If you have multiple records for one value of the index and you want to choose which one to read, I recommend creating a composite index including the key variable(s) and the variable(s) you want to subset on.

Conclusions

In using SET with KEY= it is wise to observe the following guidelines:

1) Use the UNIQUE option both on SET with KEY= and when you create the index if you expect only one observation per key value.

2) Set all the variables you are planning to bring in through direct access to missing right before the SET with KEY= statement.

3) Use the KEEP= dataset option on the lookup dataset to bring in only the desired variables.

4) Put all SET with KEY= statements in separate subroutines and LINK where necessary.

5) Store the _JOCR_ values from each SET with KEY= in separate variables or print messages to the log right after each direct access attempt that returns a non-zero code.

6) Where possible, code the index creation in the same step that creates the dataset so that every time you have to rerun the creation, and only then, you will recreate the indexes.

SET with KEY= adds a powerful technique to your SAS toolkit, especially when you have to write complex reports with different levels of detail on the same line.

References


Acknowledgments

Thanks to Billy Clifford at SAS Institute for his help with the clarity and technical accuracy of this paper.

Trademarks

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

For Further Information

The author can be reached at
P. O. Box 306
Cumberland Center, ME 04021
(207) 829-6246
USP7NN7N@IBMMAIL.COM