

Using Indexes for Direct Access to SAS Datasets

Judy Loren, Independent Consultant

What is Direct Access?

Direct access or keyed access means choosing one observation from a SAS dataset based on the values of one or more variables. (This is not to be confused with the POINT= option on SET which allows non-sequential 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 each state.

Using Direct Access

To use keyed access to a SAS dataset, you first have to create an index on the variable(s) about which you need further information. 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.

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 population of each state, with one observation per state. To use the INDEX= dataset option on a DATA statement:

```
DATA STATEPOP (INDEX=(STATE));
```

To add an index to an existing dataset:

```
PROC DATASETS LIBRARY=ddname;
  MODIFY STATEPOP;
  INDEX CREATE STATE;
```

As an example of a composite index, suppose we create a dataset with the population of each major city, with one observation per city. Using the INDEX= dataset option:

```
DATA CPOP (INDEX=(C_IDX=(STATE CITY)));
```

To add the composite index to an existing dataset:

```
PROC DATASETS LIBRARY=ddname;
  MODIFY CPOP;
  INDEX CREATE C_IDX=(STATE CITY);
```

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 C_IDX, the values of the variables STATE and CITY will be used to lookup a record in CPOP.

A Note About Re-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.

Using Indexes for Direct Access

Once an index has been defined on a dataset, you can access observations in that dataset non-sequentially, based on a value of the key variable(s). For example, suppose you had the CPOP dataset described above along with another dataset containing individual people for a mailing list. You want to qualify people for your mailing based on the population of the city in which they reside. Before indexing you would have had to sort both datasets on CITY and STATE and merge them. If your list of mailing candidates was large, this would not have been trivial. With KEY=, you can process your mailing list in its original sort and lookup the city populations as you go.

```
DATA QUALIFY;
  SET POTNTL;
  CITYPOP=.;
  SET CPOP (KEEP=CITYPOP) KEY=C_IDX /
    UNIQUE;
  IF _IORC_ NE 0 THEN DO;
    PUT 'PROBLEM WITH LOOKUP '
      STATE=
      CITY=      ;
    _ERROR_=0;
  END;
  *FURTHER CODE BELOW;
```

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 in a row, 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 Lookup Fails, below).

If you know your lookup table should have only one observation for each key value, use the UNIQUE option on index creation and on KEY= access. This ensures that there are no duplicates and that you will find the record you seek.

When the Lookup Fails: How to Prepare

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 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. One solution is to set the variables you want to read from the lookup dataset back to missing right before the SET with the KEY=.

Special note for character variables

If the variables you want to bring in from the indexed dataset are character, setting them to missing creates them with a LENGTH equal to the number of spaces between your quotes. To avoid any problems and allow you to use only one space between your quotes, 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

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 index variable(s).

The _IORC_ Variable

Whenever you use indexed access in SAS the automatic variable _IORC_ 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=. **Check the _IORC_ value after every SET with KEY= and write out the key variable values that failed.**

No WHERE

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.

References

Conner, Ben [1994] "Keys to the Data: Using the KEY= Option on the SET Statement", poster, SUGI 19.

Weiss, Philip J. [1993] "SET, MERGE, UPDATE, or MODIFY? (And What About Indexing?)", SUGI 18 Proceedings, pp 127-132.

SAS Institute, Inc. (1992) SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07 Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (1992) SAS Technical Report P-242, Changes and Enhancements to Base SAS Software, Release 6.08 Cary, NC: SAS Institute, Inc.

Acknowledgments

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

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