

An Animated Guide©: Speed Merges with Key Merging and the `_IORC_` Variable

Russ Lavery, Contractor for ASG, Inc.

ABSTRACT

The key merge (A.K.A. `_IORC_` merge) is an efficiency technique. It is a method of merging two files without having to perform a slow, and disk-space-consuming, pre-sorting of the files. Because this merge does not require pre-sorting of the files to be merged, it is faster and uses less disk space than a by merge.

The `_IORC_` merge is considered one of the "Table Lookup Techniques" and as such is a competitive technique for "by merges", formats, if-else if blocks, key-indexing, bitmapping, hashing and SQL joins. It is a useful technique for SAS programmers because it is fairly fast (faster than a "by merge") and easy to understand. Additionally, `_IORC_` merging is part of the material in the SAS certification exam.

This paper accompanies an animated presentation at NESUG and can not duplicate the animated effect. It will outline the features of the `_IORC_`, as were presented. More material can be found in the excellent articles found in the on-line SUGI proceedings and listed in the reference section of this paper. Use www.lexjensen.com top find articles

INTRODUCTION

This paper will explain details of the `_IORC_` merge and how the Program Data Vector (PDV) is modified as the `_IORC_` merge executes. An understanding of the PDV is key to understanding this technique.

PROGRAM DATA VECTOR (PDV) FACTS

The PDV can be thought of as a data storage area. It functions much like a one line Excel Spreadsheet. The PDV has a column for every variable you read in from the data set, every variable you create in the data step and some automatic variables (`_n_`, `_ERROR_` and `_IORC_`).

When SAS processes a data step set, it copies your data -ONE LINE AT A TIME- into the program data vector. All calculations in your data step will be performed in the program data vector and the results of your calculation will be stored in the PDV. When you have executed all the statements in the data step, values in the PDV will be written to the output file.

If data comes into your PDV from a SAS file (as opposed to cards or a text file), it will automatically be retained until that data set is accessed again with a set command. If a data step accesses two SAS data sets, variables from both data sets will be retained in the PDV until the set statement associated with that data set next executes.

THE BUSINESS PROBLEM

Imagine you are working at a college and you send your assistant to the gym on the first day of class. He interviews people waiting in a line to get their gym lockers and asks them if they are joggers. This information is recorded in a file called `Day_1`. At the end of term your assistant goes to the health office, in the gym, and looks at records of people who visited the health office complaining of either shin splints (a runner's problem) and tennis elbow (Tennis elbow information is not required but your assistant got carried away). This information is put in a file called `UpDt`.

Note that there is not a good match of names between the files. Also note that shinsplints are coded `S_hinsplints` and `No` and tennis elbow is coded `T_ennis elbow` and `No` (see Figure 1).

For our merging goal, we desire a file that contains all the people we interviewed on day 1 and information about their health problems that we collected at end of term. Source data sets, code, the PDV and output file are shown in Figures 1, 2, 3 and 4. These figures are also shown, in a larger size, in the appendix.

THE SAS CODE

The files we desire to merge are of different sizes. The first step in an `_IORC_` merge is to index the larger file.

As you can see in Figure 1; the data step, where the merge takes place has two set statements. Count from the top and think of them as `set1` and `set2`. `set1` executes first. The data set in the `set2` must have the `key=` option and an index. The larger file should be indexed and put in the `set2` statement - the set statement that has the `Key=` option.

The position of the small and large data sets can be reversed (put smaller file in `set2`) and the technique will still work, however the job will run faster if the larger file is indexed and used in `set2` (the set statement with the `key=` option). You might code the indexing as:

```

Proc datasets lib=work;
modify UpDt;
index create name/unique;
quit;

```

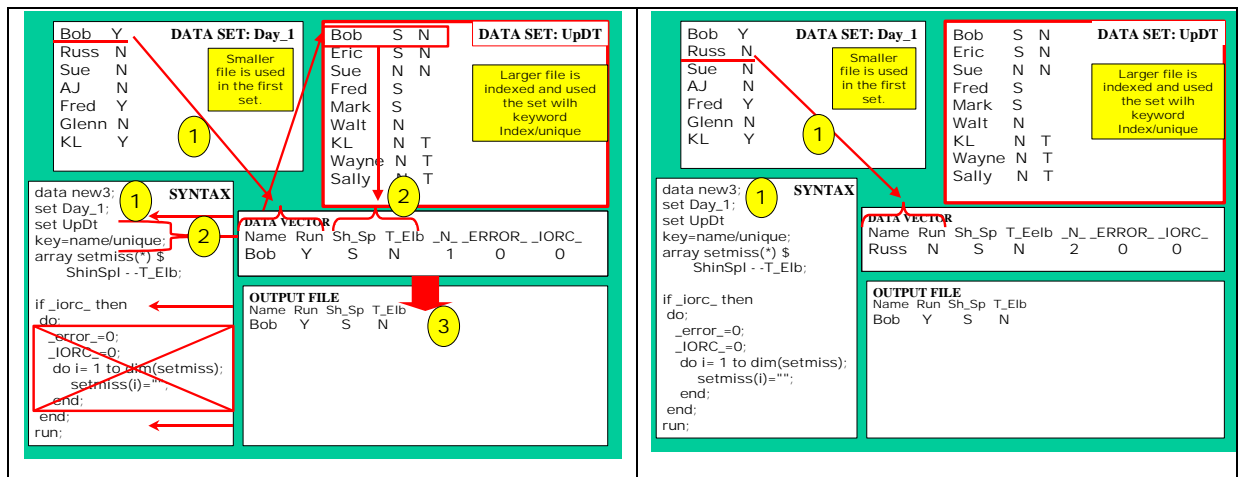


Figure 1

Figure 2

Figure 1 shows the first observation being processed. The statement data New3 creates the PDV and sets user variables to missing. At the top of the data step several things happen automatically. It sets `__n__=1` and sets `__error__` and `__IORC__` to zero (`__IORC__` is set to zero at the top of the data step ONLY for the first observation). The data step executes statements from top down and executes the following statement (circle (1) in Figure 1):

```
set day_1;
```

The above statement reads only the variables from the file Day1 into the PDV. At this time the PDV only contains values for Name (Bob) and Run(Y). `Sh_Sp` and `T_Elb` contain missing values.

Next SAS executes the statement (2):

```
set UpDt key=name/unique;
```

This second set statement performs an indexed table lookup inside UpDt. Since the command option is

```
Key=name
```

SAS looks in the PDV and gets the current value of name. It then uses the current value of name (Bob) to perform an indexed lookup in the file UpDt. SAS looks in UpDt for an observation with name= Bob. When/If it finds such an observation the second set statement executes. The attempt to perform an indexed lookup and the copying of the information to the PDV are separate steps.

When the set executes, it copies the values of the variables in UpDt from the data set into the PDV. Since there was a successful table lookup, `__error__` and `__IORC__` stay at zero. Since `__IORC__` is zero the x-ed out box of code does not execute (Figure 1) for this observation. When SAS reaches the bottom of the data set it outputs the observation to the output file (circle (3) in Figure 1). The output file contains variables for Bob from both files.

Figure 2 shows part of the processing of a "no-match" observation.

When control passes to the top of the data step, two automatic variables are modified. First, `__n__` is incremented by 1. Second, While it is not easy to see here, `__error__` is set to Zero. The value of `__IORC__` is not automatically modified at the top of the data step.

SAS processes the first set command, the line marked with a (1) in Figure 2.

```
set day_1;
```

This line will read the data, from Observation 2 of the file Day_1, into the PDV. After the above line executes, the PDV contains values "Russ" and "N" from the second observation in file Day_1. However; because of the automatic retain of data from SAS data sets, the PDV still contains information that came from Bob's record in the file UpDt. Data Step processing continues as shown below.

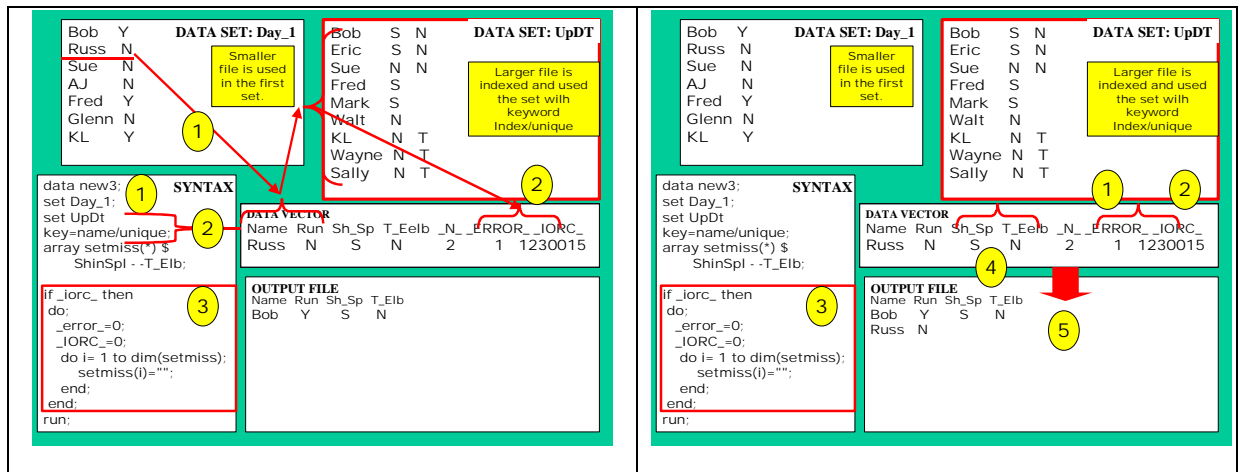


Figure 3

Figure 4

Figure three shows the execution of the second set statement, the line marked with a (2) in Figure 3:

```
set UpDt key=name/unique;
```

SAS looks in the PDV and gets the current value of name (Russ). It then attempts an indexed lookup in the file UpDt for an observation with name= Russ.

When it fails to find an observation with name=Russ in UpDt, the set does not execute. SAS sets _error_ to 1 and _IORC_ to a non-zero number. Dangerously, it does not reset the values of Sh_Sp and T_Elb to missing. The values for these variables have been retained from Bob, are not correct, and must be corrected manually. Since the value of _IORC_ is not zero the box of code (circle (3) in Figure 3) executes.

Figure 4 shows the effects of executing the box of code (circle (3) in Figure 4). The variable _error_ is set to zero (circle (1) in Figure 4) to suppress the printing on the error message in the log. The variable _IORC_ is set to zero (circle (2) in Figure 4) just to be tidy. This resetting of the _IORC_ is not required for correct execution of the merge. We use array logic (circle (3) in Figure 4) to set all the variables that came from the indexed data set (UpDt) to missing. Note that the "reset to missing logic" would have to be a bit more complex if we had brought in a mixture of numeric and character variables from UpDt (SAS arrays should be all numeric or all character). After all the code in the box finished executing, the observation would be copied to the output data set.

The code in Figures 1 through 4 created a data set that has all the observations from Day_1, regardless of the success of the matching attempt. If the match on an observation was successful, the output data set has variables from both input data sets. If the matching attempt was not successful, the observation has missing values. This output structure is often what a client wants.

SELECTING OBSERVATIONS IN BOTH FILES

The code would be slightly different if your goal were to create a data set that contains just the people that are in both files. That code is below. The pictures above can be used to examine the details of the logic.

```
*Index larger file;
proc datasets lib=work;
modify UpDt;
index create name/unique;
quit;

data new2;
set day_1;
set UpDt key=name/unique;

if _iorc_ NE 0 then
do;
_error_=0;
delete;
end;
run;
```

This code checks for "failed index lookup" by checking _IORC_. If _IORC_ is not zero, the code resets _error_ to zero and deletes the observation.

CONCLUSION

The "Table Lookup" function (i.e. merging two data sets, selecting observations from a large file that are also in a smaller file or performing a long series of if-else if processing) is a common SAS task and SAS programmers should know the best ways to perform this task.

The __IORC__ merge is a fast way of selecting observations from a large data set. It does not require sorting of the data sets thus conserves CPU time and disk space. It is material that is included in the SAS certification exams. For more details please read the excellent articles that are online at the NESUG and SUGI web sites and that are mentioned in the reference section. The articles by Sandra Aker were especially helpful to the author.

Anyone needing to perform the "Table Lookup function" quickly, and without sorting the data sets, should also investigate using formats. There are several articles on table lookup using formats in the NESUG and SUGI proceedings. Search for articles using <http://www.lexjansen.com/>

The Newest and fastest "Table Lookup" techniques (key indexing, bitmapping and hashing) are described in Paul Dorfman's articles. These techniques are a little more difficult to master than __IORC__ and format Table Lookups but are the fastest way to select observations from a large file that are in a smaller file.

REFERENCES

Aker, Sandra Lynn, "Table Look-up using Indexes, SQL, Arrays and formats without using Matched Merge Data Steps " Proceedings of the Tenth Annual New England SAS User's Group Conference", 10, pg3

Aker ,Sandra Lynn, "Using Indexes to Perform Table Look-up " Proceedings of the Twelfth Annual New England SAS User's Group Conference", 12, pg 179

Carpenter, Carpenter, "Table Lookups: From IF-THEN to key-indexing
" Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference", 26, paper 158

Croonen and Theuwissen.Author, "Table Look-up: Techniques Beyond the Obvious " Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference", 27, paper 11

Foley , Malachy J., "Advanced MATCH-MERGING: Techniques, Tricks, and Traps " Proceedings of the Twenty-Second Annual New England SAS User's Group Conference", 22, paper 39

Gober, John C., "Understanding Indexed Datasets and Using Direct Access Queries " Proceedings of the Twenty_Third Annual SAS Users Group International Conference", 23, pg 64

McAllister, Doug, "Proceedings of the Eleventh Annual New England SAS User's Group Conference", 11, pg 40

Rafiee , Dana, "NO MORE MERGE: Alternative Table Lookup Techniques " Proceedings of the Twenty-Second Annual SAS Users Group International Conference", 22, pg paper 88

Riba , S. David, "Table Look-up Techniques Other Than the Matched Merge DATA Step " Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference", 27 paper 27

Stinson ,Walter, "Indexing: My new best friend for table lookup " Proceedings of the Thirteenth Annual New England SAS User's Group Conference", 13, pg 293

Zdeb ,Mike, "Five (or more) Alternatives for Record Selection From One File Based On Information In Another " Proceedings of the Fourteenth Annual New England SAS User's Group Conference", `14, pg 355

ACKNOWLEDGMENTS

Thanks to the following skilled reviewers: John Maurer, Saad Anbari and Musa Nsereko.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Russell Lavery
 9 Station Ave. Apt 1,
 Ardmore, PA 19003,
 610-645-0735 # 3
 Email: russ.lavery@verizon.net
 Contractor for ASG, Inc. WWW.ASG-INC.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

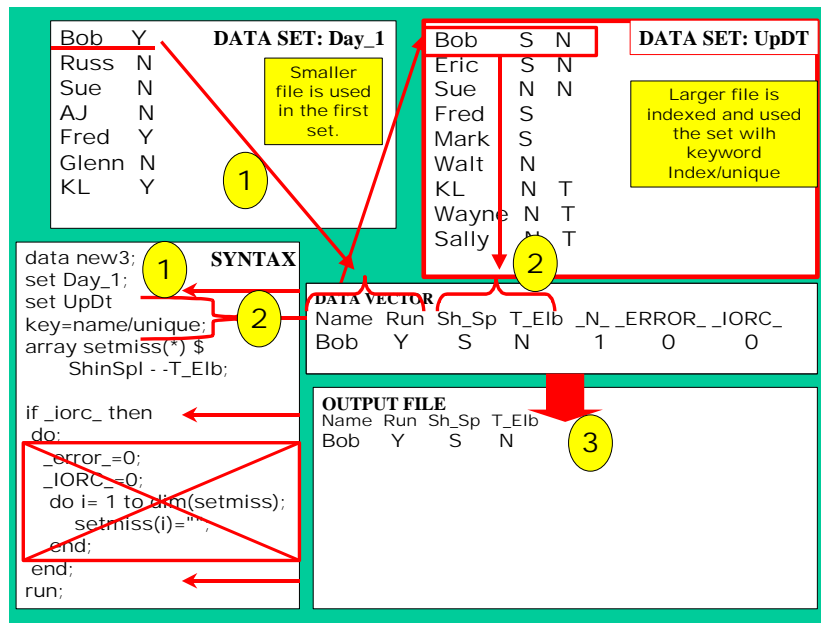


Figure 1

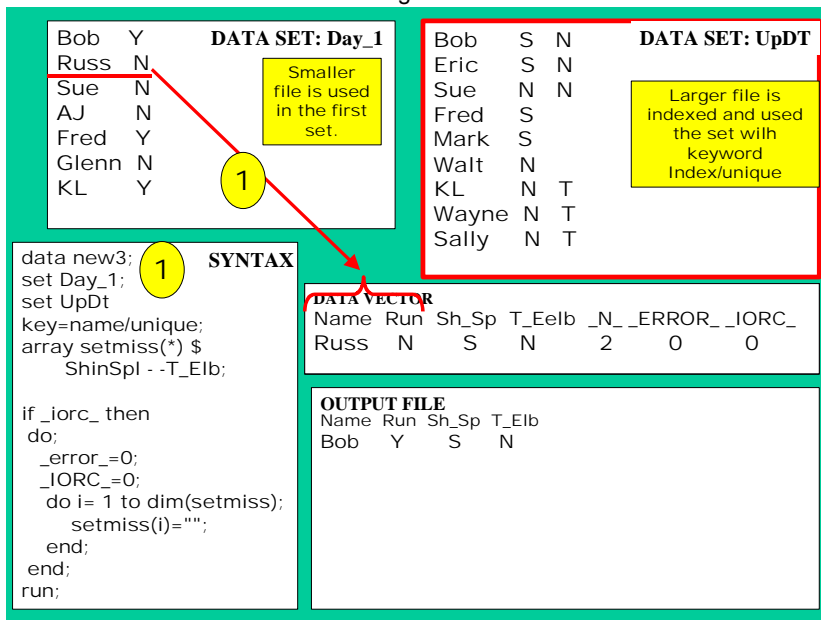


Figure 2

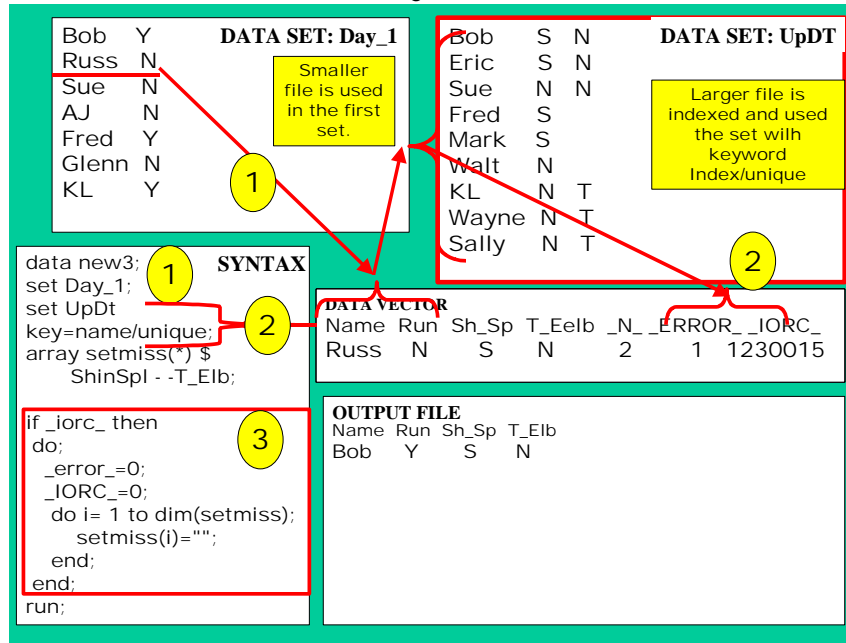


Figure 3

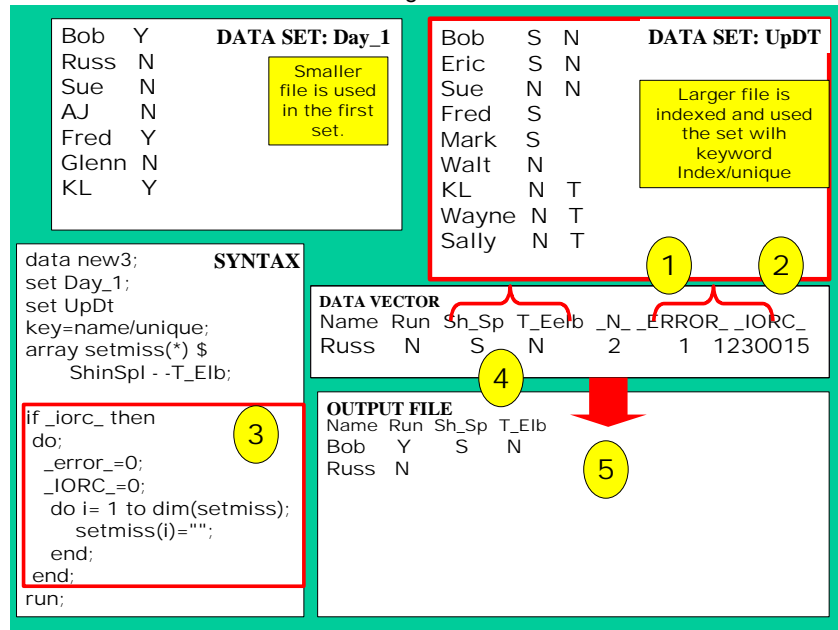


Figure 4