PharmaSUG China 2018 – Paper 18

Basic SAS® Hash Programming Techniques Applied in Our Daily Work in Clinical Trials Data Analysis

Fanyu Li, MSD, Beijing, China

ABSTRACT

With the development of SAS programming techniques, more and more methods could be used in the SAS. As a statistical programmer, we always need to merge some information from one dataset to another dataset. The simple cases are one-to-one merge and one-to-many merge. However, sometime, we may face the many-to-many merge, what should we do? We could utilize a hash object to perform lookups within the DATA step with easier method. The hash object is a powerful technique and this technique could improve the efficiency of table lookup. In this paper, it presents some practical examples which the programmers could encounter in their daily work. In addition, it also brings up some points that need to pay attention to when using the Hash programming techniques.

INTRODUCTION

The Hash programming technique provides an efficient, convenient mechanism for quick data storage and retrieval. It stores and retrieves data based on lookup keys. There are three steps to create a hash object.

1. Declare the hash object.
2. Create an instance of (instantiate) the hash object.
3. Initialize lookup keys and data.

Data step is the basic step of SAS programming. However, when dealing with large volumes of data, the ordinary data steps can be slow. A hash object is a type of array that a program accesses using keys. A hash object consists of key items and data items. The programming language applies a hash function that maps the keys to positions in the array. There are a lot of papers that had introduction about how to create a hash object. This paper will not present the details again. Below it presents some examples to show how to use it in clinical trials.

EXAMPLE 1

The following example used the SASHELP dataset to show how to implement the hash iterator object. Using the hash iterator object could store and search data based on lookup keys. The hash iterator object enables you to retrieve the hash object data in either forwarding or reversing key order. There is dataset called CARS in SASHELP. In this dataset, it has a lot of information such as the brand of cars, the model of cars, what is the type of cars and so on. Let's see the snapshot of this dataset.

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3	6	220
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3	6	220
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3	6	220
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250
18	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports	Europe	Front	$35,940	$32,512	1.8	4	180
23	Audi	TT 1.8 Quattro 2dr (convertible)	Sports	Europe	All	$37,390	$33,891	1.8	4	225
24	Audi	TT 3.2 coupe 2dr (convertible)	Sports	Europe	All	$40,590	$36,739	3.2	6	250
25	Audi	A6 3.0 Avant Quattro	Wagon	Europe	All	$40,840	$37,060	3	6	220
26	Audi	S4 Avant Quattro	Wagon	Europe	All	$49,090	$44,446	4.2	8	340
27	BMW	X3 3.0i	SUV	Europe	All	$37,000	$33,873	3	6	225
28	BMW	X5 4.4i	SUV	Europe	All	$52,195	$47,720	4.4	8	325
29	BMW	325i 4dr	Sedan	Europe	Rear	$28,495	$26,155	2.5	6	184

Assume that one customer needs to find records with horsepower greater than 350. We used the following statements.

```
data car;
      if 0 then set sashelp.cars;
      if _n_=1 then do;
          declare hash h(dataset:"sashelp.cars", ordered:'yes');
          declare hiter iter('h');
          h.definekey('Horsepower');
          h.definedata(all:'YES');
          h.definedone();
          call missing(of _all_);
      end;
      rc=iter.first();
      do while (rc=0);
          if Horsepower>350 then output;
          rc=iter.next();
      end;
  run;
```

After running the above code, there are six such records and the outputs are ordered in ascending sequence. The log is clear.

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
1	Jaguar	S-Type R 4dr	Sedan	Europe	Rear	$63,120	$57,499	4.2	8	390
2	Volkswagen	Phaeton W12 4dr	Sedan	Europe	Front	$75,000	$69,130	6	12	420
3	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450
4	Porsche	911 GT2 2dr	Sports	Europe	Rear	$192,465	$173,560	3.6	6	477
5	Mercedes-Benz	CL600 2dr	Sedan	Europe	Rear	$128,420	$119,600	5.5	12	493
6	Dodge	Viper SRT-10 convertible 2dr	Sports	USA	Rear	$81,795	$74,451	8.3	10	500

However, after using the simplest way to test, this result is not correct! Why is this result not correct? How to correct the above code?

```
data car1;
    if 0 then set sashelp.cars;
    if _n_=1 then do;
        declare hash h(dataset:"sashelp.cars", ordered:'yes', multidata:'Y');
        declare hiter iter('h');
        h.definekey('Horsepower');
        h.definedata(all:'YES');
        h.definedone();
        call missing(of _all_);
    end;
    rc=iter.first();
    do while (rc=0);
        if Horsepower>350 then output;
        rc=iter.next();
    end;
run;
```

After modifying the above code, the correct output below shows all the records' with horsepower great than 350.

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
1	Jaguar	S-Type R 4dr	Sedan	Europe	Rear	$63,120	$57,499	4.2	8	390
2	Jaguar	XJR 4dr	Sedan	Europe	Rear	$74,995	$68,306	4.2	8	390
3	Jaguar	XKR coupe 2dr	Sports	Europe	Rear	$81,995	$74,676	4.2	8	390
4	Jaguar	XKR convertible 2dr	Sports	Europe	Rear	$86,995	$79,226	4.2	8	390
5	Volkswagen	Phaeton W12 4dr	Sedan	Europe	Front	$75,000	$69,130	6	12	420
6	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450
7	Porsche	911 GT2 2dr	Sports	Europe	Rear	$192,465	$173,560	3.6	6	477
8	Mercedes-Benz	CL600 2dr	Sedan	Europe	Rear	$128,420	$119,600	5.5	12	493
9	Mercedes-Benz	SL55 AMG 2dr	Sports	Europe	Rear	$121,770	$113,388	5.5	8	493
10	Mercedes-Benz	SL600 convertible 2dr	Sports	Europe	Rear	$126,670	$117,854	5.5	12	493
11	Dodge	Viper SRT-10 convertible 2dr	Sports	USA	Rear	$81,795	$74,451	8.3	10	500

Let's see the difference of the two outputs. There are 5 records didn't appear in our previous output. The first output only shows one record for Jaguar and Mercedes-Benz. The hash object process defines the horsepower as the hash key however it is not unique, that's why there is only one record for Jaguar and Mercedes-Benz. After adding the option "Multidata: 'Y'", all the records with horsepower greater than 350 were selected. This example is very simple. You need pay attention to whether the key is unique, please remember to add the "Multidata:'Y'" option. If the customer wants to find the records with horsepower>350 and the cheapest retail price, could we only revised our previous program and get the result? The answer is yes and the modified part is very small.

```
proc sort data=sashelp.cars out=carall; by descending msrp ; run;

data car2;
     if 0 then set carall;
     if _n_=1 then do;
         declare hash h(dataset:"carall", ordered:'yes', duplicate:'replace');
         declare hiter iter('h');
         h.definekey('Horsepower');
         h.definedata(all:'YES');
         h.definedone();
         call missing(of _all_);
     end;
     rc=iter.first();
     do while (rc=0);
         if Horsepower>350 then output;
         rc=iter.next();
     end;
 run;
```

After rerun the above code, the result is shown as following.

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
Jaguar	S-Type R 4dr	Sedan	Europe	Rear	$63,120	$57,499	4.2	8	390
Volkswagen	Phaeton W12 4dr	Sedan	Europe	Front	$75,000	$69,130	6	12	420
Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450
Porsche	911 GT2 2dr	Sports	Europe	Rear	$192,465	$173,560	3.6	6	477
Mercedes-Benz	SL55 AMG 2dr	Sports	Europe	Rear	$121,770	$113,388	5.5	8	493
Dodge	Viper SRT-10 convertible 2dr	Sports	USA	Rear	$81,795	$74,451	8.3	10	500

There are six types of horsepower which is greater than 350. If the horsepower is 390, the cheapest price of the brand is Jaguar with the retail price $63,120. If the horsepower is 493, we could find 3 records in for Mercedes-Benz, the cheapest one is $121,770.

If the key is not unique and there are several records, we could only list the first or last record. The option "duplicate" determines whether to ignore duplicate keys when loading a data set into the hash object. The default is to store the first key and ignore all subsequent duplicates. If we want to store the last duplicate key record, we should use "replace" as the option's value. If we want to check whether the key is unique or not unique, we could use "error" as the option's value and it reports an error to the log if key is not unique.

EXAMPLE 2

In this example, it presents how the hash programming technique is applied in the EPOCH derivation. In SE domain, we have already had the

EPOCH. Suppose that we need to derive the EPOCH in VS domain. Let's use SE domain dataset first. We just need consider the date when deriving EPOCH.

SUBJID	EPOCH	SESTDTC	SEENDTC
110116	SCREENING	1943-07-15	
110109	SCREENING	1940-10-15	2018-04-12T13:03
110109	TREATMENT	2018-04-12T13:03	2018-05-22
110109	FOLLOW-UP	2018-05-22	
110107	SCREENING	1948-02-15	2018-04-05T10:26
110107	TREATMENT	2018-04-05T10:26	2018-05-15
110107	FOLLOW-UP	2018-05-15	
110103	SCREENING	1987-05-15	2018-03-28T12:35
110103	TREATMENT	2018-03-28T12:35	2018-05-09
110103	FOLLOW-UP	2018-05-09	
110105	SCREENING	1958-06-15	2018-03-30T12:33
110105	TREATMENT	2018-03-30T12:33	2018-05-10
110105	FOLLOW-UP	2018-05-10	
110110	SCREENING	1965-07-15	2018-04-13T12:27
110110	TREATMENT	2018-04-13T12:27	2018-05-21
110110	FOLLOW-UP	2018-05-21	

In SE domain, the above snapshot just keep four variables: SUBJID, EPOCH, SESTDTC, SEENDTC. In VS domain, we just keep 4 variables: SUBJID, VSTEST, VSDTC, VSSTRESN. The below snapshot is just part of the records in VS domain.

SUBJID	VSTEST	VSDTC	VSSTRESN
110109	Body Mass Index	2018-04-04	18.9
110109	Diastolic Blood Pressure	2018-04-04	79
110109	Diastolic Blood Pressure	2018-04-11	83
110109	Diastolic Blood Pressure	2018-04-12	75
110109	Diastolic Blood Pressure	2018-05-09	74
110109	Height	2018-04-04	148.4
110109	Heart Rate	2018-04-04	81
110109	Heart Rate	2018-04-11	83
110109	Heart Rate	2018-04-12	84
110109	Heart Rate	2018-05-09	100
110109	Respiratory Rate	2018-04-04	18
110109	Respiratory Rate	2018-04-11	20
110109	Respiratory Rate	2018-04-12	20
110109	Respiratory Rate	2018-05-09	20
110109	Systolic Blood Pressure	2018-04-04	131
110109	Systolic Blood Pressure	2018-04-11	134
110109	Systolic Blood Pressure	2018-04-12	122
110109	Systolic Blood Pressure	2018-05-09	121
110109	Temperature	2018-04-04	36.6
110109	Temperature	2018-04-11	36.7
110109	Temperature	2018-04-12	37.1
110109	Temperature	2018-05-09	36.9
110109	Weight	2018-04-04	41.7
110109	Weight	2018-05-09	41.4
110107	Body Mass Index	2018-03-28	20.5
110107	Diastolic Blood Pressure	2018-03-28	71
110107	Diastolic Blood Pressure	2018-04-04	71
110107	Diastolic Blood Pressure	2018-04-05	80
110107	Diastolic Blood Pressure	2018-05-02	59

If VSDTC is between in SESTDTC and SEENDTC within the same subject, the EPOCH in SE domain should be the EPOCH in VS domain. Let's see how to implement it.

```
data se;
     set lptss.se;
     keep usubjid subjid epoch sestdtc seendtc sestdt seendt;
 run;
proc sort; by usubjid sestdt seendt; run;
data vs;
     set lptsdd.ar_vs02_vs;
     keep usubjid subjid vstest vstestcd vsstresn vsstresc vsdtc vsdt vsseq;
 run;
proc sort; by usubjid vsdtc vstestcd ; run;

data vs_1;
     length usubjid $30 epoch epoch_ $40 sestdt seendt 8;
     if _n_=1 then do;
         declare hash matchepoch(dataset:"se", ordered:'yes', multidata:'Y');
         declare hiter myiter('matchepoch');
         matchepoch.definekey('USUBJID');
         matchepoch.definedata('SESTDT', 'SEENDT', 'EPOCH');
         matchepoch.definedone();
         call missing(epoch, sestdt, seendt);
     end;
     set vs;
     rc = myiter.first();
     do while (rc = 0);
         if sestdt<vsdt<seendt or (missing(seendt) and vsdt>=sestdt) then do;
         epoch_=epoch;
         output;
         end;
         rc = myiter.next();
     end;
 run;
```

After running the above code, the log is clear. The result presents as belowed.

SUBJID	VSTEST	VSDTC	VSSTRESN	epoch	sestdt	seendt
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1943-07-15	.
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1940-10-15	2018-04-12
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1948-02-15	2018-04-05
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-28	2018-05-09
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-30	2018-05-10
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1965-07-15	2018-04-13
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1966-01-15	2018-04-10
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1951-08-15	.
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-28	2018-05-07
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-04-04	2018-05-15
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-31	2018-05-11
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-29	2018-05-10
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1961-10-15	.
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1964-10-15	.
110109	Body Mass Index	2018-04-04	18.9	TREATMENT	2018-03-27	2018-05-07
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1960-03-15	.
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1989-01-15	.
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1967-08-15	2018-04-11

Obviously, the result is not correct. We could find that the highlighted records (subject 110109) merged SESTDTC and SEENDTC records did not belong to subject 110109 in SE domain. It merged with other patient's SESTDTC and SEENDTC. If the VSDTC is between in the SESTDTC and SEENDTC, the qualified records were created. The hash iterator passes through all the SE domain dataset ignoring the key. The data step hash iterator is a companion for the DATA step hash object. It enables programs to step through hash object records without performing key lookups. If the record's VSDTC falls between SESTDTC and SEENDTC, it is considered a 'match' and output to VS_1 dataset. How to revise it so that it could get correct result? Let's see below revised code.

```
data vs_2;
      length usubjid $30 epoch epoch_ $40 sestdt seendt 8;
      if _n_=1 then do;
          declare hash matchepoch(dataset:"se", ordered:'yes', multidata:'Y');
          matchepoch.definekey('USUBJID');
          matchepoch.definedata('SESTDT', 'SEENDT', 'EPOCH');
          matchepoch.definedone();
          call missing(epoch, sestdt, seendt);
      end;
      set vs;
      rc = matchepoch.find();
      do while (rc = 0);
          if sestdt<=vsdt<seendt or (missing(seendt) and vsdt>=sestdt) then do;
          epoch_=epoch;
          output;
          end;
          rc =matchepoch.find_next();
      end;
  run;
```

We chose to use the Hash Object programming, not hash iterator. After running the revised code, the correct result.

SUBJID	VSTEST	VSDTC	VSSTRESN	epoch	sestdt	seendt
110109	Body Mass Index	2018-04-04	18.9	SCREENING	1940-10-15	2018-04-12
110109	Diastolic Blood Pressure	2018-04-04	79	SCREENING	1940-10-15	2018-04-12
110109	Diastolic Blood Pressure	2018-04-11	83	SCREENING	1940-10-15	2018-04-12
110109	Diastolic Blood Pressure	2018-04-12	75	TREATMENT	2018-04-12	2018-05-22
110109	Diastolic Blood Pressure	2018-05-09	74	TREATMENT	2018-04-12	2018-05-22
110109	Height	2018-04-04	148.4	SCREENING	1940-10-15	2018-04-12
110109	Heart Rate	2018-04-04	81	SCREENING	1940-10-15	2018-04-12
110109	Heart Rate	2018-04-11	83	SCREENING	1940-10-15	2018-04-12
110109	Heart Rate	2018-04-12	84	TREATMENT	2018-04-12	2018-05-22
110109	Heart Rate	2018-05-09	100	TREATMENT	2018-04-12	2018-05-22
110109	Respiratory Rate	2018-04-04	18	SCREENING	1940-10-15	2018-04-12
110109	Respiratory Rate	2018-04-11	20	SCREENING	1940-10-15	2018-04-12
110109	Respiratory Rate	2018-04-12	20	TREATMENT	2018-04-12	2018-05-22
110109	Respiratory Rate	2018-05-09	20	TREATMENT	2018-04-12	2018-05-22
110109	Systolic Blood Pressure	2018-04-04	131	SCREENING	1940-10-15	2018-04-12
110109	Systolic Blood Pressure	2018-04-11	134	SCREENING	1940-10-15	2018-04-12
110109	Systolic Blood Pressure	2018-04-12	122	TREATMENT	2018-04-12	2018-05-22
110109	Systolic Blood Pressure	2018-05-09	121	TREATMENT	2018-04-12	2018-05-22
110109	Temperature	2018-04-04	36.6	SCREENING	1940-10-15	2018-04-12
110109	Temperature	2018-04-11	36.7	SCREENING	1940-10-15	2018-04-12
110109	Temperature	2018-04-12	37.1	TREATMENT	2018-04-12	2018-05-22
110109	Temperature	2018-05-09	36.9	TREATMENT	2018-04-12	2018-05-22
110109	Weight	2018-04-04	41.7	SCREENING	1940-10-15	2018-04-12
110109	Weight	2018-05-09	41.4	TREATMENT	2018-04-12	2018-05-22

Let's see the above results. In SE domain, Subject 110109 had three records, the screening epoch started from 1940-10-15 to 2018-04-12. The treatment epoch started from 2018-04-12 to 2018-05-22. The period after 2018-05-22 was the follow-up epoch. In this example, if the date is the same with the end date of next epoch's start date, the next epoch should be given for this record epoch. If all records in VS had EPOCH, VS_2 records should be the same with VS. If some records did not have EPOCH, the programmer should re-check why the EPOCH was missing for those records.

CONCLUSION

SAS hash programming is a powerful and efficient approach for table lookups and many to many merges. When you face the many to many merge, you could try this method instead of Cartesian. Although it maybe seems a litter bit difficult at first, you could write a simpler code to complete this process. Some examples are my lessons when I learn to use this method. After you get familiar with basic statement and meet the issues, you should look up the details and apply the different options for this process.

REFERENCES

SAS® 9.4 Language Reference: Concepts, Sixth Edition. Using the Hash Object.

Getting Started with the DATA Step Hash Iterator. Janice Bloom, SAS Institute Inc., Cary, NC and Jason Secosky, SAS Institute Inc., Cary, NC

SCSUG 2012, SAS® HASH Programming basics, Daniel Sakya, PPD Inc., Austin, TX

ACKNOWLEDGMENTS

I would like to thank Peng Wan to encourage and support to participant this conference and present this paper. In addition, I also should thank Yong Cao to give me some instruction and help to complete this paper.

CONTACT INFORMATION <HEADING 1>

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

Name: Fanyu Li
Enterprise: MSD
Address: 8F, Building 21 Rongda Road, Wangjing R&D Base, Zhongguancun Electronic Zone West Zone, Chaoyang District.
City, State ZIP: Beijing, 100012
Work Phone: 58609282
E-mail: Fanyu.li@merck.com