**PhUSE 2014**

**Paper CS04**

# Overview of HASH Objects

Swarnalatha Gaddam, Cytel Inc. Hyderabad, India

**Abstract:**
This topic is intended to provide more exposure to beginner or experienced SAS programmers who are looking for alternative to data steps in SAS. The concept of HASH programming is similar to the definition of an array in SAS. Several SAS users have benefitted from HASH programming by considerably reducing the processing time for compound data merging tasks. We are going to discuss introduction to HASH programming, syntax, and a few examples which are more relevant to clinical industry. Also benefits on using HASH programming versus regular data step would be discussed.

## Introduction:

Data step programming is the vital part of SAS programming for most of the programmers. However, when dealing with large volumes of data, data step programming can become so handy.

SAS HASH programming is a powerful and efficient object oriented approach for table lookups, merges, data summarization, and sorting purposes.

It support users to perform and compare data step merges versus HASH merges in terms of compilation and execution time.

## THE DEFINITION

Hashing is a search algorithm based on a mathematical technique to convert a key into an array index.
It exists only within the DATA step in which it creates the HASH object. When the DATA step ends, SAS deletes the HASH object.

## Concept :

Hashing incorporates 3 elements:
hash function - converts keys into array indices.
hash table - is a memory-resident table that stores the data to be searched
buckets - locations in the hash table, referenced by the indices created by the hash function.

## Syntax:

We will walk through the syntax of hash by illustratiing 2 datasets.
Now a days, clinical industry and FDA are mostly interested to use CDISC standards. Let us illustate one of the examples related to SDTM i.e. with Lab data.
There is a master lab dataset which has subject lab information with lab tests, visits and corresponding result values for each lab test for each subject.
As mentioned earlier, as we are using SDTM example, we need to have LBTESTCD, LBCAT and LBSCAT for each lab test. We need to ge this information from another dataset.
Let us see how easy to load the transaction dataset details into master dataset using hash object.

As we are looking below, it is the main dataset which has all the lab tests per subject, per visit and with the respective lab results.

Data: LAB

| subjid | visit | lbdate | lbtest | lab_result |
|---|---|---|---|---|
| AAAAA | DAY1 | 2014-02-07 | ALBUMIN | 45 |
| AAAAA | DAY1 | 2014-02-07 | ALKALINE_PHOSPHATASE | 84 |
| AAAAA | DAY1 | 2014-02-07 | APOB_APO_A1 | 0.61 |
| AAAAA | DAY1 | 2014-02-07 | APOLIPOPROTEIN_A1 | 1.66 |
| AAAAA | DAY1 | 2014-02-07 | APOLIPOPROTEIN_B | 1.02 |
| AAAAA | DAY2 | 2014-03-14 | BASOPHILS | 0.2 |
| AAAAA | DAY2 | 2014-03-14 | BASOPHILS_ABSOLUTE | 0.01 |
| BBBBB | DAY1 | 2014-02-07 | ALBUMIN | 45 |
| BBBBB | DAY1 | 2014-02-07 | ALKALINE_PHOSPHATASE | 34 |
| BBBBB | DAY1 | 2014-02-07 | BASOPHILS | 0.5 |
| BBBBB | DAY1 | 2014-02-07 | BASOPHILS_ABSOLUTE | 0.03 |
| BBBBB | DAY1 | 2014-02-07 | BICARBONATE | 23 |
| BBBBB | DAY1 | 2014-02-07 | BILIRUBIN_DIRECT | 2 |
| BBBBB | DAY2 | 2014-03-14 | BILIRUBIN_TOTAL | 5 |
| BBBBB | DAY2 | 2014-03-14 | BILIRUBIN_URINE | . |
| BBBBB | DAY2 | 2014-03-14 | BLOOD_URINE | . |
| BBBBB | DAY2 | 2014-03-14 | BUN | 6.78 |

This is the another dataset from which LBTESTCD, LBCAT and LBSCAT could be loaded to the main LAB data.

Data: LAB_TESTCD

| lbtestcd | lbcat | lbscat | lbtest |
|---|---|---|---|
| ALB | CHEMISTRY | PROTEIN | ALBUMIN |
| ALP | CHEMISTRY | ENZYME,LIVER FUNCTION | ALKALINE_PHOSPHATASE |
| APOBA1 | CHEMISTRY | LIPOPROTEIN | APOB_APO_A1 |
| APO_A1 | CHEMISTRY | PROTEIN | APOLIPOPROTEIN_A1 |
| APO_B | CHEMISTRY | PROTEIN | APOLIPOPROTEIN_B |
| BASO | HEMATOLOGY | DIFFERENTIAL | BASOPHILS |
| BASOABS | HEMATOLOGY | DIFFERENTIAL | BASOPHILS_ABSOLUTE |
| BICARB | CHEMISTRY | ELECTROLYTE | BICARBONATE |
| BILI_DR | CHEMISTRY | LIVER FUNCTION | BILIRUBIN_DIRECT |
| BILI_TL | CHEMISTRY | LIVER FUNCTION | BILIRUBIN_TOTAL |
| BILI_U | URINALYSIS | LIVER FUNCTION | BILIRUBIN_URINE |
| BLOODU | URINALYSIS | RENAL FUNCTION | BLOOD_URINE |
| BUN | CHEMISTRY | RENAL FUNCTION | BUN |

Hash code to load LAB_TESTCD dataset into LAB dataset.

```
Data lab_hash(Drop=Rc);

/*Initialize HASH object variables*/
Length lbtest $400 lbtestcd lbcat lbscat $500;

/*Declare HASH*/
If _N_ = 1 Then Do;

Declare Hash lbhash(HashExp:8,Dataset:'lab_testcd');❶ ❷

/*defining key variable which is used as Key for both the datasets*/
lbhash.DefineKey('lbtest');❸

/*defining data variable which needs to be loaded into HASH object
and then to main data with match of key variable*/
lbhash.DefineData('lbtestcd', 'lbcat', 'lbscat');❹

lbhash.DefineDone();

/*Initialize HASH object variables to missing*/
Call Missing(lbtest, lbtestcd, lbcat, lbscat);❺
End;

/*Set master dataset*/
Set lab;❻

/*Find the existance of hash and then load matching data into master
dataset*/
Rc = lbhash.Find(Key:lbtest);❼

If Rc = 0 Then Do;❽

Output lab_hash;❾

End;
run;
```

❶ Create hash object called LBHASH.

❷ Load observations from LAB_TESTCD dataset into hash object LBHASH .

❸ Identify variable LBTEST as the key to find data in hash object LBHASH.

❹ Load variables i.e. "`lbtestcd`", "`lbcat`" and "`lbscat`" into LBHASH. We need to have comma seperator while adding list of varibles.

❺ Initialize to missing the DATA step variables with the same names as the items that SAS loads into hash object LBHASH.

❻ Read each observation from LAB dataset.
❼ Look for hash object and find the matching observation from the LBHASH, depending on LBTEST in the LAB dataset observation currently being processed.

❽ When SAS finds a match in hash object , it retrieves from hash object into LAB dataset.

❾ When all the observations are being processed then output the results into LAB_HASH dataset.

Final output dataset LAB_HASH:

**Hash Object Methods :**
Like FIND method, we have used in our previous example, there are different methods in hash objects. Those include:
ADD, CHECK, CLEAR, DELETE, EQUALS, FIND_NEXT, FIND_PREV, REF etc. You could use these methods depending on your requirement.

**How to Store and Retrieve Data:**
As an example, let us see how we can Store and Retrieve Data using different methods of hash objects:

Below is the illustration of generating a dataset, hash_add by loading the data manually into the hash object through ADD method. Here, if we don't use FIND method with output statement, after each entry (for each patid), and if we use only at the last entry, then the final output will have only the record of final entry.

```
data hash_add;

length patid 8. disease $20.;

/* Declare the hash object and key and data variables */
if _N_ = 1 then do;
   declare hash hadd();

   rc = hadd.defineKey('patid');

   rc = hadd.defineData('disease');

   rc = hadd.defineDone();
end;

/* Define constant value for key and data */
patid = 1001001;

disease = 'Leukemia';
/* Use the ADD method to add the key and data to the hash object
*/
rc =hadd.add();
```

```
if (rc ne 0) then
   put 'Add failed.';

else if (rc = 0) then output;
/* Define constant value for key and data */
patid = 1001002;

disease = 'Alziemers';
/* Use the ADD method to add the key and data to the hash object
*/
rc = hadd.add();

if (rc ne 0) then
   put 'Add failed.';

else if (rc = 0) then output;
run;
```

HANDLING DUPLICATE KEYS:

By default, all of the keys in a hash object are unique. This means one set of data variables exists for each key. In some situations, you might want to have duplicate keys in the hash object, that is, associate more than one set of data variables with a key.

For example, assume that the key is a patient ID and the data is a visit date. If the patient were to visit multiple times, multiple visit dates would be associated with the patient ID. When you create a hash object with the MULTIDATA:“YES” argument tag, multiple sets of the data variables are associated with the key.

If the data set contains duplicate keys, by default, the first instance is stored in the hash object and subsequent instances are ignored. To store the last instance in the hash object, use the DUPLICATE argument tag. The DUPLICATE argument tag also writes an error to the SAS log if there is a duplicate key.

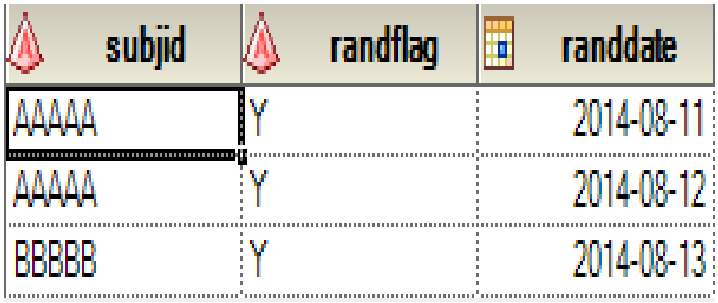
However, the hash object allows storage of multiple values for each key if you use the MULTIDATA argument tag in the DECLARE statement or _NEW_ operator.

Most often, in clinical reports, we neee to consider the subjects which are randomized.

Lets assume, the ENROLMENT data has duplicate records for a subject (might be due to data error).

As in below snapshot, subject: AAAAA has been randomized on 11AUG2014. However, we do have another record for this subject with different randomized date (which is by mistake).

DATA: ENROLMENT



| subjid | randflag | randdate |
|---|---|---|
| AAAAA | Y | 2014-08-11 |
| AAAAA | Y | 2014-08-12 |
| BBBBB | Y | 2014-08-13 |

DUPLIACATE OPTION : By using this option, we can determine, whether to

1. Ignore duplicate keys or
2. Consider the last occurance of the records : DUPLICATE: 'R' or
3. Through an error message in the log if there are duplicate keys DUPLICATE: 'E'

```
If _N_ = 1 Then Do;

 Declare Hash lbhash;

lbhash = _new_ hash(HashExp:8,Dataset:'trans' ,duplicate:'e');

lbhash.DefineKey('subjid');

lbhash.DefineData('randdate');

lbhash.DefineDone();

Call Missing(subjid, randdate);

End;
```

```
ERROR in log:
ERROR: Duplicate key found when loading data set trans at line 61
column 1.
ERROR: Hash data set load failed at line 61 column 1.
ERROR: DATA STEP Component Object failure.  Aborted during the
EXECUTION phase.
```

MULTIDATA OPTION :
Sometimes, we do need to have both the records for the above subject i.e. "AAAAA", if there are 2 periods in a clinical trail and randomization happens for both the periods. For such instances, we need to check whether the subject has randomized for both the periods of the study.

By using this option, we can load all the records of a key variable (including the duplicates) in to the HASH lookup.

```
If _N_ = 1 Then Do;
 Declare Hash lbhash;

lbhash = _new_ hash(HashExp:8,Dataset:'trans' ,multidata:'y');

lbhash.DefineKey('subjid');

lbhash.DefineData('randdate');

lbhash.DefineDone();

Call Missing(subjid, randdate);

End;
```

Advantages over regular data step merging:

1. The hash object has a great advantage of speed and so it is very helpful while dealing with bulk volume of data. Typically, hash objects do the programming tasks faster than traditional data step or SQL programming (depending on the data, the available memory, your environment, etc.).

2. Typing a series of IF-THEN statements or using merge statement requires more time and requires more steps (like sort).There is no need of further steps to be written such as SORT,FORMAT or dataset indexes.

3. Compared to PROC FORMAT, it returns numeric and /or character results without the requirement of any data conversions, specifically, character-to-numeric whereas PROC FORMAT, only returns character results.

4. Hash objects are also very good for data summarization and can typically execute the job up to twice as fast while utilizing a third of the memory when compared with data step programming.

5. Though hash object is a type of array, it has more dynamic and is relatively faster than arrays. Sometimes, we might need to include multiple arrays or multi-dimensional arrays in our code. However, when it comes to hash objects, it grows and expands as we add or remove items from the table.

**References:**

http://www.sas.com/storefront/aux/en/sphashobjects/62230_excerpt.pdf

http://analytics.ncsu.edu/sesug/2011/BB08.Lafler.pdf

http://support.sas.com/documentation/cdl/en/lrcon/65287/HTML/default/viewer.htm#n1b4cbtmb049xtn1vh9x4waiioz4.htm

http://www.scsug.org/wp-content/uploads/2012/11/HASH-Programming-basics.pdf

**Contact Information:**

Swarnalatha G
Cytel Inc.
Hyderabad, India-500081
Email: Swarna.latha3162@gmail.com