

Re-programming a many-to-many merge with Hash Objects

CS05 PhUSE 2012 Budapest

D. J. Garbutt

17 October 2012



Why? 1

In this talk I will demonstrate how you can re-program a many-to-many merge using hash objects instead of the various other methods already known.

- to illustrate the advantages of hash object for typical SAS programming



What?

- Two worked examples in the paper and on Wiki page
 - 1 a many to many merge — CM back into AE
 - 2 a CDISC data validation — check vocabularies across domains and variables

One data step with no macro variables at all.
- In this presentation I will look at just the second example which comes from a paper at PhUSE last year by Guido Wendland. Check it out.



How and When to use a hash object

How?

Tips and syntax details — See the paper.
lots of further reading

When?

Hashes can be imported into any data step and can be used in many instances where macro variables would be normally be needed.

Hashes are *not* just for look up of large tables



Up Next

- 1 Introduction and History
- 2 Check a CDISC Domain for invalid values
 - The problem
 - The data
 - The programming
 - Adding other variables
 - Adding other datasets
- 3 SAS programming after the hash object
 - SAS Memory model
 - Hash Object myths
 - Issues with SAS program development
 - Solved!
 - Questions



Hash arrays and Content Addressable Arrays

1977

AWK and Snobol4 in 1977

Currently in KornShell 93, bash, Ruby, PowerShell, .NET, Java, R, etc.,

Dorfman showed hashing could be programmed with a data step in the 90's but it is not simple and only worth the effort for very large datasets.

Now since 2008 (9.2) SAS has built-in support for the hash object.



AWK program 1

```
xargs file |
awk '{
    $1=NULL;
    t[$0]++;
}
END {
    for (i in t) printf("%d\t%s\n", t[i], i) }
' | sort -nr
```



How does the hash array look?

```
name["Dave"] + 1 ;  
index: [Dave] [dave] [Greg] [Urs] [n/a]  
address: 2334   504       31   230   42  
Count:    3       1       4     2   33
```

An Array: is a storage area *and* a rule :
Allocate area of contiguous memory then
to access it calculate storage location from the index.



How does it work? 1

normal array rule: take the memory base address and add $index * (bytes - in - integer)$

hash array rule: take the memory base address, now run `index = "character string"` through hash function to get an integer, add it to base address.



How does it work? 2

A hash function is a bit like the MD-5 key or random number generator — it does bit operations on the characters, in this case to get an integer.



How does it work? 3

A hash array is not a list data structure

It has no links between members and cannot grow indefinitely. And insertion / deletion operations take *constant time* $O(1)$.



Check variables for correct values

- Some SDTM variables (data elements) have fixed content
 - so we would like to check and make sure they conform.

Ingredients

- 1 Data domain e.g. AE
- 2 list of variables to check with the name of their code-list
- 3 SDTM controlled data list
<http://evs.nci.nih.gov/ftp1/CDISC/SDTM/SDTM/Terminology.odm.xml>



Solution with macro

http://www.lexjansen.com/phuse/2011/cs/CS02_ppt.ppt

Generate a data step fragment like

```
data violations (keep=variable error) ;  
    
  set ae ;  
    
  if aeacn not in  
    ('DOSE INCREASED','DOSE NOT CHANGED',  
     'DOSE REDUCED',...)  
  then do;  
    variable = 'AEACN' ;  
    error = cats('value of AEACN'  
                 ,aeacn  
                 , ' not found in dictionary'  
                 ) ;  
    
  output ;  
end;  
/* next test follows ...*/
```



The problem

To get this code

Transpose the code list
then substitute it into an `in()` test

Repeat for each variable.

Thousands of lines of code for SAS to interpret and execute



The data flow

VIEWTABLE: Work.Codelist

	Codelist Code	Codelist Name	CDISC Submission Value	CDISC Synonym(s)	CT
1	C66767	Action Taken with Study Treatment	DOSE INCREASED		An wa ch am
2	C66767	Action Taken with Study Treatment	DOSE NOT CHANGED		An wa An

VIEWTABLE: Work.Codelist_md

	Codelist Code	Codelist Extensible (Yes/No)	Codelist Name	CDISC Submission Value	CDISC Synonym(s)
1	C66767	No	Action Taken with Study Treatment	ACN	Action Taken with Study Treatment
2	C66768	No	Outcome of Event		
3	C66780	Yes	Age Span		
4	C66781	No	Age Unit		
5	C66797	No	Category for Inclusion/Exclusion		
6	C66784	No	Common Terminology Criteria Events		

VIEWTABLE: Work.Maplist

	ruleID	variable	Codelist
1	CT0001	AEACN	ACN
2	CT0002	AESEV	AESEV
3	CT0009	DOMAIN	DOMAIN
4	CT0027	AEOUT	OUT
5	CT0037	AEBODSYS	SOC
6	CT0064	AESER	NY

The goal is to check values 1

```
data violations (keep=variable error) ;
```

```
  [redacted]
```

```
set ae ;
```

```
  [redacted]
```



The goal is to check values 2

```
if aeacn not in('DOSE INCREASED','DOSE NOT CHANGED',
               'DOSE REDUCED',...)
then do;
    variable = 'AEACN' ;
    error = cats('value of AEACN'
                ,aeacn
                , ' not found in dictionary'
                ) ;
    
    output ;
end;
```



The goal is to check values 3

```
run;          /* next test follows ...*/
```



The goal is to check values 4

Could we do a fast lookup of the right code?

We would put all the vocabulary in one hash object
and

use the `check()` method to test for presence



Something like this: 1

```
notfound = codelistst.check(  
    key: CODELIST_CODE,  
    Key: AEACN  
);
```



Something like this: 2

```
if notfound then do;
  variable = 'AEACN' ;
  error = cats('Value of AEACN'
              ,aeacn
              , ' not found in dictionary'
              );
  
  output ;
/* next variable... */
```



Something like this: 3

Notes: no quotes round AECN
We need the code lists....



NCI Code lists

Code	Code List Code	Codelist Name	CDISC Subm
C66767		Action Taken with Study Treatment	ACN
C49503	C66767	Action Taken with Study Treatment	DOSE INCR
C49504	C66767	Action Taken with Study Treatment	DOSE NOT
C49505	C66767	Action Taken with Study Treatment	DOSE REDU
C49501	C66767	Action Taken with Study Treatment	DRUG INTE
C49502	C66767	Action Taken with Study Treatment	DRUG WITH
C48660	C66767	Action Taken with Study Treatment	NOT APPLI

Split into two datasets

read the XML file

```
filename SDTMCT
    'H:\garbutt\DATA\Projects\CS05-djg\SDTM_
filename SXLEMAP
    'H:\garbutt\DATA\Projects\CS05-djg\CDISC
libname SDTMCT xml
    xmlmap=SXLEMAP
    access=READONLY;
```



manipulate the datasets

```
/* the map defines two tables */  
data codeList_M;  
    SET SDTMCT.CodeList_M;  
run;  
data codelist;  
    set sdtmct.codelist;  
    codelist_name= scan(oid,3,'.') ;  
run;
```



Read the control dataset

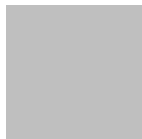
maplist data as used by Wendland

```
data cs051.maplist ;
    length ruleID $6 variable $8
           Codelist $411 ;
input ruleid @10 variable @20 codelist $10. ;
datalines ;
CT0001      AEACN          ACN
CT0002      AESEV          AESEV
CT0009      DOMAIN        DOMAIN
CT0027      AEOUT          OUT
CT0037      AEBODSYS       SOC
CT0064      AESER          NY
;
```

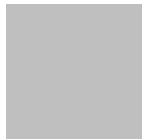


Setup the hash objects; Lookup the data

```
varleft = vars_to_check.first();
```



```
rc = codelist_md.find();
```



```
notfound = codelist.check(  
    key: CODELIST_CODE,  
    Key: AEACN );
```



Program flow and PDV contents

Source	Dataset :AE	Maplist	Maplist	Maplist	Codelist	Codelist	PDV
			Codelist_md		Codelist_md		
Variable	AEACN	RuleID	Variable	Codelist	Codelist_code	codelist_value	RC
Sample Value	Treatment	CT00001	AEACN	ACN	C66797	INCREASED	0
set ae ;	Treatment						.
variable = 'AEACN';	Treatment		AEACN				.
rc = maplist.find();	Treatment	CT00001	AEACN	ACN			0
rc = codelist_md.find();	Treatment	CT00001	AEACN	ACN	C66797		0
codelist_value= vvaluex(variable);	Treatment	CT00001	AEACN	ACN	C66797	Treatment	0
rc = codelist.check();	Treatment	CT00001	AEACN	ACN	C66797	Treatment	2365

Iterating through the datasets

add hash iterator to the maplist hash object

```
/* Maplist matches variable and codelist (num id) */  
declare hash maplist(dataset: "cs051.maplist"  
    , hashexp:4  
    , ordered: 'yes');  
rc + maplist.DEFINEKEY('VARIABLE') ;  
rc + maplist.DEFINEDATA('VARIABLE','RULEID','CODELIST');  
rc + maplist.DEFINEDONE();  
call missing(VARIABLE, ruleid, codelist) ;  
if rc then  
    putlog 'ERRO'  
        'R: problem creating hash obj maplist' ;  
  
/* define iterator for maplist to enable looping */  
declare hiter vars_to_check('maplist') ;
```



Iterating with `first()` and `next()`

```
varleft = vars_to_check.first();  
do while (varleft = 0 ) ;  
  RC = 0;  
  notfound = 1 ;  
    
  call missing(ruleid,  
               ,codelist  
               ,codelist_value  
               ,codelist_code  
               ) ;  
  varleft = vars_to_check.next() ;  
end;
```





Looping across datasets 1

Compile the data step and generate with call execute (1 of 4 ways)

```
        *--- compile program ---- ;  
data violations /PGM=work.violations ;  
  
run;  
        *--- and generate list of members  
                in vmembers from maplist--;
```



Looping across datasets 2

Compile the data step and generate with call execute (1 of 4 ways)

```
data _null_ ;  
  set vmembers ;  
  call execute (  
    'data PGM=work.violations ;' ) ;  
  call execute('execute ;') ;
```



Looping across datasets 3

Compile the data step and generate with call execute (1 of 4 ways)

```
call execute(  
  catt('redirect INPUT AE='  
    , memname  
    , ');'  
  ) ;
```



Looping across datasets 4

Compile the data step and generate with call execute (1 of 4 ways)

```
call execute(  
    catt('redirect output  violations =v  
        , memname  
        ,'; run;'  
    )  
);  
run;
```



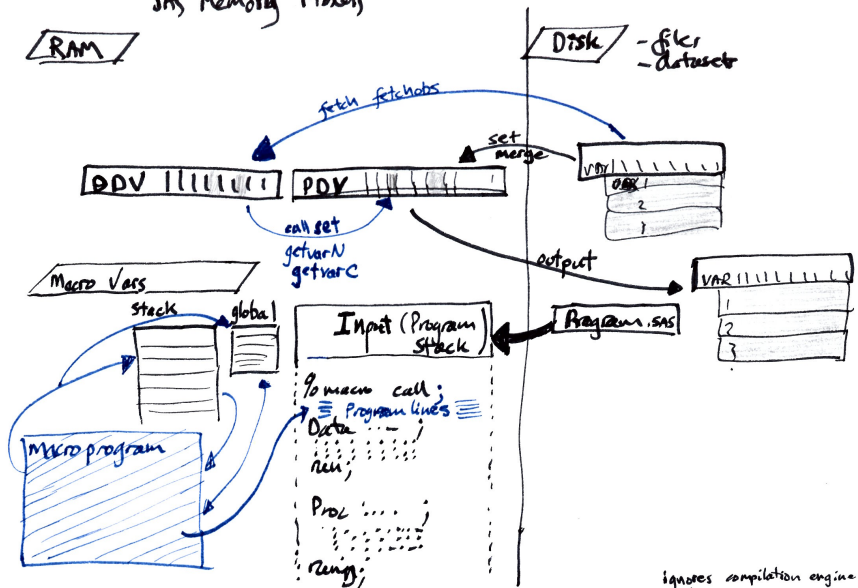
Full code is

- about 120 lines
 - 50 lines of hash object declaration
- not a *single* macro variable
- headed for the wiki...

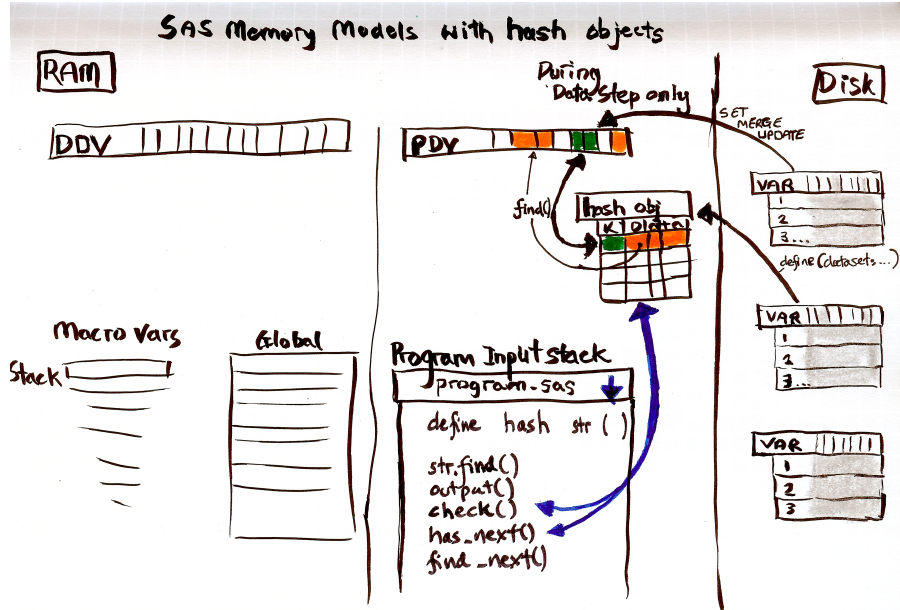


SAS memory model — Till now

SAS Memory Models



SAS memory model — After hash objects



8 Hash Object myths dispelled 1

- 1 They are just for lookup with millions of rows —
I don't need that syntax mess!
- 2 Useless — you can't have duplicate keys!
_____→multidata: 'yes'
- 3 If keys *are* duplicated you get no warning!
_____→duplicate: 'error'



8 Hash Object myths dispelled 2

- ④ You can't sort them!
—————→ordered: 'yes'
- ⑤ You'll run out of memory!
—————→4 GB is 5.3 million 800 byte records
- ⑥ You have to build the hash with add(), delete()
etc. Tedious!
—————→dataset: ' work.dave (where myths > 9)'



8 Hash Object myths dispelled 3

- ⑦ Can't be used outside a data step!
————→output() method will write to dataset
- ⑧ Merge, Proc SQL, set key=, and formats work fine for merging
————→hash is faster and advantage increases with size of data



8 Hash Object myths dispelled 4

9

10

Syntax is verbose and opaque

11

Can't be used with procs, macros, the DDV or
proc IML



Fundamental Issues with SAS program development 1

Longstanding issues since Version 6 and earlier

Why is developing large scale systems so hard with SAS? My list of reasons is:

- 1 separation of procs and data steps from each other by the disk transfer wall



Fundamental Issues with SAS program development 2

Longstanding issues since Version 6 and earlier

- 2 macro is purely a text generator (cf Korn Shell, Perl, Ruby)
- 3 macro variables are simple, individual structures. Other structures have to be emulated by naming conventions and collision avoidance. Or not.



Fundamental Issues with SAS program development 3

Longstanding issues since Version 6 and earlier

- 4 in data steps only scalars are allowed (ie single named elements of PDV). Arrays are just list of variable names, not data structures!
- 5 lack of truly encapsulated functions or subroutines (goto label: – are you kidding me?).



Fundamental Issues with SAS program development 4

Longstanding issues since Version 6 and earlier

- 6 lack of an `exec()` function that can execute code in place and return the result



Solved!

Issues with SAS program development 1

— solved!

In Version 9.2 SAS programming has jumped into the present and the previous issues addressed:

Hash Object adds matrix objects to data step, with
fast search, size independent performance
and flexible data types



Solved!

Issues with SAS program development 2

— solved!

Proc FCMP allows functions to encapsulate user code and for it to be used as normal SAS function. It supports arrays and call by reference.

run_macro, dosub & dosubl allow data steps and procs to be called from a data step and pass results back via macro variables



Solved!

Issues with SAS program development 3

— solved!

J Secosky, Executing a PROC from a DATA step,
SAS GF 2012,
[http://support.sas.com/resources/papers/
proceedings12/227-2012.pdf](http://support.sas.com/resources/papers/proceedings12/227-2012.pdf)



Solved!

The hope for meta data

Meta data programming and meta data systems are the next wave of change



The hash object opens new doors

- shorter code
- easy to debug (can use data step debugger)
- easier to maintain

For designing and building systems, especially based on metadata, these three new features will make an enormous difference to SAS programming and how it is done.

It is the 21st century, SAS programming is fun again
datasets and data steps can drive processing without
macro



Solved!

And the next doors:

- please SAS:
 - 1 Allow hash objects to be passed to proc FCMP functions
 - 2 Allow hash objects in open code and
 - 3 to persist outside data steps
 - 4 Allow procs to be passed hash objects as if datasets
 - 5 Allow procs to write hash objects as output



Questions

?

david@garbuttconsult.ch



References, further reading

PhUSE Wiki

http://www.phusewiki.org/wiki/index.php?title=Reprogramming_a_many-to-many_merge_with_Hash_Objects

Where you can also add code, examples, and discussions.

NOTE:

Watch out for papers written before V9.2

- They may now be misleading
- duplicates are allowed,
- they may also have workarounds that are no longer needed.

