

*An Antidote for Huge Clinical Trial Data Processing:
Enclosing Hash in FCMP*

Premanand Kumaravel

Symbiance Inc

Email: pkumaravel@symbiance.com

Overview

- SAS® Component Objects: Hash and Hash Iterator objects.
- SAS® PROC FCMP (Function Compiler Procedure).
- How Hashing is integrated in FCMP with examples.
- Use of RUN_MACRO function within FCMP to execute SAS macros based on HASH with example.
- How above methods can be utilized to tackle large datasets and what will be the scope for using this method.
- Advantages and disadvantages
- References.

Hash and Hash Iterator Objects:

- Hash and hash iterator objects of SAS® enables swift and effective searching, retrieval and storing data based on lookup keys created.
- Its major functionality is in-memory processing with lookup keys which can be character or numeric or composite.
- To define a Hash Object there are four major steps:

A. Declaring the Hash Object.

B. Defining the key variables.

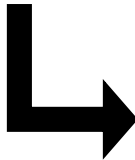
C. Defining the data variables.

D. Completing the definition.



```
declare hash mrg(dataset:'sdtn_t.dm',ordered:'Y',hashexp:6);  
rc1 = mrg.definekey('usubjid');  
rc2 = mrg.definedata('usubjid','rfstdtc');  
rc3 = mrg.definedone();
```

- Hash Iterator Object allows retrieving hash object data in either ascending or descending order.
- To define Hiter Object it is declared with a hiter referencing it to the respective hash object.



```
declare hash smh(suminc:"n", ordered:"y");  
declare hiter htit("smh");
```

PROC FCMP (Function Compiler Procedure)

- FCMP (Function Compiler Procedure) of SAS® enables programmer to create user defined SAS functions which can be used in SAS DATA steps or some procedures.
- Once function created from this procedure is compiled from their storage location it can be reused in any of the SAS DATA steps or procedures which has access to its location.

```
proc fcmp outlib = work.functions.conversions;
    function bmi(wt,ht);
        return ((wt)/(ht*ht));
    endsub;
run;

/** user defined function with subroutine **/
proc fcmp outlib = sdtm_f.func.trial;
    subroutine daycal(rfst $,dt $,dy);
        outargs dy;
        if not missing(rfst) and length(rfst) > 10 then rfstdt = input(substr(strip(rfst),1,10),E8601DA10.);
        else if not missing(rfst) and length(rfst) = 10 then rfstdt = input(strip(rfst),E8601DA10.);

        if not missing(dt) and length(dt) > 10 then idt = input(substr(strip(dt),1,10),E8601DA10.);
        else if not missing(dt) and length(dt) = 10 then idt = input(strip(dt),E8601DA10.);

        if nmmiss(rfstdt,idt) = 0 then do;
            if idt >= rfstdt then dy = (idt - rfstdt)+1;
            else if idt < rfstdt then dy = idt - rfstdt;
        end;
    endsub;
run;
```

```
options cmplib = (work.functions);
```

```
data bm_cal;
    set vs_trp(keep = usubjid weight height);
    bmsin = bmi(weight,height);
run;

options cmplib = (sdtm_f.func);
```

```
data ae_dy;
    merge ae_dcal(in = a) dm_rfs(in = b);
    by usubjid;
    if a;
    aestdy = .;
    aeendy = .;
    call daycal(rfstdtc,aestdte,aestdy);
    call daycal(rfstdtc,aeendtc,aeendy);
run;
```

Hashing in FCMP:

- In SAS 9.3, Hash and Hash Iterator Objects are available in FCMP procedure.
- Within FCMP some of the statements and methods supported for Hash Object Hash Iterator Objects are listed below.

Supported Hash Objects statements and methods:

DECLARE statement	To declare a hash object by creating instance and initializing it
DEFINEDATA method	Dataset specified to be sorted is defined
DEFINEKEY method	Key variables are defined for the hash object
DEFINEDONE method	Its specified when all data definitions and keys are complete
ADD method	Specified data of the key is added to the hash object
DELETE method	Associated hash or hash iterator object gets deleted
CHECK method	Specified key is checked in hash object if stored
FIND method	Specific key is determined if stored in hash object
CLEAR method	Without deleting hash object instance, all items from it is cleared
REMOVE method	For the specified key, associated data are removed from the hash object
REPLACE method	For the specified key, associated data gets replaced to new data

Supported Hash Iterator Objects statements and methods :

DECLARE statement	To declare a hash iterator object by creating instance and initializing it
FIRST method	First value is returned for the primary hash object
LAST method	Last value is returned for the primary hash object
NEXT method	Next value is returned for the primary hash object
PREV method	Previous value is returned for the primary hash object

Hashing in FCMP: (Example1)

```

    /** Defining HASH within PROC FCMP to create SAS User Defined Function */
    /** User Defined function to populate First Dose Date */

```

```

%macro idts();
    %let ids = &ip;
    %let odats = &op;
    data _null_;
        if 0 then set &ids (keep = _all_);
        declare hash srt (dataset:"&ids", ordered:"&ascdsc", hashexp:12);
        srt.definekey (&invar);
        srt.definedata (all:"Yes");
        srt.definedone ();
        srt.output(dataset:"&odats");
        stop;
    run;
%mend idts;

```

```

proc fcmp outlib = work.sfn.bd_srt;
    function srtids(ids $,odats $);
        rc = run_macro('idts',ids,odats);
        return(rc);
    endsub;

run;

```

```

proc fcmp outlib=work.sfn.fnd_fval;
    function fnd_fval(subjid $) $40;
        declare hash fnd(dataset: "work.exdose"); /** Declaring Hash 'fnd' */
        declare hiter fst("fnd"); /** Declaring Hash Iterator 'fst' */
        rc=fnd.definedata("exstdtc"); /** Defining variable to store */
        rc=fnd.definekey("subjid"); /** Defining key variable */
        rc=fnd.definedone(); /** Completion of defining Hash 'fnd' */
        rc = fst.first();
        rc=fnd.find();
        if rc = 0 then return(exstdtc);
        else return('NO FIRST DOSE DATE'); endsub;
quit;

```

```

    /** Compiling User Defined Function from its Respective Library */

```

```

options cmplib = (work.sfn);

```

```

    /** Calling User Defined Function within Datastep */

```

```

%let invar = "SUBJID", "EXSEQ", "EXSTDTC", "EXENDTC";

```

```

%let ascdsc = A;

```

```

%let ip = exdose_us; %let op = exdose;

```

```

data _null_; ids = "&ip"; odats = "&op"; rc = srtids(ids,odats); run;

```

```

data dm_ex; set demog; EXSTDTC = fnd_fval(subjid); run;

```

Sample datasets Demog and exdose

	SUBJID	SITEID	AGE	SEX	RACE	COUNTRY
1	1202-1001	1202	8 F	WHITE	BEL	
2	1202-1002	1202	5 M	WHITE	BEL	
3	1202-1003	1202	10 F	WHITE	BEL	
4	1204-1001	1204	5 F	WHITE	BEL	
5	1501-1001	1501	11 M		FRA	
6	1501-1002	1501	8 F		FRA	
7	1501-1003	1501	4 M		FRA	
8	1502-1001	1502	6 F	BLACK OR AFRICAN AMERICAN	FRA	
9	1503-1001	1503	6 M		FRA	
10	1503-1002	1503	9 M		FRA	
11	1505-1001	1505	10 M		FRA	
12	1506-1001	1506	7 F		FRA	
13	1506-1002	1506	4 F		FRA	
14	1801-1001	1801	10 M	WHITE	HUN	
15	1801-1002	1801	7 M	WHITE	HUN	
16	1801-1003	1801	8 M	WHITE	HUN	

	SUBJID	EXSEQ	EXDOSE	EXSTDTC	EXENDTC
1	1202-1001	1	4	2017-10-02	2017-10-08
2	1202-1001	2	8	2017-10-09	2017-10-15
3	1202-1001	3	12	2017-10-16	2017-11-02
4	1202-1001	4	8	2017-11-03	2017-11-26
5	1202-1001	5	8	2017-11-27	2017-12-17
6	1202-1002	1	4	2017-10-13	2017-10-19
7	1202-1002	2	8	2017-10-20	2017-10-26
8	1202-1002	3	12	2017-10-27	2017-11-09
9	1202-1002	4	16	2017-11-10	2017-11-23
10	1202-1002	5	16	2017-11-24	2017-12-07
11	1202-1003	1	4	2017-10-13	2017-10-19
12	1202-1003	2	8	2017-10-20	2017-10-27
13	1202-1003	3	12	2017-10-28	2017-11-10
14	1202-1003	4	16	2017-11-11	2017-11-19
15	1202-1003	5	16	2017-11-20	2017-11-26
16	1202-1003	6	20	2017-11-27	2017-12-07

Output dataset DM_EX

	SUBJID	SITEID	AGE	SEX	RACE	COUNTRY	EXSTDTC
1	1202-1001	1202	8 F	WHITE	BEL		2017-10-02
2	1202-1002	1202	5 M	WHITE	BEL		2017-10-13
3	1202-1003	1202	10 F	WHITE	BEL		2017-10-13
4	1204-1001	1204	5 F	WHITE	BEL		2017-09-05
5	1501-1001	1501	11 M		FRA		2017-09-06
6	1501-1002	1501	8 F		FRA		2017-09-06
7	1501-1003	1501	4 M		FRA		2017-10-06
8	1502-1001	1502	6 F	BLACK OR AFRICAN AMERICAN	FRA		2017-10-13
9	1503-1001	1503	6 M		FRA		2017-09-25
10	1503-1002	1503	9 M		FRA		2017-11-03
11	1505-1001	1505	10 M		FRA		NO FIRST DOSE DATE

Hashing in FCMP: (Example2)

```
proc fcmp outlib=work.sfn.fnd_lval;
  function fnd_lval(subjid $) $40;
  declare hash fnd(dataset: "work.exdose");
  declare hiter lst("fnd");
  rc=fnd.definedata("exendtc");
  rc=fnd.definekey("subjid");
  rc=fnd.definedone();
  rc = lst.first();
  rc=fnd.find();
  if rc = 0 then return(exendtc);
  else return('NO LAST DOSE DATE'); endsub;
quit;

options cmplib = (work.sfn);

%let ascdsc = D; /** Macro parameter value for descending order **/
%let ip = exdose_us; %let op = exdose;

data _null_; ids = "&ip"; odats = "&op"; rc = srtids(ids,odats); run;

data dm_ex; set dm_ex; RFENDTC = fnd_oval(subjid); run;
```

Sample datasets Demog and exdose

	SUBID	SITEID	AGE	SEX	RACE	COUNTRY
1	1202-1001	1202	8 F	WHITE		BEL
2	1202-1002	1202	5 M	WHITE		BEL
3	1202-1003	1202	10 F	WHITE		BEL
4	1204-1001	1204	5 F	WHITE		BEL
5	1501-1001	1501	11 M			FRA
6	1501-1002	1501	8 F			FRA
7	1501-1003	1501	4 M			FRA
8	1502-1001	1502	6 F	BLACK OR AFRICAN AMERICAN		FRA
9	1503-1001	1503	6 M			FRA
10	1503-1002	1503	9 M			FRA
11	1505-1001	1505	10 M			FRA
12	1506-1001	1506	7 F			FRA
13	1506-1002	1506	4 F			FRA
14	1801-1001	1801	10 M	WHITE		HUN
15	1801-1002	1801	7 M	WHITE		HUN
16	1801-1003	1801	8 M	WHITE		HUN

	SUBID	EXSEQ	EXDOSE	EXSTDTC	EXENDTC
1	1202-1001	1	4	2017-10-02	2017-10-08
2	1202-1001	2	8	2017-10-09	2017-10-15
3	1202-1001	3	12	2017-10-16	2017-11-02
4	1202-1001	4	8	2017-11-03	2017-11-26
5	1202-1001	5	8	2017-11-27	2017-12-17
6	1202-1002	1	4	2017-10-13	2017-10-19
7	1202-1002	2	8	2017-10-20	2017-10-26
8	1202-1002	3	12	2017-10-27	2017-11-09
9	1202-1002	4	16	2017-11-10	2017-11-23
10	1202-1002	5	16	2017-11-24	2017-12-07
11	1202-1003	1	4	2017-10-13	2017-10-19
12	1202-1003	2	8	2017-10-20	2017-10-27
13	1202-1003	3	12	2017-10-28	2017-11-10
14	1202-1003	4	16	2017-11-11	2017-11-19
15	1202-1003	5	16	2017-11-20	2017-11-26
16	1202-1003	6	20	2017-11-27	2017-12-07

Output dataset DM_EX with respective lookup values for EXENDTC based on SUBJID as key

	SUBJID	SITEID	AGE	SEX	RACE	COUNTRY	EXSTDTC	EXENDTC
1	1202-1001	1202	8 F	WHITE		BEL	2017-10-02	2017-12-17
2	1202-1002	1202	5 M	WHITE		BEL	2017-10-13	2017-12-07
3	1202-1003	1202	10 F	WHITE		BEL	2017-10-13	2017-12-07
4	1204-1001	1204	5 F	WHITE		BEL	2017-09-05	2018-01-25
5	1501-1001	1501	11 M			FRA	2017-09-06	2018-01-25
6	1501-1002	1501	8 F			FRA	2017-09-05	2017-12-21
7	1501-1003	1501	4 M			FRA	2017-10-06	2017-12-11
8	1502-1001	1502	6 F	BLACK OR AFRICAN AMERICAN		FRA	2017-10-13	2017-12-05
9	1503-1001	1503	6 M			FRA	2017-09-25	2017-10-16
10	1503-1002	1503	9 M			FRA	2017-11-03	2017-12-05
11	1505-1001	1505	10 M			FRA	NO FIRST DOSE DATE	NO LAST DOSE DATE

Hashing in FCMP via RUN_MACRO:

- Within PROC FCMP by using RUN_MACRO routine, provides a means to call the macro within this procedure. RUN_MACRO function executes the macro and returns a value to DATA step.
- To make Hash and Hash Iterator object customised within FCMP, RUN_MACRO function is used.
- When a user defined function of this kind executed, the function creates local macro variable with identical name and value of the variables in the parameter list from RUN_MACRO.
- Following it macro gets executes and each macro variable values are written back to equivalent variables in the function. The return code (rc) variable specifies whether the method was success or failure.

Syntax of RUN_MACRO:

rc = RUN_MACRO('macro_name' <, variable_1, ..., variable_n>);

Macro_name – name of the macro to be executed. Variables – specifies optional PROC FCMP variables

Hashing in FCMP via RUN_MACRO: (Example3)

```

/** Creating User Defined Function for splitting a dataset based on given category variable distinct values */
/** Defining Macro with HASH Objects */
%macro data_splt;
  /** Input parameters */
  %let ip = &inprt;
  %let asrt = &keyvar;
  %let pfx = &prfxval;

  proc sort data = &inpds out = &ip;
    by &asrt;
  run;

  data _null_;
    if 0 then set &ip;
    declare hash splt(ordered:'Y', hashexp:12);
    splt.definekey(&asrtvar);
    splt.definetime(&keepvar);
    splt.definedone();
    do _n_ = 1 by 1 until(last.&asrt);
      set &ip;
      by &asrt;
      splt.add();
    end;
    /** Saving Hash Object Data into multiple datasets */
    splt.output (dataset: compress("&pfx"||&asrt));

  run;
%mend;

```

M
a
c
r
o

D
e
f
i
n
i
t
i
o
n

```

/** User Defined Function with RUN_MACRO routine */
%proc fcmp outlib = work.sfn.split_ds;
  /** Defining Macro with RUN_MACRO routine */
  function split_ds(ip $, asrt $, pfx $);
    /** RUN_MACRO function with macro name and macro variable parameters */
    rc = run_macro('data_splt', ip, asrt, pfx);
    return(rc);
  endsub;
run;

/** Assigning Macro parameter variable values */
%let inpds = work.lab;
%let inprt = labs;
%let asrtvar = "subjid", "lbseq", "n_n";
%let keepvar = "subjid", "lbseq", "lbtested", "lbtest", "lborres", "lborresu", "lbstat";
%let keyvar = lbcats;
%let prfxval = CAT_;

/** Compiling User Defined Function from its Respective Library */
options cmplib = (work.sfn);

data _null_;
  ip = "&inprt";
  asrt = "&keyvar";
  pfx = "&prfxval";
  rc = split_ds(ip, asrt, pfx); /** Calling Function */
run;

```

Sample dataset LAB

	SUBID	LBSEQ	LBTESTCD	LBTEST	LBCAT	LBORRES	LBORRESU	LBSTAT
1	1202-1001	1	LBC1AMP	Laboratory Test Sample	URINALYSIS			NOT DON
2	1202-1001	2	HCG	Choriongonadotropin Beta	URINALYSIS			NOT DON
3	1202-1001	3	HCG	Choriongonadotropin Beta	URINALYSIS			NOT DON
4	1202-1001	4	HCG	Choriongonadotropin Beta	URINALYSIS			NOT DON
5	1202-1001	5	HCG	Choriongonadotropin Beta	URINALYSIS			NOT DON
6	1202-1001	6	CL	Chloride	CHEMISTRY	100	mEq/L	
7	1202-1001	7	CL	Chloride	CHEMISTRY	100	mEq/L	
8	1202-1001	8	CL	Chloride	CHEMISTRY	100	mEq/L	
9	1202-1001	9	CL	Chloride	CHEMISTRY	100	mEq/L	
10	1202-1001	10	K	Potassium	CHEMISTRY	3.3	mEq/L	
11	1202-1001	11	K	Potassium	CHEMISTRY	3.3	mEq/L	
12	1202-1001	12	K	Potassium	CHEMISTRY	3.3	mEq/L	
13	1202-1001	13	K	Potassium	CHEMISTRY	3.3	mEq/L	
14	1202-1001	14	SODIUM	Sodium	CHEMISTRY	143	mEq/L	
15	1202-1001	15	SODIUM	Sodium	CHEMISTRY	143	mEq/L	
16	1202-1001	16	SODIUM	Sodium	CHEMISTRY	143	mEq/L	
17	1202-1001	17	SODIUM	Sodium	CHEMISTRY	143	mEq/L	
18	1202-1001	18	ALP	Alkaline Phosphatase	CHEMISTRY	184	U/L	
19	1202-1001	19	ALP	Alkaline Phosphatase	CHEMISTRY	184	U/L	
20	1202-1001	20	ALP	Alkaline Phosphatase	CHEMISTRY	184	U/L	
21	1202-1001	21	ALP	Alkaline Phosphatase	CHEMISTRY	184	U/L	
22	1202-1001	22	ALT	Alanine Aminotransferase	CHEMISTRY	16	U/L	
23	1202-1001	23	ALT	Alanine Aminotransferase	CHEMISTRY	16	U/L	
24	1202-1001	24	ALT	Alanine Aminotransferase	CHEMISTRY	16	U/L	
25	1202-1001	25	ALT	Alanine Aminotransferase	CHEMISTRY	16	U/L	
26	1202-1001	26	AST	Aspartate Aminotransferase	CHEMISTRY	18	U/L	
27	1202-1001	27	AST	Aspartate Aminotransferase	CHEMISTRY	18	U/L	

Output datasets split based on LBCAT

Explorer		
Contents of 'Work'		
Name	Size	Type
Cat_chemistry	2.5MB	Tab
Cat_hematology	829.0KB	Tab
Cat_urinalysis	829.0KB	Tab
Cat_urinedrugscreen	49.0KB	Tab
Lab	4.7MB	Tab
Labs	4.7MB	Tab
Sasmacr	5.0KB	Cat
Sfn	49.0KB	Tab

Scope for using Hashing in FCMP :

- Hash objects is much preferred to be used with larger datasets with lesser variables. Since its highly efficient with such type of datasets.
- It also works for datasets with extremely large data and many variables yet the efficiency which we get won't be like larger datasets with lesser variables.
- when a hash object is integrated into a user defined function via FCMP procedure, it drastically shortens the use of conventional steps and procedures instead a simple function will be called.
- As hash object entries are placed in system memory, major benefit is increase in efficiency which directly reduces system resource and processing time taken.
- If the given memory is not sufficient as data corresponding to the hash object loaded is extremely large, SAS will stop executing DATA step and error message will be written log concerning memory failure. Hence system allocated memory also to be taken into consideration.

Advantages versus Disadvantages:

ADVANTAGES:

1. Simplification and streamlining of programmer's code.
2. Multiple DATA steps or procedures can be integrated into FCMP procedure.
3. SAS® Hash and Hash Iterator objects readily available to be used within FCMP procedure.
4. Easy to store and compile functions from desired location.
5. Can also be used within a macro by using %SYSFUNC or %SYSCALL.
6. Functions created based on hash objects can handle large volume of data by reducing processing time and system memory used.
7. User defined functions created by integrating macros via RUN_MACRO routine allows the function to operate more flexible and dynamic.

DISADVANTAGES:

1. When Hash and Hash Iterator used complexity of the process increases.
2. System memory might cause issues if insufficient memory is allocated for hash object processing within a user defined function.
3. Not a game changer in processing speed for small or moderate sized data when utilizing hash object-based functions.
4. Hashing in FCMP can't be used for all scenarios as like macros hence it is based on necessity.

References:

1. SAS Institute Inc. 2012. Base SAS® 9.3 Procedures Guide, Second Edition. Cary, NC: SAS Institute Inc.
2. SAS Institute Inc. 2011. SAS® 9.3 Component Objects: Reference. Cary, NC: SAS Institute Inc.
3. PROC FCMP and DATA Step Component Objects:
<http://documentation.sas.com/?docsetId=proc&docsetVersion=9.4&docsetTarget=n03uc8c8fkguxqn1i5iapv1auqrz.htm&locale=en>
4. Qing Liu (2017), “Streamline Table Lookup by Embedding HASH in FCMP”, Proceedings of PharmaSUG China 2017: <https://www.pharmasug.org/proceedings/china2017/AD/PharmaSUG-China-2017-AD02.pdf>
5. Paul M. Dorfman, Don Henderson (2017), “Beyond Table Lookup: The Versatile SAS® Hash Object”, Proceedings of the SAS Global 2017: <http://support.sas.com/resources/papers/proceedings17/0821-2017.pdf>
6. Schacherer, Chris (2015), “Introduction to SAS® Hash Objects”, Proceedings of the SAS Global 2015: <https://support.sas.com/resources/papers/proceedings15/3024-2015.pdf>
7. Qixia Shi, Guanyu, su (2016), “Custom useful SAS functions using the PROC FCMP procedure”, Proceedings of PharmaSUG China 2016: <https://www.lexjansen.com/pharmasug-cn/2016/PS/PharmaSUG-China-2016-PS06.pdf>
8. Richann Watson, Lynn Mullins (2016), “Exploring HASH Tables vs. SORT/DATA Step vs. PROC SQL”, Proceedings of PharmaSUG 2016: <https://www.pharmasug.org/proceedings/2016/TT/PharmaSUG-2016-TT11.pdf>
9. Dylan Ellis, Mathematica Policy Research, Washington, DC, (2013), “RUN_MACRO Run! With PROC FCMP and the RUN_MACRO Function from SAS® 9.2, Your SAS® Programs Are All Grown Up”, Proceedings of the SAS Global Forum 2013: <https://support.sas.com/resources/papers/proceedings13/033-2013.pdf>
10. Paul M. Dorfman, Lessia S. Shajenko, Koen Vyverman (2008), “Hash Crash and Beyond”, Proceedings of the SAS Global Forum 2008: <http://www2.sas.com/proceedings/forum2008/037-2008.pdf>

THANK YOU!!

QUERIES?