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.

- It 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.

- 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.

```plaintext
declare hash mrg(dataset:'sdtm_t.dm', ordered:'Y', hashexp:6);
  rcl = mrg.definekey('usubjid');
  rci = mrg.definedata('usubjid','rfsdsc');
  rc2 = mrg.definedone();
```

```plaintext
declare hash smh(suminc: "n", ordered: "y");
display 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.

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

/* user defined function with subroutine **/
proc fcmp outlib = sdm_f.func.trial;
    subroutine daycal(rfstdt,ad,dy);
        outargs dy;
        if not missing(rfstdt) and length(rfstdt) > 10 then rfstdt = input(strip(rfstdt),1,10,38601DA10.);
        else if not missing(rfstdt) and length(rfstdt) = 10 then rfstdt = input(strip(rfstdt),38601DA10.);
        if not missing(dt) and length(dt) > 10 then idc = input(strip(dt),1,10,38601DA10.);
        else if not missing(dt) and length(dt) = 10 then idc = input(strip(dt),38601DA10.);
        if not missing(idc) and length(idc) > 10 then idc = input(strip(idc),38601DA10.);
        else if not missing(idc) and length(idc) = 10 then idc = input(strip(idc),38601DA10.);
        if not missing(idc) and length(idc) = 9 then do;
            if idc > rfstdt then dy = (idc - rfstdt-1);
            else if idc < rfstdt then dy = idc - rfstdt;
        end;
    endsub;
run;
```

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

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

options cmplib = (sdm_f.func);

data ae_dy;
    merge ae_dcal(in = a) dm_rfs(in = b);
    by usubjid;
    if a;
        aestdy = .;
        aeendy = .;
        call daycal(rfstdtc,aestdttc,aestdy);
        call daycall(rfstdttc,aendttc,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:

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

Supported Hash Iterator Objects statements and methods:

<table>
<thead>
<tr>
<th>DECLARE statement</th>
<th>To declare a hash iterator object by creating instance and initializing it</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST method</td>
<td>First value is returned for the primary hash object</td>
</tr>
<tr>
<td>LAST method</td>
<td>Last value is returned for the primary hash object</td>
</tr>
<tr>
<td>NEXT method</td>
<td>Next value is returned for the primary hash object</td>
</tr>
<tr>
<td>PREV method</td>
<td>Previous value is returned for the primary hash object</td>
</tr>
</tbody>
</table>
Hashing in FCMP: (Example1)

```sas
### Defining HASH within PROC FCMP to create SAS User Defined Function ###
### User Defined function to populate First Dose Date ###
%macro idts();
  %let id = &ip;
  %let odates = &op;
  data_null;
  if 0 then set &ids (keep = _all_);
  declare hash srt (dataset=&id, ordered="#aadsc", hashexp=12);
  srt.definekey (#invar);
  srt.definedata (all="Yes");
  srt.definedone ();
  srt.output(dataset="#odats");
  stop;
run;
%mend idts;

proc fcmp outlib = work.sftrbd.srt;
function srtds(ids $,odats $);
  rc = run_macro('idts',ids,odats);
  return(rc);
endsub;
run;

proc fcmp outlib = work.sftrbd.fnd_fval;
function fnd_fval(subjid $) $40;
  declare hash fnd(dataset="work.exdose"); /* Declaring Hash ‘fnd’ */
  declare hitter fst("fnd"); /* Declaring Hash Iterator ‘fst’ */
  rc=fnd.definedata("extstdtc"); /* 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(extstdtc);
  else return('NO FIRST DOSE DATE');
endsub;
quit;
/** Compiling User Defined Function from its Respective Library **/
options cmplib = (work.sftrbd);
/** Calling User Defined Function within Datasep ****/
%let invar = "SUBJID","EXSEQ","EXTSTDTC","EXENSTDTC";
%let asdsc = A;
%let ip = exdose_us; %let op = exdose;

data_null ; id = "&ip"; odates = "&op"; rc = srtds(ida,odats); run;
data dm_ex; set demog; EXSTDTC = fnd_fval(subjid); run;
```

Sample datasets Demog and exdose

Output dataset DM_EX
Hashing in FCMP: (Example2)

Sample datasets Demog and exdose

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

```sas
/* *** Creating User Defined Function for splitting a dataset based on given category variable distinct values ***/
/* *** Defining Macro with HASH Objects ***/
%macro data_split;
  /* ** Input parameters ***/
  %let ip = aimpert;
  %let aact = akeyvar;
  %let prfx = apfrval;
  %let nsplit = nsplitval;
  %let ncat = ncatval;
  proc sort data = inputdata out = %syssummer;
    by aact;
    run;
  data _null_; 
    if 0 then set inputdata; 
    declare hash split(parameter='Y', hashsplit); 
    split.definekey(aimpert);
    split.defindata(akeyvar); 
    split.defindone(); 
    do _n_ = 1 by 1 until(last.aact);
      set inputdata;
      by aact;
      split.add();
    end;
  /* *** Saving Hash Object Data into multiple datasets ***/
  split.output(dataset = compress("%sysfunc(putn(ip))",aact));
  run;
%mend;

/* *** User Defined Function with RUN_MACRO routine***/
%macro user_defined_function; 
  /* ** Assigning Macro variable values ***/
  %let ip = 1; %let aact = "A"; %let prfx = "P";
  /* *** User Defined Function with macro name and macro variable parameters ***/
  %let rc = run_macro('data_split',ip,aact,prfx);
  return(rc);
%mend;

/* *** Assigning Macro parameter variable values ***/
%let impdt = work; 
%let input = labs; 
%let aact = "subject", "level","_n_"; 
%let kkeyvar = "subject","level","levelcode","lbcstat","lborres","lborresu","lbcstat"; 
%let keyvar = "lbcstat"; 
%let prfxval = CAT_;
/* *** Compiling User Defined Function from its Respective Library ***/
options cmplib = (work.sfn); 

/*data _null_; 
  %let ip = "impdt";
  %let aact = "keyvar";
  %let prfx = "pfrval";
  %let nsplit = nsplitval; /* *** Calling Function ***/
run;
```

Sample dataset LAB

Output datasets split based on LBCAT
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:


3. PROC FCMP and DATA Step Component Objects: [http://documentation.sas.com/?docsetId=proc&docsetVersion=9.4&docsetTarget=n03uc8e8fkguxqn1i5iapvlaugrz.htm&locale=en](http://documentation.sas.com/?docsetId=proc&docsetVersion=9.4&docsetTarget=n03uc8e8fkguxqn1i5iapvlaugrz.htm&locale=en)


THANK YOU!!

QUERIES?