PharmaSUG China 2017 - Paper 55

Could you make join datasets more efficient, especially many to many join?

Wei Duan, PPD, Beijing, China

ABSTRACT

Joining datasets to get variables is a very common thing in SAS programming. Usually we implemented the data join by DATASTEP MERGE and SQL join. However, they are not always efficient. Depending on whether the join key is unique, join dataset can be divided into two parts. The first part includes one to one, one to many join. The second part covers many to many join. This paper focus on using FORMAT and SET/SET in one to one and one to many join. Also provide POINT CONTROL and HASH OBJECTS methods to complete many to many join. The advantages, disadvantages and their conditions are summarized at the end of the paper.

INTRODUCTION

Almost all SAS users join datasets in their programs. A Join dataset is basically using key variables connect two datasets to map new variables from new dataset to base dataset and return join outputs. It usually uses to calculate a new value as a function. This paper shows some less common but more efficient join techniques: FORMAT/SETSET for one-one join and POINT CONTROL/ HASH for many-many join. After comparing efficiency of each method, high efficient reasons are also explored.

ONE TO ONE/ONE TO MANY JOIN

JOINING USE FORMAT

Formats are one of the most powerful and ubiquitous features in SAS. The FORMAT procedure is used to create a format from an input data set, and we can use the defined format to convert an input value to output value without joining two dataset. For some common variables or flags, if we define them and store them in libraries where everyone in our group can have access to them and prevent from repeated programs. From this point, it can simplify program and enhance efficiency. Now The FCMP procedure allows us to write our own functions and routines[1]. It is very flexible and provides us with a tool that can be applied in any number of ways.

This example creates a format converting Vital Signs test date (VSDTC) to EPOCH. Firstly, define a user format (EPOCH) in SV dataset: connect subject identification number(USUSBJID) with his visit start date(SVSTDTC) as START PARAMTER and connect subject identify number(USUBJID) with his visit end date(SVENDTC) as END PARAMTER. This defines a period from START to END for different experiment procedure (EPOCH) of each subject. Then in VS dataset, this format could be used to derive EPOCH variable through mapping subject identify number(USUBJID) and Vital Signs test date (VSDTC) information.

	USUBJID	EPOCH	SVSTDTC	SVENDTC
1	302-105-0001	FOLLOW-UP	2016-06-14	2016-11-29
2	302-105-0001	SCREENING	2016-05-17	2016-05-20
3	302-105-0001	TREATMENT	2016-05-20	2016-06-01
4	302-112-0001	FOLLOW-UP	2016-12-14	2017-02-22
5	302-112-0001	SCREENING	2016-11-17	2016-11-21
6	302-112-0001	TREATMENT	2016-11-21	2016-12-03
7	302-113-0001	FOLLOW-UP	2017-02-24	2017-03-17
8	302-113-0001	SCREENING	2016-12-09	2016-12-22
9	302-113-0001	TREATMENT	2016-12-22	2017-01-03
10	302-116-0001	FOLLOW-UP	2016-08-29	2017-02-17
11	302-116-0001	SCREENING	2016-08-01	2016-08-05
12	302-116-0001	TREATMENT	2016-08-05	2016-08-17
13	302-120-0001	SCREENING	2016-08-11	2016-08-11
14	302-120-0002	FOLLOW-UP	2017-01-16	2017-03-20
15	302-120-0002	SCREENING	2016-12-22	2016-12-22
16	302-120-0002	TREATMENT	2016-12-23	2017-01-04
17	302-301-0001	SCREENING	2017-02-08	2017-02-17
18	302-301-0001	TREATMENT	2017-02-17	2017-02-17
19	302-307-0001	FOLLOW-UP	2017-03-23	2017-03-23
20	302-307-0001	SCREENING	2017-02-22	2017-03-01
21	302-307-0001	TREATMENT	2017-03-01	2017-03-13
22	302-601-0001	SCREENING	2017-03-19	2017-03-19

Display 1 SV DATASET

	USUBJID	vsorres	vstest	vstestcd	vsdtc
1	302-105-0001	82	Diastolic Blood Pressure	DIABP	2016-05-17
2	302-105-0001	45	Pulse Rate	PULSE	2016-05-17
3	302-105-0001	137	Systolic Blood Pressure	SYSBP	2016-05-17
4	302-112-0001	89	Diastolic Blood Pressure	DIABP	2016-11-17
5	302-112-0001	78	Pulse Rate	PULSE	2016-11-17
6	302-112-0001	155	Systolic Blood Pressure	SYSBP	2016-11-17
7	302-113-0001	53	Diastolic Blood Pressure	DIABP	2016-12-22
8	302-113-0001	50	Pulse Rate	PULSE	2016-12-22
9	302-113-0001	118	Systolic Blood Pressure	SYSBP	2016-12-22
10	302-116-0001	70	Diastolic Blood Pressure	DIABP	2016-08-04
11	302-116-0001	80	Pulse Rate	PULSE	2016-08-04
12	302-116-0001	118	Systolic Blood Pressure	SYSBP	2016-08-04
13	302-120-0002	96	Diastolic Blood Pressure	DIABP	2016-12-22
14	302-120-0002	88	Pulse Rate	PULSE	2016-12-22
15	302-120-0002	128	Systolic Blood Pressure	SYSBP	2016-12-22
16	302-301-0001	78	Diastolic Blood Pressure	DIABP	2017-02-17
17	302-301-0001	63	Pulse Rate	PULSE	2017-02-17
18	302-301-0001	144	Systolic Blood Pressure	SYSBP	2017-02-17
19	302-307-0001	62	Diastolic Blood Pressure	DIABP	2017-02-28
20	302-307-0001	67	Pulse Rate	PULSE	2017-02-28
21	302-307-0001	140	Systolic Blood Pressure	SYSBP	2017-02-28
22	302-112-0001	89	Diastolic Blood Pressure	DIABP	2016-11-21

Display 2 VS DATASET

/** Create a format in SV dataset **/

```
data fmt;
  set sv end=last;
  start=compress(usubjid||'-'||svstdtc);
  end=compress(usubjid||'-'||svendtc);
  label=EPOCH;
  eexcl='Y';
  fmtname='$epoch';
  output;
  if last then do;
      call missing(label);
      hlo='o';
      output;
  end;
  keep start end label hlo fmtname;
run;
proc format cntlin=fmt;
run;
```

Doing mapping in a data step is as simple as using the PUT function:

```
data final;
  length epoch $50;
  set vs;
  EPOCH=put(compress(usubjid)||'-'||vsdtc,$epoch.);
```

```
run;
```

	epoch	USUBJID	vsorres	vstest	vstestcd	vsdtc
1	SCREENING	302-105-0001	82	Diastolic Blood Pressure	DIABP	2016-05-17
2	SCREENING	302-105-0001	45	Pulse Rate	PULSE	2016-05-17
3	SCREENING	302-105-0001	137	Systolic Blood Pressure	SYSBP	2016-05-17
4	SCREENING	302-112-0001	89	Diastolic Blood Pressure	DIABP	2016-11-17
5	SCREENING	302-112-0001	78	Pulse Rate	PULSE	2016-11-17
6	SCREENING	302-112-0001	155	Systolic Blood Pressure	SYSBP	2016-11-17
7	SCREENING	302-113-0001	53	Diastolic Blood Pressure	DIABP	2016-12-22
8	SCREENING	302-113-0001	50	Pulse Rate	PULSE	2016-12-22
9	SCREENING	302-113-0001	118	Systolic Blood Pressure	SYSBP	2016-12-22
10	SCREENING	302-116-0001	70	Diastolic Blood Pressure	DIABP	2016-08-04
11	SCREENING	302-116-0001	80	Pulse Rate	PULSE	2016-08-04
12	SCREENING	302-116-0001	118	Systolic Blood Pressure	SYSBP	2016-08-04
13	SCREENING	302-120-0002	96	Diastolic Blood Pressure	DIABP	2016-12-22
14	SCREENING	302-120-0002	88	Pulse Rate	PULSE	2016-12-22
15	SCREENING	302-120-0002	128	Systolic Blood Pressure	SYSBP	2016-12-22
16	TREATMENT	302-301-0001	78	Diastolic Blood Pressure	DIABP	2017-02-17
17	TREATMENT	302-301-0001	63	Pulse Rate	PULSE	2017-02-17
18	TREATMENT	302-301-0001	144	Systolic Blood Pressure	SYSBP	2017-02-17
19	SCREENING	302-307-0001	62	Diastolic Blood Pressure	DIABP	2017-02-28
20	SCREENING	302-307-0001	67	Pulse Rate	PULSE	2017-02-28
21	SCREENING	302-307-0001	140	Systolic Blood Pressure	SYSBP	2017-02-28
22	SCREENING	302-112-0001	89	Diastolic Blood Pressure	DIABP	2016-11-21

Display 3 Result of FORMAT

It is also possible to define format as a macro and use it to define common flag or date.Then this custom format could be use anywhere. It will simplify your program greatly especially deriving complicated flag.

```
/**create a format macro**/
%macro fmt( indsn=,start=,end= ,label= ,fmtname= ,type=C ,
missingvalue=%str(''));
   data fmt;
      set &indsn. end=last;
      start=catx('|',&start.);
      end=catx('|',&end.);
      label=&label.;
     eexcl='Y';
      fmtname="&fmtname.";
      type="&type.";
      output;
      if last then do;
         call missing(start,end);
         label=&missingvalue.;
         hlo='O';
```

```
        output;
      end;
   run;
   proc sort data=fmt dupout=dupfmt nodupkey;
      by start;
   run;
   proc format cntlin=fmt;
   run;
%mend fmt;
/**define treatment flag as a format**/
%fmt(indsn=adb.adsl,start=usubjid,end=usubjid,label=TRTFL,fmtname=TRTFL);
/**define treatment date as a format**/
%fmt(indsn=adb.adsl,start=usubjid,end=usubjid,label=trtsdt,fmtname=trtsdt
,type=I,
missingvalue=.);
```

Once defined FORMAT, we could map “USUBJID” to treatment flag or treatment date in any dataset without joining dataset. It can use to one-one mapping or many-one mapping, but can’t use to one-many. “Many ”could be a range or multiple values here.

Please note that for the same subject, period from visit start date to visit end date of different experiment procedure have no overlap possibility. This point is precondition of using FORMAT. In trial data issues of START DATE or END DATE often lead to period overlap. It could be easy found in Format log, which means using FORMAT join dataset could help us to find data issue of visit date.

JOINING USE SET SET

SET is a common command which is used to get records from one dataset or many datasets to complete end to end join. Many SET command could exist in a DATASTEP, and each SET command is assigned a point. So being different with single SET, SETSET actually is a double points command. When we use it in data steps, Comparing with single point command MERGE, double points SETSET is more efficient and don’t need to sort beforehand. SQL which is belonging to Cartesian product, executes more times and is less efficient than SETSET.Please see the following example.

A company want to calculate all staff’s bonus. Bonus = salary*factor. Factor is decided by their amount of sales every month. Dataset bonus_criteria offer criterias, if someone’s sales amount fall in intervals from low to high, he will get correspond Factor. Dataset sale_amount record everyone’s salary and sales performance in this month. We need to join two dataset to calculate staff’s bonus.

```
/**sample dataset**/
data bonus_criteria;
input  low high factor @@;
cards;
0    500   0.1
501  1000  0.2
1001 2000  0.4
2001 5000  0.6
5001 10000 0.8
```

	low	high	factor
1	0	500	0.1
2	501	1000	0.2
3	1001	2000	0.4
4	2001	5000	0.6
5	5001	10000	0.8

```
;
run;
data sale_amount;
  input name $ salary amount @@;
  cards;
Jim   500   200
Tom   1000  700
Lily  2000  1500
Saly  700   300
Gery  3500  3000
Coliy 4500  6000
Jony  5000  3000
lary  2000  300
Sam   1040  200
Jamy  5600  1200
Bob     3400  500
Salla 2600  24
;
run;
/** Use SET/SET join two datasets to derive new variable **/
proc sort data=bonus_criteria; by low high;run;
proc sort data=sale_amount; by amount;run;
data bonus;
  set sale_amount;
  do while(not(low le amount le high));
      set bonus_criteria;
  end;
  bonus=salary*factor;
run;
```

Display 4 DATASET BONUS_CRITERIA

	name	salary	amount
1	Jim	500	200
2	Tom	1000	700
3	Lily	2000	1500
4	Saly	700	300
5	Gery	3500	3000
6	Coliy	4500	6000
7	Jony	5000	3000
8	lary	2000	300
9	Sam	1040	200
10	Jamy	5600	1200
11	Bob	3400	500
12	Salla	2600	24

Display 5 DATASET SALE_AMOUNT

	name	salary	amount	low	high	factor	bonus
1	Salla	2600	24	0	500	0.1	260
2	Jim	500	200	0	500	0.1	50
3	Sam	1040	200	0	500	0.1	104
4	Saly	700	300	0	500	0.1	70
5	lary	2000	300	0	500	0.1	200
6	Bob	3400	500	0	500	0.1	340
7	Tom	1000	700	501	1000	0.2	200
8	Jamy	5600	1200	1001	2000	0.4	2240
9	Lily	2000	1500	1001	2000	0.4	800
10	Gery	3500	3000	2001	5000	0.6	2100
11	Jony	5000	3000	2001	5000	0.6	3000
12	Coliy	4500	6000	5001	10000	0.8	3600

Display 6 Result of SET SET

```
/** Use SQL join two datasets to derive new variable**/
proc sql;
```

```
  create table bonus_s as
  select name, salary, amount,low, high, factor,salary*factor as bonus
  from bonus_criteria as a
  join sale_amount as b
  on  low le amount le high
  ;
quit;
```

Comparing execute times of two different methods. SQL belongs to Cartesian product, execute 12 ×5=60 times; SET/SET use together with DO WHILE, execute 12 + 5=17 times.

To comparing run time of each method, dataset Test a and test.b are dummy with 60000000 records. Using MERGE , SQL, SETSET,FORMAT join two datasets respectively, real time and cup time are recorded. Please see screen shot below.

```
proc format;
    picture val
    low-high="99999999";
run;
data test.a(keep=usubjid saffl);
    length saffl $1;
    do i=1 to 60000000 ;
        if (i/3) >1000 then saffl="Y";
        else saffl="N";
        usubjid=put(i,val.);
        output;
    end;
run;
```

Display 7 Dummy datasets test.a

```
data test.b(keep=usubjid trtfl);
    length trtfl $1;
    do i=1 to 60000000 ;
        if (i/2) >1000 then trtfl="Y";
        else trtfl="N";
        usubjid=put(i,val.);
        output;
    end;
run;
```

Display 8 Dummy datasets test.b

```
**MREGE;

proc sort data=test.a; by usubjid;run;
proc sort data=test.b; by usubjid;run;
data merge_rel;
    merge test.a test.b;
    by usubjid;
run;
```

Display 9 Join datasets by MERGE

```
**SQL;
proc sql noprint;
    create table sql_rel as
    select a.usubjid, saffl, trtfl
    from test.a a
    inner join test.b b
    on a.usubjid=b.usubjid
    ;
quit;
```

Display 10 Join datasets by SQL

```
**SETSET;
data setset_rel;
    set test.a;
    set test.b;
run;
```

Display 11 Join datasets by SETSET

Methods	Real time (seconds)	CPU time (seconds)
MERGE	13.83*	11.71*
SQL	54.77	53.16
SETSET	10.45	6.47

Table 1 Run time and CPU time of different methods

*Note:Before using merge join dataset, always need to sort dataset. Run time of sort is longer than merge process.

MANY- MANY JOIN

Many to many means key variable of join can't identify records unique in both datasets. Most of time users consider SQL is the only solution. If users want to complete many to many join in DATASTEP, which solutions they could choose? This paper offer two solutions besides SQL. Please see example below.

In ADAE dataset, Infusion-related adverse events (AEIRR) need to derive. AEIRR defined as 'Related' if met condition(1) and condition(2).

(1) AEREL in (''Definitely Related', 'Possibly Related', 'Probably Related') in dataset AE

(2) begins either during or within 24 hours after the start of the infusion in dataset EX

Merge with EX where EXDOSE ne missing on AESTDTC <= 24 hours after EXSTDTC.

or datepart(AESTDTC)=datepart(EXSTDTC) If timepart in AESTDTC or EXSTDTC is completely missing ;or datepart(AESTDTC)=datepart(EXSTDTC) and hourpart AESTDTC <= 24 hours after EXSTDTC if minute part of AESTDTC is missing.

In this example, we need to join EX and AE, SUBJECT have many adverse events and also be injected to drug A for many times. The same subject identify number (usubjid) have many records in both datasets. SQL, POINT CONTROL , HASH methods will be use to complete many to many join respectively.

/***sample dataset ***/

	USUBJID	EXSTDTC
1	401-001-0001	2016-03-22
2	401-001-0001	2016-03-22
3	401-001-0001	2016-03-29
4	401-001-0001	2016-04-05
5	401-001-0001	2016-04-12
6	401-001-0001	2016-04-19
7	401-001-0001	2016-04-26
8	401-001-0001	2016-05-03
9	401-001-0001	2016-05-10
10	401-001-0001	2016-05-17
11	401-001-0001	2016-05-24
12	401-001-0001	2016-05-31
13	401-001-0001	2016-06-07
14	401-001-0001	2016-06-14
15	401-001-0001	2016-06-21
16	401-001-0001	2016-06-28
17	401-001-0001	2016-07-05
18	401-001-0001	2016-07-12
19	401-001-0001	2016-07-19
20	401-001-0001	2016-07-26
21	401-001-0001	2016-08-02
22	401-001-0001	2016-08-09
23	401-001-0001	2016-08-15
24	401-001-0001	2016-08-23
25	401-001-0001	2016-08-30
26	401-001-0001	2016-09-06
27	401-001-0001	2016-09-13
28	401-001-0001	2016-09-20
29	401-001-0001	2016-09-27
30	401-001-0001	2016-10-04

Display 12 EX DATASET

	USUBJID	AETERM	AESTDTC	ASTDT
1	401-001-0001	COUGH	2016-11-20T00:00	2016-11-20
2	401-001-0001	ACROCIANOSIS	2016-06-07T11:20	2016-06-07
3	401-001-0001	ACROCIANOSIS	2016-06-14T14:45	2016-06-14
4	401-001-0001	ACROCIANOSIS	2016-06-21T12:00	2016-06-21
5	401-001-0001	GASTROENTERITIS	2016-06-20	2016-06-20
6	401-001-0001	HYPERTHERMIA	2016-06-07T11:20	2016-06-07
7	401-001-0001	HYPERTHERMIA	2016-06-14T15:25	2016-06-14
8	401-001-0001	HYPERTHERMIA	2016-06-14T17:05	2016-06-14
9	401-001-0001	HYPOGLICEMIA	2016-05-31T11:05	2016-05-31
10	401-001-0001	NASAL CONGESTION	2016-11-20T00:00	2016-11-20
11	401-001-0001	ORAL THRUSH	2016-07-06T15:00	2016-07-06
12	401-001-0001	VOMIT	2016-06-21T12:00	2016-06-21
13	401-001-0001	BRONCHIAL HYPERACTIVITY / WHEEZING	2016-03-30T00:00	2016-03-30
14	401-005-0001	FEVER	2016-04-05	2016-04-05
15	401-005-0001	UPPER RESPIRATORY INFECTION	2016-01-18	2016-01-18
16	401-005-0001	UPPER RESPIRATORY INFECTION	2016-02-09	2016-02-09
17	401-005-0001	UPPER RESPIRATORY INFECTION	2016-02-23	2016-02-23
18	401-005-0001	VIRAL ILLNESS	2016-01-02	2016-01-02
19	401-005-0002	MILDLY THICKENED AORTIC VALVE	2016-11-02	2016-11-02
20	401-005-0002	CARPAL TUNNEL SYNDROME	2016-11-02	2016-11-02
21	401-005-0002	BILATERAL MILD TO MODERATE HEARING LOSS	2016-11-02	2016-11-02
22	401-005-0002	EAR INFECTION	2017-01-17	2017-01-17
23	401-005-0002	MITRAL VALVE PROLAPSE	2016-11-02	2016-11-02
24	401-005-0002	THICKENED MITRAL VALVE	2016-11-02	2016-11-02
25	401-005-0002	UNILATERAL OPTIC NERVE SWELLING	2017-01-18	2017-01-18
26	401-005-0002	PAIN SECONDARY TO SURGICAL PROCEDURE	2016-12-13	2016-12-13
27	401-005-0002	ANISOCORIA	2017-01-18	2017-01-18
28	401-005-0002	VOMITING	2016-11-30	2016-11-30
29	401-009-0002	REDNESS OVER THE NOSE	2016-05-20T14:40	2016-05-20
30	401-009-0002	PNEUMONIA	2016-06-27T09:00	2016-06-27

Display 13 AE DATASET

SQL

Most of time, users use SQL join EX and AE datasets to get necessary variables firstly. Secondly, derive new variables AEIRR in DATASTEP. Please see code below.

```
proc sql;
  create table _aeirr as
  select distinct a.usubjid, aeterm, astdt, aestdtc, exstdtc
  from adae_2 as a left join trans.EX(where=(^missing(exdose))) as b on
a.usubjid=b.usubjid;
quit;
data _aeirr;
  set _aeirr;
```

```
  _aestm = ifn(index(aestdtc,"T"),input(scan(aestdtc,2,"T"),??time5.),.);
  _aestmh =
ifn(index(aestdtc,"T"),input(scan(scan(aestdtc,2,"T"),1,":"),??best.),.);

  _exstdt = ifn(index(exstdtc,"T"),input(scan(exstdtc,1,"T"),??yymmdd10.),
input(exstdtc,??yymmdd10.));
  _exstm = ifn(index(exstdtc,"T"),input(scan(exstdtc,2,"T"),??time5.),.) ;
  _exstmh =
ifn(index(exstdtc,"T"),input(scan(scan(exstdtc,2,"T"),1,":"),??best.),.);

  if n(astdt,_exstdt)=2 then do;
      if n(_aestm,_exstm)=2 then do;
          if (astdt*24*60+_aestm) - (_exstdt*24*60+_exstm) <=24*60 then
AEIRR="Related";
      end;
      else if n(_aestm,_exstm)<2 then do;
          if astdt=_exstdt then AEIRR="Related";
      end;
      else if n(_exstmh,_aestmh)=2 then do;
          if _aestmh - _exstmh <=24 then AEIRR="Related";
      end;
  end;

  if ^missing(AEIRR);
run;
```

POINT CONTROL

In first paragraph, SETSET method is introduced. Point control is using multiple point control options associate with SETSET method. Options like point=/nobs=. To understand this procedure, we have to know data step processing in SAS. Please see picture below.

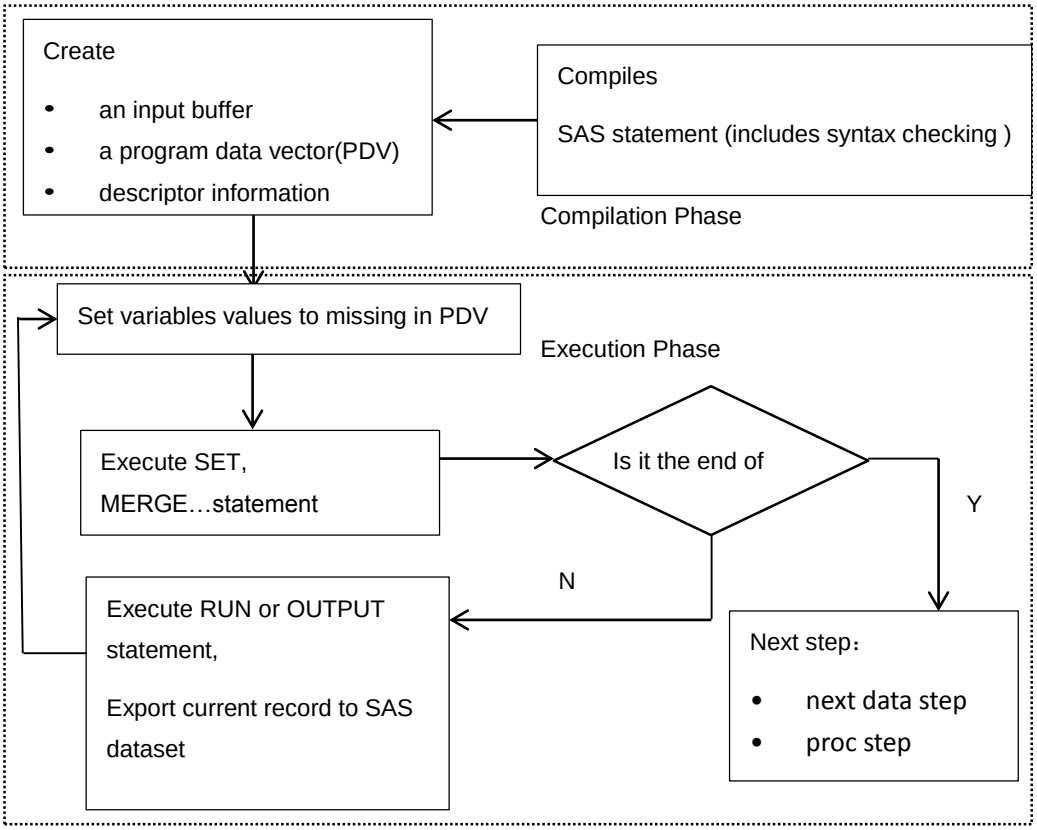

Figure 1 Data step processing[2]

SAS statement is divided into two phase: compilation phase and execution phase and compilation phase firstly run and then execution phase. Then we see the following statement:

1. In compilation phase, assign two data point to two SET command respectively. At the same time create a PDV, set all variables from both dataset to missing. Option NOBS= represent total records number of dataset AE, set total records number to temporary variable NUSUBJID. Option RENAME= rename variable name from USUBJID of AE dataset to USUBJID_AE in PDV.
2. In execution phase, data point of EX point to first record of EX, data point of AE search record from first record to last record of AE dataset because of option POINT=. If condition is met, then export a record to dataset POINT_RE. If not, then data point of EX point to second record…until to the last record of EX dataset.

```
data point_re(drop=usubjid_ae);
   set ex;
   do i=1 to nusubjid;
       set ae(rename=(usubjid=usubjid_ae)) nobs=nusubjid point=i;
       if usubjid=usubjid_ae  then output;
  end;
 run;

data point_re;
  set point_re;
_aestm=ifn(index(aestdtc,"T"),input(scan(aestdtc,2,"T"),??time5.),.);
```

```
_aestmh=ifn(index(aestdtc,"T"),input(scan(scan(aestdtc,2,"T"),1,":"),??be
st.),.);
_exstdt=ifn(index(exstdtc,"T"),input(scan(exstdtc,1,"T"),??yymmdd10.),
input(exstdtc,??yymmdd10.));
_exstm=ifn(index(exstdtc,"T"),input(scan(exstdtc,2,"T"),??time5.),.) ;
_exstmh=ifn(index(exstdtc,"T"),input(scan(scan(exstdtc,2,"T"),1,":"),??be
st.),.);

if n(astdt,_exstdt)=2 then do;
if n(_aestm,_exstm)=2 then do;
  if (astdt*24*60+_aestm) - (_exstdt*24*60+_exstm) <=24*60 then
AEIRR="Related";
  end;
  else if n(_aestm,_exstm)<2 then do;
      if astdt=_exstdt then AEIRR="Related";
  end;
  else if n(_exstmh,_aestmh)=2 then do;
            if _aestmh - _exstmh <=24 then AEIRR="Related";
  end;
end;
if ^missing(AEIRR) and exdose^=. then output;
run;
```

Efficient of point control is equal to SQL because they all belong to Cartesian product. When not use in big data, it's a good method to substitute to SQL.

HASH

When handle many to many join in big data, the hash object provides an efficient method for quick join tables of data[3]. Why hash join is obviously faster than MERGE or SQL? It is because if we define a big dataset to hash object, it will be stored in memory rather than on disk. Either write into memory or read from it, Speed is much faster than disk. The only limitation is the amount of memory available to SAS. The following code illustrates how to join two dataset by using hash object.

In a DATA STEP, We will create the hash object out of the EX dataset and join it with AE dataset.

```
data ex;
  set ex;
  _exstdt = ifn(index(exstdtc,"T"),input(scan(exstdtc,1,"T"),??yymmdd10.),
input(exstdtc,??yymmdd10.));
  _exstm = ifn(index(exstdtc,"T"),input(scan(exstdtc,2,"T"),??time5.),.) ;
  _exstmh =
ifn(index(exstdtc,"T"),input(scan(scan(exstdtc,2,"T"),1,":"),??best.),.);
run;

data adae;
  if 0 then set ex;
  set ae;
```

```
  _aestm = ifn(index(aestdtc,"T"),input(scan(aestdtc,2,"T"),??time5.),.);
  _aestmh =
ifn(index(aestdtc,"T"),input(scan(scan(aestdtc,2,"T"),1,":"),??best.),.);
/*upon the first iteration of the input dataset, declare and create the hash
table*/
  if _n_=1 then do;
/*declare the hash table*/
/*DATASET designates the source dataset for the hash table*/
      declare hash ex_h(dataset:'ex',multidata:'y');
/*define key variable of join*/
      ex_h.definekey('usubjid');
/*define other variables besides key variable of join*/
      ex_h.definedata('_exstdt','_exstm','_exstmh');
  /*define done completes the declaration of the hash table*/
      ex_h.definedone();
  end;
/*end of declare the hash table*/

/*Initialize the variables used in hash to missing values */

  call missing(_exstdt,_exstm,_exstmh);
/*whether key variable is stored in the hash object. If it is, the return code
is set to zero. If not, the return code is non-zero. */

  fl=ex_h.find();
  do while(fl=0);
      if n(astdt,_exstdt)=2 then do;
      if n(_aestm,_exstm)=2 then do;
          if (astdt*24*60+_aestm) - (_exstdt*24*60+_exstm) <=24*60 then
AEIRR="Related";
      end;
      else if n(_aestm,_exstm)<2 then do;
          if astdt=_exstdt then AEIRR="Related";
      end;
      else if n(_exstmh,_aestmh)=2 then do;
          if _aestmh - _exstmh <=24 then AEIRR="Related";
      end;
  end;
/*Set the key variable to the next item, other variables also move to the
correspond next record */
      fl=ex_h.find_next();
  end;
run;
```

Please be careful about option multidata:'y' , it enable key variable is not unique in hash table. If

multidata:'N' or don't define, there will be error in log.

To comparing run time of SQL and hash, AE and EX dataset are dummy with 300000 records with repeat key variables(USUBJID). Using, SQL, HASH join two datasets respectively, real time and cup time are recorded. Please see run time below.

Methods	Real time (seconds)	CPU time (seconds)
SQL	3:36.17	31.66
HASH	9.71	1.24

Table 2 Run time of SQL and HASH in many to many join

CONCLUSION

What is more efficient programs？Simplify program，short execute time，more choices and easy to implement. Comparing with MERGE, SQL which we usually use, FORMAT could simplify our program,

POINT CONTROL offer more choices to solve many to many join datasets. SETSET and HASH make execute time of join big data dataset more fast. In fact, Understand compilation and execution phase of SAS language could help you choose best efficient method to join dataset in future.

REFERENCES

[1] Arthur L, Carpenter. 2013.Using PROC FCMP to the Fullest: Getting Started and Doing More. SAS Global Forum 2013 Paper 139-2013. California Occidental Consultants, Anchorage, AK

[2]Yao, Zhiyong. 2016.SAS programing and data analysis business cases.page7.Beijing,China: Machinery Industry.

[3] Dari Mazloom. 2017.SAS Hash Objects, Demystified . SAS Global Forum 2017 Paper 1479-2017.USAA

CONTACT INFORMATION

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

Name: Wei Duan
Enterprise: PPD
Address: 25 F Raffles City Beijing Office Tower, No. 1 Dongzhimen South Street, Dongcheng District City, State ZIP: Beijing, China
Work Phone:
Fax:
E-mail: Wei.Duan@ppdi.com
Web:
Twitter: