SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

The Roads We Take

Let’s Hash It Out!

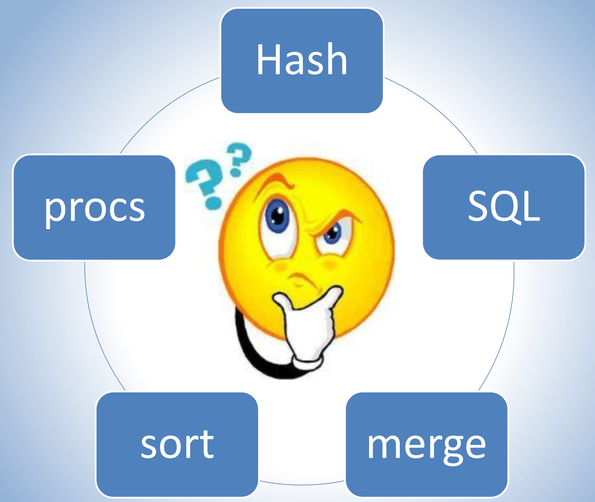
#SASGF

The Roads We Take: Let's Hash it Out!

David Izrael & Elizabeth Axelrod

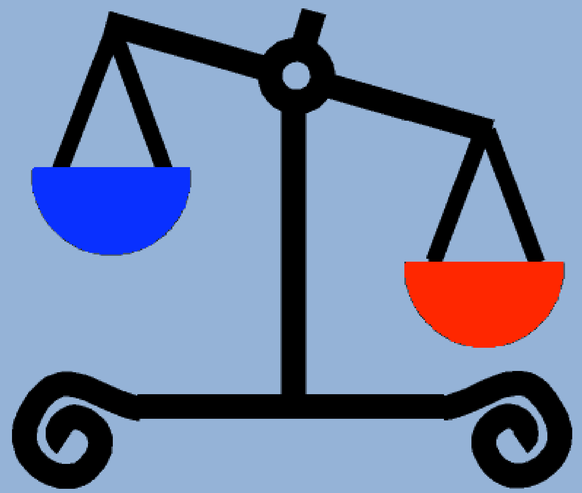
Abt Associates, Inc.

So many problems, Even more solutions!



For a class of problems where Hash can be applied, is it always the best tool to use? Here are some problems and solutions, using both Hash tables and more traditional methods. Comparing the solutions, one must consider computing time as well as *your* time to code and validate results.

How do you choose?



Which solution is the best?

The one that runs the fastest?

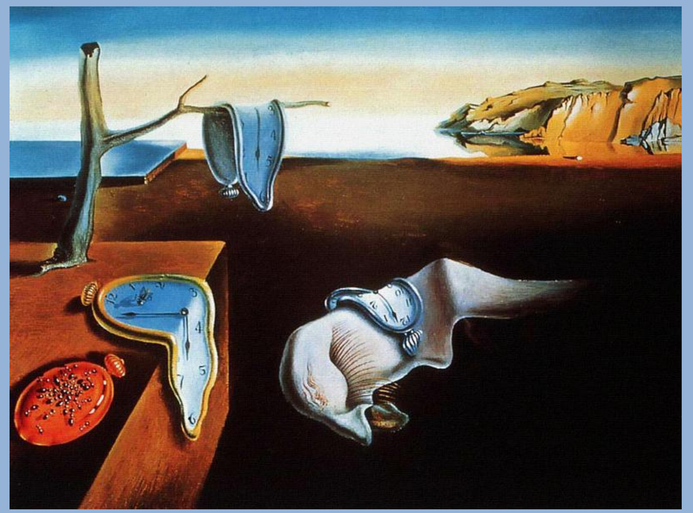
The one that's easiest to code and validate?

Or maybe….The most elegant?

You decide!

But first… a word about time…

It's about time!



How long does it take for your jobs to run?

It's a quagmire! Many factors influence this, especially if you work in a shared environment, competing for resources with other programmers and system activities.

Our jobs are run on a 64-bit windows server, using SAS 9.4

We present our times (REAL and CPU) not as formal benchmarking, but to demonstrate the relative performance of our solutions.

The Roads We Take: Let's Hash it Out!

David Izrael & Elizabeth Axelrod

Abt Associates, Inc.

To use MERGE we first need to SORT

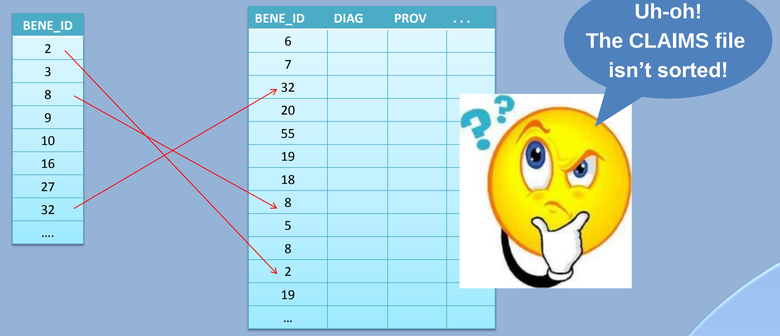
```
proc sort data=claims out=clm_sort;
     by bene_id;
run;
data task1_merge;
     merge clm_sort (in=in1)
           finder (in=inf);
     by bene_id;
     if in1 & inf;
run;
```

Try a User-Defined FORMAT statement… No sorting needed

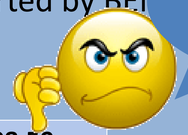
```
/* code to prepare the cntlin file */
proc format cntlin = find_fmt;
run;

data task1_format;
     set claims;
     if put(bene_id,$keeper)='KEEP';
run;
```

Task 1 – Simple Lookup

Finder file (n=3m) Claims File (n=32m)



Our task is to pull records from the claims file that match to our finder file on BENE_ID.
But notice that the claims file isn't sorted by BE



SORT	2:08.59
MERGE	0:39.60
User-defined format	0.05.98
FORMAT / PUT	0.47.90
PROC SQL	0.47.55
HASH	0.42.80

PROC SQL
Another method where no sorting is needed

```
proc sql;
    create table task1_sql as
        select *
        from  finder          as f,
              inpat_claims as i
        where f.bene_id = i.bene_id
        ;
quit;
```

HASH solution
No sorting needed and done in a single datastep

```
data task1_hash;
   if _n_=0 then set finder;
   if _n_=1 then do;
        declare hash f (dataset:"finder");
        f.definekey('bene_id');
        f.definedone();
        call missing(bene_id);
   end;
   set claims;
   rc = f.find();
   if rc=0;
run;
```

The Roads We Take: Let's Hash it Out!

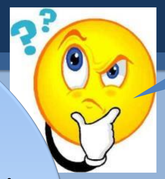
David Izrael & Elizabeth Axelrod

Abt Associates, Inc.

To use MERGE we must sort the Claims file 3 times…

There's gotta be a better way!!

```
proc sort data=claims out=clm_sort;
     by bene_id;
run;
data task1_m;
     merge clm_sort (in=in1)
           finder (in=inf);
     by bene_id;
     if in1 & inf;
run;
```



Oh noooo! I'll have to sort the Claims file 3 times!

Task 2 – Multi-Table Lookup

Finder file (n=3m)

BENE_ID
2
1
8
9
10
16
27
32
….

Claims File (n=32m)

BENE_ID	DIAG	PROV	…
6	AAA	1234	
7	BBB	7659	
32	XXX	1111	
20	ZZZ	2222	
55	123	5555	
19	456	5555	
18	879	9999	
8	304	1234	
5	CD4	2345	
8	X1G	6473	
2	456	7342	
19	703	5354	
…	…	…	

Diags

DIAG
XXX
YYY
ZZZ
123
456
965
39C
X1G
…

Provider IDs

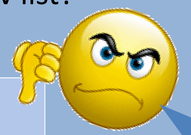
PROV
2222
2345
6666
9999
4567
1546
1234
6473
…

As before, we want to pull records from the claims file that match to our finder file on BENE_ID.
And for the selected records, we'll create two new vars:
Is the DIAG code from the claim in the Diag list?
Is the PROV ID from the claim in the Prov list?

PROC SQL

```
proc sql _method;
   create table task2_sql as
      select i.*,
          case
             when (i.prim_dx in(select
                   prim_dx from diag_list))
                then 1 else 0
                end as dx_found,
          case
             when (i.provider in(select
                   provider from prov_list))
                then 1 else 0
                end as prov_found,
         f.date1,
         f.date2
      from inpat_claims i, finder f
      where i.bene_id = f.bene_id;
quit;
```

SORT/MERGE	X.XX.XX
SORT/MERGE	X.XX.XX
SORT/MERGE	X.XX.XX
PROC SQL	1:54.94
HASH	0:50.04



No Way! Sort a file 3 times? I won't even bother trying this method.

HASH solution
All done in a single datastep!

```
data task2_hash;
    if _n_=0 then do;
       set finder;set diag_list;set prov_list;
    end;
    if _n_=1 then do;
        declare hash f (dataset: "finder");
        f.definekey('bene_id');
        f.definedata('date1','date2');
        f.definedone();
        declare hash d (dataset: "diag_list");
        d.definekey('prim_dx');
        d.definedone();
        declare hash p (dataset: "prov_list");
        p.definekey('provider');
        p.definedone()
        declare
    end;
    set inpat_cl
    rc_ben = f.
    if rc_b
    rc_dx = d.
    found_dx=(rc
    rc_prov = p.
    found_pro
run;
```

For multi-table lookups, we *really* like HASH!

The Roads We Take: Let's Hash it Out!

David Izrael & Elizabeth Axelrod

Abt Associates, Inc.

Solution using a 2-step Macro

```
%macro split;
   proc sql noprint ;
      select distinct state into: statelst
        separated by ' '
        from drugs (keep = state);
   quit;

   data &statelst;
      set drugs (keep=state main_cat weight
                      daysupply dose);
         %do i=1 %to &sqlobs;
           %let curst=%scan(&statelst, &i,%str( ));
           %if &i = 1 %then
               if state = "&curst" then
                   output &curst;
           %else
               else if state = "&curst" then
                   output &curst;
         %end;
   run;
%mend;
%split;
```

Task 3 – Splitting a SAS Dataset

State	Var1	Var2	Var3
MI			
MI			

State	Var1	Var2	Var3
GA			
GA			

State	Var1	Var2	Var3
MA			
MA			
MA			

State	Var1	Var2	Var3
MI			
MA			
MA			
MI			
GA			
MA			
GA			

Here we want to split a SAS Dataset set by state, writing out a SAS file for each value of *state*. We do not know beforehand what states are in the data sets, nor do we know if the data set is sorted by *state*.

Macro solution	**9:05.17 (Real)** **2:31.74 (CPU)**
Hash of Hashes	**7:13.04 (Real)** **3:05.93 (CPU)**

Solution using Hash of Hashes!

```
data _null_ ;
   dcl hash states (ordered: 'a');
   dcl hiter his('states');
   states.definekey('state');
   states.definedata('state', 'states_inst');
   states.definedone();

   dcl hash states_inst (); **will have all states;

   do _n_ = 1 by 1 until ( eof ) ;
      set drugs (keep = state main_cat
                        weight daysupply dose)
                end = eof ;
      if states.find () ne 0 then do ;
          states_inst = _new_ hash (ordered: 'a');
          states_inst.definekey ('state','_n_');
          states_inst.definedata ('state','main_cat',
                                  'weight','daysupply','dose');
          states_inst.definedone();
          states.replace();
      end;
      states_inst.replace();
   end;
   rc = his.first();

   do while (rc=0);
      states_inst.output (dataset: 'out'|| state );
      rc = his.next();
   end;
   stop;
run;
```

The Roads We Take: Let's Hash it Out!

David Izrael & Elizabeth Axelrod

Abt Associates, Inc.

Conclusion

Is hash always the best solution? For simple single-table lookups, hash was pretty equivalent to some of other methods we tried. For multi-table lookups hash was the winner. But in all lookup situations, the authors prefer methods that don't require pre-sorting, and we particularly like using hash solutions for their elegance and ease of coding.

Besides striving for gain in the data processing, one should consider the programmer's time required to develop, debug, and test the program. If there is no great performance advantage to one method over another, the deciding factor should probably be how skilled you are with the method you choose. You should be able to write the code, validate your results, and understand what's really going on.

Acknowledgements

The authors wish to thank Jack Shoemaker and Paul Grant for providing us with valuable input and technical guidance.

References

1. Dorfman, Paul. 2001. "Table Look-Up by Direct Addressing: Key-Indexing -- Bitmapping -- Hashing." *Proceedings of the Twenty-sixth SAS Users Group International Meeting.*
2. Paul M Dorfman, Koen Vyverman. 2005. "Data Step Hash Objects as Programming Tools." *Proceedings of the SAS Global Forum 2005 Conference.*
3. Eberhardt, Peter. 2011. "The SAS® Hash Object: It's Time to .find() Your Way Around." *Proceedings of the 2011 SAS Global Forum Conference.*
4. Loren, Judy. 2008. "*How* Do I Love Hash Tables? Let Me Count The Ways!" *Proceedings of the 2008 SAS Global Forum Conference.*

Contact

David Izrael
Abt Associates
E-Mail: david_izrael@abtassoc.com

Elizabeth Axelrod
Abt Associates
E-mail: elizabeth_axelrod@abtassoc.com

SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

LAS VEGAS | APRIL 18-21

#SASGF