Hash Objects – Why Bother?

Barb Crowther
SAS Technical Training Specialist

Purpose

- The purpose of this presentation is not to teach you how to program Hash Objects
  - That’s a two hour topic in PRG3.
- The purpose of this presentation is to show you how other people are using them.
- And once you know how powerful they can be, you’ll start looking at how to use them in your environment..

In PRG3….

- We introduce Hash Objects primarily as a table lookup technique like Arrays and User Formats.
- Whereas the array has to have a series of consecutive integers to point to elements, hash objects can use any character and numeric data.

In PRG3 (cont’d)

- And whereas the array can only return one value, the hash object can return multiple items
- And another bonus is the data does not have to be sorted for a hash object

In PRG3 (cont'd)

- BUT the code looks STRANGE and scary

```
data supplier_info;
    length Supplier_Name $40 Supplier_Address $ 45 Country $ 2;
    if _N_=1 then do;
      declare hash S(dataset:'orion.supplier');
      S.definekey('Supplier_ID');
      S.definedata(‘Supplier_Name’,’Supplier_Address’,
        'Country‘);
      S.definedone();
      call missing(Supplier_name,Supplier_Address,
         Country);
    end;
```

What can be done with Hash Objects?

- *DefineKey, DefineData, DefineDone*, and *Delete* define and delete hash objects
- *Add*, *Replace*, and *Remove* manipulate the data
- *Find* and *Check* determine if specified data exists
- *Output* writes the data values to a SAS dataset
- *Num_Items* and *Sum* allow summarization of the data

But…

- There is a lot more to hash objects than just table lookups.
- They can become a programming technique in their own right….
  - Summary-less summary
  - Chained look-ups
  - Replicate PROC SurveySelect
- So maybe it's worth the effort to learn a new programming technique

And…

- Many times, the Hash Object can do these programming tasks faster than traditional SAS programming or SQL techniques (see Note 1)

Note 1: **Depends** on your data, your environment, and what you are trying to do so you have to benchmark….

Summary-less Summarization

- Hash-Crash and Beyond; Paul Dortman et al;
  - http://www2.sas.com/proceedings/forum2008/037-2008.pdf
  - Reminder: using NWAY specifies that the output data set contain only statistics for the observations with the highest _TYPE_ value.

Summary-less Summarization

- Paul has 11 different ways to program with Hash Objects including table lookups and sorting data.
- One of the advanced techniques he programs is how to use Hash Objects to split a SAS dataset input into multiple files DYNAMICALLY

```
Data out1 out2..outX;
If id = 1 then output out1;
Else if id = 2 then output out2;

                    . . .

Else if id = x then output outX;
```

Summary-less Summarization

- Another technique was to replace a summarization in the DATA step.
- Compared required computer resources of PROC SUMMARY with NWAY option to Hash Object

```
proc summary data = input nway ;
class k1 k2 ; var num ;
output out = summ_sum (drop = _:) sum =
sum ;
```

- The Hash Object did “the job more than twice as fast at the same time utilizing ⅓ the memory”

Chained Look-ups

- How Do I Love Hash Tables? Let Me Count The Ways! Judy Loren
  - http://www2.sas.com/proceedings/forum2008/029-2008.pdf
- Judy shows six Hash Object programs

Chained Look-ups

- One of my favorites is how to merge 3 tables that do not have a common variable and without sorting the data
  - If you did it using traditional SAS programming techniques, you had to SORT, SORT, MERGE, SORT, SORT, MERGE
  - You could also use SQL to code the join which is much simpler but SQL is more likely to run into resource issues
  - So this gives a 3rd option

Chained Look-ups

- Another favourite shows how to handle scenarios where somebody's ID changed over time but we still wanted to process their data as one person.
  - Could be the someone loses their health or credit cards but you want to maintain their history…

Chained Lookups (cont'd)

Members

Member ID	Plan ID	Group ID
164-234	XYZ	G123
297-123	ABC	G123
344-123	JKL	G456
395-123	XYZ	G123
495-987	ABC	G456
562-987	ABC	G123
697-12	XYZ	G456

Old->New conversion

Old Id	New ID
164-234	N164-234
297-123	N297-123
344-123	C344-123
N164-234	M164-234
N297-123	B297-123
M164-234	P164-234
P164-234	A164-234

Replicate SURVEYSELECT

- Better Hashing in SAS9.2; Robert Ray and Jason Secosky
  - http://support.sas.com/rnd/base/datastep/dot/better-hashing-sas92.pdf
- Select observations from a table without replacement.
- Used a Hash Object that selected songs from a playlist with no repetition AND added another twist: time constraint. Not only were there no repeats, the total duration was fixed

Remove the top and bottom 10% of data values

- Need to get rid of the data extremes of each by-group for data analysis.
  - http://support.sas.com/kb/25/990.html
- Could easily be changed to write the top and bottom rows to separate datasets

But .....

- Isn’t *fill in your preferred technique* faster? Will my job run as fast as before?

Hash Object Performance

- I cut my processing time by 90% using hash tables - You can do it too!; Jennifer K. Warner-Freeman
  - **http://www.nesug.info/Proceedings/nesug07/bb/bb16.pdf**
- Jennifer looked at different ways to merge tables.
- "In my own experience I took a process … that was taking between 2 and 4 hours (depending on network traffic) to run using a PROC SQL join, and using hash tables cut the execution time to a consistent 11 minutes."

My experiment

- I created some test data: “Patient” and “Drug” and identified some generic manipulation:
  - Merge the tables together
  - If there were no problems with the Drug data, write the row out to the ‘Good’ table; else write it out to the ‘Bad’ Table.
  - So not only was I doing a table look-up, I was creating two different tables

Sample Code

```
proc sort data=patientgroup;  by patientid;
proc sort data=patientdata;   by patientid;
run;
data good bad  ;
  merge patientgroup(in=a) patientdata(in=b);
  by patientid;
  if (missing(drug) or missing(dose)) then
   output bad;
  else output good;
 run;
```

3 Techniques & 3 Data Volumes

- Used data step merge, SQL join, and Hash Object programming
- Number of rows:

Patient Data	Drug Data
50	18
950,000	9,200
6,250,000	9,200

The results (based on 5 runs per technique per data volume)

	Data Step Merge		Hash Object		SQL Join	
# Row	Clock	CPU	Clock	CPU	Clock	CPU
50	0.50	0.24	0.49	0.25	0.52	0.21
1M	**11.18**	2.94	5.77	3.34	6.55	4.66
6M	96.02	23.01	106.04	26.48	**265.45**	46.48

So, what do I think?

- On my laptop (dual processor, 3 GB RAM) I have to check out using Hash Objects as a programming technique.

But…

Different scenarios, different conclusions

- Another programmer compared (1) Formats, (2) hash objects, (3) merges, (4) key=, and (5) SQL joins and concluded that Hash Objects were good up to 1.9M rows and SQL performed very well up to 10M rows as a table lookup
- Scalability of Table Lookup Techniques, Rick Langston
  - http://support.sas.com/resources/papers/proceedings09/037-2009.pdf

Different scenarios, different conclusions

- I was using Hash Objects for more than just table look-ups – I was programming with them….

Another performance advantage

- When we use Hash Objects to replace PROC steps our programming is in **a data step**. One of the implications is we can do further data manipulation without having to read the data again.

Conclusion

- Hash Objects have a lot to offer – and there are lots of examples out there on how they can improve our programs:
  - http://support.sas.com/events/sasglobalforum/previous/online.html
  - http://support.sas.com/notes/index.html
- In the PRG3 class notes you can see examples of:
  - Chapter 7.2: Create a Cartesian product
  - Chapter 7.5: Conditionally combining tables

Questions?

Cool SAS 9.2 Tidbit

- SAS 9.2 now supports colon lists that allow you to list out all the data sets. So, if I had 5 datasets named PD1, PD2 … PD5, I could read them all in with:

```
Data pd_all;
  set pd:;
Run;
```