



# Scalability of Table Lookup Techniques

Rick Langston, SAS Institute Inc.

# Outline

- 5 different table lookup techniques shown
- Their performance is compared as we scale up
- Recommendations from that test scenario
- Macro for computing format memory usage
- More info on PROC FORMAT memory usage

# Table Lookup Techniques

- Many different ways available
- Common question: which is fastest?
- We wanted to test scalability
- 1K, 10K, 100K, 500K, 1M, 5M, 10M obs
- 5 different methods chosen...

## Using a format

```
/*-----PROC FORMAT and PUT function-----*/  
proc format cntlin=temp; run;  
data _null_; set temp(rename=(label=shouldbe)) end=eof;  
  label=put(start,$testfmt.);  
  if label=shouldbe then matched+1;  
  if eof;  
  if matched=_n_ then put 'all matched';  
  else put 'not all matched';  
run;
```

## Hash Object

```
/*-----hash object-----*/
data _null_; set temp(rename=(label=shouldbe)) end=eof;

    length label $20;
    retain label ' ';

    if _n_=1 then do;
        declare hash ht(dataset:"temp");
        ht.defineKey("start");
        ht.defineData("label");
        ht.defineDone();
    end;

    rc = ht.find();
    if rc = 0 then do;
        if label=shouldbe then matched+1;
    end;
    if eof;
    if matched=_n_ then put 'all matched';
    else put 'not all matched';
run;
```

## DATA step merge

```
/*-----merge-----*/  
proc sort data=temp out=temp2(drop=random rename=(label=shouldbe)); by start;  
run;  
proc sort data=temp out=lookup(drop=random); by start;  
run;  
data _null_; merge temp2(in=want) lookup end=eof; by start;  
  if label=shouldbe then matched+1;  
  if eof;  
  if matched=_n_ then put 'all matched';  
  else put 'not all matched';  
run;
```

## KEY= option for lookup

```
/*-----key= usage-----*/  
data lookup(index=(start)); set temp(keep=start label);  
  run;  
data _null_; set temp(keep=start label rename=(label=shouldbe)) end=eof;  
  set lookup key=start;  
  if label=shouldbe then matched+1;  
  if eof;  
  if matched=_n_ then put 'all matched';  
  else put 'not all matched';  
  run;
```

## SQL inner join

```
/*-----SQL inner join-----*/  
proc sql;  
  create table merged as  
    select label,shouldbe from temp a inner join temp(rename=(label=shouldbe)) b  
      on a.start = b.start;  
quit;  
  
data _null_; set merged end=eof;  
  if label=shouldbe then matched+1;  
  if eof;  
  if matched=_n_ then put 'all matched';  
  else put 'not all matched';  
run;
```

# Operating systems used

- 64-bit Unix system
- 64-bit Windows system with 16G
- 32-bit Windows system with 2G

## On the Unix box

- Up to 500K obs – formats and hash better
- At 700K – formats ran out of memory
- At 1.9M – hash runs out of memory
- Others OK up to 10M
- KEY= the slowest
- SQL join the best

## On both Windows boxes

- All completed up to 10M
- SQL join seemed to do best
- Let's look at the numbers shall we?

UNIX	1K	10K	100K	500K	1M	5M	10M
Format execution	0.01	0.02	0.25	1.56	1.96	-	-
Hash usage	0.06	0.05	0.26	1.58	3.61	-	-
MERGE merging	0.04	0.08	0.44	1.63	4.36	21.70	44.09
SQL table creation	0.06	0.05	0.21	1.81	3.88	21.26	45.45
Key lookup	0.01	0.13	1.46	10.56	21.96	129.57	266.95
Windows 64-bit	1K	10K	100K	500K	1M	5M	10M
Format execution	0.00	0.01	0.25	1.79	4.73	30.45	72.90
Hash usage	0.35	0.03	0.28	2.15	5.03	36.34	84.76
MERGE merging	0.05	0.02	0.37	1.33	3.01	25.44	49.60
SQL table creation	0.73	0.01	0.15	0.92	1.90	13.64	32.01
Key lookup	0.01	0.10	1.46	8.92	18.14	98.60	209.01
Windows 32-bit	1K	10K	100K	500K	1M	5M	10M
Format execution	0.01	0.01	0.21	1.68	4.82	27.39	62.84
Hash usage	1.20	0.01	0.26	2.42	4.90	35.68	82.59
MERGE merging	0.10	0.03	0.20	1.47	3.33	16.66	27.49
SQL table creation	0.59	0.01	0.12	0.76	1.68	12.25	27.60
Key lookup	0.07	0.09	1.37	8.14	16.75	93.17	189.50

# Memory constraints

- Windows 32 bit, Linux 32 bit, z/OS
- All at a max of 4G due to architecture
- 2G on Windows and z/OS
- 64-bit still constrained due to ....
- hardware memory
- SAS option choices
- administration

# z/OS Constraints

- 32-bit system
- typically 64M is the max
- chargebacks convert memory to EXCPs

# SAS options

- `proc options group=memory define; run;`

# PROC FORMAT memory usage

- You want to create a set of large formats
- Or you already have these formats
- You need to know how much memory they will need
- Can they be simultaneously loaded?

# Macro for computation

- `%get_fmt_memsizes(mylib.formats);`
- Creates an output data set with the memory needs for each member
- Available via <http://support.sas.com>

## For formats not yet created

- Suppose you have 500,000 ranges/labels
- Create with the first 100 ranges/labels
- Run the macro
- Multiply the memory needed by 5000

# Remember creation vs. usage

- More memory is needed to create a format than to use it
- But every associated format is loaded in the DATA or PROC step
- This applies regardless of explicit usage



2009 Washington, DC  
22-25 March 2009