

Overview Of HASH Objects

-Swarna Latha Gaddam

Agenda

- **Introduction**
- **Syntax and Methods**
- **Implementation**
 - ✓ **Example using Lab dataset**
- **Hiter-Traversing HASH object data**
- **Comparison of Performance with Different Techniques**
- **Advantages**
- **Conclusion**

Introduction

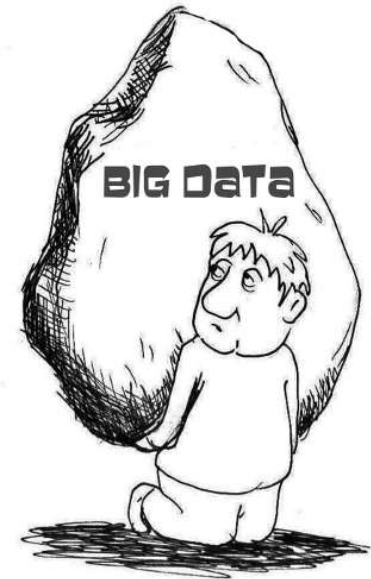
SAS
DATA step
& PROCs

- Large & Complex Data
- Data Analysis

Lengthy Coding

Long Run-time

....HASH OBJECT?



Introduction Continues.....

What is HASH object ?

A data structure that contains an array of items that are used as keys (e.g.. USUBJID) to map their associated values

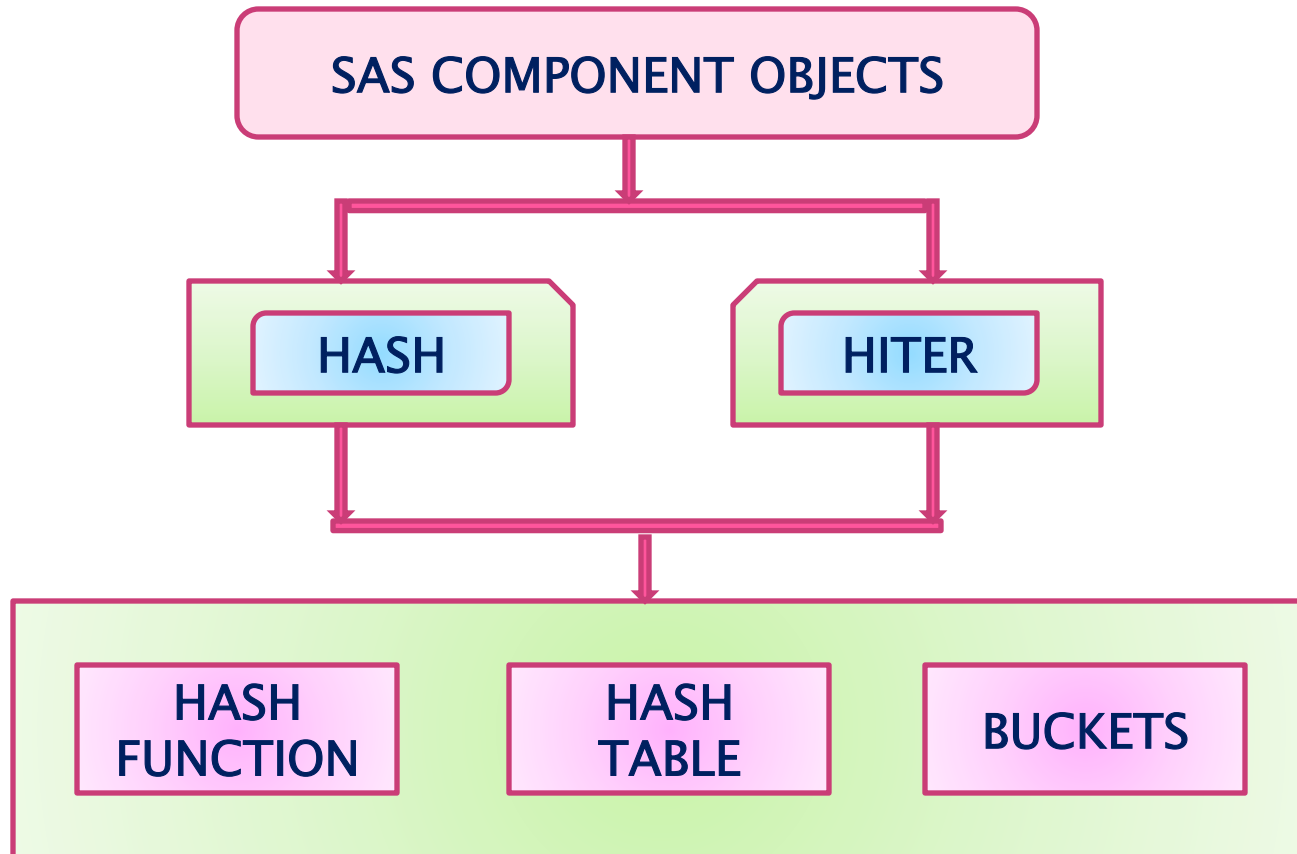
What is Hashing?

Hashing is a search algorithm based on a mathematical technique to convert a key into an array index.

Why Hashing ?

- Key lookup occurs in memory which reduces run time
- Perform the standard operations with better efficiency

Components & Elements



How To Work With HASH Objects

Declare HASH

Define HASH

Load HASH

Access/Modify HASH

Using Dot. Notation:

A = Object.Attribute

Or

Rc = Object.Method(tag1:'value1',)

```
declare hash h(argument1:'value1',...argumentn:'valuen');  
  h.definekey('key1', 'key2,...'keyn');  
  h.definedata('data1', 'data2', ...'datan');  
  h.definedone();
```

Methods & Arguments

HASH Methods

ADD	HAS_NEXT
CHECK	HAS_PREV
CLEAR	OUTPUT
DEFINEDATA	REF
DEFINEDONE	REMOVE
DEFINEKEY	REMOVEDUP
DELETE	REPLACE
EQUALS	REPLACEDUP
FIND	SUM
FIND_NEXT	SUMDUP
FIND_PREV	

HITER Methods

FIRST
LAST
NEXT
PREV
SETCUR

Arguments

DATASET
DUPLICATE
HASHEXP
ORDERED
MULTIDATA

Implementation

*Illustration by loading lab_testcd lookup
into LAB dataset*

Interaction Between Data And HASH Object

```
Data lab_hash(Drop=Rc);
If _N_ = 1 Then Do;
```

```
Length lbtest $400 lbtestcd lbcac lbscat $500;
```

```
Declare Hash lbhash(HashExp:
8, Dataset: 'lab_testcd');
lbhash.DefineKey('lbtest');
lbhash.DefineData('lbtestcd', 'lbcac', 'lbscat');
lbhash.DefineDone();
Call Missing(lbtest, lbtestcd, lbcac,
lbscat);
```

```
End;
```

```
Set lab;
Rc = lbhash.Find(Key:lbtest);
lbhash.output(dataset: 'hash_data');
If Rc = 0 Then Do;
Output lab_hash;
End;
```

```
run;
```

Creating Variables

lbtest	lbtestcd	lbcac	lbscat

Creating Hash

lbtest	lbtestcd	lbcac	lbscat

Checking the existence of HASH object and load the HASH into main dataset

LAB Dataset

⚠ SUBJID	⚠ VISITNUM	⚠ VISIT	⚠ LBTEST	123 LBORRES	⚠ LBDTC
0123	D1	DAY 1	Albumin	45	2014-02-07
0123	D1	DAY 1	Alkaline Phosphatase	84	2014-02-07
0124	D1	DAY 1	Albumin	45	2014-02-07
0124	D1	DAY 1	Alkaline Phosphatase	34	2014-02-07

LAB_TESTCD Lookup

⚠ LBTESTCD	⚠ LBTEST	⚠ LBCAT	⚠ LBSCAT
ALB	Albumin	CHEMISTRY	PROTEIN
ALP	Alkaline Phosphatase	CHEMISTRY	ENZYME,LIVER FUNCTION
BASO	Basophils	HEMATOLOGY	DIFFERENTIAL
BICARB	Bicarbonate	CHEMISTRY	ELECTROLYTE

Output LAB_HASH

⚠ SUBJID	⚠ VISIT	⚠ VISITNUM	⚠ LBDTC	123 LBORRES	⚠ LBTEST	⚠ LBTESTCD	⚠ LBCAT	⚠ LBSCAT
0123	DAY 1	D1	2014-02-07	45	Albumin	ALB	CHEMISTRY	PROTEIN
0123	DAY 1	D1	2014-02-07	84	Alkaline Phosphatase	ALP	CHEMISTRY	ENZYME,LIVER FUNCTION
0124	DAY 1	D1	2014-02-07	45	Albumin	ALB	CHEMISTRY	PROTEIN
0124	DAY 1	D1	2014-02-07	34	Alkaline Phosphatase	ALP	CHEMISTRY	ENZYME,LIVER FUNCTION

Hiter-Traversing HASH object data

Use the hash iterate object to store and search data based on lookup keys. The hash iterate object enables to retrieve the hash object data in either forward or reverse key order.

Example

```
data crit;  
length usubjid $20 ady aval 8;  
if _n_=1 then do;  
  Declare hash hck(dataset:"adlb(where=(paramcd='CKS')",ordered:'ascending');
```

```
  declare hiter hick('hck');
```

**Declare Hiter to access
data from Hash object**

```
  hck.definekey('usubjid', 'ady');  
  hck.definedata('usubjid', 'ady', 'aval');  
  hck.definedone();  
  call missing(usubjid,ady,aval);  
end;
```

```
  rc=hick.first();  
  do while(rc=0);  
    rc1=hck.find();  
    if aval>150 then output;  
    rc=hick.next();  
  end;
```

**Access Hash Object
data using Hiter**

```
run;
```

Comparison of
Performance with
Different Techniques

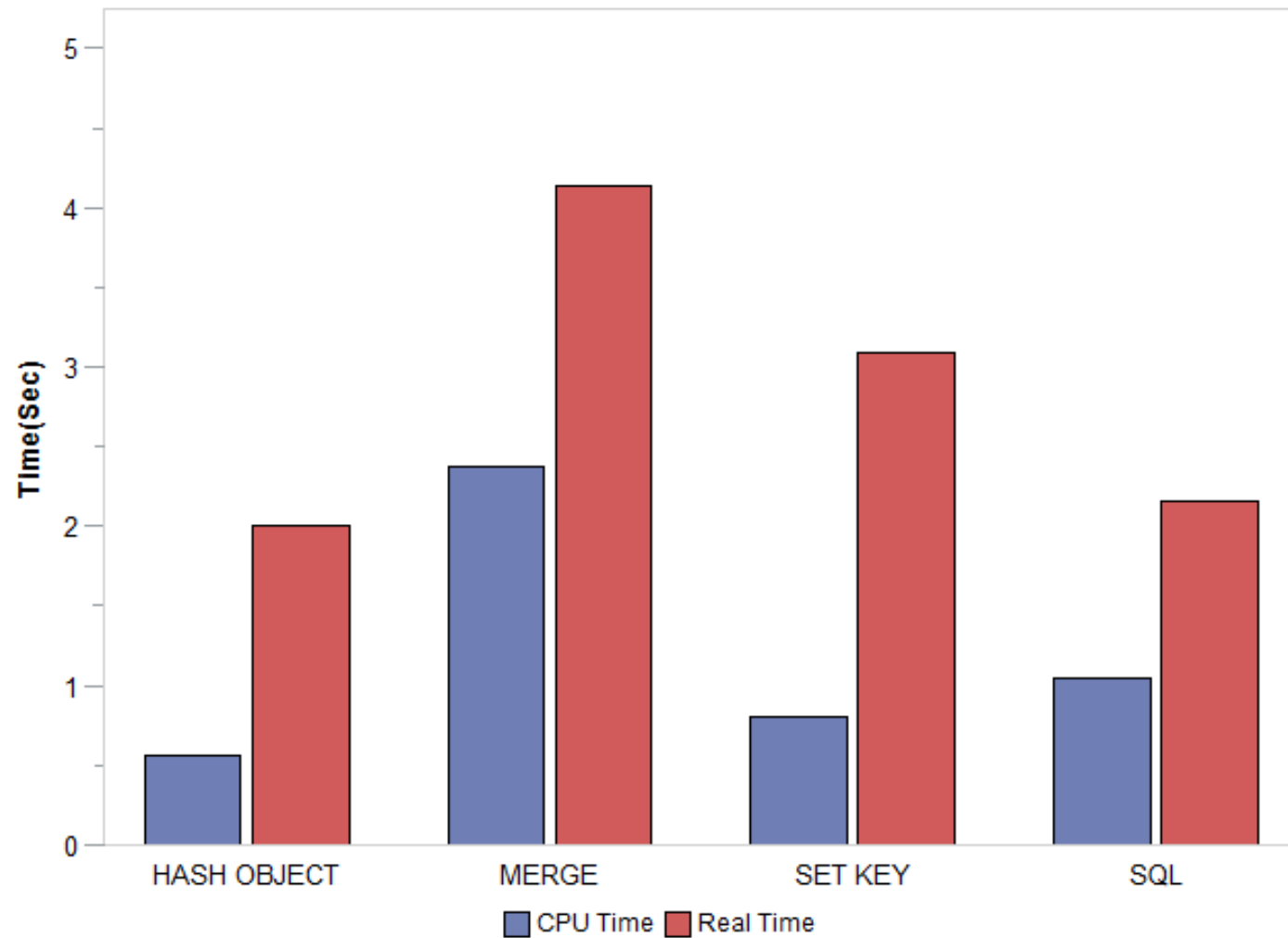
Compare the Efficiency



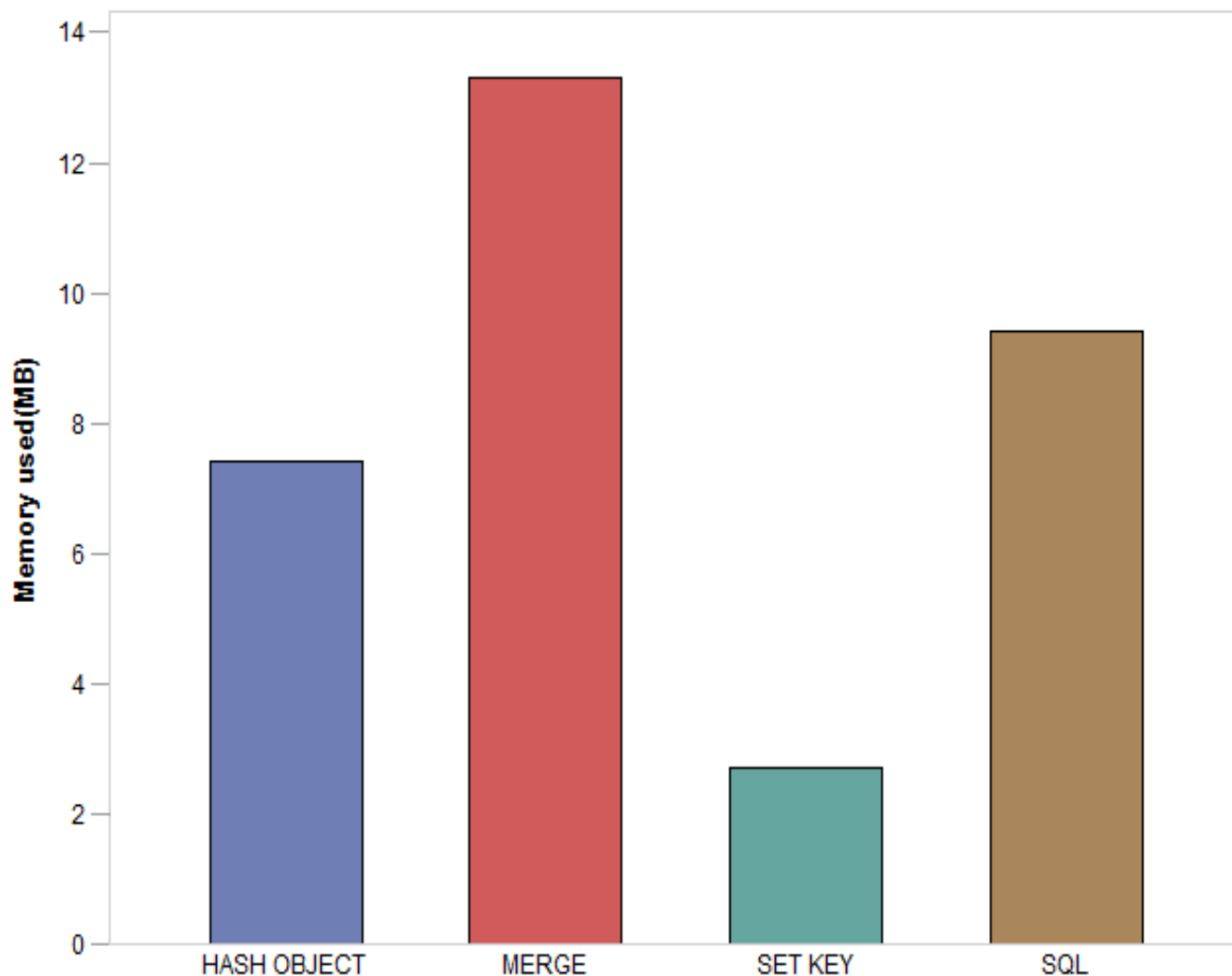
**How efficient is
this compared to
other techniques**

The performance of four techniques (Merge, Set key, SQL and HASH) has been benchmarked on a laptop with Window 7 professional operating system, Intel Core I3 2.4 GHz CPU, 4 GB memory (RAM) and SAS 9.3 installed. The average of these tests is used to measure the performance.

PERFORMANCE OF CPU AND REAL TIME



PLOT REPRESENTING THE MEMORY USED BY DIFFERENT TECHNIQUES



Advantages

- **Memory Resident**
- **Direct -addressing**
- **Natural way to sort/distinct values**
- **Fast and flexible table look-up**
- **No need to predict size in advance**
- **Can return multiple variables at once**
- **Store data like an array**
- **Summarize in a DATA step on-the-fly**
- **Can have character keys and composite keys**

Conclusion

Users have a powerful hash DATA-step construct to sort data, search datasets, and perform table lookup operations in the SAS system. SAS users everywhere can begin to take advantage of this powerful memory-based programming technique to improve the performance of sorts, searches and table lookup operations.



