

Blink and You Will Miss it !

Introduction to SAS® Hash Tables
Presentation and reference paper CT06

Daniel Lindenbaum



Hash table operations are fast!

10 minutes of focused energy!

Background & Overview

3

The Hash Table (onwardly referred to as HT) :

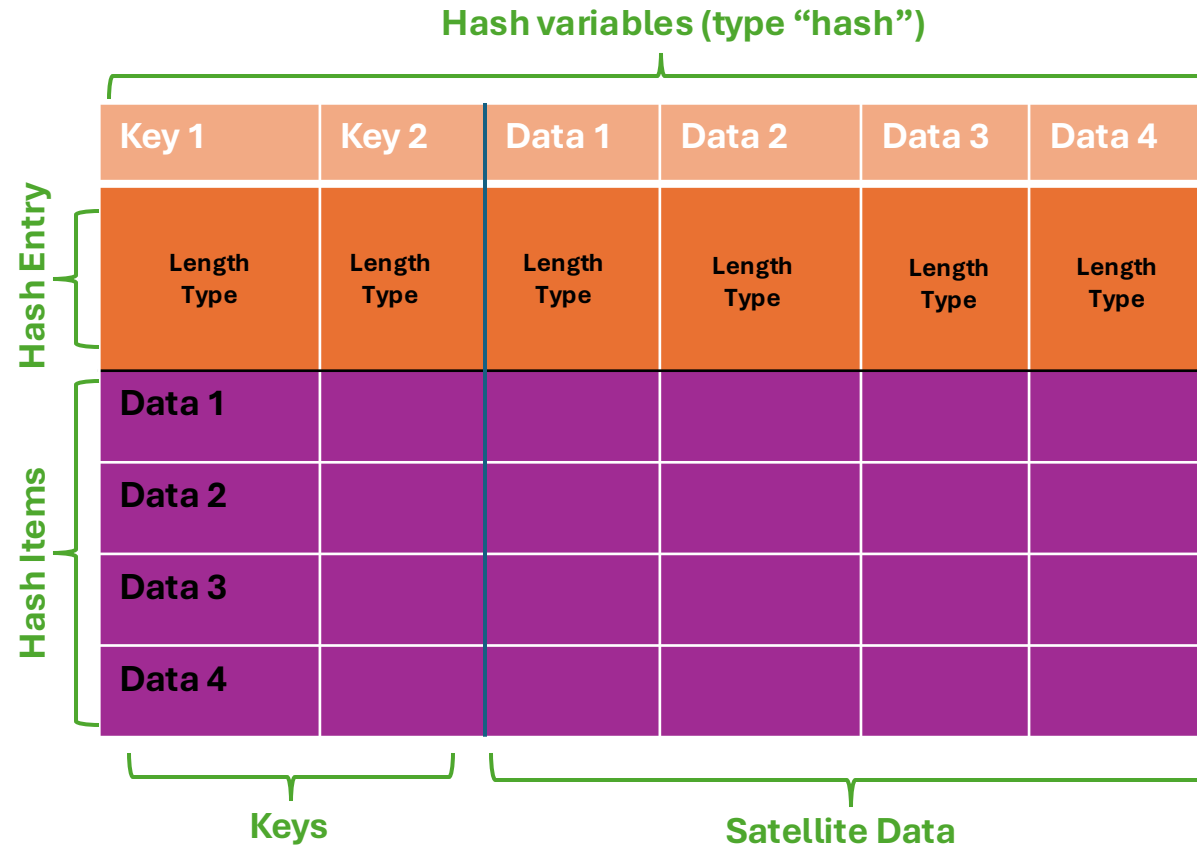
- Extension to the Datastep
- Fast data operations
- Parallels with indexed datasets (key and satellite data portion)
- Resides completely in memory
- Controlled during execution time
- Driven by runtime expressions
- Dynamic structure – memory management
- Temporary nature – exists for the duration of the data step
- Searches are performed using buckets (trees) of key values and corresponding data
- Speed is almost totally independent of the number of items in the table

[!] PDV must have initialized variables with appropriate attribute information, for each HT variable that will be used

Background & Overview

4

HT Visual Representation of Structure:

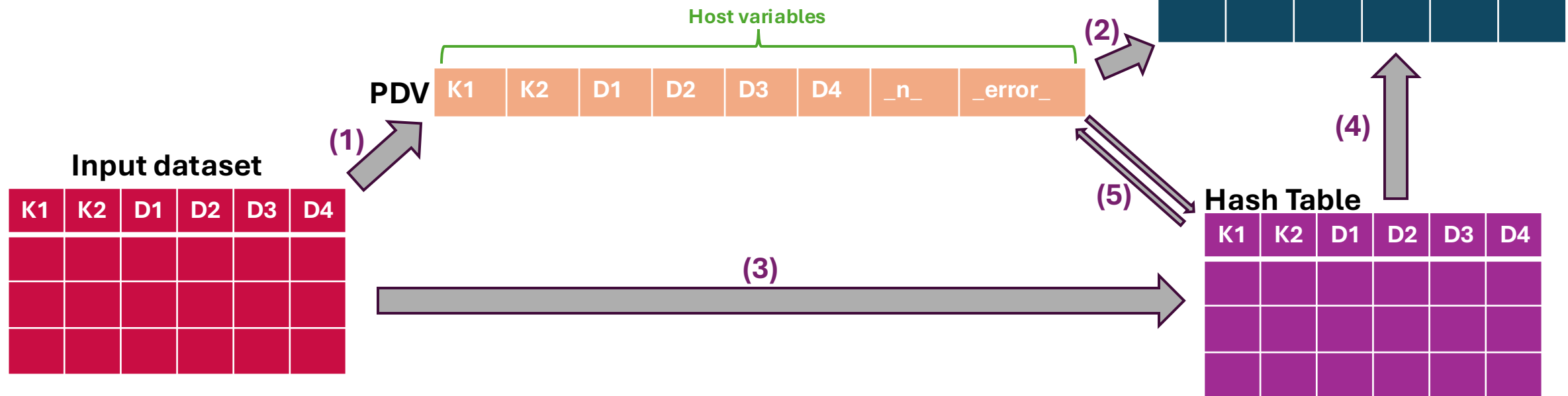


Key fields can be repeated in the satellite data portion.

Data Flow

5

- (1.) Data step loads one record at a time into the program data vector (PDV)
- (2.) Data step outputs one record at a time
- (3.) The entire input dataset is loaded into memory
- (4.) Entire HT can be written to a dataset in a single operation
- (5.) Data can travel between HT and PDV



Sequence of Operations:

- Update the PDV with attribute information: names, types & lengths of fields*
- Declare the HT (+create an instance of the HT)
- Define the hash variables (keys & satellite data variables)
- Terminate the HT definition process
- Assign values to hash variables
- Apply sorting (optional)
- Process duplicate keys
- Perform data operations/derivations
- Write to a dataset (optional)

* Allows the data step compiler see the variables that are to be given to the HT and places them in the PDV

Hash Table Methods

- DEFINE KEY : specify the names of the key fields
- DEFINE DATA : specify the names of the satellite data fields
- DEFINE_DONE : signal the end of the HT definition phase
- ADD : insert data row to the named HT (PDV to HT)
- CHECK : evaluate whether a value for a key field exists in the HT
- FIND : as per CHECK, also returns the corresponding data variable values (HT to PDV)
- OUTPUT : send HT contents to a dataset
- CLEAR : Delete HT values (does not delete the HT)

Example Code

8

Elementary HT syntax and structure

```
data _null_;  
  length k1 d1 8;  
  call missing(k1,d1);  
  declare hash hob();  
    hob.definekey('k1');  
    hob.definedata('k1','d1');  
  hob.definedone();  
  stop;  
run;
```

Initialise PDV variables

Declare & create an instance of the HT

Define hash key fields

Define hash data fields

End the hash definition

Prevent infinite dataset looping



The abbreviation for the DECLARE keyword is DCL – the terms are interchangeable when coding.

Blink and You Will Miss it !

Note on Attributes

9

- HT *cannot* determine variable attributes from the context of a data step.
- Attributes must be defined *before* the HT is declared.
- Attributes can be assigned “long-hand” (length statement) or inherited from an existing dataset.

```
data dseta;  
  length k1 8 d1 $10;  
  k1=62;  
  d1='age';  
  output;  
run;
```

```
data _null_;  
  if 0 then set dseta;  
  dcl hash hob();  
  hob.definekey('k1');  
  hob.definedata('k1','d1');  
  hob.definedone();  
run;
```

“if 0 then set <dsname>” can be used as a simple method to force “inheritance” of variable attributes from a “parent” dataset.

Method: ADD

10

Add the current PDV values to a record in the the stated HT

```
data _null_;  
  length k1 8 d1 $10;  
  declare hash hob();  
    hob.definekey('k1');  
    hob.definedata('d1');  
  hob.definedone();  
  k1=1;  
  d1='a';  
  rc=hob.add();  
run;
```

Method: CHECK

11

Determine whether the PDV key values are found in the named HT

```
data _null_;  
  length k1 8 d1 $10;  
  declare hash hob();  
    hob.definekey('k1');  
    hob.definedata('k1','d1');  
  hob.definedone();  
  k1=1;  
  d1='a';  
  hob.add();  
  d1='';  
  rc1=hob.check();  
  put 'NOTE: Before key value change ' _all_;  
  k1=2;  
  rc2=hob.check();  
  put 'NOTE: After key value change ' _all_;  
run;
```

```
NOTE: Before key value change k1=1 d1= rc1=0 rc2=_ERROR_=0 _N_=1  
NOTE: After key value change k1=2 d1= rc1=0 rc2=160038 _ERROR_=0 _N_=1  
NOTE: DATA statement used (Total process time):  
    real time           0.01 seconds  
    cpu time            0.00 seconds
```

Method: FIND

12

Determine whether the PDV key(s) are found in the named HT

```
data _null_;  
  length k1 8 d1 $10;  
  declare hash hob();  
    hob.definekey('k1');  
    hob.definedata('k1','d1');  
  hob.definedone();  
  k1=1;  
  d1='a';  
  rc1=hob.add();  
  k1=2;  
  d1='b';  
  rc1=hob.add();  
  rc2=hob.find();  
  put 'NOTE: Pre key value change: ' _all_;  
  if k1=2 then k1=1;  
  rc2=hob.find();  
  put 'NOTE: Post key value change: ' _all_;  
run;
```

```
NOTE: Pre key value change: k1=2 d1=b rc1=0 rc2=0 _ERROR_=0 _N_=1  
NOTE: Post key value change: k1=1 d1=a rc1=0 rc2=0 _ERROR_=0 _N_=1  
NOTE: DATA statement used (Total process time):  
      real time           0.00 seconds  
      cpu time            0.01 seconds
```

Return the data values if the key is located from HT to PDV

Duplicate Keyed Records

13

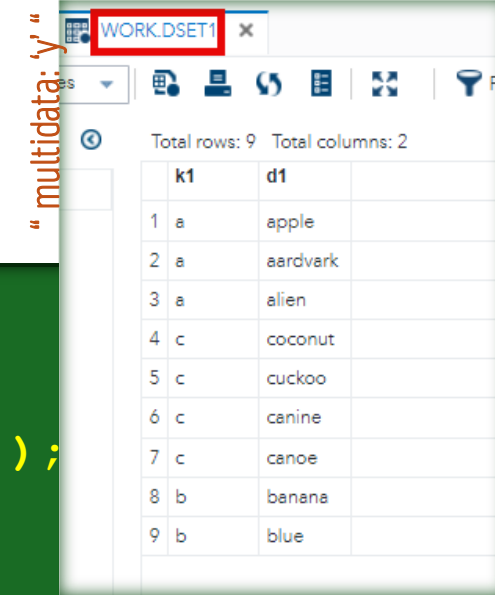
Default is to only keep the first record if duplicate keys are encountered

```
data multi;  
  input k1 $ d1 $;  
  cards;  
a apple  
a aardvark  
a alien  
b banana  
b blue  
c coconut  
c cuckoo  
c canine  
c canoe  
;  
run;
```

Import data source and allow repeated key values (method 1)

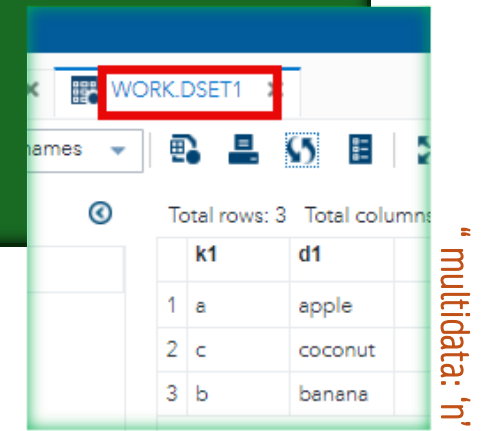
```
data _null_;  
  if 0 then set multi;  
  dcl hash hob(dataset: 'multi', multidata: 'Y');  
  hob.definekey('k1');  
  hob.definedata('d1', 'd1');  
  hob.definedone();  
  hob.output(dataset: 'dset1');  
  stop;  
run;
```

“multidata: ‘y’”



	k1	d1
1	a	apple
2	a	aardvark
3	a	alien
4	c	coconut
5	c	cuckoo
6	c	canine
7	c	canoe
8	b	banana
9	b	blue

“multidata: ‘n’”



	k1	d1
1	a	apple
2	c	coconut
3	b	banana

Argument tags are parameters that can be assigned, influencing HT operations.
The DATASET argument tag allows data to be read/written from/to a dataset.
The MULTIDATA argument tag forces the acceptance of duplicate keys.

Sorting Data

14

Load data from a source dataset, sort in memory (by keys) and output to a separate dataset

```
%macro tosort(inds=, outds=, keys=, dvar=, keepdup=);  
  
data _null_;  
  if 0 then set &inds;  
    dcl hash hob(dataset:"&inds", ordered: 'a', multidata:"&keepdup");  
    hob.definekey(&keys);  
    hob.definedata(&dvar);  
    hob.definedone();  
    hob.output(dataset:"&outds");  
  stop;  
run;  
  
%mend tosort;  
  
* different combinations of key variables and duplicate handling;  
%tosort(inds=sashelp.class, outds=classaa, keys=%str('age','sex'), dvar=%str('sex','age','name'), keepdup=n);  
%tosort(inds=sashelp.class, outds=classab, keys=%str('sex'), dvar=%str('sex'), keepdup=y);  
%tosort(inds=sashelp.class, outds=classac, keys=%str('sex'), dvar=%str('sex'), keepdup=n);
```

ORDERED argument tag

Sorting Data

15

%tosort – call I
(keys: age, sex, multidata:'n')

The screenshot shows a spreadsheet window titled 'WORK.CLASSAA'. The 'Columns' panel on the left has 'Select all', 'Sex', 'Age', and 'Name' checked. The main table has 11 rows and 3 columns: Sex, Age, and Name. The data is sorted by Age (ascending) and then by Sex (ascending).

	Sex	Age	Name
1	F	11	Joyce
2	M	11	Thomas
3	F	12	Jane
4	M	12	James
5	F	13	Alice
6	M	13	Jeffrey
7	F	14	Carol
8	M	14	Alfred
9	F	15	Janet
10	M	15	Ronald
11	M	16	Philip

%tosort – call II
(keys: sex, multidata:'y')

The screenshot shows a spreadsheet window titled 'WORK.CLASSAB'. The 'Columns' panel on the left has 'Select all' and 'Sex' checked. The main table has 19 rows and 2 columns: Sex and an empty column. The data is sorted by Sex (ascending).

	Sex	
1	F	
2	F	
3	F	
4	F	
5	F	
6	F	
7	F	
8	F	
9	F	
10	M	
11	M	
12	M	
13	M	
14	M	
15	M	
16	M	
17	M	
18	M	
19	M	

%tosort – call III
(keys: sex, multidata:'n')

The screenshot shows a spreadsheet window titled 'WORK.CLASSAC'. The 'Columns' panel on the left has 'Select all' and 'Sex' checked. The main table has 2 rows and 2 columns: Sex and an empty column. The data is sorted by Sex (ascending).

	Sex	
1	F	
2	M	

DATA LOOKUP

16

Lookup a value contained in one dataset, returning a record from another dataset, matched on key

```
%macro bigdata;  
  data main(keep=invoice make model);  
    set sashelp.cars;  
    %do k=1 %to 100000;  
      invoice+&k;  
      output;  
    %end;  
  run;  
%mend bigdata;  
%bigdata;  
  
data lkup;  
  length invoice 8;  
  invoice=380698;  
  output;  
  stop;  
run;
```

```
data main2;  
  if 0 then set lkup;  
  if _n_=1 then do;  
    * load the smaller dataset into the HT;  
    dcl hash hob(dataset:"lkup");  
    hob.definekey('invoice');  
    hob.definedone();  
  end;  
  set main;  
  if (hob.check()=0) then do;  
    * if rc is zero then result is true;  
    output;  
  end;  
run;
```

CPU time used:
Hash Method : 8s

	Make	Model	Invoice
1	Acura	MDX	\$380,698
2	Mercury	Grand Marquis LS Ultimate 4dr	\$380,698

```
proc sort data=main;  
  by invoice;  
run;  
  
data main3;  
  merge main(in=aa) lkup(in=bb);  
  by invoice;  
  if bb;  
run;  
  
proc datasets lib=work nolist;  
  delete main: lkup;  
run;
```

CPU time used:
Merge Method : 40s

5 fold difference in CPU time used

HASH ITERATOR OBJECT

17

The hash iterator object facilitates record by record navigation within a hash object

```
proc sort data=sashelp.class out=class(keep=name age height);  
  by name;  
run;
```

```
data class;  
  if 0 then set class;  
  declare hash hasha(dataset:"class", ordered:'a');  
  declare hiter hitera('hasha');  
  hasha.definekey('name');  
  hasha.definedata('name','age','height');  
  hasha.definedone();  
  call missing(name, age, height);  
  put _all_;  
  hitera.first();  
  put _all_;  
  hitera.next();  
  put _all_;  
  hitera.last();  
  put _all_;  
  hitera.prev();  
  put _all_;  
stop;  
run;
```

```
NOTE: There were 19 observations read from the data set WORK.CLASS.  
Name= Age= Height= _ERROR_=0 _N_=1  
Name=Alfred Age=14 Height=69 _ERROR_=0 _N_=1  
Name=Alice Age=13 Height=56.5 _ERROR_=0 _N_=1  
Name=William Age=15 Height=66.5 _ERROR_=0 _N_=1  
Name=Thomas Age=11 Height=57.5 _ERROR_=0 _N_=1  
NOTE: The data set WORK.CLASS has 0 observations and 3 variables.  
NOTE: DATA statement used (Total process time):
```

Load named dataset into hash object

Declare hash iterator object (and the hash table to which it belongs)

Set PDV values to missing

Retrieve first record from hash object

Retrieve next record from hash object

Retrieve last record from hash object

Retrieve previous record from hash object

N remains equal to 1 for all hash iterator data reads

Thank You

Enjoy incorporating Hash Tables into
your programming solutions!

✉ daniel.lindenbaum@phastar.com

