

Chrestos – Die grüne CRO

Hash-ing It Out: Bringing SAS to the Next Level

Main Author: Dr. Lara Suckut
lara.suckut@chrestos.de

Co-Author: Veronika Schmidt
veronika.schmidt@chrestos.de

CT01

Phuse EU Connect 2025



The problem

- Larger, more complex and more data sets
- Combination of information needed from different SAS data sets
- SAS merge is slow, requires sorting
- Quick merge and look-up needed
- Also for different structure of data sets
- Looking for disk space and run-time efficient solution

Big Data



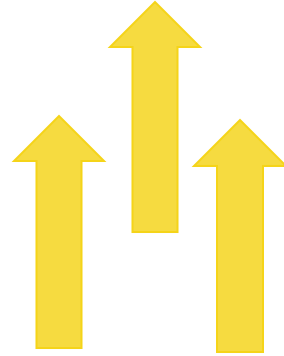
The solution

Hash tables!

Level Up!



Base



Advanced

What is a hash object?

- Data step construct
 - No sorting required
 - Memory-based
 - Connects data sets using key values
 - Based on binary search tree
- fast look-up possible

Keyvar1	Keyvar2	...	Datavar1	Datavar2	...
Key 1	Key A	...	Value 1	Value A	...
Key 1	Key B	...	Value 2	Value B	...
Key 1	Key C	...	Value 3	Value C	...
Key 2	Key B	...	Value 4	Value D	...
Key 2	Key D	...	Value 5	Value E	...
Key 3	Key A	...	Value 6	Value F	...
Key 3	Key B	...	Value 7	Value G	...
Key 4	Key A	...	Value 8	Value H	...
...	...	...	...	...	...

How do I code this?

```

1 data hashbuild;
2   if 0 then set hashdata;
3   if _N_ eq 1 then do;
4     declare hash h (dataset: 'hashdata (where = (datavar1 > 10) ) ');
5     h.definedata('datavar1', 'datavar2');
6     h.definedone();
7   end;
8
9 run;
```

Introduce variables of hash object to program data vector, can also be length or attrib statements with call missing

Initialize hash object of name h, optionally

Hash object name.method() structure

Define data variable(s) for variable(s)

Tell SAS hash object definition ends here

possible

How do I use this?

Example 1: check values

- Check if data from one data set is in other data set

Variable 1	Variable 2	Variable 3	Variable 4
...	...	Key 3	...
...	...	Key 1	...
...	...	Key 4	...
...	...	Key 2	...
...	...	Key 3	...

Data table

Hash table		
Keyvar1	Datavar1	Datavar2
Key 1	Value 1	Value A
Key 3	Value 3	Value C
Key 2	Value 2	Value B



al programming example:
laboratory tests in CDISC
Controlled Terminology

Example1: check values

```

1  data checked issues;
2    set tocheck; Data set to be checked
3    if 0 then set sdtm_ct(keep=cdisc_submission_value);
4    if _N_ eq 1 then do;
5      declare hash h (dataset: `sdtm_ct`); Declare hash object and load CDISC
6      h.definekey(`cdisc_submission_value` S Define key variable for look-up
7      h.definedone(); No data variable needed, end hash object
8    end; initialization
9    rc=h.check(key:lbtest); ! The look-up is case-sensitive !
10   row_num=_N_; Check if values of lbtest column in key column
11   if rc ne 0 then output issues; Output values not found in issues data set
12   output checked;
13 run;

```

What else can I use this for?

- Retrieve matching data from one data set to other data set

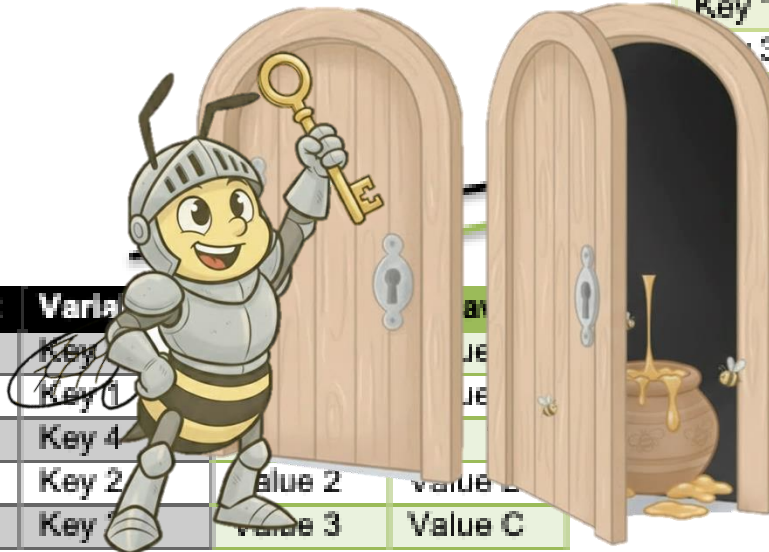
Hash table

Keyvar1	Datavar1	Datavar2
Key 1	Value 1	Value A
Key 3	Value 3	Value C
Key 2	Value 2	Value B

Data table

Variable 1	Variable 2	Variable 3
...	...	Key 1
...	...	Key 1
...	...	Key 4
...	...	Key 2
...	...	Key 3

statistical programming example:
calculate ADaM On Treatment
Record Flag in Concomitant
Medications



Example 2: Fetch values

```
1  data adcm;
2      set indata.cm(where=(length(cmstdtc) eq 10 and length(cmendtc) eq 10));
3      if 0 then set indata.adsl(keep=
4      if _N_ eq 1 then do;
5          declare hash h (dataset: `indata.adsl(keep=usubjid trtsdt trtedt)`);
6          h.definekey(`usubjid`);
7          h.definedata(`trtsdt`,`trtedt`);
8          h.definedone();
9      end;
10
11
12
13
14
15  run;
```

Ignoring incomplete date entries here

Example 2: Fetch values

```
1 data adcm;
```

```
...
```

```
10 call missing(trtsdt,trtedt);
```

```
11 Call missing to avoid retaining of previous find by default
```

```
12 Not necessary to state key as data set variable named the same as key variable
```

```
13 cmend = input(cmendc,11MDD10.);
```

```
14 ontrtfn = ifn(cmstd ge trtsdt and cmstd le (trtedt + 14) or (cmend ge  
trtsdt and cmend le (trtedt + 14), 1, 0);
```

```
15 run;
```

```
15 Fetched column value can be used in calculations
```

Beware of ...



- Searches are case-sensitive
 - Key variables need to lead to a unique match or code needs to be adjusted for multiple matches
 - Data variable names from hash table and data set must not be the same
- possibly renaming needed
- Option hashexp can be set to optimize performance based on number of searches, size of data, and memory availability
 - If you add multiple hash tables in your data set choose names wisely so as not to confuse them
 - Most run-time benefit in comparison to data set merge, less SQL joins → think about whether to change a running system

There is more!

- Can be used with metadata tables and call execute
- Our paper has a summary of additional use cases with reference papers
 - Manipulating hash tables themselves based on current data sets or entered values
 - Dealing with duplicate keys
 - Statistical programming examples like defining baseline, detecting repeated adverse events
- ...
- You can find the paper after the conference on the Phuse website under CT01





Thank you for your
attention!

