How Many Ways Can You Join SAS® Tables

Ask The Expert

19 October 2023

Charu Shankar

SAS Education

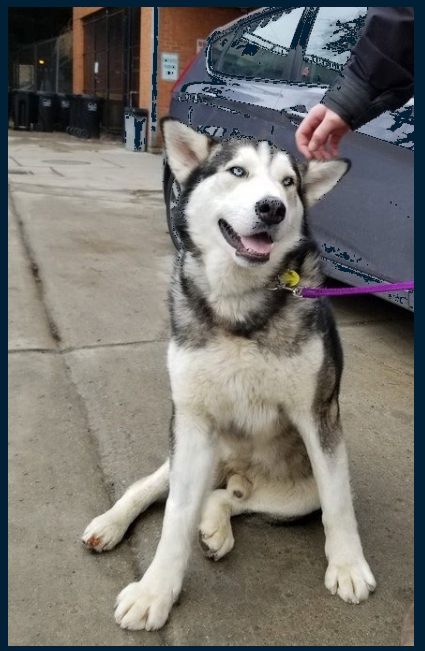
Bio

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking.

A SAS Instructor since 2007, she curates and delivers unique content via the SAS YouTube channel, SAS global forum, SAS Ask the Expert Series, SAS Training Post Blog, etc.

Charu loves to support users by teaching at conferences on topics related to SAS, SQL, Efficiencies, PERL, Macros, Python, Viya, etc.

When she’s not coding, Charu is A Yoga Instructor who loves to explore Canadian trails with her husky Miko.



Agenda

Nuts and Bolts

Data Step Merge

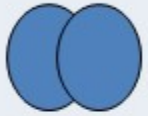
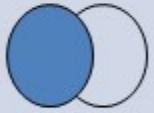
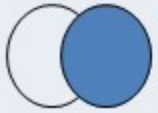
Data Step Hash Join

PROC SQL join with implicit and explicit pass-through

PROC FEDSQL join in SAS® Viya®/ PROC DS2 Merge

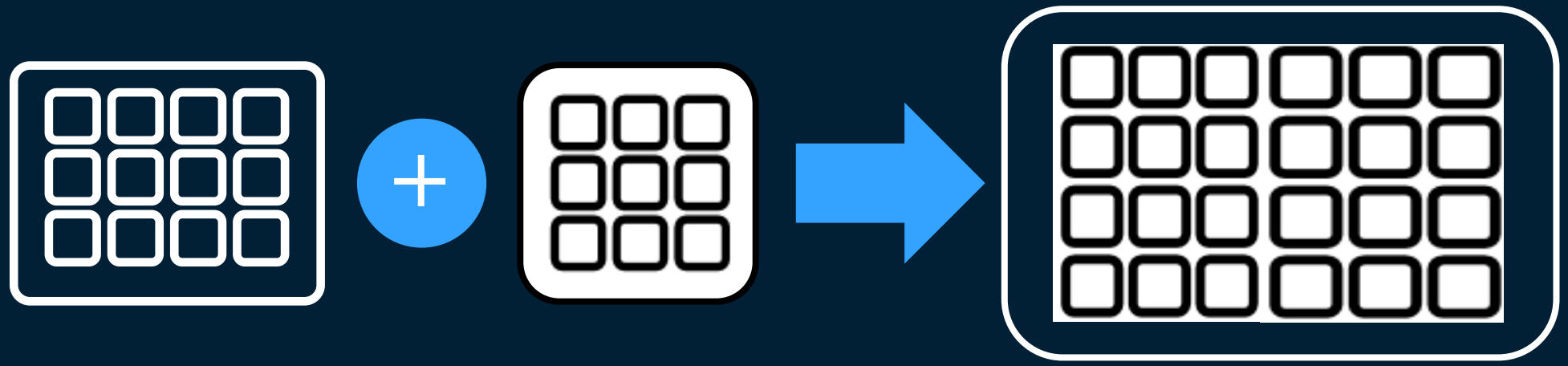
Handy Links

Merging -SQL vs. Data Step

VISUAL	SQL	DATA STEP
All rows from both tables 	**Full Outer Join** Select * from tableA Full outer Join tableB On tableA.id=tableB.id;	**Match Merge** Data tableC; Merge tableA tableB; By id; Run;
All rows from left table & matching rows from right table 	**Left Join** Select * from tableA Left Join tableB On tableA.id=tableB.id;	**Data Step Merge** Use IN=data set option Data tableD; Merge tableA(in=INA) tableB; By id; If INA; Run;
All rows from right table & matching rows from left table 	**Right Join** Select * from tableA Right Join tableB On tableA.id=tableB.id;	**Data Step Merge** Data tableE; Merge tableA tableB(in=INB); By id; If INB; Run;

1 Data Step Merge

Merging Tables



Merging Tables

one-to-one

A	B	C
		1
		2
		3

C	D	E
1		
2		
3		

nonmatching rows

A	B	C
		1
		2
		4

C	D	E
2		
3		
4		

one-to-many

A	B	C
		1
		2

C	D	E
1		
1		
2		

Discussion

Invited
Drew
Lani
Mansfield
Caroline
Kristin
Michael
George

Attending
Caroline
Drew
Michael
Lani
Kristin

Invited
Caroline
Drew
George
Kristin
Lani
Mansfield
Michael

Attending
Caroline
Drew
Kristin
Lani
Michael

Merging Tables

class

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

With the input tables ordered by **Name**, SAS can compare rows sequentially to efficiently match rows.

Merging Tables

```
DATA output-table;
    MERGE input-table1 input-table2 ...;
    BY BY-column(s);
RUN;
```

list any number of input tables with one or more common columns

list the common column or columns

The input tables must be sorted by the column (or columns) listed in the BY statement.

Merging Tables

```
data class2;
    merge sashelp.class ATE.class_teachers;
    by Name;
run;
```

Columns are combined in the new table by matching values of **Name**.

sashelp.class

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98

ATE.class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

class2

Name	Sex	Age	Height	Weight	Grade	Teacher
Alfred	M	14	69	112.5	8	Thomas
Alice	F	13	56.5	84	7	Evans
Barbara	F	13	65.3	98	6	Smith

Merging Tables: Compilation

```
data class2;
    merge sashelp.class ATE.class_teachers;
    by Name;
run;
```

All columns from the first table are added to the PDV.

PDV				
Name	Sex	Age	Height	Weight

Merging Tables: Compilation

```
data class2;
    merge sashelp.class ATE.class_teachers;
    by Name;
run;
```

Additional columns from the second table are added to the PDV.

PDV

Name	Sex	Age	Height	Weight	Grade	Teacher

The BY column is already in the PDV.

Merging Tables: Execution

```
data class2;
     merge sashelp.class ATE.class_teachers;
     by Name;
run;
```

Rows are read sequentially from both tables. When the BY values match, they are both read into the PDV.

sashelp.class

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98

ATE.class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

PDV

Name	Sex	Age	Height	Weight	Grade	Teacher
Alfred	M	14	69	112.5	8	Thomas

One-to-Many Merge

```
data class2;
     merge teachers_sort test2_sort;
     by name;
run;
```

The BY values match, and both rows are read into the PDV.

teachers_sort

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

test2_sort

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67

PDV

Name	Grade	Teacher	Subject	TestScore	_N_
Alfred	8	Thomas	Math	82	1

One-to-Many **Merge**

```
data class2;
     merge teachers_sort test2_sort;
     by name;
run;
```

The BY values do not match, but one value matches the PDV. That row is read into the PDV and overwrites previous values.

teachers_sort

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

test2_sort

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67

PDV

Name	Grade	Teacher	Subject	TestScore	_N_
Alfred	8	Thomas	Reading	79	2

One-to-Many Merge

```
data class2;
     merge teachers_sort test2_sort;
     by name;
run;
```

Neither BY value matches the PDV.

teachers_sort

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

test2_sort

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67

PDV

Name	Grade	Teacher	Subject	TestScore	_N_
Alfred	8	Thomas	Math	82	3

One-to-Many Merge

```
data class2;
     merge teachers_sort test2_sort;
     by name;
run;
```

The PDV is reset to missing values when a new BY group begins.

teachers_sort

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

test2_sort

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67

PDV

Name	Grade	Teacher	Subject	TestScore	_N_
	.			.	3

One-to-Many Merge

```
data class2;
     merge teachers_sort test2_sort;
     by name;
run;
```

The BY values match, and both rows are read into the PDV.

teachers_sort

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith

test2_sort

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67

PDV

Name	Grade	Teacher	Subject	TestScore	_N_
Alice	7	Evans	Math	71	3

Identifying Matching and Non-Matching Rows

```
data class2;
     merge ATE.class_update ATE.class_teachers;
     by name;
run;
```

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas

class2

Name	Sex	Age	Height	Weight	Grade	Teacher
Alfred	M	14	69	112.5	8	Thomas
Alice	F	13	56.5	84	7	Evans
Barbara	F	13	65.3	98	6	Smith
Carol		.	.	.	8	Thomas
David	M	11	55.3	73	.	
Henry	M	14	63.5	102.5	8	Thomas

The new table includes matches and nonmatches.

Merging Tables: Execution

```
data class2;
     merge ATE.class_update ATE.class_teachers;
     by name;
run;
```

The BY values do not match. SAS reinitializes the PDV and reads the next row in sequence.

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas

PDV

Name	Sex	Age	Height	Weight	Grade	Teacher
Carol		.	.	.	8	Thomas

Merging Tables: Execution

```
data class2;
     merge ATE.class_update ATE.class_teachers;
     by name;
run;
```

The BY values do not match. SAS reinitializes the PDV and reads the next row in sequence.

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas

PDV

Name	Sex	Age	Height	Weight	Grade	Teacher
David	M	11	55.3	73	.	

Merging Tables: Execution

```
data class2;
     merge ATE.class_update ATE.class_teachers;
     by name;
run;
```

Sequential matching continues until all rows are read from each input table.

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas

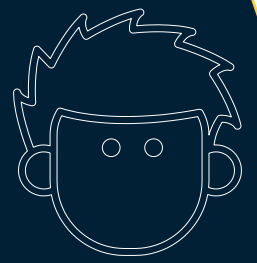
PDV

Name	Sex	Age	Height	Weight	Grade	Teacher
Henry	M	14	63.5	102.5	8	Thomas

Merging Tables with Nonmatching Rows

```
DATA output-table;
      MERGE input-table1(IN=variable)
            input-table2(IN=variable) ...;
      BY BY-column(s);
RUN;
```

The IN= data set option can be used to identify matching and nonmatching rows.



Merging Tables with Nonmatching Rows

```
data class2;
    merge ATE.class_update(in=inUpdate)
          ATE.class_teachers(in=inTeachers);
    by name;
run;
```

PDV

Name	Sex	Age	Height	Weight	Grade	Teacher	inUpdate	inTeachers
							D	D

The IN= variables are 0 if the BY value is *not in* the corresponding input table and 1 if the BY value is *in* the corresponding input table.

Merging Tables with Nonmatching Rows

```
data class2;
    merge ATE.class_update(in=inUpdate)
          ATE.class_teachers(in=inTeachers);
    by name;
run;
```

class2

Name	Sex	Age	Height	Weight	Grade	Teacher	inUpdate	inTeachers
Alfred	M	14	69	112.5	8	Thomas	1	1
Alice	F	13	56.5	84	7	Evans	1	1
Barbara	F	13	65.3	98	6	Smith	1	1
Carol		.	.	.	8	Thomas	0	1
David	M	11	55.3	73	.		1	0

How can we include only matching rows in the output table?

Values are assigned in the PDV during execution but not written to the output table.

Multiple Choice Question

Which statement writes only matching rows to the output table?

```
data class2;
     merge ATE.class_update(in=inUpdate)
           ATE.class_teachers(in=inTeachers);
     by name;
     ???
run;
```

a. `where inUpdate=1 and inTeachers=1;`
b. `where inUpdate=1 or inTeachers=1;`
c. `if inUpdate=1 and inTeachers=1;`
d. `if inUpdate=1 or inTeachers=1;`

Multiple Choice Question – Correct Answer

Which statement writes only matching rows to the output table?

```
data class2;
     merge ATE.class_update(in=inUpdate)
           ATE.class_teachers(in=inTeachers);
     by name;
     if inUpdate=1 and inTeachers=1;
run;
```

a. `where inUpdate=1 and inTeachers=1;`
b. `where inUpdate=1 or inTeachers=1;`
c. `if inUpdate=1 and inTeachers=1;`
d. `if inUpdate=1 or inTeachers=1;`

The subsetting IF statement must be used because values for the IN= variables are assigned during execution.

Merging Tables with Matching & Nonmatching Rows

1. Demonstration illustrates using the DATA step MERGE to combine two tables with matching rows.
2. Demonstration illustrates using the DATA step MERGE to combine two tables and identify nonmatching rows.

2 Data Step Hash Join

Performing a Table Lookup

Calculate the percentage of US city population within the state population, storing the results by descending city population.

```
PctPop=CityPop2017/StatePop2017;
```

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop
New York	Albany	19,849,399	New York	NY	8,622,698	43.4%
California	Sacramento	39,536,653	Los Angeles	CA	3,999,759	10.1%
Illinois	Springfield	12,802,023	Chicago	IL	2,716,450	21.2%
Texas	Austin	28,304,596	Houston	TX	2,312,717	8.2%
Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	23.2%

Performing a Table Lookup

ATE.population_usstates (50 rows)

StateName	Capital	StatePop2017
California	Sacramento	39,536,653
Texas	Austin	28,304,596
Florida	Tallahassee	20,984,400
New York	Albany	19,849,399
Pennsylvania	Harrisburg	12,805,537

ATE.population_uscities (19,500 rows)

CityName	StateCode	CityPop2017
New York	NY	8,622,698
Los Angeles	CA	3,999,759
Chicago	IL	2,716,450
Houston	TX	2,312,717
Phoenix	AZ	1,626,078

Before calculating the population percentage, the state populations must be combined with the city populations.

```
StateName=stname1(StateCode);
```

Discussion

How many steps are needed for this example if you want to use a DATA step with a MERGE statement?

ATE.population_usstates (50 rows)

StateName	Capital	StatePop2017
California	Sacramento	39,536,653
Texas	Austin	28,304,596
Florida	Tallahassee	20,984,400
New York	Albany	19,849,399
Pennsylvania	Harrisburg	12,805,537

+

ATE.population_uscities (19,500 rows)

CityName	StateCode	CityPop2017
New York	NY	8,622,698
Los Angeles	CA	3,999,759
Chicago	IL	2,716,450
Houston	TX	2,312,717
Phoenix	AZ	1,626,078

↓

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop
New York	Albany	19,849,399	New York	NY	8,622,698	43.4%
California	Sacramento	39,536,653	Los Angeles	CA	3,999,759	10.1%
Illinois	Springfield	12,802,023	Chicago	IL	2,716,450	21.2%
Texas	Austin	28,304,596	Houston	TX	2,312,717	8.2%
Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	23.2%

DATA Step MERGE Versus DATA Step Hash Object

DATA step with SET	read cities table to create **StateName**
PROC SORT step	sort cities table by **StateName**
PROC SORT step	sort states table by **StateName**
DATA step with MERGE	merge tables to calculate **PctPop**
PROC SORT step	sort new table by descending **CityPop2017**

DATA Step MERGE Versus DATA Step Hash Object

DATA step with SET	read cities table to create **StateName**
PROC SORT step	sort cities table by **StateName**
PROC SORT step	sort states table by **StateName**
DATA step with MERGE	merge tables to calculate **PctPop**
PROC SORT step	sort new table by descending **CityPop2017**

DATA step with SET and Hash Object

- load states table into hash object
- read cities table and create **StateName**
- look up **StateName** in hash object and if found return states data
- calculate **PctPop**

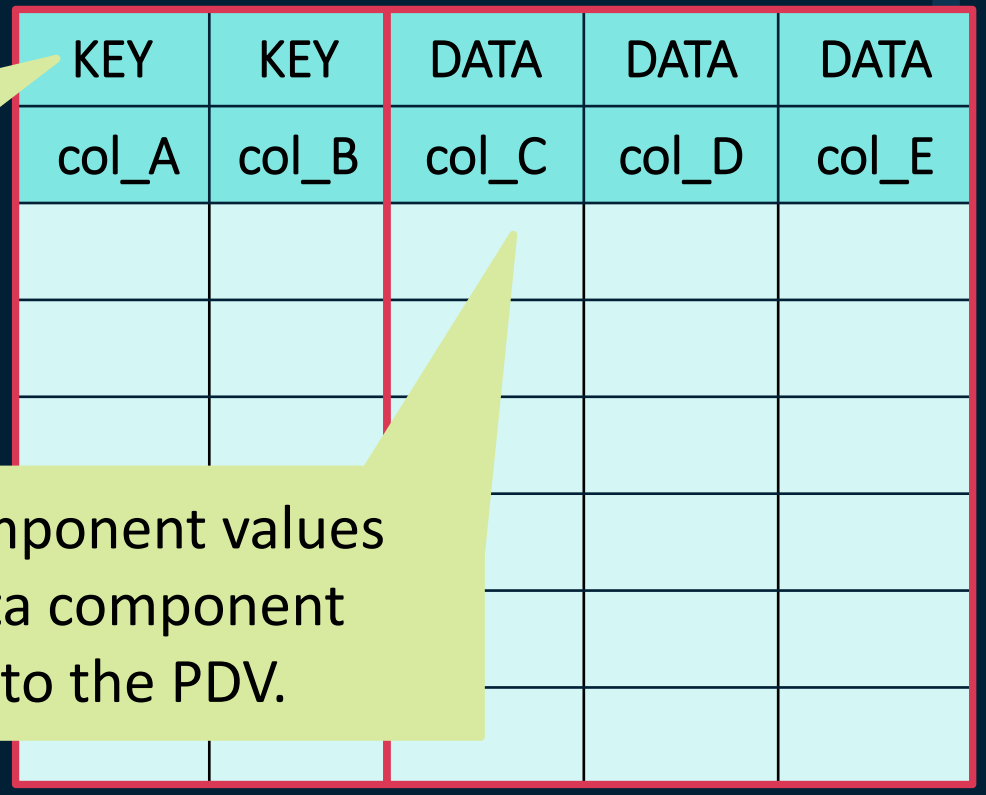
What Is a Hash Object?

A *hash object* is an in-memory table that contains key and data components.

KEY	KEY	DATA	DATA	DATA
col_A	col_B	col_C	col_D	col_E

What Is a Hash Object?

You can quickly and efficiently store, search, and retrieve data based on the key components.



When the key component values are found, the data component values are copied to the PDV.

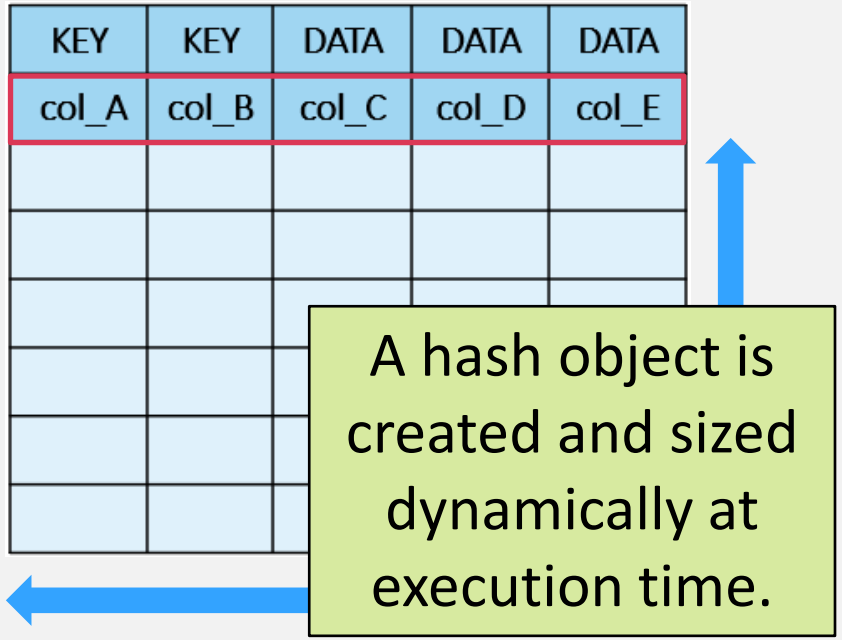
What Is a Hash Object?

```
data ... ;

    ...

run;
```

A hash object exists only for the duration of the DATA step.

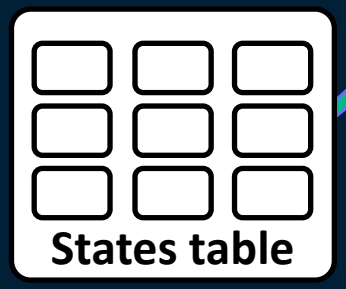


A hash object is created and sized dynamically at execution time.

PDV

col_A	col_B	col_C	col_D	col_E	col_F	col_G	col_H	col_I	_N_

Hash Object Process



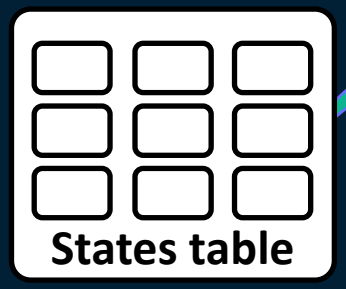
Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
		.			.	.	1

Hash Object Process



Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

Remember that key and data components must also be defined as columns in the PDV.

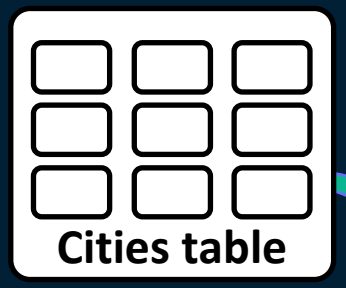
PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
		.			.	.	1

Hash Object Process

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .



PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
		.	Los Angeles	CA	3999759	.	2

Hash Object Process

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

```
StateName=stname1(StateCode);
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
California		.	Los Angeles	CA	3999759	.	2

Hash Object Process

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

The hash object is efficiently searched by the key component.

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
California		.	Los Angeles	CA	3999759	.	2

Hash Object Process

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
California	Sacramento	39536653	Los Angeles	CA	3999759	.	2

Hash Object Process

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

```
PctPop=CityPop2017/StatePop2017;
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	PctPop	_N_
California	Sacramento	39536653	Los Angeles	CA	3999759	0.4344	2

Declaring a Hash Object

DECLARE *object object-name*(<*argument_tag-1*: *value-1*, … >);

hash

```
declare hash States( );
```

name of the hash object

The DECLARE statement creates an instance of and initializes data for an object.

Declaring a Hash Object

(DATASET: '*data-set-name* **<(***data-set-option***)>')**

```
declare hash States(data set:'ATE.population_usstates');
```

name of the table to load into the hash object

```
declare hash States(dataset:'ATE.population_usstates
                    (where=(StatePop2017>20000000))');
```

data set option

Hash Object Process

declare hash object

DECLARE *object object-name*(<*argument_tag-1*: *value-1*, … >);

define hash object

object-name.**DEFINEKEY**('*key-1*' < , … '*key-n*' >);

object-name.**DEFINEDATA**('*data-1*' < , … '*data-n*' >);

object-name.**DEFINEDONE**( );

use hash object

object-name.**FIND**(<**KEY:** *value-1*, … **KEY:** *value-n*>)

Defining a Hash Object - Methods

Hash object methods are operations performed on a hash object using dot notation.

```
object-name.definekey()
```

```
object-name.definedata()
```

```
object-name.definedone()
```

```
object-name.find()
```

```
object-name.add()
```

```
object-name.output()
```

Defining a Hash Object

object-name.**DEFINEKEY(**'*key-1*' < , … '*key-n*' >**);**

object-name.**DEFINEDATA(**'*data-1*' < , … '*data-n*' >**);**

object-name.**DEFINEDONE( );**

To define a hash object, three DEFINE methods are needed.

Defining a Hash Object

```
length StateName $ 20 Capital $ 14 StatePop2017 8;

if _N_=1 then do;

   declare hash States(dataset: 'ATE.population_usstates');

   States.definekey('StateName');

   States.definedata('Capital','StatePop2017');

   States.definedone();

   call missing(StateName, Capital, StatePop2017);

end;
```

declare and define the hash object only one time

defines the key components

defines the data components

loads the table into the hash object

Defining Key and Data Components as PDV Columns

```
length StateName $ 20 Capital $ 14 StatePop2017 8;

if _N_=1 then do;

   declare hash States(dataset: 'ATE.population_usstates');

   States.definekey('StateName');

   States.definedata('Capital','StatePop2017');

   States.definedone();

   call missing(StateName, Capital, StatePop2017);

end;
```

defines the key and data components as PDV columns

sets the key and data columns to missing

Finding Key Values in a Hash Object - FIND Method

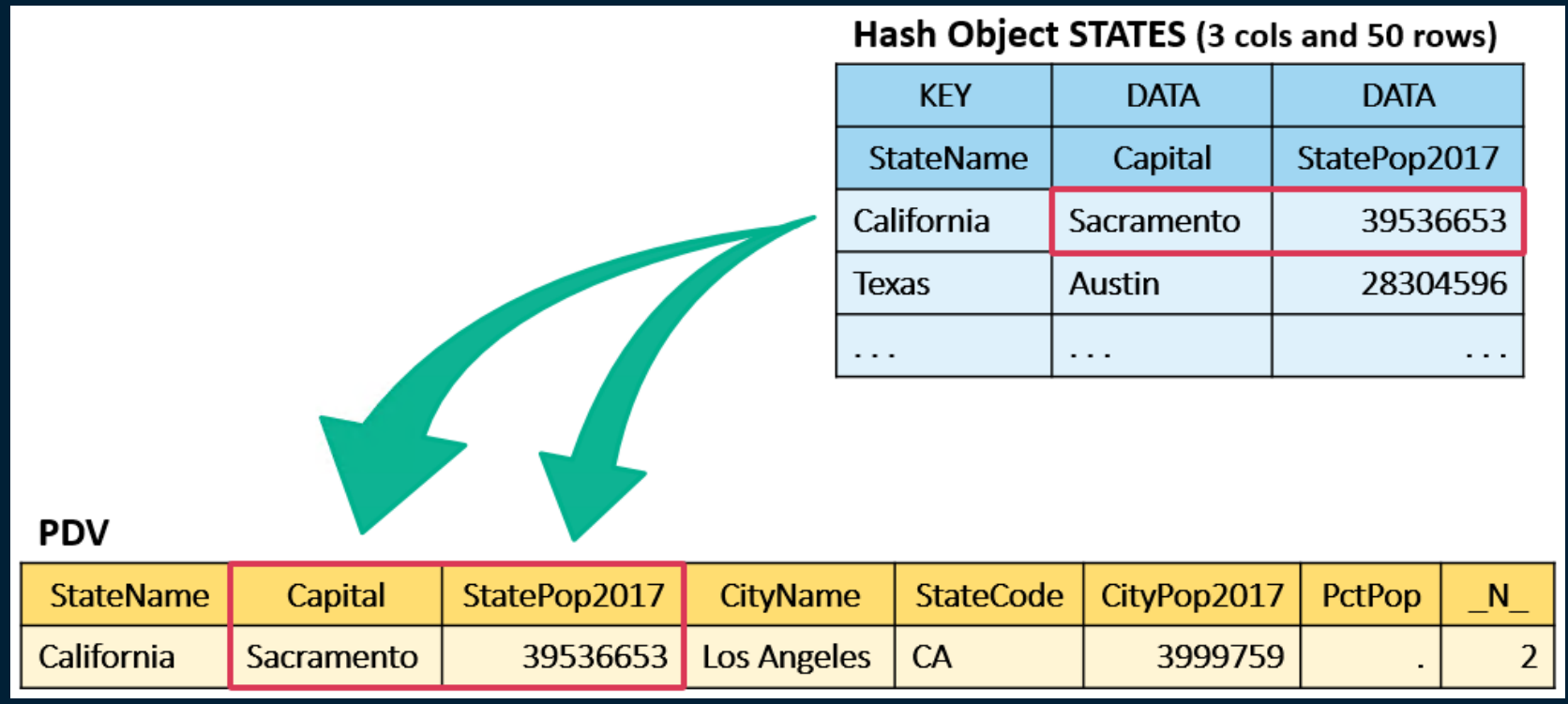
object-name.**FIND(**<**KEY:** *value-1*, … **KEY:** *value-n*>**)**

- returns a return code of zero if the key value (or values) is found (a match)
- returns a return code of a nonzero value if the key value (or values) is not found (a non-match)

The FIND method searches the hash object for the current key value or values.

FIND Method

If the value is found, the FIND method copies the data component values to the PDV.



Match

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

The key value is found.

```
RC=States.find(key:StateName);
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop	_N_
California	Sacramento	39536653	Los Angeles	CA	3999759	0	0.4344	2

Non-Match

Hash Object STATES (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

The key value is ***not*** found.

```
RC=States.find(key:StateName);
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop	_N_
District of Columbia		.	Washington	DC	693972	160038	.	20

KEY Argument

```
States.find()
```

The KEY argument is not needed if the PDV column has the same name as the key component.

```
States.find(key:StateName)
```

The KEY argument is required if the PDV column has a different name than the key component.

```
States.find(key:stnamel(StateCode))
```

An expression can be specified as the value to the KEY argument.

Performing a Hash Table Lookup with The Find Method

```
data work.StateCityPopulation;
    length StateName $ 20 Capital $ 14 StatePop2017 8;
    if _N_=1 then do;
       declare hash States(dataset: 'ATE.population_usstates');
       States.definekey('StateName');
       States.definedata('Capital','StatePop2017');
       States.definedone();
       call missing(StateName, Capital, StatePop2017);
    end;
    set ATE.population_uscities;
    StateName=stnamel(StateCode);

    *PctPop=CityPop2017/StatePop2017;
    *format StatePop2017 comma14. PctPop percent8.1;
run;
```

Performing a Table Lookup with the FIND Method

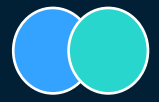
1. Demonstration illustrates declaring and defining a hash object.
2. Demonstration illustrates performing a table lookup by using the FIND method on a hash object.

Handy Links

- Data Step Merge Statement
- Speed Comparison Table Lookups
- Stacking Up - Horizontal or Vertical with PROC SQL or DATA Step – Charu Shankar
- Life Saver Tip for Comparing PROC SQL Join with SAS Data Step Merge – Charu Shankar
- Why choose between SAS Data Step and PROC SQL when you can have both? Charu Shankar

Upcoming 2024

PROC SQL Join with Implicit and Explicit Pass-through



PROC FEDSQL join in SAS® Viya®/ PROC DS2 Merge

Thank You For Your Time

Questions

Contact
Charu Shankar
Senior Technical Training Consultant
SAS Institute, Toronto

EMAIL Charu.shankar@sas.com
BLOG https://blogs.sas.com/content/author/charushankar/
TWITTER CharuYogaCan
LINKEDIN https://www.linkedin.com/in/charushankar/