Paper 1450-2014

Exploring DATA Step Merges and PROC SQL Joins

Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, California

Abstract

Explore the various DATA step merge and PROC SQL join processes. This presentation examines the similarities and differences between merges and joins, and provides examples of effective coding techniques. Attendees examine the objectives and principles behind merges and joins, one-to-one merges (joins), and match-merge (equi-join), as well as the coding constructs associated with inner and outer merges (joins) and PROC SQL set operators.

Introduction

This paper illustrates the similarities and differences between the Base-SAS® software DATA step merge and PROC SQL join techniques. We'll examine two "key" topics that most users are confronted with when working with their tables of data, conditional logic scenarios and merges/joins. This paper introduces brief explanations, guidelines and "simple" techniques for users to consider when confronted with conditional logic scenarios and merges/joins. You are encouraged to explore these and other techniques to make your SAS® experience an exciting one.

Example Tables

The data used in all the examples in this paper consist of a selection of movies that I've viewed over the years. The Movies table contains four character columns: title, category, studio, and rating, and two numeric columns: length and year. The data stored in the Movies table is shown in Figure 1 below.

	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Casablanca	103	Drama	1942	MGM / UA	PG
3	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
4	Coming to America	116	Comedy	1988	Paramount Pictures	R
5	Dracula	130	Horror	1993	Columbia TriStar	R
6	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
7	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13
8	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
9	Jaws	125	Action Adventure	1975	Universal Studios	PG
10	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
11	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
12	Michael	106	Drama	1997	Warner Brothers	PG-13
13	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13
14	Poltergeist	115	Horror	1982	MGM / UA	PG
15	Rocky	120	Action Adventure	1976	MGM / UA	PG
16	Scarface	170	Action Cops & Robber	1983	Universal Studios	R
17	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
18	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
19	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	PG
20	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
21	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
22	Titanic	194	Drama Romance	1997	Paramount Pictures	PG-13

Figure 1. MOVIES Table

The data stored in the ACTORS table consists of three columns: title, actor_leading, and actor_supporting, all of which are defined as character columns. The data stored in the Actors table is illustrated in Figure 2 below.

	Title	Actor_Leading	Actor_Supporting
1	Brave Heart	Mel Gibson	Sophie Marceau
2	Christmas Vacation	Chevy Chase	Beverly D'Angelo
3	Coming to America	Eddie Murphy	Arsenio Hall
4	Forrest Gump	Tom Hanks	Sally Field
5	Ghost	Patrick Swayze	Demi Moore
6	Lethal Weapon	Mel Gibson	Danny Glover
7	Michael	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	Chevy Chase	Beverly D'Angelo
9	Rocky	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	Anthony Hopkins	Jodie Foster
11	The Hunt for Red October	Sean Connery	Alec Baldwin
12	The Terminator	Arnold Schwarzenegge	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet

Figure 2. ACTORS Table

The Process of Merging and Joining

A merge or join is the process of combining two or more tables' side-by-side (horizontally). Its purpose is to gather and manipulate data from across tables for exciting insights into data relationships. The process consists of a matching process between a table's rows bringing together some or all of the tables' contents, as illustrated in Figure 3.

Figure 3. The Process of Merging and Joining Tables

The ability to define relationships between multiple tables and retrieve information based on these relationships is a powerful feature of the relational model. A merge or join of two or more tables provides a means of gathering and manipulating data. Merges and joins are specified on a minimum of two tables at a time, where a column from each table is used for the purpose of connecting the two tables. Connecting columns should have ***"like"*** values and the same column attributes since the processes' success is dependent on these values.

Contrasting Merges and Joins

The difference between a DATA step merge and a join are subtle, but differences do exist.

Merge Features

1. Relevant only to the SAS System – not portable to other vendor data bases.
2. More steps are often needed than with the SQL procedure.
3. Data must first be sorted using by-value.
4. Requires common variable name.
5. Duplicate matching column is automatically overlaid.
6. Results are not automatically printed.

Join Features

1. Portable to other vendor data bases.
2. Data does not need to be sorted using BY-value.
3. Does not require common variable name.
4. Duplicate matching column is not automatically overlaid.
5. Results are automatically printed unless NOPRINT option is specified.

Cartesian Product

A Cartesian Product is defined as a result set of all the possible rows and columns contained in two or more data sets or tables. The DATA step doesn't really lend itself to easily creating a Cartesian Product – PROC SQL is the desired approach. Its most noticeable coding characteristic is the absence of a WHERE-clause. The resulting set of data resulting from a Cartesian Product can be extremely large and unwieldy as illustrated below, that is a set of 286 rows. Although rarely produced, a Cartesian Product join nicely illustrates a base (or internal representation) for all joins.

Code

```
PROC SQL;
 SELECT *
  FROM MOVIES(KEEP=TITLE LENGTH RATING),
     ACTORS(KEEP=TITLE ACTOR_LEADING);
QUIT;
```

Results

The SAS System

Title	Length	Rating	Title	Actor Leading
Brave Heart	177	R	Brave Heart	Mel Gibson
Casablanca	103	PG	Brave Heart	Mel Gibson
Christmas Vacation	97	PG-13	Brave Heart	Mel Gibson
Coming to America	116	R	Brave Heart	Mel Gibson
Dracula	130	R	Brave Heart	Mel Gibson
Dressed to Kill	105	R	Brave Heart	Mel Gibson
Forrest Gump	142	PG-13	Brave Heart	Mel Gibson
Ghost	127	PG-13	Brave Heart	Mel Gibson
Jaws	125	PG	Brave Heart	Mel Gibson
Jurassic Park	127	PG-13	Brave Heart	Mel Gibson
Lethal Weapon	110	R	Brave Heart	Mel Gibson
Michael	106	PG-13	Brave Heart	Mel Gibson
National Lampoon's Vacation	98	PG-13	Brave Heart	Mel Gibson
Poltergeist	115	PG	Brave Heart	Mel Gibson
Rocky	120	PG	Brave Heart	Mel Gibson
Scarface	170	R	Brave Heart	Mel Gibson
...	...	...	< Some Data Omitted >	...	...	...
Forrest Gump	142	PG-13	Titanic	Leonardo DiCaprio
Ghost	127	PG-13	Titanic	Leonardo DiCaprio
Jaws	125	PG	Titanic	Leonardo DiCaprio
Jurassic Park	127	PG-13	Titanic	Leonardo DiCaprio
Lethal Weapon	110	R	Titanic	Leonardo DiCaprio
Michael	106	PG-13	Titanic	Leonardo DiCaprio
National Lampoon's Vacation	98	PG-13	Titanic	Leonardo DiCaprio
Poltergeist	115	PG	Titanic	Leonardo DiCaprio
Rocky	120	PG	Titanic	Leonardo DiCaprio
Scarface	170	R	Titanic	Leonardo DiCaprio
Silence of the Lambs	118	R	Titanic	Leonardo DiCaprio
Star Wars	124	PG	Titanic	Leonardo DiCaprio
The Hunt for Red October	135	PG	Titanic	Leonardo DiCaprio
The Terminator	108	R	Titanic	Leonardo DiCaprio
The Wizard of Oz	101	G	Titanic	Leonardo DiCaprio
Titanic	194	PG-13	Titanic	Leonardo DiCaprio

Match Merging or Joining

Merging or joining two or more tables together is a relatively easy process in the SAS System. The most reliable way to merge or join two or more tables together, and to avoid creating a Cartesian product, is to reduce the resulting set of data using one or more common columns. *The result of a **Matched merge or join** is illustrated by the shaded area (AB) in the Venn diagram in Figure 4 below.*

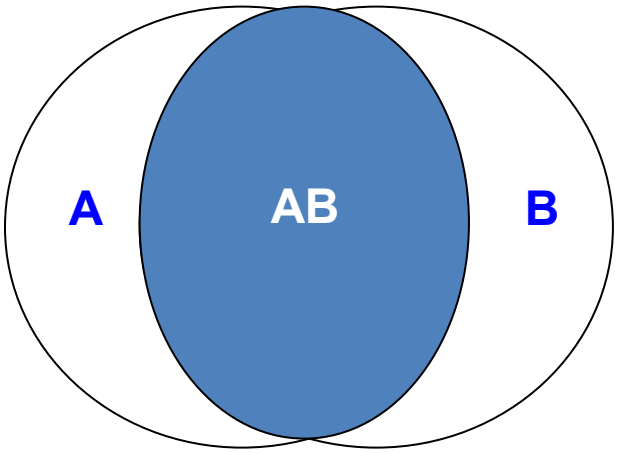


Figure 4. Venn Diagram – Matched Merge or Join

To illustrate how a match merge or join works, the MOVIES and ACTORS tables are linked together using the movie title (TITLE), as the key, as shown in Figure 5.

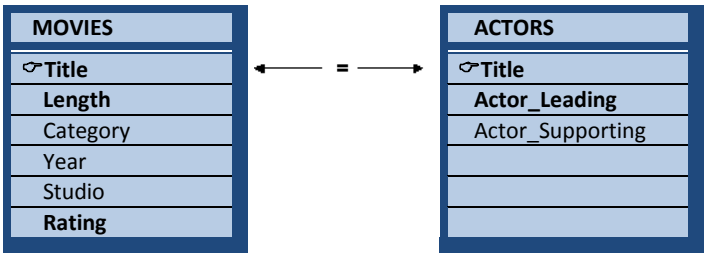


Figure 5. Matched Merge or Join using the MOVIES and ACTORS Tables

The code used to perform a match merge of the MOVIES and ACTORS data sets is shown below.

Match Merge Code

```
PROC SORT DATA=MOVIES;
  BY TITLE;
RUN;

PROC SORT DATA=ACTORS;
 BY TITLE;
RUN;

DATA MERGED;
  MERGE  MOVIES (IN=M KEEP=TITLE LENGTH RATING)
         ACTORS (IN=A KEEP=TITLE ACTOR_LEADING);
  BY TITLE;
  IF M AND A;
RUN;

PROC PRINT DATA=MERGED NOOBS;
RUN;
```

Results

```
                                The SAS System

Title                          Length    Rating    Actor_Leading

Brave Heart                       177    R         Mel Gibson
Christmas Vacation                 97    PG-13     Chevy Chase
Coming to America                 116    R         Eddie Murphy
Forrest Gump                      142    PG-13     Tom Hanks
Ghost                             127    PG-13     Patrick Swayze
Lethal Weapon                     110    R         Mel Gibson
Michael                           106    PG-13     John Travolta
National Lampoon's Vacation        98    PG-13     Chevy Chase
Rocky                             120    PG        Sylvester Stallone
Silence of the Lambs              118    R         Anthony Hopkins
The Hunt for Red October          135    PG        Sean Connery
The Terminator                    108    R         Arnold Schwarzenegge
Titanic                           194    PG-13     Leonardo DiCaprio
```

The SQL procedure code to produce a "matched" row result set from the MOVIES and ACTORS tables is shown below.

SQL Code

```
PROC SQL;
  CREATE  TABLE  JOINED  AS
    SELECT   *
      FROM MOVIES(KEEP=TITLE LENGTH RATING),
           ACTORS(KEEP=TITLE ACTOR_LEADING)
      WHERE  MOVIES.TITLE = ACTORS.TITLE;

   SELECT * FROM JOINED;
QUIT;
```

Results

```
                                The SAS System

Title                          Length    Rating    Actor_Leading

Brave Heart                       177    R         Mel Gibson
Christmas Vacation                 97    PG-13     Chevy Chase
Coming to America                 116    R         Eddie Murphy
Forrest Gump                      142    PG-13     Tom Hanks
Ghost                             127    PG-13     Patrick Swayze
Lethal Weapon                     110    R         Mel Gibson
Michael                           106    PG-13     John Travolta
National Lampoon's Vacation        98    PG-13     Chevy Chase
Rocky                             120    PG        Sylvester Stallone
Silence of the Lambs              118    R         Anthony Hopkins
The Hunt for Red October          135    PG        Sean Connery
The Terminator                    108    R         Arnold Schwarzenegge
Titanic                           194    PG-13     Leonardo DiCaprio
```

Asymmetrical Merging and Joining

A typical merge or join consists of a process of relating rows in one table with rows in another symmetrically. But occasionally, rows from one or both tables that have no related rows can be retained. This approach is sometimes referred to as an asymmetric type of join because its primary purpose is row preservation. This type of processing is a significant feature offered by the outer join construct.

There are syntax and operational differences between inner (natural) and outer joins. The obvious difference between an inner and outer join is the way the syntax is constructed. Outer joins use keywords such as LEFT JOIN, RIGHT JOIN, and FULL JOIN, and has the WHERE clause replaced with an ON clause. These distinctions help identify outer joins from inner joins. But, there are operational differences as well.

Unlike an inner join, the maximum number of tables that can be specified in an outer join construct is two. Similar to an inner join, an outer join relates rows in both tables. But this is where the similarities end because the resulting set of data also includes rows with no related rows from one or both of the tables. This special handling of "matched" and "unmatched" rows of data is what differentiates a symmetric inner join from an asymmetric outer join. Essentially the resulting set of data from an outer join process contains rows that "match" the ON-clause plus any "unmatched" rows from the left, right, or both tables.

The result of a ***Left Outer merge or join*** is illustrated by the shaded areas (A and AB) in the Venn diagram illustrated in Figure 6.

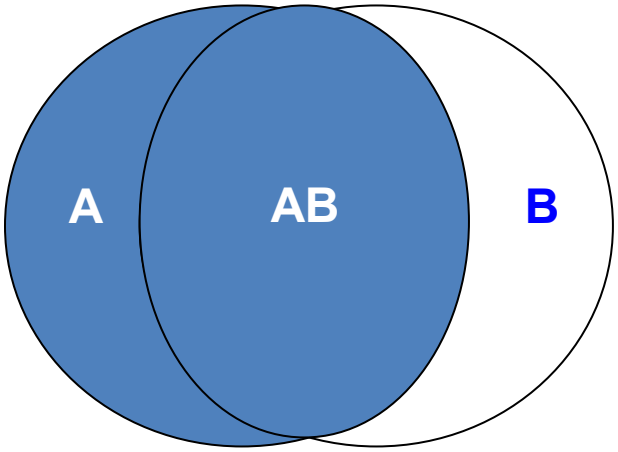


Figure 6. Venn Diagram – Left Outer Merge or Join

Left Outer Merge or Join

The result of a Left Outer merge or join produces matched rows from both tables while preserving all unmatched rows from the left table. The following merge code illustrates a left outer merge construct that selects "matched" movies based on their titles from the MOVIES and ACTORS tables, plus all "unmatched" movies from the MOVIES table.

Merge Code

```
PROC SORT DATA=MOVIES;
  BY TITLE;
RUN;

PROC SORT DATA=ACTORS;
  BY TITLE;
RUN;

DATA LEFT_OUTER_MERGE;
   MERGE  MOVIES (IN=M KEEP=TITLE LENGTH RATING)
          ACTORS (IN=A KEEP=TITLE ACTOR_LEADING);
     BY  TITLE;
       IF M;
RUN;

PROC PRINT DATA=LEFT_OUTER_MERGE NOOBS;
RUN;
```

Results

The SAS System

Title	Length	Rating	Actor_Leading
Brave Heart	177	R	Mel Gibson
Casablanca	103	PG	
Christmas Vacation	97	PG-13	Chevy Chase
Coming to America	116	R	Eddie Murphy
Dracula	130	R	
Dressed to Kill	105	R	
Forrest Gump	142	PG-13	Tom Hanks
Ghost	127	PG-13	Patrick Swayze
Jaws	125	PG	
Jurassic Park	127	PG-13	
Lethal Weapon	110	R	Mel Gibson
Michael	106	PG-13	John Travolta
National Lampoon's Vacation	98	PG-13	Chevy Chase
Poltergeist	115	PG	
Rocky	120	PG	Sylvester Stallone
Scarface	170	R	
Silence of the Lambs	118	R	Anthony Hopkins
Star Wars	124	PG	
The Hunt for Red October	135	PG	Sean Connery
The Terminator	108	R	Arnold Schwarzenegge
The Wizard of Oz	101	G	
Titanic	194	PG-13	Leonardo DiCaprio

The corresponding SQL procedure code to produce a left outer join row result set is shown below.

SQL Code

```
PROC SQL;
  CREATE  TABLE  LEFT_OUTER_JOIN  AS
    SELECT   *
      FROM MOVIES(KEEP=TITLE LENGTH RATING)
        LEFT JOIN
            ACTORS(KEEP=TITLE ACTOR_LEADING)
        ON MOVIES.TITLE = ACTORS.TITLE;

  SELECT * FROM LEFT_OUTER_JOIN;
QUIT;
```

Results

The SAS System

Title	Length	Rating	Actor Leading
Brave Heart	177	R	Mel Gibson
Casablanca	103	PG	
Christmas Vacation	97	PG-13	Chevy Chase
Coming to America	116	R	Eddie Murphy
Dracula	130	R	
Dressed to Kill	105	R	
Forrest Gump	142	PG-13	Tom Hanks
Ghost	127	PG-13	Patrick Swayze
Jaws	125	PG	
Jurassic Park	127	PG-13	
Lethal Weapon	110	R	Mel Gibson
Michael	106	PG-13	John Travolta
National Lampoon's Vacation	98	PG-13	Chevy Chase
Poltergeist	115	PG	

```
Rocky                          120     PG        Sylvester Stallone
Scarface                       170     R
Silence of the Lambs           118     R         Anthony Hopkins
Star Wars                      124     PG
The Hunt for Red October       135     PG        Sean Connery
The Terminator                 108     R         Arnold Schwarzenegge
The Wizard of Oz               101     G
Titanic                        194     PG-13     Leonardo DiCaprio
```

The result of a ***Right Outer merge or join*** is illustrated by the shaded areas (B and AB) in the Venn diagram in Figure 7.

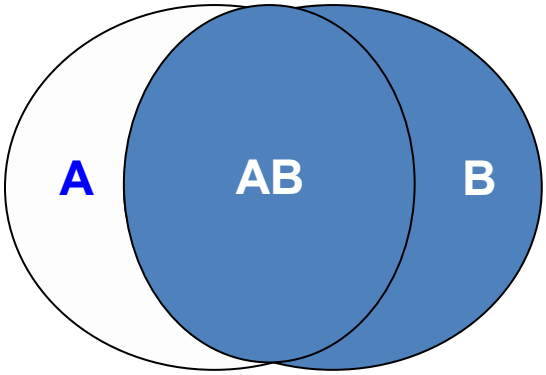


Figure 7. Venn Diagram – Right Outer Merge or Join

Right Outer Merge or Join

The result of a Right Outer merge or join produces matched rows from both tables while preserving all unmatched rows from the right table. The following merge code illustrates a right outer merge construct that selects "matched" movies based on their titles from the MOVIES and ACTORS tables, plus all "unmatched" movies from the ACTORS table.

Merge Code

```
PROC SORT DATA=MOVIES;
 BY TITLE;
RUN;

PROC SORT DATA=ACTORS;
 BY TITLE;
RUN;

DATA RIGHT_OUTER_MERGE;
  MERGE  MOVIES (IN=M KEEP=TITLE LENGTH RATING)
         ACTORS (IN=A KEEP=TITLE ACTOR_LEADING);
   BY  TITLE;
    IF A;
RUN;

PROC PRINT DATA=RIGHT_OUTER_MERGE NOOBS;
RUN;
```

Results

```
                                The SAS System

Title                          Length    Rating    Actor Leading

Brave Heart                       177    R         Mel Gibson
Christmas Vacation                 97    PG-13     Chevy Chase
Coming to America                 116    R         Eddie Murphy
Forrest Gump                      142    PG-13     Tom Hanks
Ghost                             127    PG-13     Patrick Swayze
Lethal Weapon                     110    R         Mel Gibson
Michael                           106    PG-13     John Travolta
National Lampoon's Vacation        98    PG-13     Chevy Chase
Rocky                             120    PG        Sylvester Stallone
Silence of the Lambs              118    R         Anthony Hopkins
The Hunt for Red October          135    PG        Sean Connery
The Terminator                    108    R         Arnold Schwarzenegge
Titanic                           194    PG-13     Leonardo DiCaprio
```

The corresponding SQL procedure code produces a right outer join row result set as shown below.

SQL Code

```
PROC SQL;
  CREATE  TABLE  RIGHT_OUTER_JOIN  AS
    SELECT   *
      FROM MOVIES(KEEP=TITLE LENGTH RATING)
        RIGHT JOIN
            ACTORS(KEEP=TITLE ACTOR_LEADING)
        ON MOVIES.TITLE = ACTORS.TITLE;

  SELECT * FROM RIGHT_OUTER_JOIN;
QUIT;
```

Results

```
                                The SAS System

Title                          Length    Rating    Actor Leading

Brave Heart                       177    R         Mel Gibson
Christmas Vacation                 97    PG-13     Chevy Chase
Coming to America                 116    R         Eddie Murphy
Forrest Gump                      142    PG-13     Tom Hanks
Ghost                             127    PG-13     Patrick Swayze
Lethal Weapon                     110    R         Mel Gibson
Michael                           106    PG-13     John Travolta
National Lampoon's Vacation        98    PG-13     Chevy Chase
Rocky                             120    PG        Sylvester Stallone
Silence of the Lambs              118    R         Anthony Hopkins
The Hunt for Red October          135    PG        Sean Connery
The Terminator                    108    R         Arnold Schwarzenegge
Titanic                           194    PG-13     Leonardo DiCaprio
```

An Introduction to Hash Programming

One of the more exciting and relevant programming techniques available to SAS users today is the Hash object. Available as a DATA step construct, users are able to construct relatively simple code to perform match-merge and/or join operations. The purpose of this paper and presentation is to introduce the basics of what a hash table is and to illustrate practical applications so SAS users everywhere can begin to take advantage of this powerful Base-SAS programming feature.

What is a Hash Object?

A hash object is a data structure that contains an array of items that are used to map identifying values, known as keys (e.g., employee IDs), to their associated values (e.g., employee names or employee addresses). As implemented, it is designed as a DATA step construct and is not available to any SAS PROCedures. The behavior of a hash object is similar to that of a SAS array in that the columns comprising it can be saved to a SAS table, but at the end of the DATA step the hash object and all its contents disappear.

How Does a Hash Object Work?

A hash object permits table lookup operations to be performed considerably faster than other available methods found in the SAS system. Unlike a DATA step merge or PROC SQL join where the SAS system repeatedly accesses the contents of a table stored on disk to perform table lookup operations, a hash object reads the contents of a table into memory once allowing the SAS system to repeatedly access it, as necessary. Since memory-based operations are typically faster than their disk-based counterparts, users generally experience faster and more efficient table lookup operations. Figure 8 illustrates the process of performing a table lookup using the Movie Title (i.e., key) in the MOVIES table matched against the Movie Title (i.e., key) in the ACTORS table to return the ACTOR_LEADING and ACTOR_SUPPORTING information.

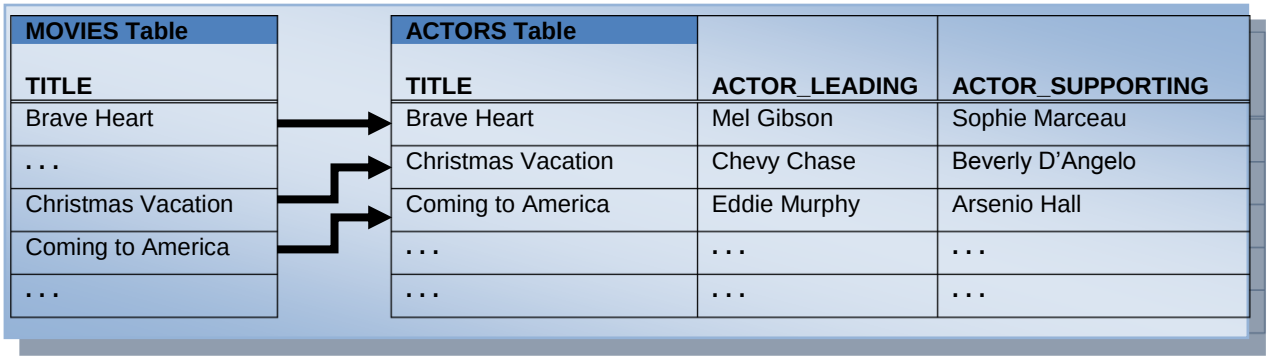


Figure 8. Table Lookup Operation with Simple Key

Although one or more hash tables may be constructed in a single DATA step that reads data into memory, users may experience insufficient memory conditions preventing larger tables from being successfully processed. To alleviate this kind of issue, users may want to load the smaller tables as hash tables and continue to sequentially process larger tables containing lookup keys.

Hash Object Syntax

Users with DATA step programming experience will find the hash object syntax relatively straight forward to learn and use. Available in all operating systems running SAS 9 or greater, the hash object is called using methods. The syntax for calling a method involves specifying the name of the user-assigned hash table, a dot (.), the desired method (e.g., operation) by name, and finally the specification for the method enclosed in parentheses. The following example illustrates the basic syntax for calling a method to define a key.

```
HashTitles.DefineKey ('Title');
```

where:

HashTitles is the name of the hash table, DefineKey is the name of the called method, and 'Title' is the specification being passed to the method.

Hash Object Methods

In SAS 9, the author has identified twenty six (26) known methods. Figure 9 illustrates an alphabetical list of the available methods.

Method	Description
ADD	Adds data associated with key to hash object.
CHECK	Checks whether key is stored in hash object.
CLEAR	Removes all items from a hash object without deleting hash object.
DEFINEDATA	Defines data to be stored in hash object.
DEFINEDONE	Specifies that all key and data definitions are complete.
DEFINEKEY	Defines key variables to the hash object.
DELETE	Deletes the hash or hash iterator object.
EQUALS	Determines whether two hash objects are equal.
FIND	Determines whether the key is stored in the hash object.
FIND_NEXT	The current list item in the key's multiple item list is set to the next item.
FIND_PREV	The current list item in the key's multiple item list is set to the previous item.
FIRST	Returns the first value in the hash object.
HAS_NEXT	Determines whether another item is available in the current key's list.
HAS_PREV	Determines whether a previous item is available in the current key's list.
LAST	Returns the last value in the hash object.
NEXT	Returns the next value in the hash object.
OUTPUT	Creates one or more data sets containing the data in the hash object.
PREV	Returns the previous value in the hash object.
REF	Combines the FIND and ADD methods into a single method call.
REMOVE	Removes the data associated with a key from the hash object.
REMOVEDUP	Removes the data associated with a key's current data item from the hash object.
REPLACE	Replaces the data associated with a key with new data.
REPLACEDUP	Replaces data associated with a key's current data item with new data.
SETCUR	Specifies a starting key item for iteration.
SUM	Retrieves a summary value for a given key from the hash table and stores the value to a DATA step variable.
SUMDUP	Retrieves a summary value for the key's current data item and stores the value to a DATA step variable.

Figure 9. Alphabetical Listing of Hash Object Methods

Match Merge with Hash

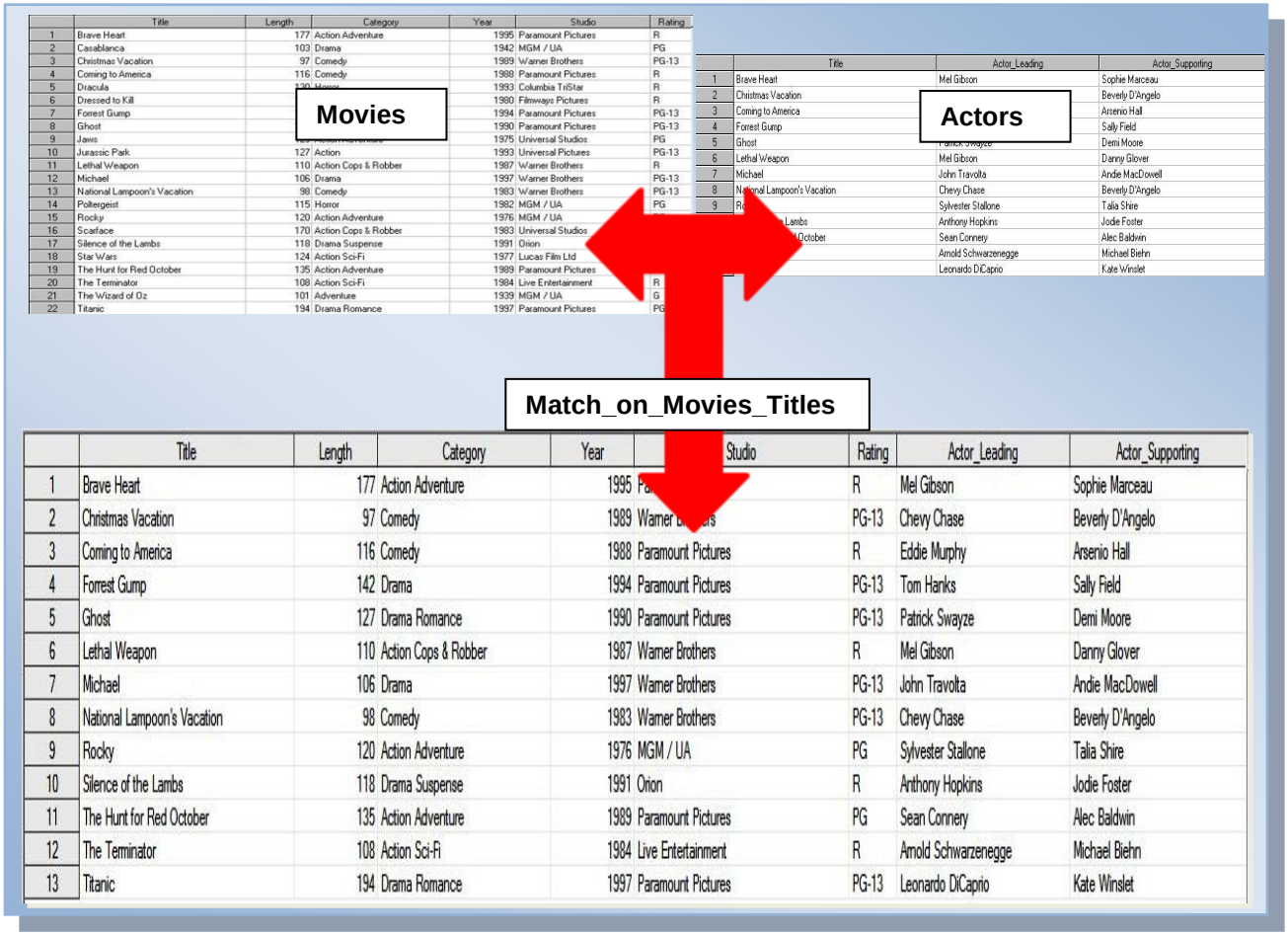
A match merge can be performed with a DATA step hash construct. The hash object, as implemented in the DATA step, provides users with an approach to match-merge (or join) two or more tables together. An important distinction with the hash approach is that data does not have to be sorted or be in a designated sort order before use as it does with the DATA step merge process. The following code illustrates a hash object with a simple key (TITLE) to merge (or join) the MOVIES and ACTORS tables to create a new table (MATCH_ON_MOVIE_TITLES) with matched observations.

DATA Step Hash Code

```
data match_on_movie_titles(drop=rc);

❶ if 0 then set movies actors;  /* load variable properties into hash tables */

  if _n_ = 1 then do;
❷   declare Hash MatchTitles (dataset:'actors'); /* declare the name MatchTitles
                                                    for hash */

❸   MatchTitles.DefineKey ('Title'); /* identify variable to use as key */
    MatchTitles.DefineData ('Actor_Leading',
                            'Actor_Supporting'); /* identify columns of data */
     MatchTitles.DefineDone (); /* complete hash table definition */
   end;

   set movies;

❹ if MatchTitles.find(key:title) = 0 then output;  /* lookup TITLE in MOVIES table
                                                      using MatchTitles hash */
```

Results

The match-merge (or join) process is illustrated below.



	Title	Length	Category	Year	Studio	Rating	Actor_Leading	Actor_Supporting
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R	Mel Gibson	Sophie Marceau
2	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13	Chevy Chase	Beverly D'Angelo
3	Coming to America	116	Comedy	1988	Paramount Pictures	R	Eddie Murphy	Arsenio Hall
4	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13	Tom Hanks	Sally Field
5	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13	Patrick Swayze	Demi Moore
6	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R	Mel Gibson	Danny Glover
7	Michael	106	Drama	1997	Warner Brothers	PG-13	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13	Chevy Chase	Beverly D'Angelo
9	Rocky	120	Action Adventure	1976	MGM / UA	PG	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	118	Drama Suspense	1991	Orion	R	Anthony Hopkins	Jodie Foster
11	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	PG	Sean Connery	Alec Baldwin
12	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R	Arnold Schwarzenegge	Michael Biehn
13	Titanic	194	Drama Romance	1997	Paramount Pictures	PG-13	Leonardo DiCaprio	Kate Winslet

Conclusion

The Base-SAS DATA step and SQL procedure are wonderful languages for SAS users to explore and use in a variety of application situations. This paper has presented explanations, guidelines and "simple" techniques for users to consider when confronted with conditional logic scenarios and merges/joins. You are encouraged to explore these and other techniques to make your SAS experience an exciting one.

References

Lafler, Kirk Paul (2013). *PROC SQL: Beyond the Basics Using SAS, Second Edition*, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul (2012), *"Exploring DATA Step Merges and PROC SQL Joins,"* Proceedings of the 2012 SAS Global Forum (SGF) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2011), *"Exploring DATA Step Merges and PROC SQL Joins,"* Proceedings of the 2011 PharmaSUG Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2010), *"DATA Step and PROC SQL Programming Techniques,"* Ohio SAS Users Group (OSUG) 2010 One-Day Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"DATA Step and PROC SQL Programming Techniques,"* South Central SAS Users Group (SCSUG) 2009 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"DATA Step versus PROC SQL Programming Techniques,"* Sacramento Valley SAS Users Group 2009 Meeting, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul, Advanced SAS® Programming Tips and Techniques; Software Intelligence Corporation, Spring Valley, CA, USA; 1987-2007.

Lafler, Kirk Paul (2007), *"Undocumented and Hard-to-find PROC SQL Features,"* Proceedings of the PharmaSUG 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul and Ben Cochran (2007), *"A Hands-on Tour Inside the World of PROC SQL Features,"* Proceedings of the SAS Global Forum (SGF) 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, and The Bedford Group, USA.

Lafler, Kirk Paul (2006), *"A Hands-on Tour Inside the World of PROC SQL,"* Proceedings of the 31st Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2005), *"Manipulating Data with PROC SQL,"* Proceedings of the 30th Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2004). *PROC SQL: Beyond the Basics Using SAS*, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul (2003), "*Undocumented and Hard-to-find PROC SQL Features*," Proceedings of the Eleventh Annual Western Users of SAS Software Conference.

Acknowledgments

I want to thank Marje Fecht, SGF 2014 Conference Chair and the Hands-On Workshop (HOW) Section Chairs for accepting my abstract and paper, as well as the SGF Executive Committee, Conference Leaders, and SAS Institute for organizing a great conference!

Trademark Citations

Author Information

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been using SAS since 1979. He is a SAS Certified Professional, provider of IT consulting services, trainer to SAS users around the world, and sasCommunity.org emeritus Advisory Board member. As the author of five books including PROC SQL: Beyond the Basics Using SAS, Second Edition (SAS Press 2013), Kirk has written more than five hundred papers and articles, been an Invited speaker and trainer at four hundred-plus SAS International, regional, special-interest user, local and in-house group conferences and meetings, and is the recipient of 23 “Best” contributed paper, hands-on workshop (HOW), and poster awards.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Senior SAS Consultant, Application Developer, Data Analyst, Trainer and Author
Software Intelligence Corporation
E-mail: KirkLafler@cs.com
LinkedIn: http://www.linkedin.com/in/KirkPaulLafler
Twitter: @sasNerd