

DATA Step and PROC SQL Programming Techniques

Kirk Paul Lafler, Software Intelligence Corporation

ABSTRACT

Are you considering whether to use a DATA step or PROC SQL step in your next project? This presentation explores the similarities and differences between DATA step and PROC SQL programming techniques. Topics include IF-THEN-ELSE, SELECT-WHEN, and PROC SQL CASE expressions conditional logic concepts and constructs; and the techniques for constructing effective merges and joins. Attendees explore examples that contrast DATA step versus PROC SQL programming techniques to conduct conditional logic scenarios, one-to-one match-merges and match-joins, and an assortment of inner and outer join programming constructs.

INTRODUCTION

This paper illustrates the similarities and differences between the Base-SAS software DATA step and SQL procedure. 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 below.

MOVIES Table

	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

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 below.

ACTORS Table

	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 Schwarzenegger	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet

CONDITIONAL LOGIC SCENARIOS

A powerful feature of the SAS software as a programming language is its ability to perform different actions depending on whether a programmer-specified condition evaluates to true or false. The method of accomplishing this is to use one or more conditional statements, conditional expressions, and conditional constructs to construct a level of intelligence in a program or application. In this section, we'll discuss and illustrate the various conditional logic scenarios IF-THEN / ELSE, SELECT, and CASE Expressions available in the DATA step and PROC SQL.

CONDITIONAL LOGIC WITH IF-THEN / ELSE

The IF-THEN / ELSE construct is available to users for logic scenarios in a DATA step. Its purpose is to enable a sequence of conditions to be assigned that when executed proceeds through the sequence of IF-THEN / ELSE conditions until either a match in an expression is found or until all conditions are exhausted. The example shows a character variable `Movie_Length` being assigned a value of either "Shorter Length", "Average Length", or "Longer Length" based on the mutually exclusive conditions specified in the IF-THEN and ELSE conditions. Although not required, an ELSE condition, when present, is an effective technique for continuing processing to the next specified condition when a match is not found for the current condition. An ELSE condition can also be useful as a "catch-all" to prevent a missing value from being assigned.

Code:

```
DATA IF_THEN_EXAMPLE;  
  ATTRIB Movie_Length LENGTH=$14 LABEL='Movie Length';  
  SET MOVIES;  
  IF LENGTH < 120 THEN Movie_Length = 'Shorter Length';  
  ELSE IF LENGTH > 160 THEN Movie_Length = 'Longer Length';  
  ELSE Movie_Length = 'Average Length';  
RUN;  
PROC PRINT DATA=IF_THEN_EXAMPLE NOOBS;  
  VAR TITLE LENGTH Movie_Length;  
RUN;
```

Results

The SAS System		
Title	Length	Movie_Length
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

CONDITIONAL LOGIC WITH SELECT

Another form of conditional logic available to users is a **SELECT** statement. Its purpose is to enable a sequence of logic conditions to be constructed in a DATA step by specifying one or more **WHEN** conditions and an optional **OTHERWISE** condition. When executed, processing continues through each WHEN condition until a match is found that satisfies the specified expression. Typically one or more WHEN conditions are specified in descending frequency order representing a series of conditions. The next example shows a value based on the mutually exclusive conditions specified in the sequence of logic conditions of "Shorter Length", "Average Length", or "Longer Length" being assigned to the character variable Movie_Length. Although not required, the OTHERWISE condition can be useful in the assignment of a specific value or as a "catch-all" to prevent a missing value from being assigned.

Code:

```
DATA SELECT_EXAMPLE;  
  ATTRIB Movie_Length LENGTH=$14 LABEL='Movie Length';  
  SET MOVIES;  
  SELECT;  
    WHEN (LENGTH < 120) Movie_Length = 'Shorter Length';  
    WHEN (LENGTH > 160) Movie_Length = 'Longer Length';  
    OTHERWISE Movie_Length = 'Average Length';  
  END;  
RUN;  
PROC PRINT DATA=SELECT_EXAMPLE NOOBS;  
  VAR TITLE LENGTH Movie_Length;  
RUN;
```

Results

The SAS System		
Title	Length	Movie_Length
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

CONDITIONAL LOGIC WITH CASE EXPRESSIONS

Another form of conditional logic available to users is a case expression. Its purpose is to provide a way of conditionally selecting result values from each row in a table (or view). Similar to an IF-THEN/ELSE or SELECT construct in the DATA step, a case expression can only be specified in the SQL procedure. It supports a WHEN-THEN clause to conditionally process some but not all the rows in a table. An optional ELSE expression can be specified to handle an alternative action should none of the expression(s) identified in the WHEN condition(s) not be satisfied. A case expression must be a valid SQL expression and conform to syntax rules similar to DATA step SELECT-WHEN statements. Even though this topic is best explained by example, a quick look at the syntax follows.

```
CASE <column-name>
  WHEN when-condition THEN result-expression
  <WHEN when-condition THEN result-expression> ...
  <ELSE result-expression>
END
```

A column-name can optionally be specified as part of the CASE-expression. If present, it is automatically made available to each when-condition. When it is not specified, the column-name must be coded in each when-condition. Let's examine how a case expression works.

If a when-condition is satisfied by a row in a table (or view), then it is considered "true" and the result-expression following the THEN keyword is processed. The remaining WHEN conditions in the CASE expression are skipped. If a when-condition is "false", the next when-condition is evaluated. SQL evaluates each when-condition until a "true" condition is found or in the event all when-conditions are "false", it then executes the ELSE expression and assigns its value to the CASE expression's result. A missing value is assigned to a CASE expression when an ELSE expression is not specified and each when-condition is "false".

In the next example, a simple case expression is illustrated. The next example shows a character variable Movie_Length being assigned with the AS keyword. Assigned values based on the mutually exclusive conditions specified in the sequence of logic conditions of either "Shorter Length" for movie lengths less than 120 minutes, "Longer Length" for movie lengths greater than 160 minutes, or "Average Length" for all other movie lengths. Although not required, an ELSE condition can be useful in the assignment of a specific value or as a "catch-all" to prevent a missing value from being assigned.

SQL Code

```
PROC SQL;
  SELECT TITLE,
         LENGTH,
         CASE
           WHEN LENGTH < 120 THEN 'Shorter Length'
           WHEN LENGTH > 160 THEN 'Longer Length'
           ELSE 'Average Length'
         END AS Movie_Length
  FROM MOVIES;
QUIT;
```

Results

The SAS System		
Title	Length	Movie Length
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

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 below.



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;
```

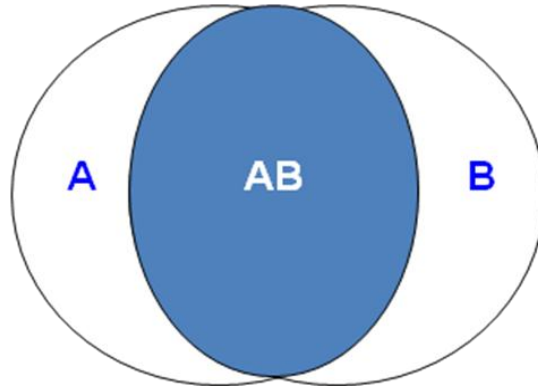
Results

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

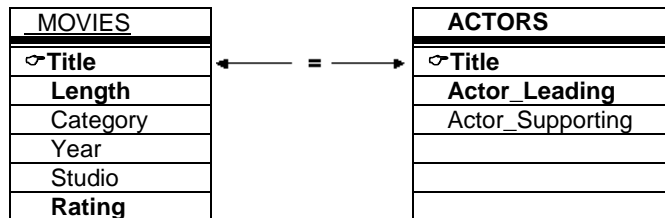
MATCH MERGING OR JOINING

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 following Venn diagram.*

Venn Diagram – Matched Merge or Join



To illustrate how a match merge or join works, two tables are linked together using the movie title (TITLE) in the following diagram.



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 Schwarzenegger
Titanic	194	PG-13	Leonardo DiCaprio

The corresponding SQL procedure code to produce a "matched" row result set 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			
<u>Title</u>	<u>Length</u>	<u>Rating</u>	<u>Actor Leading</u>
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 Schwarzenegger
Titanic	194	PG-13	Leonardo DiCaprio

ASYMMETRIC MERGING AND JOINING

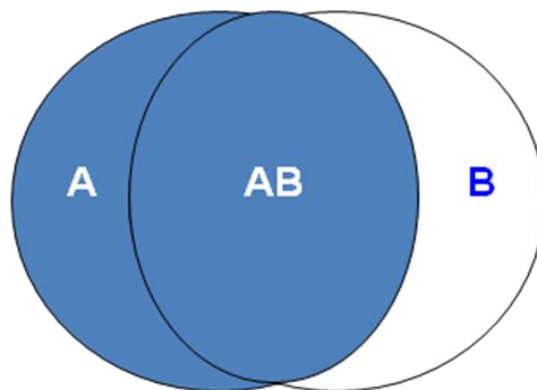
A typical merge or join consists of a process of relating rows in one table with rows in another symmetrically. But occasionally, rows from one or both tables that have no related rows can be retained. This approach is sometimes referred to as an asymmetric type of join because its primary purpose is 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 following Venn diagram.

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 Schwarzenegger
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;

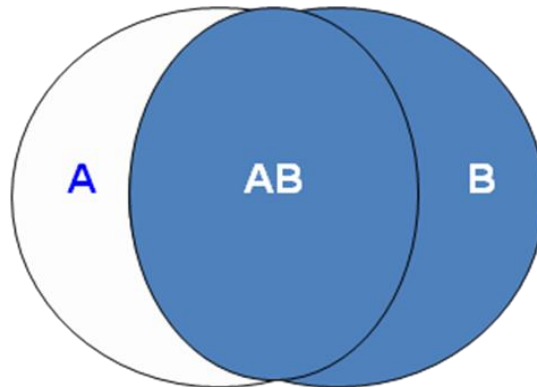
  SELECT * FROM LEFT_OUTER_JOIN;
QUIT;
```

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 Schwarzenegger
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 following Venn diagram.

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 Schwarzenegger	
Titanic	194	PG-13	Leonardo DiCaprio	

The corresponding SQL procedure code to produce a right outer join row result set is 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			
<u>Title</u>	<u>Length</u>	<u>Rating</u>	<u>Actor Leading</u>
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 Schwarzenegger
Titanic	194	PG-13	Leonardo DiCaprio

CONCLUSION

The Base-SAS DATA step and PROC 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 (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*.

Lafler, Kirk Paul, PROC SQL Programming for Beginners; Software Intelligence Corporation, Spring Valley, CA, USA; 1992-2007.

Lafler, Kirk Paul, Intermediate PROC SQL Programming; Software Intelligence Corporation, Spring Valley, CA, USA; 1998-2007.

Lafler, Kirk Paul, Advanced PROC SQL Programming; Software Intelligence Corporation, Spring Valley, CA, USA; 2001-2007.

Lafler, Kirk Paul, PROC SQL Programming Tips; Software Intelligence Corporation, Spring Valley, CA, USA; 2002-2007.

SAS[®] Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition; SAS Institute, Cary, NC, USA; 1990.

SAS[®] SQL Procedure User's Guide, Version 8; SAS Institute Inc., Cary, NC, USA; 2000.

TRADEMARK CITATIONS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

ABOUT THE AUTHOR

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been using SAS since 1979. Kirk provides IT consulting services and training to SAS users around the world. As a SAS Certified Professional, Kirk has written four books including PROC SQL: Beyond the Basics Using SAS, and more than four hundred peer-reviewed papers and articles. He has also been an Invited speaker and trainer at more than three hundred SAS International, regional, local, and special-interest user group conferences and meetings throughout North America. His popular SAS Tips column, "Kirk's Korner of Quick and Simple Tips", appears regularly in several SAS User Group newsletters and Web sites, and his fun-filled SASword Puzzles is featured in SAScommunity.org.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Software Intelligence Corporation
World Headquarters
P.O. Box 1390
Spring Valley, California 91979-1390
E-mail: KirkLafler@cs.com

