Tutorial on SQL and Data Federation for Genomics Researchers

a presentation by

Kirk Paul Lafler
SAS® Consultant, Author, and Trainer
E-mail: KirkLafler@cs.com
Copyright © Kirk Paul Lafler, 1992-2010. All rights reserved.

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. SAS Certified Professional is the trademark of SAS Institute Inc., Cary, NC, USA.

All other company and product names mentioned are used for identification purposes only and may be trademarks of their respective owners.
Objectives

• What is data federation?
• Characteristics associated with data federation
• What is a join?
• Why join?
• SAS® DATA step merge versus a Join
• Cartesian product joins
• Two table joins
• Table aliases to reference tables
• Three table joins
• Left and Right outer joins
• What happens during a join?
• Available join algorithms
# Tables Used in Examples

<table>
<thead>
<tr>
<th>Title</th>
<th>Length</th>
<th>Category</th>
<th>Year</th>
<th>Studio</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brave Heart</td>
<td>177</td>
<td>Action Adventure</td>
<td>1995</td>
<td>Paramount Pictures</td>
<td>R</td>
</tr>
<tr>
<td>Casablanca</td>
<td>103</td>
<td>Drama</td>
<td>1942</td>
<td>MGM / UA</td>
<td>PG</td>
</tr>
<tr>
<td>Christmas Vacation</td>
<td>97</td>
<td>Comedy</td>
<td>1989</td>
<td>Warner Brothers</td>
<td>PG-13</td>
</tr>
<tr>
<td>Coming to America</td>
<td>116</td>
<td>Comedy</td>
<td>1980</td>
<td>Paramount Pictures</td>
<td>R</td>
</tr>
<tr>
<td>Dracula</td>
<td>130</td>
<td>Horror</td>
<td>1993</td>
<td>Columbia TriStar</td>
<td>R</td>
</tr>
<tr>
<td>Dressed to Kill</td>
<td>106</td>
<td>Drama Mystery</td>
<td>1980</td>
<td>Filmways Pictures</td>
<td>R</td>
</tr>
<tr>
<td>Forrest Gump</td>
<td>142</td>
<td>Drama</td>
<td>1994</td>
<td>Paramount Pictures</td>
<td>PG-13</td>
</tr>
<tr>
<td>Ghost</td>
<td>127</td>
<td>Drama Romance</td>
<td>1990</td>
<td>Paramount Pictures</td>
<td>PG-13</td>
</tr>
<tr>
<td>Jaws</td>
<td>125</td>
<td>Action Adventure</td>
<td>1975</td>
<td>Universal Studios</td>
<td>PG</td>
</tr>
<tr>
<td>Jurassic Park</td>
<td>127</td>
<td>Action</td>
<td>1993</td>
<td>Universal Pictures</td>
<td>PG-13</td>
</tr>
<tr>
<td>Lethal Weapon</td>
<td>110</td>
<td>Action Cop &amp; Robber</td>
<td>1987</td>
<td>Warner Brothers</td>
<td>R</td>
</tr>
<tr>
<td>Michael</td>
<td>106</td>
<td>Drama</td>
<td>1997</td>
<td>Warner Brothers</td>
<td>PG-13</td>
</tr>
<tr>
<td>National Lampoon's Vacation</td>
<td>98</td>
<td>Comedy</td>
<td>1993</td>
<td>Warner Brothers</td>
<td>PG-13</td>
</tr>
<tr>
<td>Potsergeist</td>
<td>115</td>
<td>Horror</td>
<td>1982</td>
<td>MGM / UA</td>
<td>PG</td>
</tr>
<tr>
<td>Rocky</td>
<td>120</td>
<td>Action Adventure</td>
<td>1976</td>
<td>MGM / UA</td>
<td>PG</td>
</tr>
<tr>
<td>Scarface</td>
<td>170</td>
<td>Action Cop &amp; Robber</td>
<td>1963</td>
<td>Universal Studios</td>
<td>R</td>
</tr>
<tr>
<td>Silence of the Lambs</td>
<td>118</td>
<td>Drama Suspense</td>
<td>1951</td>
<td>Brian</td>
<td>R</td>
</tr>
<tr>
<td>Star Wars</td>
<td>124</td>
<td>Action Sci-Fi</td>
<td>1977</td>
<td>Lucas Film Ltd</td>
<td>PG</td>
</tr>
<tr>
<td>The Hunt for Red October</td>
<td>135</td>
<td>Action Adventure</td>
<td>1969</td>
<td>Paramount Pictures</td>
<td>PG</td>
</tr>
<tr>
<td>The Wizard of Oz</td>
<td>101</td>
<td>Adventure</td>
<td>1995</td>
<td>MGM / UA</td>
<td>G</td>
</tr>
<tr>
<td>Titanic</td>
<td>194</td>
<td>Drama Romance</td>
<td>1997</td>
<td>Paramount Pictures</td>
<td>PG-13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Actor_Leading</th>
<th>Actor_Supporting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brave Heart</td>
<td>Mel Gibson</td>
<td>Sophie Marceau</td>
</tr>
<tr>
<td>Christmas Vacation</td>
<td>Cherly Chase</td>
<td>Beverly D'Angelo</td>
</tr>
<tr>
<td>Coming to America</td>
<td>Eddie Murphy</td>
<td>Arsenio Hall</td>
</tr>
<tr>
<td>Forrest Gump</td>
<td>Tom Hanks</td>
<td>Sally Field</td>
</tr>
<tr>
<td>Ghost</td>
<td>Patrick Swazie</td>
<td>Demi Moore</td>
</tr>
<tr>
<td>Lethal Weapon</td>
<td>Mel Gibson</td>
<td>Danny Glover</td>
</tr>
<tr>
<td>Michael</td>
<td>John Travolta</td>
<td>Andie MacDowell</td>
</tr>
<tr>
<td>National Lampoon's Vacation</td>
<td>Cherly Chase</td>
<td>Beverly D'Angelo</td>
</tr>
<tr>
<td>Rocky</td>
<td>Sylvester Stallone</td>
<td>Talia Shire</td>
</tr>
<tr>
<td>Silence of the Lambs</td>
<td>Anthony Hopkins</td>
<td>Jodie Foster</td>
</tr>
<tr>
<td>The Hunt for Red October</td>
<td>Sean Connery</td>
<td>Alec Baldwin</td>
</tr>
<tr>
<td>The Terminator</td>
<td>Arnold Schwarzenegge</td>
<td>Michael Biehn</td>
</tr>
<tr>
<td>Titanic</td>
<td>Leonardo Dicaprio</td>
<td>Kate Winslet</td>
</tr>
</tbody>
</table>
What is Data Federation?

- Process of integrating data from many different sources
- Leaves data in place without using resources to copy data
- Makes access to data sources as easy as possible
- Provides a degree of reusability
- A data federation approach often replaces a data warehouse
Data Federation Characteristics

- It represents an integration approach
- Ability to aggregate data from many different sources
- Data sources can be in any location
- It can be implemented within any lifecycle methodology
- Provides flexibility
- It is user-centric
- Federated data does not copy and store data like a data warehouse
- It contains metadata (information about the actual data and its location)
- Is NOT always designed with optimality in mind
What is a Join?

- Process of combining tables side-by-side (horizontally)
- Consists of a matching process between rows in tables
- Some or all of the tables’ contents are brought together
- Gather and manipulate data from across tables

Visually, it would look something like this:
Why Join?

- Data in a database is often stored in separate tables
- Joins allow data to be combined as if it were stored in one huge file
- Provide exciting insights into data relationships
- Types of joins:
  - Inner joins – a maximum of 256 tables can be joined
  - Outer joins – a maximum of 2 tables can be joined
DATA Step Merge versus a Join

- Merges process data differently than a standard join
- The merge process overlays the duplicate by-column
- Joins adhere to ANSI guidelines
- Joins do not automatically overlay the duplicate matching column
DATA Step Merge Process

Customers

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ryan</td>
</tr>
<tr>
<td>5</td>
<td>Anna-liese</td>
</tr>
<tr>
<td>10</td>
<td>Ronnie</td>
</tr>
</tbody>
</table>

Movies

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Adventure</td>
</tr>
<tr>
<td>5</td>
<td>Comedy</td>
</tr>
<tr>
<td>7</td>
<td>Suspense</td>
</tr>
</tbody>
</table>

Merged

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ryan</td>
<td>Adventure</td>
</tr>
<tr>
<td>5</td>
<td>Anna-liese</td>
<td>Comedy</td>
</tr>
</tbody>
</table>

DATA merged;
MERGE customers (IN=c)
movies (IN=m);
BY cust_no;
IF c AND m;
RUN;
Join Process

Customers

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ryan</td>
</tr>
<tr>
<td>5</td>
<td>Anna-liese</td>
</tr>
<tr>
<td>10</td>
<td>Ronnie</td>
</tr>
</tbody>
</table>

Movies

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Adventure</td>
</tr>
<tr>
<td>5</td>
<td>Comedy</td>
</tr>
<tr>
<td>7</td>
<td>Suspense</td>
</tr>
</tbody>
</table>

```
PROC SQL;
SELECT *
FROM customers, movies
WHERE
  customers.cust_no = movies.cust_no;
QUIT;
```
Merge versus Join Results

### Features

**Merge**
1. Data must be sorted using by-value.
2. Requires variable names to be same.
3. Duplicate matching column is overlaid.
4. Results are not automatically printed.

**Join**
1. Data does not have to be sorted using by-value.
2. Does not require variable names to be same.
3. Duplicate matching column is not overlaid.
4. Results are automatically printed unless NOPRINT option is specified.

### Examples

#### Merge

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ryan</td>
<td>Adventure</td>
</tr>
<tr>
<td>5</td>
<td>Anna-liese</td>
<td>Comedy</td>
</tr>
</tbody>
</table>

#### Join

<table>
<thead>
<tr>
<th>Cust_no</th>
<th>Name</th>
<th>Cust_no</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ryan</td>
<td>3</td>
<td>Adventure</td>
</tr>
<tr>
<td>5</td>
<td>Anna-liese</td>
<td>5</td>
<td>Comedy</td>
</tr>
</tbody>
</table>
Cartesian Product Join (Cross Join)

Result represents all possible combinations of rows and columns

Absence of WHERE clause produces a Cartesian product

PROC SQL;
SELECT *
FROM customers, movies;
QUIT;
A Cartesian product join, sometimes referred to as a cross join, can be very large because it represents all the possible combinations of rows and columns.

PROC SQL;
SELECT *
FROM MOVIES,
    ACTORS;
QUIT;
Equi-Join with Two Tables

The result of an **Equi-join** is illustrated by the shaded area (AB) in the Venn diagram.
Equi-Join with Two Tables

PROC SQL;
SELECT *
FROM customers, movies
WHERE
  customers.cust_no = movies.cust_no;
QUIT;

Customers
- Cust_no  Name
  3     Ryan
  5     Anna-liese
  10    Ronnie

Movies
- Cust_no  Category
  3     Adventure
  5     Comedy
  7     Suspense

Cust_no  Name    Cust_no  Movie_no  Category
3        Ryan      3        1011      Adventure
5        Anna-liese 5        3090      Comedy
Example – WHERE-clause

The most reliable way to join two tables together, and to avoid creating a Cartesian product, is to use a WHERE clause with common columns or keys.

PROC SQL;
SELECT MOVIES.TITLE, RATING, ACTOR_LEADING
FROM MOVIES,
    ACTORS
WHERE MOVIES.TITLE = ACTORS.TITLE;
QUIT;
Table Aliases

PROC SQL;
SELECT *
FROM customers c,
    movies m
WHERE c.cust_no = m.cust_no;
QUIT;

Customers
- Cust_no: 3, Name: Ryan
- Cust_no: 5, Name: Anna-liese
- Cust_no: 10, Name: Ronnie

Movies
- Cust_no: 3, Category: Adventure
- Cust_no: 5, Category: Comedy
- Cust_no: 7, Category: Suspense

Aliases = Equi Join

Cust_no  Name    Cust_no  Movie_no  Category
3        Ryan     3        1011      Adventure
5        Anna-liese 5        3090      Comedy
Example – Table Alias

Assigning a table alias is not only a useful way to reference a table, but can reduce the number of keystrokes typed.

PROC SQL;
  SELECT M.TITLE, RATING, ACTOR_LEADING
  FROM MOVIES M,
       ACTORS A
  WHERE M.TITLE = A.TITLE;
QUIT;
Joining Three Tables

```
PROC SQL;
    SELECT c.cust_no, c.name,
         m.movie_no, c.category,
         a.lead_actor
    FROM customers c,
         movies m,
         actors a
    WHERE
        c.cust_no = m.cust_no AND
        m.movie_no = a.movie_no;
QUIT;
```
The result of a **Left Outer join** is illustrated by the shaded areas (A and AB) in the Venn diagram.
Example – Left Outer Join

The result of a Left Outer join produces both matched rows from both tables plus any unmatched rows from the left table.

PROC SQL;
  SELECT MOVIES.TITLE, RATING, ACTOR_LEADING
  FROM MOVIES LEFT JOIN ACTORS
    ON MOVIES.TITLE = ACTORS.TITLE;
QUIT;
Right Outer Joins

The results of a Right Outer join is illustrated by the shaded areas (B and AB) in the Venn diagram.
Example – Right Outer Join

The result of a Right Outer join produces matched rows from both tables while preserving unmatched rows from the right table.

PROC SQL;
  SELECT MOVIES.TITLE, RATING, ACTOR_LEADING
  FROM MOVIES RIGHT JOIN
    ACTORS
  ON MOVIES.TITLE = ACTORS.TITLE;
QUIT;
What Happens during a Join?

When joining two tables:
- An intermediate Cartesian product is built from the two tables
- Rows are selected that match the WHERE clause, if present

When joining more than two tables:
- SQL query optimizer evaluates the available methods for retrieving the data and attempts to use the most efficient method
- The join is reconstructed into several two-way joins
- Removes unwanted rows and columns from the intermediate tables
- Determines the order of processing to reduce the size of the intermediate Cartesian product
Join Algorithms

Users supply the list of tables for joining along with the join conditions, and the PROC SQL optimizer determines which join algorithm to use for performing the join. The algorithms include:
Users supply the list of tables for joining along with the join conditions, and the PROC SQL optimizer determines which of the join algorithms to use for performing the join. The algorithms include:

- **Nested Loop Join (brute-force join)** – When an equality condition is not specified, a read of the complete contents of the right table is processed for each row in the left table.
Nested Loop Join - Features

- Used with join relations of two tables
- One or both of the tables is relatively small
- I/O intensive
- This join generally performs fairly well with smaller tables, but generally performs poorly with larger join relations
Join Algorithms

Users supply the list of tables for joining along with the join conditions, and the PROC SQL optimizer determines which of the join algorithms to use for performing the join. The algorithms include:

✓ Nested Loop Join – When an equality condition is not specified, a read of the complete contents of the right table is processed for each row in the left table.

✓ Sort-Merge Join – When the specified tables are already in the desired sort order, resources are not expended for resorting.
Sort-Merge Join - Features

• Used with joins of two tables
• Works best when one or both of the join relations are in the desired order
• One or both of the tables are of moderate size
• If the optimizer determines a sort is not needed – no sort will be performed, otherwise sort resources are expended:
  - using an explicit sort operation <or>
  - by taking advantage of pre-existing ordering
• Generally performs well, particularly when the majority of the rows are being joined
Join Algorithms

Users supply the list of tables for joining along with the join conditions, and the PROC SQL optimizer determines which of the join algorithms to use for performing the join. The algorithms include:

✓ Nested Loop Join – When an equality condition is not specified, a read of the complete contents of the right table is processed for each row in the left table.

✓ Sort-Merge Join – When the specified tables are already in the desired sort order, resources are not expended for resorting.

✓ Indexed Join – When an index exists on >=1 variable(s) to represent a key, matching rows may be accessed using the index.
Indexed Join - Features

- Used with joins of two tables
- An index must be defined that produces a small subset of the total number of rows in a table
- Matching rows are accessed directly using the index
- One or both of the tables are of moderate to large size
- Generally performs well, particularly when a small number of rows are being joined
Join Algorithms

Users supply the list of tables for joining along with the join conditions, and the PROC SQL optimizer determines which of the join algorithms to use for performing the join. The algorithms include:

☑ Nested Loop Join – When an equality condition is not specified, a read of the complete contents of the right table is processed for each row in the left table.

☑ Sort-Merge Join – When the specified tables are already in the desired sort order, resources are not expended for resorting.

☑ Indexed Join – When an index exists on ≥1 variable(s) to represent a key, matching rows may be accessed using the index.

☑ Hash Join – When an equality relationship exists and the smaller table is able to fit in memory, set-matching operations generally perform well.
Hash Join - Features

- Used with joins of two tables
- The SQL optimizer attempts to estimate the amount memory required to build a hash table in memory
- A hash table data structure associates keys with values
- The optimizer tries to use the smaller of the tables as the hash table
- Its purpose is to perform more efficient lookups
- Requires an equi-join predicate
- This algorithm generally performs well with small to medium join relations
By specifying the MSGLEVEL=I option, helpful notes describing index usage, sort utilities, and merge processing are displayed on the SAS Log.

```sql
OPTIONS MSGLEVEL=I;
PROC SQL;
   SELECT MOVIES.TITLE, RATING, LENGTH, ACTOR_LEADING
   FROM MOVIES,
       ACTORS
   WHERE MOVIES.TITLE = ACTORS.TITLE AND
       RATING = 'PG';
QUIT;
```
OPTIONS MSGLEVEL=I;
PROC SQL;
  SELECT MOVIES.TITLE, RATING, LENGTH, ACTOR_LEADING
  FROM MOVIES,
    ACTORS
  WHERE MOVIES.TITLE = ACTORS.TITLE AND
    RATING = 'PG';

INFO: Index Rating selected for WHERE clause optimization.
QUIT;
__METHOD Option

A __METHOD option can be specified on the PROC SQL statement to display the hierarchy of processing that takes place. Results are displayed on the Log.

<table>
<thead>
<tr>
<th>Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqxcrta</td>
<td>Create table as Select</td>
</tr>
<tr>
<td>sqxslct</td>
<td>Select</td>
</tr>
<tr>
<td>sqxjsl</td>
<td>Step loop join (Cartesian)</td>
</tr>
<tr>
<td>sqxjm</td>
<td>Merge join</td>
</tr>
<tr>
<td>sqxjndx</td>
<td>Index join</td>
</tr>
<tr>
<td>sqxjhsh</td>
<td>Hash join</td>
</tr>
<tr>
<td>sqxsort</td>
<td>Sort</td>
</tr>
<tr>
<td>sqxsorc</td>
<td>Source rows from table</td>
</tr>
<tr>
<td>sqxfil</td>
<td>Filter rows</td>
</tr>
<tr>
<td>sqxsumg</td>
<td>Summary statistics with GROUP BY</td>
</tr>
<tr>
<td>sqxsumn</td>
<td>Summary statistics with no GROUP BY</td>
</tr>
</tbody>
</table>
Program Example using _METHOD

PROC SQL _METHOD;

TITLE '2-Way Equi Join';
SELECT MOVIES.TITLE, RATING, ACTOR_LEADING FROM MOVIES,
    ACTORS
WHERE MOVIES.TITLE = ACTORS.TITLE;
QUIT;

SAS Log Results
NOTE: SQL execution methods chosen are:
sqxsict
    sqxjhs
    sqxsrt( MOVIES )
    sqxsrt( ACTORS )
Conclusion

• Data federation characteristics
• A merge and join do not process data the same way
• A join combines tables side-by-side (horizontally)
• Joins adhere to ANSI guidelines
• Cartesian product represents all possible combinations of rows from the underlying tables
• 256 tables can be joined using an inner join construct
• 2 tables can be joined using an outer join construct
• Four types of join algorithms:
  ✓ Nested loop join
  ✓ Sort-Merge join
  ✓ Indexed join
  ✓ Hash join
A Book of Data Transfer Methods with Examples

Coming Winter 2004!

Coming in September 2010!

SAS® and Excel®

Transferring Data between SAS and Excel Using SAS

Kirk Paul Lafler
William E. Benjamin, Jr.

sas

Copyright 1992-2010 by Kirk Paul Lafler
Thank you for attending!

Questions?

Joining Worlds of Data is exciting and fun!

Kirk Paul Lafler
SAS® Consultant, Author, and Trainer
Software Intelligence Corporation
E-mail: KirkLafler@cs.com
We kindly thank the WUSS Leadership and the following organizations for their assistance and support for this symposium:
Mark your calendars for WUSS 2010!

Three days of classes, workshops and presentations introducing the latest in SAS® technology and applications.

November 3-5
Hyatt Regency Mission Bay
San Diego, CA

More information will be available at www.wuss.org