

Combining Multiple Date-Ranged Historical Data Sets with Dissimilar Date Ranges into a Single Change History Data Set

Potential
of One

Power
of
All

James Moon,
County of Fairfax VA

Combining Multiple Date-Ranged Historical Data Sets

James Moon

County of Fairfax VA

Abstract

- This paper describes a method that uses some simple SAS® macros and SQL to merge data sets having related data containing rows with varying effective date ranges.
- The data sets are merged into a single data set that represents a serial list of snapshots of the merged data as of a change in any of the effective dates.
- While simple conceptually, this type of merge is often problematic when the effective date ranges are not consecutive or consistent, when the ranges overlap, or when there are missing ranges from one or more of the merged data sets.

In our simplified example, we start with three Date-Ranged Historical datasets:

Dataset 1

Id	Start Date	End Date	DS1 Value
120	Date1	Date2	A
120	Date2	Date5	B
120	Date5	12/31/9999	C

Dataset 2

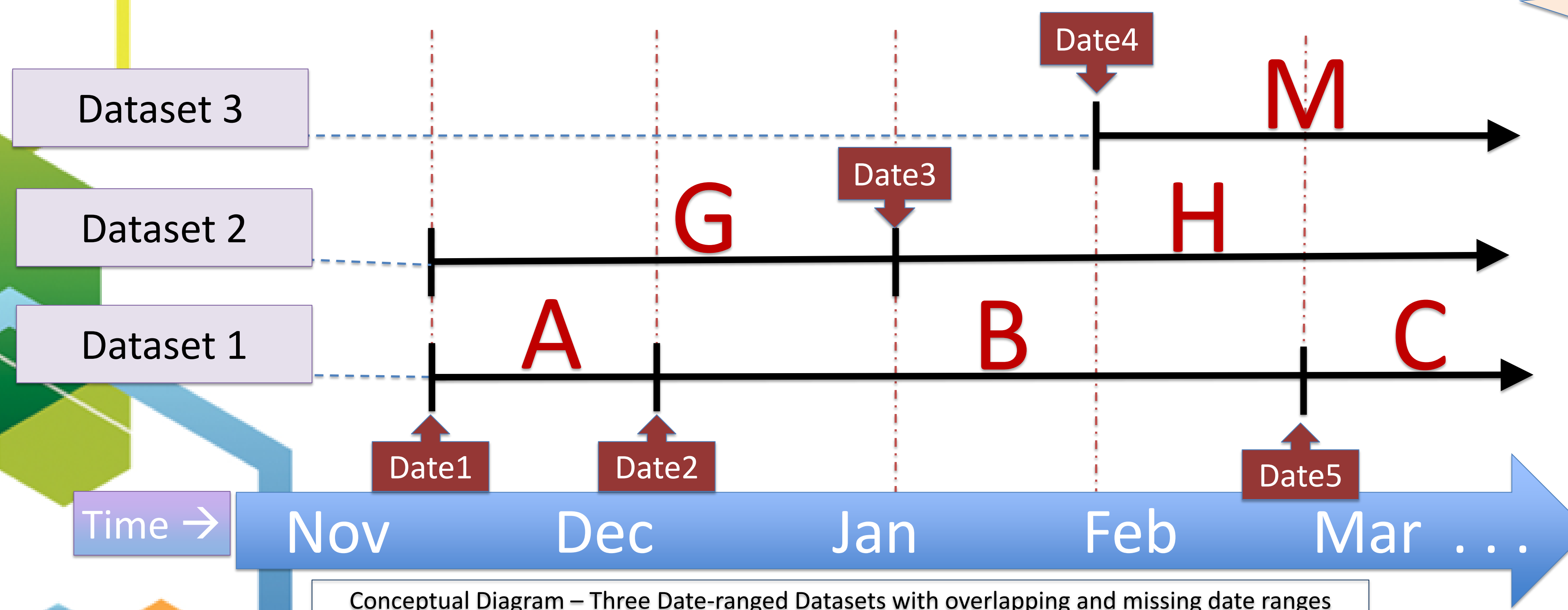
Id	Start Date	End Date	DS2 Value
120	Date1	Date3	G
120	Date3	12/31/9999	H

Dataset 3

Id	Start Date	End Date	DS3 Value
120	Date4	12/31/9999	M

...and transform the datasets into one comprehensive Change History dataset having a "snapshot" of the values from each dataset on the Change Date

Id	Change Date	DS1 Value	DS2 Value	DS3 Value
120	Date1	A	G	. (missing)
120	Date2	B	G	. (missing)
120	Date3	B	H	. (missing)
120	Date4	B	H	M
120	Date5	C	H	M



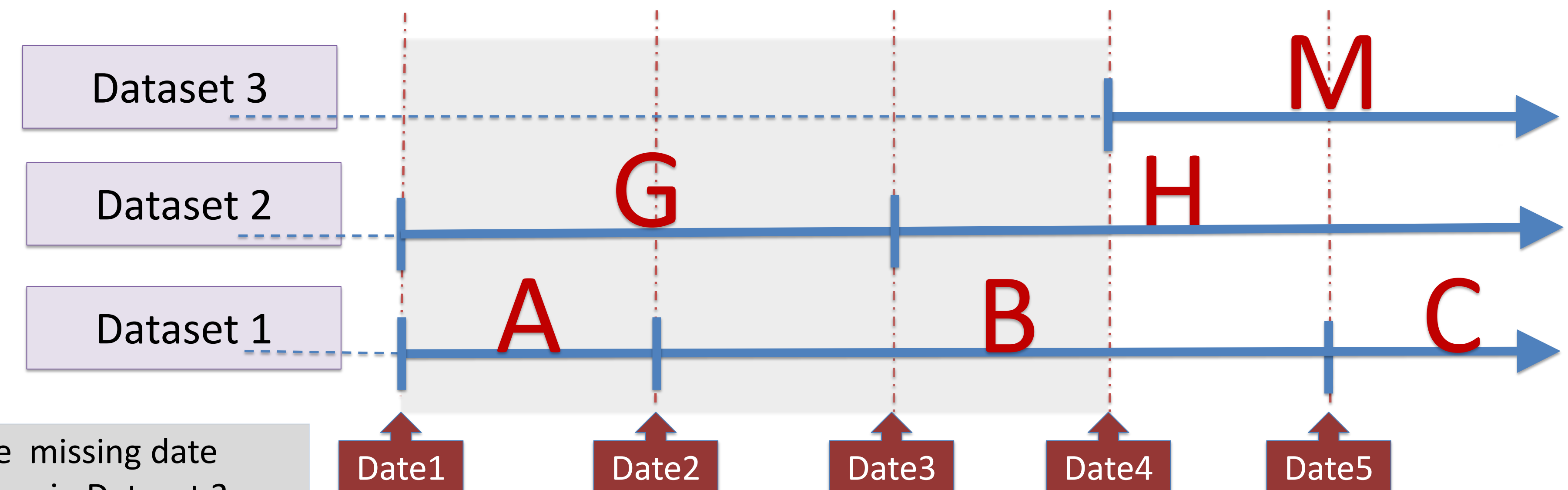
Why is this so HARD?

The Trouble with Traditional SQL Techniques for joining across date ranges:

1. **It's complicated!** Here's a sample of a Date Range join for selecting matching id/date ranges across just three tables:

```
SELECT *
FROM TABLEA a, TABLEB b, TABLEC c
WHERE a.Id = b.Id AND
b.Id = c.Id AND
( ( b.BegDat <= a.BegDat and b.EndDat >= a.BegDat ) OR
  b.BegDat between a.BegDat and a.EndDat ) AND
( ( c.BegDat <= b.BegDat and c.EndDat >= b.BegDat ) OR
  c.BegDat between b.BegDat and b.EndDat ) AND
( ( c.BegDat <= a.BegDat and c.EndDat >= a.BegDat ) OR
  c.BegDat between a.BegDat and a.EndDat )
```

2. **There can be no gaps!** A traditional multi-table join (like above) will combine all the desired rows only if all the date ranges are consecutive, and there are no gaps or missing ranges in any range in any of the contributing tables. Each table must have rows that are effective for all ranges. A missing range in any table causes missing rows in the result set for all the tables.



The missing date range in Dataset 3 overlaps ranges that are present in other datasets (shaded area). However, when using a traditional multi-table join technique, those ranges will not be considered a matching range.

Dataset 3			
Id	Start Date	End Date	DS3 Value
120	Date4	12/31/9999	M

These rows will be omitted from the result set. We would like them to be present with the "snapshot" values for Datasets 1 and 2, and to have "missing values" in the value column for Dataset 3.

Change Date	DS1 Value	DS2 Value	DS3 Value
Date1	A	G	. (missing)
Date2	B	G	. (missing)
Date3	B	H	. (missing)
Date4	B	H	M
Date5	C	H	M

Combining Multiple Date-Ranged Historical Data Sets

James Moon

County of Fairfax VA

The Solution – Overall Approach

Step #1

Let's approach the problem from a simpler perspective. First, create a list of specific dates where the values in any of the datasets could have changed for each "Id". The only possible dates the data in any of the datasets could have changed are the endpoints of the data ranges, i.e. the start and end dates. Therefore, if we extract all the range start and end dates for each "Id" into a single list, we have distilled a finite list of possible change dates for that "Id", across all the datasets.

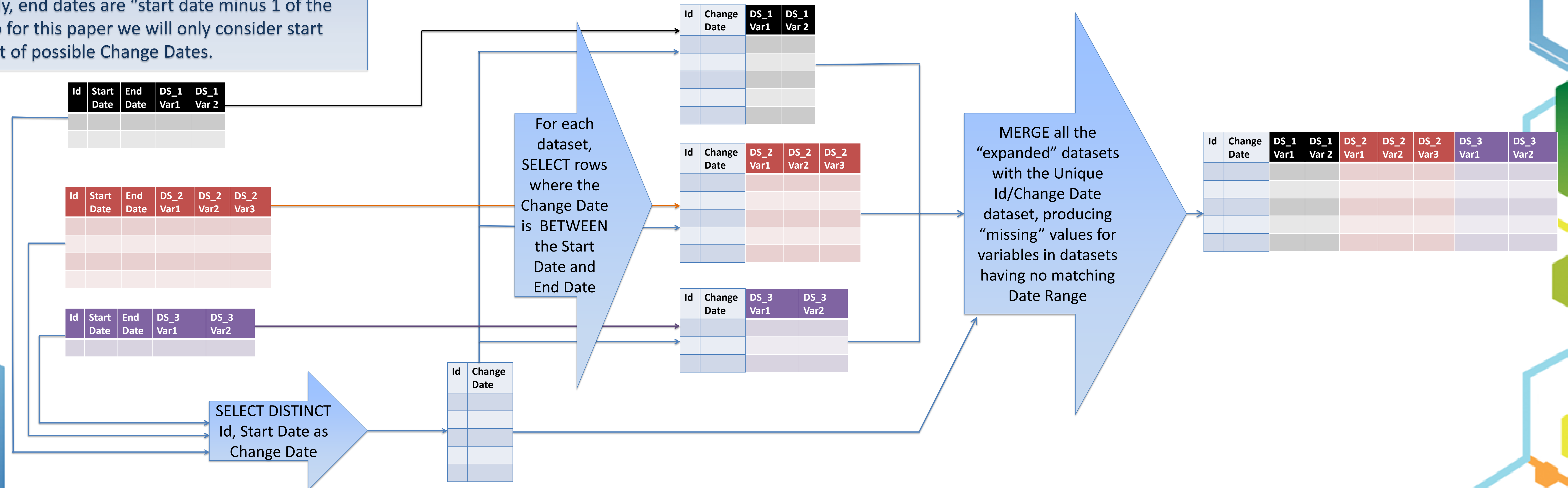
NOTE: Typically, end dates are "start date minus 1 of the next range", so for this paper we will only consider start dates in our list of possible Change Dates.

Step #2

Once we have a finite list of change dates for each "id", the rest is simple! We select the rows from each dataset where the Change Date is BETWEEN the range Start Date and End Date. If a particular row spans multiple Change Dates, then multiple rows will be produced, one for each Change Date encompassed by its Date Range.

Step #3

We then MERGE all the datasets by Id/Change Date, producing variables with "missing" values in observations with missing date ranges. We, thereby achieve our desired result of a comprehensive Change History dataset, keyed by Id and Change Date, having all variables from all the input datasets, "as they were" on the Change Date.



Step #1 – Select the unique Change Dates

Create a dataset that contains all the unique Ids and Begin Dates. This is the “universe” of all the Id/Change Dates on which a change could possibly have occurred.

For each date-ranged table, using SELECT DISTINCT, we create a table of unique Id and Begin Dates. Combine all the ID/Start Date tables using UNION ALL and eliminate duplicates. We are left with a table of all the unique Id/Change Dates across all the tables.

```
LIBNAME histlib ... /* location of date-ranged historical datasets */
PROC SQL;
CREATE TABLE all_id_activity_dates AS
SELECT DISTINCT
id      AS act_id
begdate AS act_date FORMAT=MMDDYY10. LABEL="Activity Date"
FROM
(
  SELECT id, begdate FROM histlib.file1
    UNION ALL
  SELECT id, begdate FROM histlib.file2
    UNION ALL
  SELECT id, begdate FROM histlib.file3
    UNION ALL
  SELECT id, begdate FROM histlib.file4
    UNION ALL
  SELECT id, begdate FROM histlib.file5 );
```

Extract the Ids and Change Dates from each historical dataset and eliminate duplicates. In this sample code, “Id” is called “Activity Id” and “Change Date” is called “Activity Date”.

Id	Start Date	End Date	DS1 Value
120	Date1	Date2	A
120	Date2	Date5	B
120	Date5	12/31/9999	C

Id	Start Date	End Date	DS2 Value
120	Date1	Date3	G
120	Date3	12/31/9999	H

Id	Start Date	End Date	DS3 Value
120	Date4	12/31/9999	M

SELECT DISTINCT to produce a table of unique Change Dates for each Id.

Id	Change Date
120	Date1
120	Date2
120	Date3
120	Date4
120	Date5

Step #2 – Select rows “between” date range

For each date-ranged table, create a new table with rows that match the “universe” of Change Dates for that Id (from Step #1). In other words, select rows from each table where the Id matches **and the Change Date is BETWEEN the Begin and End Dates**. Each date-ranged row produces one or more rows, **a row for each Change Date encompassed by its Date Range**.

Here’s a macro, “sortact”, that uses WHERE...BETWEEN to produce a row for each Id/Change Date within each row’s Begin/End Date range.

```
%macro sortact(dsname, sortkey, uniqkey);
PROC SQL;
CREATE TABLE actdate_&dsname as
SELECT
    a.act_id
    , a.act_date
    , i.*
FROM all_id_activity_dates a ,
    histlib.&dsname i
WHERE a.act_id = i.id and
    a.act_date BETWEEN i.begdate
    and i.enddate;
QUIT;

PROC SORT DATA=actdate_&dsname
OUT=sort1_&dsname;
BY &sortkey;

PROC SORT DATA=sort1_&dsname nodupkey
OUT=alldate_&dsname;
BY &uniqkey;

%mend sortact;
run;
```

Execute the sortact macro for each date-ranged dataset.

```
%sortact(file1, act_id act_date
descending begdate descending
enddate, act_id act_date);

%sortact(file2, act_id act_date
descending begdate descending
enddate, act_id act_date);

%sortact(file3, act_id act_date
descending begdate descending
enddate descending seqnr, act_id
act_date);

%sortact(file4, act_id act_date
descending begdate descending
enddate, act_id act_date) ;
```

Id	Start Date	End Date	DS1 Value
120	Date1	Date2	A
120	Date2	Date5	B
120	Date5	12/31/9999	C

Id	Change Date
120	Date1
120	Date2
120	Date3
120	Date4
120	Date5

Id	Change Date	DS1 Value
120	Date1	A
120	Date2	B
120	Date3	B
120	Date4	B
120	Date5	C

Id	Start Date	End Date	DS2 Value
120	Date1	Date3	G
120	Date3	12/31/9999	H

Id	Change Date
120	Date1
120	Date2
120	Date3
120	Date4
120	Date5

Id	Change Date	DS2 Value
120	Date1	G
120	Date2	G
120	Date3	H
120	Date4	H
120	Date5	H

Id	Start Date	End Date	DS3 Value
120	Date4	12/31/9999	M

Id	Change Date
120	Date1
120	Date2
120	Date3
120	Date4
120	Date5

Id	Change Date	DS3 Value
120	Date4	M
120	Date5	M

Each date-ranged table is matched with the Unique Id/Change Dates from Step #1.

Unique Id/Change Dates

Each resulting table has a row for each Id/Change Date that is within row’s the Date Range.

Combining Multiple Date-Ranged Historical Data Sets

James Moon

County of Fairfax VA

Step #3 – Great Big MERGE!

Now, we MERGE all the tables from Step #2 with the table of unique Id/Change Dates in a multi-file MERGE BY Id/Change Date. SAS does the hard work of matching all the datasets in one simple step and ensures that observations with “missing” values are generated. The result is a “Big Flat Table” with all the variables from all the date-ranged tables in a single dataset, having a single row for each ID/Change Date, with values “as of” the Change Date.

```
/* Merge all the data together by Id and Activity Date. */
/* This may be a large merge, but since all the merge files */
/* are nicely lined up and in sequence by Id/Activity Date, */
/* it runs very efficiently. */
```

```
DATA change_action_hist_master;
```

```
MERGE
```

```
all_id_activity_dates
```

```
alldate_file1
```

```
alldate_file2
```

```
alldate_file3
```

```
alldate_file4
```

```
alldate_file5
```

```
;
```

```
BY act_id act_date;
```

```
DROP begdate enddate;
```

```
RUN;
```

Merge the Unique Id/Change Date dataset with all the date-ranged datasets from Step #2, producing “missing values” for missing date ranges .

Id	Change Date	DS1 Value
120	Date1	A
120	Date2	B
120	Date3	B
120	Date4	B
120	Date5	C

Id	Change Date	DS2 Value
120	Date1	G
120	Date2	G
120	Date3	H
120	Date4	H
120	Date5	H

Id	Change Date	DS3 Value
120	Date4	M
120	Date5	M

Id	Change Date
120	Date1
120	Date2
120	Date3
120	Date4
120	Date5

Unique Id/Change Dates

MERGE

Id	Change Date	DS1 Value	DS2 Value	DS3 Value
120	Date1	A	G	. (missing)
120	Date2	B	G	. (missing)
120	Date3	B	H	. (missing)
120	Date4	B	H	M
120	Date5	C	H	M

Combining Multiple Date-Ranged Historical Data Sets

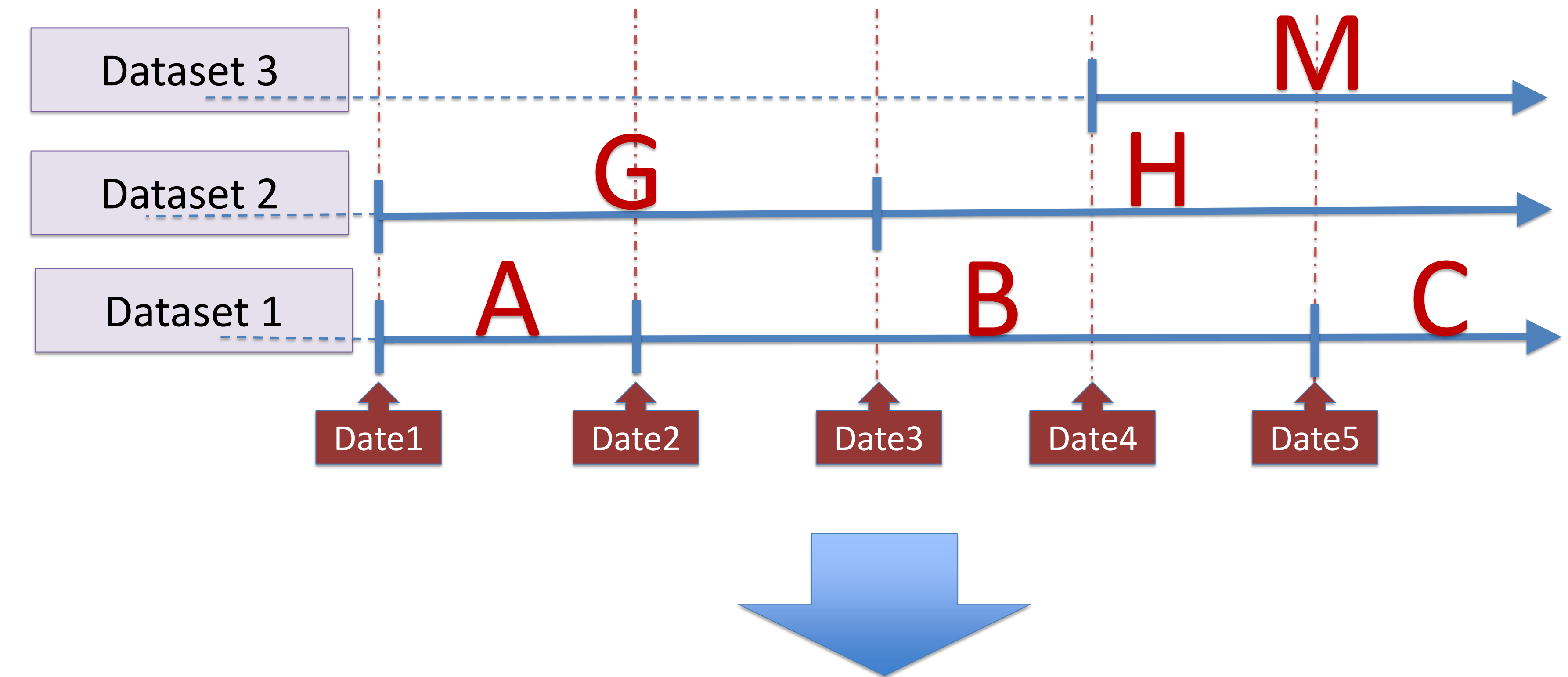
James Moon

County of Fairfax VA

Conclusion

Combining multiple date-ranged datasets using traditional methods of complicated multi-table JOIN, or a series of OUTER JOINs, will not provide reliable results when the date ranges overlap or are missing. While requiring a three-step process, the simple technique described in this paper will unequivocally provide the desired results of creating a comprehensive Change History dataset, containing a “snapshot” of the values of every observation “as of” the Change Date.

The crux of this technique is approaching the problem from a different perspective. Rather than attempting some form of iterative comparison of “date ranges to date ranges to date ranges”, we use a simple process that creates a finite aggregate list of specific “Change Dates” and then apply a traditional unambiguous selection using “WHERE...BETWEEN”, followed by an efficient MERGE process to combine the multiple merge datasets into a single Change History dataset.



Id	Change Date	DS1 Value	DS2 Value	DS3 Value
120	Date1	A	G	. (missing)
120	Date2	B	G	. (missing)
120	Date3	B	H	. (missing)
120	Date4	B	H	M
120	Date5	C	H	M

The views expressed in this article are solely those of the author in his private capacity and do not necessarily reflect the position or policy of the Fairfax County, Virginia government.



Washington, D.C.

March 23–26, 2014

About the Author

Jim Moon serves as Lead SAS® developer for Fairfax County Virginia's Human Resources Information Systems Division. Jim develops creative and practical SAS-based reporting solutions for Fairfax County's Human Resources Department, most-recently producing an innovative extract, transform and load (ETL) process for Fairfax County's SAP ERP system using SAS® Data Surveyor for SAP™.

Jim conceived and developed Fairfax County's SAS/IntrNet™ Human Capital Management reporting application, called Point-and-Click Enterprise Ad-hoc Query (PEAQ), for which he received the Local Government Personnel Association (LPGA) Innovation Award in 2006. Jim also specializes in legacy historical query and reporting using SAS® Object-Oriented ODS™ and advanced SAS® macros. Jim is a certified SAS® Advanced Programmer for SAS9.

Contact Information

Your comments and questions are valued and encouraged.
Contact the author at:

James A. Moon
Human Resources Information Systems Division
Department of Human Resources
County of Fairfax, Virginia
12000 Government Center Pkwy
Fairfax, VA 22035
Work Phone: (703) 324-3497
Email: james.moon@fairfaxcounty.gov