AT-168

Vetting Differences Between Relational Database Definitions and Actual Data with SAS®

Michael A. Raithel, Westat, Inc.

Abstract

SAS programmers are sometimes tasked with loading SAS data sets into relational databases management systems (RDBMS) such as Oracle, SQL Server, or SYBASE. Loading the data can be a painful, stop-and-go process when the relational database table definitions do not agree with the SAS data sets that are to be uploaded. Differences between SAS and the RDBMS in the number of variables, variable names, variable types, and variable lengths can cause issues with the loading or with subsequent processing. Therefore, it is prudent to vet the differences between the SAS data sets' metadata and the RDBMS tables' metadata before attempting to load SAS data sets into a database.

This paper presents a method of comparing the metadata of SAS data sets against an RDBMS's table definitions. It produces four reports: variables found in SAS but not in the RDBMS; variables found in the RDBMS but not SAS; variable type differences; and variable length differences. Working with these reports and with the RDBMS's database administrator, SAS programmers can help shape the SAS data sets and table definitions that lead to an error-free loading of their relational databases.

Introduction

It is pragmatic to compare a relational database's table definition SQL code against the metadata of the SAS data sets that you are going to load into the database *before* attempting to load. Doing so helps you to avoid frustrating load-time errors and allows you to find the mismatches between what data the database believes it is getting and what is actually in the SAS data sets. This process can help you to make early decisions on whether you need to update the database table definitions or update the SAS data sets that will be loaded.

The ***Compare Create Table SQL File Against Production SAS Data Sets.sas*** SAS program that this paper is based on was written to compare an Oracle database table definition SQL program against forty-four production SAS data sets that were to be loaded into database tables. It successfully identified mismatches and led to more precise Oracle table definitions and more complete SAS data sets that were staged for loading. The program can be found in its entirety in **Appendix A - *The Compare Create Table SQL File Against Production SAS Data Sets* SAS Program** at the end of this paper. For brevity's sake, it is simply referred to as "the SAS program" throughout the rest of this paper.

The SAS program can identify up to nine specific mismatches as documented in **Table 1 - Comparing an Oracle table definition .sql file against SAS data sets**. The columns in Table 1 denote the following:

- **Mismatch –** Provides reference numbers used in **Part 5 –Creating Reports of Mismatches** of this paper.
- **Comparison** – Describes each particular comparison.
- **Error?** – Notes whether this type of mismatch results in an ERROR in the SAS log.
- **Message** – Provides examples of the type of SAS ERROR or WARNING message this disparity produces.
- **Example Variables** – Highlights the variables in the example Oracle table definition and SAS data set that produced the text in the **Message** column.

Mismatch #	Comparison	Error?	Message	Example Variables
1	SAS data set but no Oracle table with same name	No	NOTE: BASE data set does not exist. DATA file is being copied to BASE file.	N/A
2	Column in Oracle table but no corresponding variable in SAS data set	No	WARNING: Variable TotalSalesTax was not found on DATA file.	TotalSalesTax
3	Variable in SAS data set but no corresponding column in Oracle table	Yes	WARNING: Variable NetSales was not found on BASE file. The variable will not be added to the BASE file. ERROR: No appending done because of anomalies listed above. Use FORCE option to append these files. NOTE: 0 observations added. NOTE: The SAS System stopped processing this step because of errors.	NetSales
4	Character variable in SAS but numeric column in Oracle	Yes	WARNING: Variable RegionID not appended because of type mismatch. ERROR: No appending done because of anomalies listed above. Use FORCE option to append these files. NOTE: 0 observations added. NOTE: The SAS System stopped processing this step because of errors.	RegionID
5	Character column in Oracle but numeric variable in SAS	Yes	WARNING: Variable OverStocked not appended because of type mismatch. ERROR: No appending done because of anomalies listed above. Use FORCE option to append these files. NOTE: 0 observations added. NOTE: The SAS System stopped processing this step because of errors.	OverStocked
6	Character variable in SAS longer than character column in Oracle	Yes	WARNING: Variable Product has different lengths on BASE and DATA files (BASE 10 DATA 14). ERROR: No appending done because of anomalies listed above. Use FORCE option to append these files. NOTE: 0 observations added. NOTE: The SAS System stopped processing this step because of errors.	Product
7	Character column in Oracle longer than character variable in SAS	No	WARNING: Variable Subsidiary has different lengths on BASE and DATA files (BASE 25 DATA 12).	Subsidiary
8	Numeric variable in SAS longer than Numeric column in Oracle	Yes	ERROR: ERROR: ERROR: ORACLE execute error: ORA-01438: value larger than specified precision allowed for this column. With the occurrence of the above ERROR, the error limit of 1 set by the ERRLIMIT= option has been reached. ROLLBACK has been issued(Any Rows processed after the last COMMIT are lost). NOTE: The SAS System stopped processing this step because of errors.	Returns
9	Numeric column in Oracle longer than numeric variable in SAS	No	NA	NumStores
			Note: BASE File = Oracle table, DATA File = SAS data set	

Table 1 – Comparing an Oracle table definition .sql file against SAS data sets.

Since the SAS program is divided into five "Parts", the next five sections of this paper discuss each part in succession; beginning with Part 1 and ending with Part 5. Part 1 contains a description of the three items you must specify to the program to run your analysis. Part 2 discusses the SAS code that processes the table definition file into a SAS data set. Part 3 shows how to write the SAS library metadata to a SAS data set. Part 4 examines the code for match merging the two SAS data sets together for comparison. Part 5 of this paper provides a description of the program code for creating reports of the mismatches. A Conclusions section wraps up the discussion of the utility of using the SAS program to vet differences between the database definition and the SAS data sets that will be loaded to the database.

Part 1 – Specifications and Allocations

Part 1 of the SAS program specifies SAS options and allocates the various directories and files that are used. The LIBNAME, FILENAME, and %LET statement must all be completed in order to specify the input SAS library, the Create Table SQL program, and the directory for mismatch reports, respectively. Specifically:

- **Libname SASDATA** – Specify the directory holding the SAS data sets that are to be loaded into Oracle. The program expects that directory to *only* house the data sets that will be loaded to Oracle.

- **Filename TableDes** – Specify the full path name of the Create Table SQL program. The program assumes that this SQL program only has CREATE TABLE SQL code in it for one or more tables.

- **%LET REPORTS** – Specify the directory where the mismatch reports are to be written to.

Once the three statements above are coded to the specific values for your project, you can execute the SAS program to look for mismatches.

Part 2 – Processing the RDBMS Table Definition as a Flat File

Part 2 of the SAS program uses SAS's character handling functions to process the SQL table definition file and create a SAS data set named **CreateTbl**. It can do this because a SQL table definition file is basically a flat file with a suffix of **.sql**. For example, the **CreateShoesData.sql** file used in this paper contains the following lines of code:

```
CREATE Table Shoesdata (
    RegSubID char(5) NOT NULL,
    RegionID int(2),
    OverStocked char(3),
    Region varchar(25) ,
    Product varchar(10) ,
    Subsidiary char(25) ,
    Sales number(8) ,
    Inventory number(8) ,
    Returns number(2) ,
    NumStores int(5) ,
    TotSalesTax number(8)
);
```

Since **CreateShoesData.sql** meets SQL coding best practices of specifying one column per line of code, it is straightforward to write SAS code that reads each line of the file and creates meaningful observations.

The DATA step in Part 2 creates observations with four variables from the Create Table SQL statement:

- **TABLE** – The name of the table, upcased via SAS's UPCASE function
- **VARIABLE** – The name of the column, upcased via SAS's UPCASE function
- **CreateTable_data_type** – The data type specified for a column
- **CreateTable_length** – The length specified for a column

The DATA step obtains the name of the table from the "Create Table" SQL statement and retains it in variable TABLE for the duration of the program. It discards blank lines and lines that contain ending create table delimiters: "**);**" Other lines are examined and the variable name, data type, and length are extracted and stored in discrete observations. When it is complete, the **CreateTbl** SAS data set is sorted by TABLE and VARIABLE in anticipation of its merge in Part 4.

This is what the example **CreateTbl** SAS data set looks like after Part 2 is done:

	table	variable	CreateTable_data_type	CreateTable_length
1	SHOESDATA	INVENTORY	Numeric	8
2	SHOESDATA	NUMSTORES	Int	4
3	SHOESDATA	OVERSTOCKED	Character	3
4	SHOESDATA	PRODUCT	Character	10
5	SHOESDATA	REGION	Character	25
6	SHOESDATA	REGIONID	Int	4
7	SHOESDATA	REGSUBID	Character	5
8	SHOESDATA	RETURNS	Numeric	2
9	SHOESDATA	SALES	Numeric	8
10	SHOESDATA	SUBSIDIARY	Character	25
11	SHOESDATA	TOTSALESTAX	Numeric	8

The eleven observations in the data set correspond to the eleven columns specified in the **CreateShoesData.sql** table definition file.

Part 3 – Writing the SAS Library Metadata to a SAS Data Set

Part 3 of the SAS program uses PROC CONTENTS to capture the metadata of the SAS data sets that are to be loaded into Oracle. It does so by first executing PROC CONTENTS against the SAS data library specified in Part 1 and using the OUT= option to save the results in a SAS data set named **SASDataSets**. Only variables MEMNAME, NAME, TYPE, and LENGTH are kept in in **SASDataSets**.

A DATA step is then used to update various aspects of the **SASDataSets** SAS data set:

- Variables MEMNAME and NAME are renamed to TABLE and VARIABLE, respectively
- The contents of variables TABLE and VARIABLE are upcased using SAS's UPCASE function
- A new variable, SASDataSets_data_type, is created and set to "Numeric" when TYPE =1, or "Character" when TYPE = 2.
- Variable LENGTH is renamed to SASDataSets_Length

Then, the **SASDataSets** SAS data set is sorted by TABLE and VARIABLE to prepare it for the merge with the **CreateTbl** SAS data set that was created in Part 2.

Part 4 – Match-Merging RDBMS Table Definitions and SAS Metadata

Part 4 of the SAS program match-merges the SAS data set built from the Create Table SQL file with that built from the PROC CONTENTS of the SAS data sets. The data sets are matched on variables: TABLE and VARIABLE. The merge can produce one, two, or three SAS data sets depending upon the compatibility of the files:

1. **Observations in the Createtable_and_sasdatasets SAS data set –** This data set contains observations from both data sets that have exact matches on TABLE and VARIABLE. Here is an example:

	table	variable	BuildTable_data_type	BuildTable_length	SASDataSets_length	SASDataSets_data_type
1	SHOESDATA	INVENTORY	Numeric	8	8	Numeric
2	SHOESDATA	NUMSTORES	Int	4	3	Numeric
3	SHOESDATA	OVERSTOCKED	Character	3	3	Numeric
4	SHOESDATA	PRODUCT	Character	10	14	Character
5	SHOESDATA	REGION	Character	25	25	Character
6	SHOESDATA	REGIONID	Int	4	2	Character
7	SHOESDATA	REGSUBID	Character	5	5	Character
8	SHOESDATA	RETURNS	Numeric	2	8	Numeric
9	SHOESDATA	SALES	Numeric	8	8	Numeric
10	SHOESDATA	SUBSIDIARY	Character	25	12	Character

2. **Observations in the Createtable_only SAS data set.** This data set contains observations with specific values of TABLE and VARIABLE only found in the **CreateTbl** data set. Here is an example:

	table	variable	BuildTable_data_type	BuildTable_length
1	SHOESDATA	TOTSALESTAX	Numeric	8

3. **Observations in the Sasdatasets_only SAS data set.** This data set contains observations with specific values of TABLE and VARIABLE only in the **SASDataSets** data set. Here is an example:

	table	variable	SASDataSets_length	SASDataSets_data_type
1	SHOESDATA	NETSALES	8	Numeric

Once these SAS data sets are built, we can compare the various characteristics of interest between them to determine how significant they are.

Part 5 – Creating Reports of Mismatches

Part 5 of the program can create up to five reports of mismatches between what is found in the Create Table SQL program and what is found in the SAS data sets. The program creates the "reports" as Excel files. However, you can just as easily have it use the ODS Word destination or PDF destination to provide your users with reports in the format they find the most useful.

The example SQL Create Table program and SAS data set were doctored so that all five reports were produced as Excel files. The contents of each Excel report file has been copied and pasted below.

Report 1 – Create Table File Variables Not Found in SAS Data Sets. According to this report the TOTSALESTAX variable is in the Create Table SQL program, but not in the corresponding SAS data set.

table	variable
SHOESDATA	TOTSALESTAX

This corresponds to Mismatch #2 in Table 1, which would have produced a WARNING in the SAS log if an upload had been attempted.

Report 2 – SAS Data Sets Variables Not Found in Create Table File. This report shows that the NETSALES variable is in the SAS data set, but not in the corresponding Create Table SQL program.

table	Variable
SHOESDATA	NETSALES

This corresponds to Mismatch #3 in Table 1, which would have produced an ERROR in the SAS log if an upload had been attempted.

Report 3 – Variable TYPE Mismatches between SAS Data Sets and Create Table File Variables. The first Type mismatch (on NUMSTORES) in this report seems harmless because it is declared as INT (integer) in SQL and numeric in SAS. However, the other two are not so innocuous. OVERSTOCKED is specified as character in SQL and numeric in SAS. REGIONID is specified as INT in SQL and character in SAS. Both of these variables require investigation in order to determine which Type is correct.

table	variable	CreateTable_data_type	CreateTable_length	SASDataSets_length	SASDataSets_data_type
SHOESDATA	NUMSTORES	Int	4	3	Numeric
SHOESDATA	OVERSTOCKED	Character	3	3	Numeric
SHOESDATA	REGIONID	Int	4	2	Character

These corresponds to Mismatch #4 and #5 in Table 1, which would have produced ERROR messages in the SAS log if an upload had been attempted.

Report 4 – Variable LENGTH Mismatches between SAS Data Sets and Create Table File Variables. Five variables have different lengths between the table definition and the existing SAS data set. Of those, the most troubling is REGIONID which we saw in Report 3 also has mismatched Types.

table	variable	CreateTable_data_type	CreateTable_length	SASDataSets_length	SASDataSets_data_type
SHOESDATA	NUMSTORES	Int	4	3	Numeric
SHOESDATA	PRODUCT	Character	10	14	Character
SHOESDATA	REGIONID	Int	4	2	Character
SHOESDATA	RETURNS	Numeric	2	8	Numeric
SHOESDATA	SUBSIDIARY	Character	25	12	Character

These corresponds to Mismatch #6, #7 and #8 in Table 1, which would have produced ERROR and WARNING messages in the SAS log if an upload had been attempted.

Report 5 – Character Variable LENGTH Mismatches between SAS Data Sets and Create Table File Variables. This report specifically looks for mismatched lengths in *character* variables. Incompatible character variable lengths can lead to truncation of data when it is being loaded into an RDBMS table. That is a very subtle error that can cause harm if not caught at the source.

Table	variable	CreateTable_data_type	CreateTable_length	SASDataSets_length	SASDataSets_data_type
SHOESDATA	PRODUCT	Character	10	14	Character
SHOESDATA	SUBSIDIARY	Character	25	12	Character

These corresponds to Mismatch #6 and #7 in Table 1, which would have produced ERROR and WARNING messages in the SAS log if an upload had been attempted.

Armed with these reports, SAS programmers and database administers can compare specifications and determine the correct characteristics that both the SQL tables and SAS data sets should have. Then, they can modify one or the other—or both—to ensure that they correctly reflect how the data will be stored in the database.

Conclusions

Comparing an RDBMS's table definition SQL code against the metadata for the SAS data that are to be loaded to a database can help you to avoid aggravating, and possibly time-consuming, load-time errors. The SAS program presented in this paper (***Compare Create Table SQL File Against Production SAS Data Sets.sas***) provides a reliable method for you to find the mismatches between the data your organization's database is programmed to store and what is actually found in the SAS data sets. Running the SAS program early in your database creation process can help you and your database administrator align the RDBMS table definitions and the contents of the SAS data sets that will be uploaded to the database. Consequently, it may well be a pragmatic move on your part to include this program as a QC step in your initial database loading routines.

Disclaimer

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

References

SAS Institute Inc. 2017. Base SAS® 9.4 Procedures Guide, Seventh Edition. Cary, NC: SAS Institute Inc. Available: https://go.documentation.sas.com/api/collections/pgmsascdc/9.4_3.4/docsets/proc/content/proc.pdf?locale=en#nameddest=bookinfo

SAS Institute Inc. 2016. SAS® 9.4 Companion for Windows, Fifth Edition. Cary, NC: SAS Institute Inc. Available: https://go.documentation.sas.com/api/collections/pgmsascdc/9.4_3.4/docsets/hostwin/content/hostwin.pdf?locale=en#nameddest=titlepage

Raithel, Michael A. 2014. Did You Know That? Essential Hacks for Clever SAS Programmers: Over 100 Essential Hacks to Make Your Programs Leaner, Cleaner, and More Competitive. Bethesda, Maryland: Michael A. Raithel Available: http://www.amazon.com/Michael-A.-Raithel/e/B001K8GG90/ref=ntt_dp_epwbk_0

ACKNOWLEDGMENTS

The author would like to thank Westat management for supporting his participation in SESUG 2019. Thanks also go to colleague Tovey Chen who provided the error messages that can be found in Table 1.

Contact Information

I would love to get your feedback on this paper; especially if you found it helpful. You can contact me at the following email address:

michaelraithel@westat.com

Appendix A – *The Compare Create Table SQL File Against Production SAS Data Sets* SAS Program

```
/*************************************************************************************************************/
/*  Program: Compare Create Table SQL File Against Production SAS Data Sets.sas                              */
/*                                                                                                           */
/*  Author: Michael A. Raithel                                                                               */
/*                                                                                                           */
/*  Created: 5/7/2019                                                                                        */
/*                                                                                                           */
/*  Purpose: This program compares a Create Table SQL file against the SAS data sets to determine           */
/*           if there are any mismatches in data sets, variables, data types, and lengths. It               */
/*           Creates Excel spreadsheets of any discrepancies found.                                          */
/*                                                                                                           */
/*  Parameters: Change the following three statements in Part 1:                                             */
/*                                                                                                           */
/*             libname SASDATA - Specify the directory holding the SAS data sets that are to be              */
/*                               loaded into Oracle. This program expects that directory will ONLY           */
/*                               house data sets that will be loaded to Oracle.                              */
/*                                                                                                           */
/*             filename TableDes - Specify the full path name of the Create Table SQL program.               */
/*                                                                                                           */
/*             %LET REPORTS= - Specify the directory where you want the reports to be written to.            */
/*                                                                                                           */
/*  Change Log:                                                                                              */
/*                                                                                                           */
/*************************************************************************************************************/

    /*****************************************/
    /* Part 1 - Specifications and allocations */
    /*****************************************/

options symbolgen mprint mlogic source2 nonumber;

/* Allocate the Production SAS data set directory */
libname SASDATA "C:\TEMP\SESUG2019";

/*Allocate the file with the database, table, and variable names */
filename TableDes "C:\TEMP\SESUG2019\CreateShoesData.sql";

/* Allocate the reports directory */
%LET REPORTS = C:\TEMP\SESUG2019;


    /*****************************************/
    /* Part 2 - Input the Create Table SQL file */
    /*****************************************/

/* Read the file with database, table, and variable names*/
data CreateTbl(drop=bigline);

infile TableDes truncover;

length table                $32.
       variable             $32.
       CreateTable_data_type $9.
       CreateTable_length     8.
       ;

retain table;

input @1 bigline $100.;

/* Remove unusable lines*/
if bigline = "" or find(bigline,");") then delete;

/* Determine the table name from "Create Table" statements */
if find(upcase(bigline),"CREATE TABLE") then do;
    bigline = strip(tranwrd(bigline,"CREATE Table",""));
    table = scan(bigline,1);
    table = upcase(table);
```

```
    delete;
end;
/* Determine the variable name, length, and type */
else do;
    /* Determine variable name */
    variable = scan(bigline,1);
    variable = upcase(variable);

    /* Determine variable type */
    if index(upcase(bigline),"VARCHAR(") > 0 then CreateTable_data_type = "Character";
              else if index(upcase(bigline),"CHAR(") > 0 then CreateTable_data_type = "Character";
           else if index(upcase(bigline),"NUMBER(") > 0 then CreateTable_data_type = "Numeric";
               else if index(upcase(bigline),"INT(") > 0 then CreateTable_data_type = "Int";
                   else CreateTable_data_type = "Unknown";

    /* Determine variable length */
    if index(bigline,"int") > 0 then CreateTable_length = 4;
        else CreateTable_length = put(scan(bigline,-1,,"dko"),8.);

end;

run;


proc sort data=CreateTbl;
    by table variable;
run;

    /*********************************************/
    /* Part 3 - Input the Production SAS data sets */
    /*********************************************/

/* Get the Production SAS data set names and variable names */
proc contents data=SASDATA._all_ out=SASDataSets(keep=memname name type length) noprint;
run;

/* Create SAS data types */
data SASDataSets(drop=type);
set  SASDataSets(rename=(MEMNAME=table NAME=variable));

table = upcase(table);
variable= upcase(variable);

length SASDataSets_data_type $9.;

if type = 1 then SASDataSets_data_type = "Numeric";
    else if type = 2 then SASDataSets_data_type = "Character";

rename length = SASDataSets_length;

run;


/* Sort the SAS data set file */
proc sort data=SASDataSets;
    by table variable;
run;

/*****************************************************************************************************/
/* Part 4 - Merge the Create Table file with the SAS Data Sets CONTENTS and create reports of mismatches*/
/*****************************************************************************************************/

/* Merge the data sets together to determine which tables/variables are common and which are only in one source
file */
data CreateTable_And_SASDataSets
     CreateTable_Only(drop=SASDataSets_data_type SASDataSets_length)
     SASDataSets_Only(drop=CreateTable_data_type CreateTable_length)
     ;
merge CreateTbl(in=a) SASDataSets(in=b);
    by table variable;

    if a and b then output CreateTable_And_SASDataSets;
    if a and not b then output CreateTable_Only;
```

```
    if b and not a then output SASDataSets_Only;

run;



    /*********************************************/
    /* Part 5 - Create reports of the mismatches. */
    /*********************************************/

ODS RESULTS Off;

/* Report #1 - Create Table file variables not found in the SAS Data Sets*/

ods excel file="&REPORTS\Create Table File Variables Not Found in SAS Data Sets.xlsx";

Proc print data=CreateTable_Only noobs;
    var table variable;
title1 "Variables Found in the Create Table File";
title2 "That Are Not in the SAS Data Sets";
run;

ods excel close;


/* Report #2 - SAS Data Sets variables not found in Create Table File*/

ods excel file="&REPORTS\SAS Data Sets Variables Not Found in Create Table File.xlsx";

Proc print data=SASDataSets_Only noobs;
    var table variable;
title1 "Variables Found in the SAS Data Sets";
title2 "That Are Not in the Create Table File";
run;

ods excel close;


/* Report #3 - variable TYPE mismatches between the Create Table file and SAS Data Sets*/
options orientation=landscape;

ods excel file="&REPORTS\Variable TYPE Mismatches between SAS Data Sets and Create Table File Variables.xlsx";

Proc print data=CreateTable_And_SASDataSets(where=(CreateTable_data_type ne SASDataSets_data_type)) noobs;
title1 "Variables with TYPE differences between the SAS Data Sets";
title2 "And the Create Table File";

run;

ods excel close;


/* Report #4 - variable LENGTH mismatches between Create Table file and SAS Data Sets*/
options orientation=landscape;

ods excel file="&REPORTS\Variable LENGTH Mismatches between SAS Data Sets and Create Table File Variables.xlsx";

Proc print data=CreateTable_And_SASDataSets(where=(CreateTable_length ne SASDataSets_length)) noobs;
title1 "Variables with LENGTH differences between the SAS Data Sets";
title2 "And the Create Table File";
run;

ods excel close;


/* Report #5 SECOND Report variable LENGTH mismatches between the   */
/* Create Table file and SAS Data Sets. This one takes into account */
/* differences between data types.                                  */
options orientation=landscape;

ods excel file="&REPORTS\Character Variable LENGTH Mismatches between SAS Data Sets and Create Table File
Variables.xlsx";
```

```
Proc print data=CreateTable_And_SASDataSets(where=(CreateTable_length ne SASDataSets_length and
CreateTable_data_type eq "Character")) noobs;
title1 "Character Variables with LENGTH differences between the SAS Data Sets";
title2 "And the Create Table File";
run;

ods excel close;
```