

Updating an MS SQL database from SAS

**Jim Groeneveld,
OCS Biometric Support,
's Hertogenbosch, Netherlands.**

PhUSE 2010 – CC04

AGENDA / CONTENTS

- 1. Creating an MS SQL database table**
- 2. The write access dilemma**
- 3. Limited write access solution**
- 4. Overview of basic steps and obstacles**
 - a. Emptying target table (structure and content)
 - b. Refilling target table, structure and content independently
 - i. Structure: variable types, conversions if incompatible
 - ii. Content: values, conversions if incompatible types
- 5. Conclusions**

Creating an MS SQL database table

```
LIBNAME SQLLIB +un+pw.....; LIBNAME SASLIB .....;
```

1. a data step, copying a SAS dataset to an SQL table:

```
DATA SQLlib.table;  
  SET SASlib.dataset;  
RUN;
```

2. PROC DATASETS, copying a dataset to an SQL table:

```
PROC DATASETS NOLIST; * (or outdated PROC COPY) ;  
  COPY IN=SASlib OUT=SQLlib;  
  SELECT dataset; * becoming a database table;  
QUIT;
```

3. SAS PROC SQL, copying a SAS dataset to an SQL table:

```
PROC SQL;  
  CREATE SQLlib.table AS  
  SELECT * FROM SASlib.dataset;  
QUIT;
```

The write access dilemma

- 1. needs general write access in MS SQL database;**
- 2. selective write access to not yet existing, new tables is not possible;**
- 3. full access is a security flaw;**
- 4. SAS should only have limited write access.**

Solution: limited write access

- 1. write access to only one or some specific, existing tables;**
- 2. overwriting those tables without actually recreating them;**
- 3. which is just modifying tables;**
- 4. by emptying them completely (data and structure);**
- 5. and refilling them with a new structure and data.**

Basic steps (5) in more detail

- 1. connecting to the database (with username and password);**
- 2. emptying the table, removing all existing data, still leaving the structure;**
- 3. removing the existing structure, the columns and their attributes;**
- 4. defining the new structure (variables and attributes from the SAS dataset);**
- 5. filling the table with data from the SAS dataset.**

Obstacles to overcome

The 5 basic steps seem simple but there are many obstacles to overcome.

The steps will be discussed one by one.

A SAS macro %SASStSQLt will be presented that performs all steps automatically.

Connecting to the database

1. with username and password;

2. using ODBC engine;

3. two methods, both applied:

a. * MS SQL lib ODBC engine;

LIBNAME MSSQLsrv ODBC DSN="SQL server"

USER=username PWD=password;

b. * Pass-Through Facility;

PROC SQL;

CONNECT TO ODBC (DSN="SQL server"

USER=username PWD=password);

Remove the table's contents

1. remove database table's data via the **SQL Pass-Through EXEC** statement:

```
EXEC (DELETE FROM &SQLtable) BY ODBC;
```

2. this could alternatively as well be done using the **SAS SQL** statement:

```
DELETE FROM &SQL_table;
```

(In here and the following SAS macro variables are being used to denote datasets, tables, variables and lists, like &MSSQLsrv, &SQLtable, etc.)

Determine existing columns

1. build list of existing column names:

```
SELECT Name INTO :SQLvars SEPARATED BY ' '
FROM DICTIONARY.COLUMNS
WHERE UPCASE (LIBNAME)='MSSQLSRV' AND
UPCASE (MEMNAME) ="&SQLtable";
```

2. alternatively using PROC CONTENTS:

```
%LET SQLvars = ; /* create empty variable;
%Var_List (DATA=&SQL_table, StoreVar =
SQLvars, Delim = %STR(,)) /* aux. macro */
%PUT SQLvars=&SQLvars; /* feedback of
existing columns in MS SQL table;
```

Remove the table's structure

1. unlike a SAS dataset an SQL table should minimally have one column.

Thus firstly add a Dummy column:

```
EXEC (ALTER TABLE &SQLtable ADD &Dummy  
FLOAT) BY ODBC; * &Dummy is unique name;
```

2. remove all other columns (variables):

```
EXEC (ALTER TABLE &SQLtable DROP COLUMN  
&SQLvars) BY ODBC;
```

In here the macro variable &SQLvars contains the list of existing columns.

Define the table's new structure-1

Add column declarations (&SASvars) to the SQL table (&SQLtable):

```
EXEC (ALTER TABLE &SQLtable ADD &SASvars) BY ODBC;
```

The content of &SASvars must meet MS SQL syntax, including VARCHAR(...), FLOAT, DATETIME, comma delimited.

SAS has to generate these declarations from the variable types and formats.

MS SQL has a specific DATETIME type,

Define the table's new structure-2

When directly CREATing an MS SQL table from SAS, SAS automatically and implicitly takes care of the type interpretations and conversions to MS SQL of the column declarations and its values from the SAS formats.

In this alternative case the new table structure has to be explicitly defined, including the later value conversions.

Define the table's new structure-3

So, every numerical SAS variable has to be checked for its associated format. If the format is one of the many explicitly checked DATETIME, DATE or TIME formats then the corresponding MS SQL column will be of the DATETIME type:

```
%LET DtTm_Fmt = 'DATETIME' 'DATEAMP' ;  
%LET Date_Fmt = 'DATE' 'DAY' 'DDMMYY' 'DOWNAM' ...many more... ;  
%LET Time_Fmt = 'TIME' 'TOD' 'HHMM' 'HOUR' 'MMSS' /*'TIMEAMP'*/ ;  
IF (Format IN: (&DtTm_Fmt)) THEN Attr=TRIM(Name) || ' DATETIME' ;  
ELSE IF (Format IN: (&Date_Fmt)) THEN Attr=TRIM(Name) || ' DATETIME' ;  
ELSE IF (Format IN: (&Time_Fmt)) THEN Attr=TRIM(Name) || ' DATETIME' ;
```

The formats have been determined from PROC CONTENTS.

Define the table's new structure-4

After all that the mentioned column declaration command can be executed:

```
EXEC (ALTER TABLE &SQLtable ADD &SASvars) BY ODBC;
```

Subsequently the temporary Dummy variable may be removed:

```
EXEC (ALTER TABLE &SQLtable DROP COLUMN &Dummy) BY ODBC;
```

Finally the SQL Pass-Through facility is no longer needed:

```
DISCONNECT FROM ODBC;
```

Now the table must be filled with values.

Fill the table with data values-1

The data to be fed to SQL, using the SAS SQL INSERT command, are actually unrelated to their already declared column names while inserted.

So care should be taken for transferring the data in the correct order with the correct type and with the correct DATETIME value conversion if necessary. The command is:

```
INSERT INTO &SQL_table SELECT * FROM &SAS_dataset;
```

But firstly value conversions needed

Fill the table with data values-2

Before doing that certain DATE values need conversion to the DATETIME format. DATE values must be multiplied by 86400 (seconds/day).

```
ARRAY ChronoVar _NUMERIC_ ; * all numeric variables;  
DO OVER ChronoVar; * (old style DO-loop over vars);  
    IF          (VFORMAT(ChronoVar) IN: (&DtTm_Fmt)) THEN ;  
    ELSE IF (VFORMAT(ChronoVar) IN: (&Date_Fmt)) THEN  
        ChronoVar = 86400 * ChronoVar; * Convert Date;  
END; * Only then perform the INSERT command here;  
INSERT INTO &SQL_table SELECT * FROM &SAS_dataset;  
PROC SQL may be terminated:  
QUIT; * Finish PROC SQL;
```

Conclusions

- 1. Transferring data from SAS to MS SQL seems easy, but is cumbersome while just having limited write access;**
- 2. Yet it is possible by modifying an existing table (structure and data) for which one has write access;**
- 3. Everything discussed here (& more) is part of the macro %SASstSQLt that can be used as a basic macro to transfer data from SAS to MS SQL.**

QUESTIONS & ANSWERS

SASquestions@ocs-consulting.com

Jim.Groeneveld@ocs-biometricsupport.com

<http://jim.groeneveld.eu.tf/SASstSQLt>