Paper 008-2008

SAS® Scripting of a Production Database into an Open Test Environment

Sigurd W. Hermansen, Himanshi Singh, and Laura Hrycyk,
Westat, Rockville, MD, USA

ABSTRACT

Inventing test and demonstration 'use case' data consumes more and more time and effort as complex Web application systems move into production. Databases supporting production systems accumulate richer and more realistic collection of data than testers could ever invent, yet using copies of production databases for testing could expose confidential information to a wider audience. SAS/Base Version 9.13 scripts (SAS programs) with SAS/Access (OLEdb, DB2, Oracle, etc.) can not only script the copying of databases, but they can also substitute fake names and other fake identifying information for analogous data in production databases. This method creates a rich and realistic test, demo, or training environment that does not expose confidential and private information to testers and trainees.

This paper features SAS Macros that shorten the repetitive process of referencing corresponding base relations (tables) in two database schema. The Macros also provide templates for scripting database copying and 'disclosure-proofing' of test and demonstration databases on the Web. An extended example, including database connections and programs, illustrates step-by-step how to create a realistic test and demo database using SAS/Access connections to database.

INTRODUCTION

Test data play an important role in the development of database and application systems, but as a systems become more complex, so do data and relations among data elements. Test data may become so complex that the task of creating realistic data values and relations among them, plus constructing fake data in the quantities and of the quality needed, drains time and resources needed for development.

With this in mind, we developed SAS System program 'scripts' that extract data from a production database, replace sensitive data items with fake data, and insert disclosure-proofed data into a test, demo, or training database. Scripting of a production database copying process protects confidential ID's, names, addresses, and other private information. The process limits inventing of 'use cases' and training scenarios to a prescribed scope and scale. Services available on the Web facilitate creation of sufficiently realistic and interrelated names, addresses, SSN, and other demographic information. (See first entry under References below.)

The entire scripting process relies critically on middleware (OLEdb) or native driver connections to a production, demo, test, or training database. The SAS/Access product provides the middleware. A data model of the production database guides the data transfer process. The SAS/Access product allows navigation from table to table within a database, or between different databases, and it supports SQL, SAS Data step procedural programming, SAS summary and statistical procedures, and SAS Macro scripting.

AN OVERVIEW OF TEST, DEMO, OR TRAINING DATABASE SCRIPTING

The system administrator kicks off the scripting process by copying the production database to a new instance of a database. The copy has in place all reference and other look-up tables and domains, as well as primary and referential key constraints.

In a database with primary key and foreign key constraints on key attributes, base relations must be deleted in one order and populated with data in the reverse order. Tuples in base relations with foreign key constraints must be deleted before base relations with corresponding primary keys. Inserts into databases that have primary keys created automatically do not include the primary key; the database

system creates a distinct key value during each insert. Primary key values have to be inserted before a foreign key in base relation with a foreign key constraint. A database system administrator typically provides a diagram of a data model that identifies foreign keys to be deleted before primary keys, and primary keys that have to be populated before foreign keys.

Names, addresses, Social Security Numbers (SSN), phone numbers, and, in some situations, birthdays, have original values replaced with fake values. We create tables in SAS that have empty values instead of actual values of confidential or private attributes. In SAS we create a sequential number key ranging from 1 to the maximum number of tuples. The sequential key allows us to assign a parallel sequence of key values to tables of fake data. A join of the tables on the sequential key replaces confidential and private data values with fake values.

To insert data from a production database and continue to maintain integrity of automatically created and other keys and relations among tables, we create empty base relations in the destination database LIKE (CREATE TABLE xxx LIKE yyy;) the base relations we plan to populate in part with fake data. Data from temporary tables containing fake data can then be inserted into the destination database. Once we have inserted data from temporary tables into the destination database, we drop the temporary tables.

Essential Steps

1. Extract primary keys, a sequencing number, and attributes containing private information (PI) from production base relations db[i] into SAS datasets dbPI[i] ;
2. For each dbPI[i], create a parallel SAS dataset fPI[i] that contains the same sequencing number and fake attribute values;
3. Join each dbPI[i] to its corresponding fPI[i] on sequencing number, substituting fake attribute values for real ones;
4. Copy the production database (DB) to a test DB (tDB);
5. In a SAS program define dbDelete and dbInsert Macros and order, first, dbDelete calls, and then dbInserts, as required by foreign keys and other data model constraints and business rules;
6. Execute SAS program (reset undo_policy=required to reverse updates upon violation of key constraints or business rules).

METHODS AND PROGRAMS

Replacements of real names, addresses, and other confidential and private data take place in SAS SQL joins of fake data sources to production base relations. For example,

```
Libname mhtsprdR  oledb provider=sqloledb properties=("data source"=SQL6 "Database"="DB" "User
ID"=Admin_RO "Password"='PW' "initial catalog"="SE") schema=dbo;
/* Add sequential number to DB table. */
Data t2Subject/view=t2Subject;
   Set mhtsprdR;
        ID=_N_;
Run;
/* Substitute fake identifiers for real ones. */
Proc sql;
Create table t3Subject as
Select t2Subject.SubjectID, t2Subject.ProjSubjectID, fd.Surname as LastName, fd.GivenName as
FirstName, fd.MiddleInitial as MI, t2Subject.SiteID, t2Subject.SubjectStatusID,
t2Subject.ReleaseGroupID, t2Subject.DOB, t2Subject.Gender, t2Subject.EnrollmentTypeID,
t2Subject.EnrollDate, t2Subject.EligibleYN, t2Subject.StatusDT, t2Subject.BeginDT,
t2Subject.UpdateDT, t2Subject.StaffID
From t2Subject inner join fakedata.fakedata as fd
         On t2Subject.ID = fd.Number
;
quit;
```

SAS/Access supports reading of data simultaneously from a database (using LIBNAME reference mhtsprdR) and from a SAS dataset (fakedata in library fakedata). The T2Subject table aliases refer to ID's and other data values from the production database, while fd table aliases refer to fake data

While updates, deletes, and inserts of data must occur in a specific order, the details of the operations tend to be ordinary and highly repetitive. SAS Macro programs work particularly well when basic operations repeat over a range of table names.

Each of the DELETE operations connects to a database via middleware and operates directly on the destination database. A SAS Macro accepts the name of an external database table as an argument and passes a DELETE program to the database. The EXECUTE operator on the DELETE executes on the database command shell a SQL program that does not return data to the SAS System:

```
/*******************************************
* Delete data from base relations
*******************************************/
%macro dbDelete (BRDel=);
      Proc Sql ERRORSTOP;
         connect to oledb as mCon (provider=sqloledb
           properties=("Data Source"=&srv "Initial Catalog"="&db"
           "User ID"=&loginID password="&pw"));
         execute (delete from &BRDel) by mCon;

         disconnect from mCon;
      Quit;
%mend dbDelete;
```

A call of the delete operation references a base relation in the destination database.

```
%dbDelete(tblDel=CheckRequests)
```

The INSERT operation references a destination base relation and a source. SAS SQL in this instance passes SQL statements through to the database system. A SAS SQL program extracts a list of attributes in a database table from metadata. Again, the EXECUTE operator executes SQL statements that do not yield relations to be passed back to the SAS System.

```
/*************************************************
* Insert data from temp datasets to base relations;
*************************************************/
%macro dbInsert (BRto=, BRfrom=);

proc sql ;
  select name into :varlist separated by ","
  from dictionary.columns
  where libname = upcase("deBC") and upcase(memname) = upcase("&BRto")
    ;

%put &varlist.

Proc Sql ERRORSTOP;
 connect to oledb as mCon (provider=sqloledb
      properties=("Data Source"=&srv "Initial Catalog"="&db"
                "User ID"=&loginID password="&pw"));

      EXECUTE (SET IDENTITY_INSERT &BRto ON ) BY mCon;
      EXECUTE (INSERT INTO &BRto (&varlist.) SELECT * FROM &BRfrom) by mCon;
```

```
        EXECUTE (SET IDENTITY_INSERT &BRto OFF ) BY mCon;

    disconnect from mCon;
        Quit;
%mend dbInsert

%macro dbInsert(BRto=, BRfrom=);
```

The INSERT operation above enables and triggers automatic creation of primary key values in an MS SQL Server database system. Other database systems require different settings or methods.

Repetitive SAS Macro calls reference in sequence each of the base relations in the production database:

```
%dbDelete(tblDel=CheckRequests)
%dbDelete(tblDel=InsurerAddress)
%dbDelete(tblDel=InsurerContact)
%dbDelete(tblDel=Provider)
….
%dbInsert(tblto=CheckRequests, BRfrom=tCheckRequests)
%dbInsert (tblto=Subject, BRfrom=tSubject)
%dbInsert(tblto=PrintBatch, BRfrom=tPrintBatch)
%dbInsert(tblto=SubjectEvent, BRfrom=tSubjectEvent)
```

INSERTs into the destination database reference temporary datasets into which we have already substituted fake data for confidential and private data values. Tables that do not contain private data do not have to be emptied and have data replaced.

Details of domain and integrity constraints - and of business rules in general - often appear in standard views of database metadata. In a MS SQL Server database system, for example, the INFORMATION_SCHEMA views display metadata. This program extracts these metadata into SAS WORK datasets:

```
proc sql;
   connect to oleDB as passThru
   (provider=sqloledb properties=("data source"=DZSQL6 "Database"="MHTS-SE" "User ID"=
"Password"=" "initial catalog"="MHTS-SE") schema=dbo)
;
   create table MSSDBRs as select * from
     connection to passThru
        (select * from INFORMATION_SCHEMA.TABLES)
     ;
   create table MSSDBRAttrs as select * from
     connection to passThru
        (select * from INFORMATION_SCHEMA.COLUMNS)
     ;
   create table MSSDBRRefCon as select * from
     connection to passThru
        (select * from INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS)
     ;
   create table MSSDBRTblCon as select * from
     connection to passThru
        (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS)
     ;
   create table MSSDBRsSchemata as select * from
     connection to passThru
        (select * from INFORMATION_SCHEMA.SCHEMATA)
     ;
```

```
    create table MSSDBRsChk_Constraints as select * from
      connection to passThru
          (select * from INFORMATION_SCHEMA.CHECK_CONSTRAINTS)
      ;
quit;
```

CONCLUSION

The SAS System SAS/Access product provides all the tools that database and application developers need to capture selected data from databases and use them to create rich, realistic, and secure test, demonstration, and training databases. SAS Macro programs replace long and tedious program segments with repetitive Macro calls. For databases that operate under complex attribute domain, key integrity, and business rule constraints, one may have to draw more extensively on the power of the SAS System to tailor substitutions of fake data for private data. SAS suffices to capture data from the Web and orchestrate a complete makeover of a production database.

REFERENCES

HTTP://DATALEXIS.COM/ARCHIVE/2007/08/23/FAKE-NAME-GENERATOR.ASPX

SAS/ACCESS 9.1.3 SUPPLEMENT FOR DB2 UNDER Z/OS

SAS/ACCESS 9.1.3 SUPPLEMENT FOR DB2 UNDER UNIX AND PC HOSTS (SAS/ACCESS FOR RELATIONAL DATABASES), SECOND EDITION

SAS/ACCESS 9.1 SUPPLEMENT FOR ODBC (SAS/ACCESS FOR RELATIONAL DATABASES)

SAS/ACCESS 9.1 SUPPLEMENT FOR OLE DB (SAS/ACCESS FOR RELATIONAL DATABASES)

SAS/ACCESS 9.1.3 SUPPLEMENT FOR ORACLE (SAS/ACCESS FOR RELATIONAL DATABASES)

SAS/ACCESS 9.1.3 SUPPLEMENT FOR TERADATA (SAS/ACCESS FOR RELATIONAL DATABASES), SECOND EDITION

ACKNOWLEDGEMENTS

Michael Raithel provided valuable comments and suggestions.

DISCLAIMERS

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

CONTACT INFORMATION

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

Sigurd W. Hermansen
Westat
1650 Research Blvd,
Rockville, MD 20850
Work Phone: 301.251.4268
E-mail: hermans1@westat.com