SAS® GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL

Tips for Mastering Relational Databases Using SAS/ACCESS®

Andrew Clapson
MD Financial Management (Ottawa, ON, Canada)

USERS PROGRAM

Tips for Mastering Relational Databases Using SAS/ACCESS®

#1 – Set up and to configure your ODBC connections

What does this mean?

- SAS/ACCESS interface to ODBC uses the ODBC (Open Database Connectivity) API to communicate with other databases

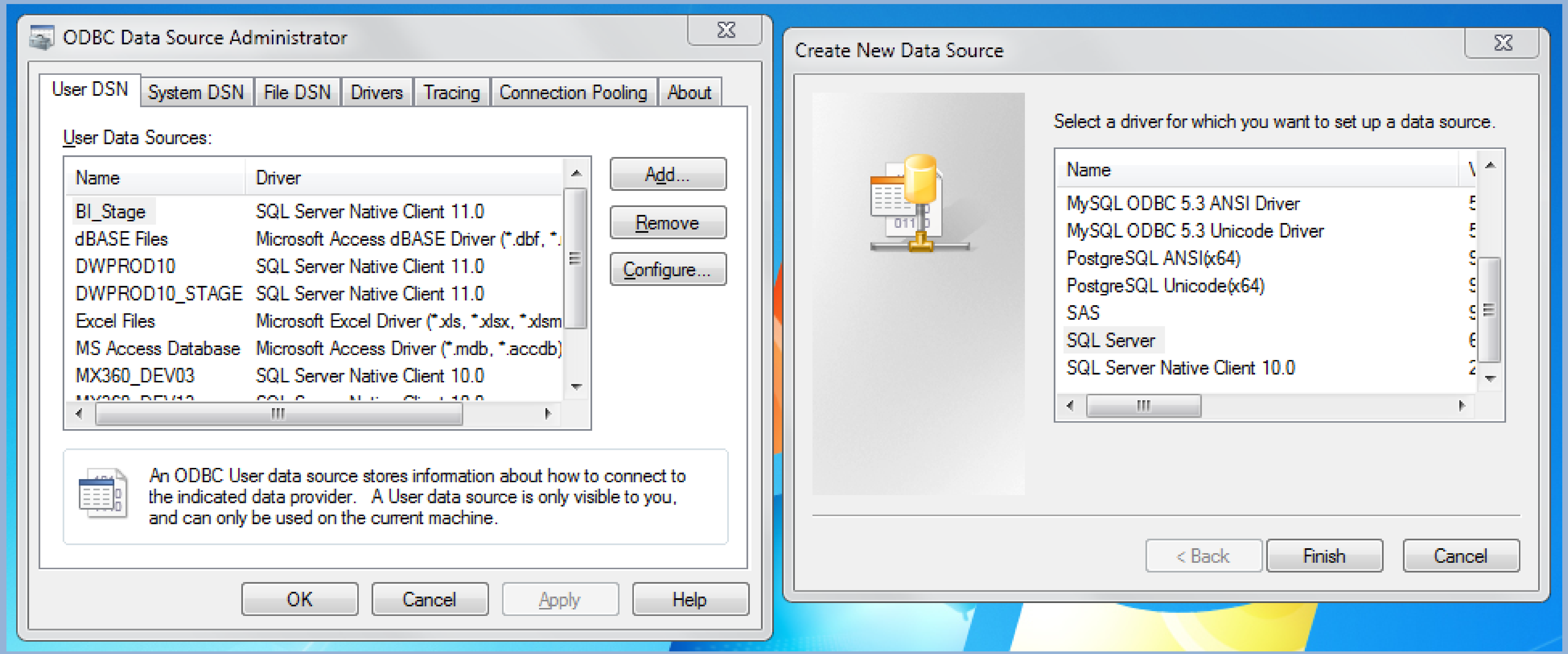
Why is this useful?

- In order for SAS to 'talk' to a particular database using ODBC, a DSN (data source name) is required
- You can tweak the settings for each server/database depending on your requirements

How does this work?

Example (using Windows 7):

- ODBC Data Source Administrator
- User vs. System DSN
- Add -> Driver -> Setup (server name, database, etc.)



User Data Sources:

Name	Driver
BI_Stage	SQL Server Native Client 11.0
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.
DWPROD10	SQL Server Native Client 11.0
DWPROD10_STAGE	SQL Server Native Client 11.0
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
MX360_DEV03	SQL Server Native Client 10.0

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Select a driver for which you want to set up a data source.

Name
MySQL ODBC 5.3 ANSI Driver
MySQL ODBC 5.3 Unicode Driver
PostgreSQL ANSI(x64)
PostgreSQL Unicode(x64)
SAS
SQL Server
SQL Server Native Client 10.0

#2 – Use PROC SQL return codes/messages from database operations

What does this mean?

- Every database has its own set of return codes and logging messages, and SAS provides automatic macro variables to use these values in your programs.

Why is this useful?

- Default error codes correspond only to the code executed by SAS – if we include information from the database itself it becomes much easier to troubleshoot errors or poor performance.

How does this work?

```
%if (&SQLXRC. NE 0) %then %do;
 %put ERROR: Return code: *&SQLXRC.*;
 %put ERROR- Error message: *&SQLXMSG.*.;
 %abort cancel;
%end;
```

Notes:

- The values reset with each statement that is executed
- 0 = no error; 4 = warning
- Must be using pass-through to an outside database

Tips for Mastering Relational Databases Using SAS/ACCESS®

#3 – Use EXECUTE() for enhanced in-database functionality

What does this mean?

- The EXECUTE() statement passes code directly to your database for execution (this is explicit pass-through).

Why is this useful?

- Will run unaltered database-native code (i.e. not SAS-style PROC SQL code)
- Permits the use of DBMS-specific functionality such as calling stored procedures, using parameters, and using CTEs.

How does this work?

```
EXECUTE (
 DECLARE @current_date_key int
 SET @current_date_key = (SELECT
   MAX(date_key) FROM example_table)

 SELECT id, sales INTO #sales_today
 FROM example_table
 WHERE (date_key = @current_date_key)
) BY DB_NAME;
```

Tips for Mastering Relational Databases Using SAS/ACCESS®

#4 – Bring your SAS data into an external database

What does this mean?

- Take SAS data (stored in WORK, or from another library) and load it into a relational database

Why is this useful?

- Often the majority of your data does not reside in SAS, and you can push the work to your database
- If you are using SAS as an ETL (extract, transform, and load) tool
- Your pass-through run-times are unacceptably long

How does this work? (MS SQL Server)

```
proc SQL;
 CREATE TABLE DB.'#temp_table_name'n AS
 SELECT varName_1, varName_2
 FROM WORK.exampleData;
QUIT;
```

Notes:

- A name-literal is required here because SAS does not consider '#' to be a valid table name
- Syntax will vary depending on database used
- Some formats may not translate

Tips for Mastering Relational Databases Using SAS/ACCESS®

#5 – Use your database's data-loader

What does this mean?

- Every database has a data-loading utility, which SAS can use to load data

Why is this useful?

- The SAS default is to use the SAS/ACCESS engine, which essentially does a row-by-row INSERT operation.
- Using your database's data-loader utility will vastly improve the speed of any append/insertion operations

How does this work?

```
proc SQL;
 CONNECT TO ODBC AS DB_NAME (
  DSN = 'DWPROD10'
  BULKLOAD = yes);

 CREATE TABLE DB.exampleData AS
 SELECT * FROM WORK.exampleData;
QUIT;
```

Notes:

- User permissions are required on the database side
- There are many other options to tweak performance, such as READBUFF and INSERTBUFF

SAS® GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL