Using SAS Views to Insure Data Consistency and Ease Maintenance

Presented by:
Patrice Cohen

New York State
Department of Taxation and Finance
Office of Tax Policy Analysis

Prepared for the Federation of Tax Administrators
Revenue Estimation and Tax Research Conference
FTA in Portland Oregon
September 19, 2006

What Were The Problems We Needed To Address

- We provide annual data sets both internally and externally
  - Each requires its own specifications
  - Some files contain only a subset of fields
  - Some files contain identifiers while others do not
  - All versions come from the same originating file
  - Once we create the originating file, we then create all the versions
- Invariably, we discover a problem with the file. Most times minor, occasionally significant (all before we distribute the file).
  - At that point the originating file is corrected then we have 2 choices
    - Separately correct each of the individual files (using the same code as against the original) OR
    - Recreate all of the secondary files.

What Were The Problems We Needed To Address – cont.

- We then have to insure all the changes are reflected on all the secondary files and that all secondary files are created appropriately.
- In addition, all of these versions, created as separate files use a lot of space on the server.
- Since we need to keep the naming convention for all the files, we also have a problem of making sure the correct version is updated.

Our Solution-- SAS Views

- While using point and click SAS queries and SQL, we discovered that we could save the results as either an output data set (which we are all familiar with) or a VIEW.
- We had not used this functionality before, and its discovery solved the aforementioned problems.

Our Solution-- SAS Views (cont.)

- For those familiar with DB2, this is similar to DB2 views. The table remains intact, and the view, while it looks like a table, simply executes a query.
- SAS views also look like SAS datasets and can be used just like a SAS dataset, but they just execute the query against the main SAS data set. (They contain no data in and of themselves)

Simple code to create SAS view

- Assume you want to store your views in R:\Views and your Main data is stored in S:\Maindata
- Your program would be something like this:

```
** CREATE VIEWS ** ;
Libname SFVIIEWS 'R:\views';

PROC SQL noprint ;
        TITLE '2003 Article 9A Study File (Study File with Identifiers)';
        CREATE VIEW SFVIEWS.Article9a_allteam_studyfile_2003 AS
        SELECT *
        FROM Maindata.article9a_studyfile_2003
         Where min_ind ne 1
                USING LIBNAME Maindata 'S:\Maindata';
```

SAS code
(cont.)

- Most important part of code is
  - "Using libname xxxx '\yyy';
  - The first time we did this, only the person who created the view could use it.
  - If the views are created without this line, then other users will get an error when they access the view if their libnames in their SAS profile are not identical to the one who created the view
  - We found this out the hard way (internally, fortunately)

Advantages of Views

- Once the views are created, if any problems are found, and the main data is corrected, all of the views are automatically updated
- Can do anything with SAS views that can do with SAS data sets
- Significant disk space saver but still allows various flavors of the data
- Even though we deal with data one year at a time, we plan to include all of our data in one data set and then use views to segregate and analyze by year