Teaching a New Dog Old Tricks - Using the EXCEPT Operator in PROC SQL and Generation Data Sets to Produce a Comparison Report

Stanley Fogleman, Harvard Clinical Research Institute, Boston, MA

ABSTRACT
It is possible to produce a comparison report on rather short notice using the EXCEPT operator in PROC SQL to list only those rows that have changed since the last time the report was run. The generation data set represents the data today versus whenever the report was last run.

INTRODUCTION
A few years back, I presented a paper at NESUG on PROC COMPARE. I was struck by how much time I spent preparing the data to “present” the data in an acceptable format. The approach of this paper was to go “back to the basics” and to use as straightforward an approach as possible.

THE PROBLEM
A database table that is updated regularly is sampled at regular intervals (in this case, bi-weekly). A new generation is created every two weeks and the current generation is compared to the previous one using PROC SQL. What could be simpler?

BACKGROUND
Once upon a time, there were these big machines called mainframes. They had an operating system called MVS. They had a directory structure called the Generation Data Group (or GDG). The number started at 0001 and went to 9999. I never remember getting anywhere near this high. There was also a version number available, but the less said about that, the better. SAS® has adopted this paradigm in a most peculiar matter. For example, the current version is without (emphasis added) any generation number (my_dataset). The most recent is the highest number present(my_dataset#004, assuming there were three datasets before it.).

Example:
my_dataset (current version)
my_dataset#001 (oldest)
my_dataset#002
my_dataset#003
my_dataset#004 (most recent)

Another way of referring to generation datasets is by generation number. For example, gennum = -1 refers to the most recent version and gennum = 0 refers to the current version. Confusingly, positive numbers are absolute generation numbers! In other words, Gennum = 3 refers to my_dataset#003 (!!!)

The beauty of using generation datasets is that the same compare code can be run every time without resorting to macro variables, or worse, making modifications on the fly.

For example the code:

```
proc sql noprint;
create table differ as
select columna, columnb
from PRODLIB.serious_au(gennum=-1) a
except
```
select columna, columnb
    from PRODLIB.serious_ae(gennum=-0) b;

requires no changes each time it is run.
The second part of the equation is to utilize the “except” operator. The definition of the “except” operator is simplicity itself. The operator produces rows that are part of the first query only. When this result table is joined with the old and new versions of the table, a report can be produced showing a side-by-side comparison. Tip: Adding “select ’1’ as table” allows us to distinguish between the two tables).

proc sql noprint;
    create table oldtable as
    select '1' as table,
        a.*
    from PRODLIB.serious_ae(gennum=-1) a,
        PRODLIB.differ b
    where (a.keyfield=b.keyfield)
    order by keyfield;

    create table newtable as
    select '2' as table,
        a.*
    from PRODLIB.serious_ae(gennum=0) a,
        PRODLIB.differ b
    where (a.keyfield=b.keyfield)
    order by keyfield;
quit;

produces a report with detail lines such as:

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>88</td>
<td>1</td>
<td>22-</td>
<td>22-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>ORIGINAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Nov-</td>
<td>Nov-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COMMENT</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>88</td>
<td>2</td>
<td>22-</td>
<td>22-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>CHANGED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Nov-</td>
<td>Nov-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COMMENT</td>
<td></td>
</tr>
</tbody>
</table>

It can be seen that in the last three columns that the comment has changed and that a missing value has been replaced by a 2 in the newer file.

To summarize, the steps that must be taken to create a comparison report are:

1) create a generation data set
2) write a query containing only the fields you wish to check for differences on.(primary keys only)
3) Match the delta file produced with the current and most recent files
4) Produce a simple listing report

The third step may not seem so intuitive. It is basically a process to select only those records that changed from both files. That way, they can be printed “side-by-side” so that differences are readily apparent.
CONCLUSIONS
Using the combination of generation data sets and the “except” operator in PROC SQL can produce a useful comparison report on short notice.

REFERENCES:

ACKNOWLEDGMENTS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Thanks to Al Verheggen for getting me in this crazy business to begin with.
Thanks to Mani Carathanasssis for teaching me everything anyone would ever want to know about JCL and Generation Data Groups.

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

    Stanley Fogleman
    Harvard Clinical Research Institute
    930 Commonwealth Ave West
    Boston, MA 02215
    Web: www.hcri.edu

********************************************************************