PROGRAMMING EFFICIENCIES USING PROC DATASETS

Introduction

There is usually more than one way to get anything done in SAS and, if you don't have time to test and compare, it is usually not obvious which method will get the job done fastest. And of course, what works most efficiently at one site may not hold true at another. There is a general rule of thumb that procedures designed to do one thing (e.g., PROC PRINT) execute faster than multipurpose procedures (e.g., using PROC SQL to print). But there is an important exception to that rule: PROC DATASETS. This workhorse of a procedure can perform a very wide range of tasks, and usually faster than other methods. You can even manage several datasets in one PROC DATASETS statement. The purpose of this presentation is to provide a brief overview of the many tasks that PROC DATASETS can perform, and why it can do them faster than other methods.

Here's some of the tasks you can do with PROC DATASETS:
Rename, copy, move or delete datasets
Append a dataset (or multiple datasets) to another
Assign, modify or remove variable informats, formats and labels
Rename variables
Assign, change or remove passwords
Create or delete indexes
Create and manage audit files
Create and manage integrity constraints
List the SAS files in a SAS library
List the attributes of a SAS dataset

Why PROC DATASETS is Faster

There are two or three logical components of a SAS data set:

1. Descriptor information
2. Data values
3. Index(es)

The descriptor component contains information about attributes of the dataset and its variables: informats, formats, labels, variables names, type, and length - the information that is output when you run PROC CONTENTS or a PROC DATASETS Contents statement.

The descriptor information ensures that SAS datasets are self-documenting.

In a large dataset, the descriptor portion is very small compared to the data values component. While datasteps can be used to complete the same tasks that can be done with PROC DATASETS, the dataset is read into the program data vector. PROC DATASETS, however, can complete these tasks without having to read in the data values - changes are made to the relatively small descriptor component. That's where the efficiency gains come in - the tasks can be completed faster and more efficiently than with a data step.

Here are some simple examples of PROC DATASETS in action:

Renaming SAS Files:

```
Libname students 'g:\confid\demographics';

Proc datasets lib=students;
  Change old_name=new name;
Run;
```

A libref is not needed if the datasets you are updating are in the work or SASUSER library. In this example, work.test_scores_current is renamed to work.test_scores_prior_year:

```
Proc datasets;
  Change test_scores_current=test_scores_prior_year;
Run;
```

Renaming Variables:

```
Libname students 'g:\confid\demographics';

Proc datasets lib=students;
  Modify LakeTravisISD;
    ID=studentid;
Run;
```

If there's already a variable named studentid an error statement will be generated.

If there is a simple index for the variable you rename, PROC DATASETS also renames the index. If the renamed variable is used in a composite index, the composite index will automatically reference the new variable name.

Copying Datasets:

```
Libname source 'g:\confid\location1';
Libname dest    'g:\confid\location2';
```

This example copies all the datasets in the source library to the destination ("dest") library:

```
Proc datasets library=source;
  Copy out=dest;
Run;
```

To copy only some of the files, add a SELECT or EXCLUDE statement:

```
Proc datasets library=source;
  Copy out=dest;
  Select attendance demographics assessment;
Run;
```

```
Proc datasets library=source;
  Copy out=dest;
  Exclude grade: ;
Run;
```

Deleting SAS Files:

```
Libname students ‘g:\confid\demographics’;

Proc datasets lib=students;
  Delete gradePK gradeKG;
Run;
```

Be careful - the datasets are deleted immediately. If the dataset you delete is indexed, PROC DATASETS also deletes the indexes.

Appending Datasets

This example appends the dataset work.newstudents to the base dataset work.students:

```
Proc datasets;
  Append base=students
             data=newstudents;
Run;
```

You can also use a WHERE statement to restrict the observations in work.newstudents that are appended to the base dataset:

```
Proc datasets;
  Append base=students
             Data=newstudents;
    Where district=‘Austin ISD’;
Run;
```

When you use a DATA step with a SET statement to append one dataset to another, SAS reads all the observations from both datasets into the Program Data Vector. But when you use PROC DATASETS, only the observations in the DATA= dataset are read into the Program Data Vector. The bigger the base dataset is, the more efficient PROC DATASETS is compared to using a DATA step.

If there are variables in the BASE= dataset that are not in the DATA= dataset, those variables will be blank for the observations from the DATA= datasets.

If there are variables in the DATA= dataset that are not in the BASE= dataset, you can use a FORCE option with the APPEND statement. The additional variables will be dropped and a warning message will be written to the log.

If the variables in the two datasets have different attributes, the attributes in the BASE= dataset will prevail.

PROC DATASETS statements are executed in the order they are written. While executing multiple statements in one procedure, you can manipulate files with multiple tasks.

Formatting Variables

The FORMAT statement, used with a MODIFY statement, is used to assign, change or remove variable formats:

```
Libname perm ‘G:\confid’;

Proc datasets;
  Modify students;
    Format total_students gifted_students voced_students comma9.2;
```

```
Run;
```

Labeling a Dataset:

```
Proc datasets;
  Modify students;
    Format total_students gifted_students voced_students comma9.2;
    Label=‘School Year 2010 - 2011 Students;
Run;
```

Labeling Variables:

```
Proc datasets;
  Modify students;
    Format total_students gifted_students voced_students comma9.2;
    Label=‘School Year 2010 - 2011 Students;
    Label total_students=‘Total Enrollment”
             Gifted_students=‘Students Participating in Gifted/Talented Program’
              Voced_students=‘Students Participating in Vocational Education Program’;
  Run;
```

You don’t have to run PROC CONTENTS now to see the modifications you’ve made – you can generate the same output from PROC DATASETS using a CONTENTS statement:

```
Proc datasets;
  Modify students;
    Format total_students gifted_students voced_students comma9.2;
    Label=‘School Year 2010 - 2011 Students;
    Label total_students=‘Total Enrollment”
             Gifted_students=‘Students Participating in Gifted/Talented Program’
              Voced_students=‘Students Participating in Vocational Education Program’;
  Run;
  Contents data=students;
  Run;
Quit;
```

Removing Labels and Formats:

```
Proc datasets;
  Modify students;
    Attrib _all_ label=’ ‘;
    Attrib _all_ format=;
Run;
```

Password Management:

To password protect a dataset with the password “pencil”:

```
Proc datasets;
```

```
  Modify students (pw=pencil);
```

To remove a password:

```
Proc datasets;
  Modify students (read=pencil);
Run;
```

To change a password (in this example, from “pencil” to “pen”):

```
Proc datasets;
Modify students (read=pencil/pen);
```

How PROC DATASETS Executes

PROC DATASETS uses RUN-group processing, so you can submit RUN statements without ending the procedure. SAS reads program statements associated with a task until it reaches a RUN statement or an implied RUN statement. Several statements execute immediately when submitted - the RUN statement is implied:

- PROC DATASETS executes immediately
- APPEND forms a single-statement RUN group
- CONTENTS forms a single-statement RUN group
- MODIFY (and any subordinate statements) run immediately (no other statement necessary)
- COPY forms a RUN group.

This example includes several run groups in one PROC DATASETS statement:

```
LIBNAME dest 'g:\confid';

proc datasets;
    /* RUN group */
  change students=students0910;
  delete students0809;
  exchange district=current_district;
    /* RUN group */
  copy out=dest;
    select report;
    /* RUN group */
  modify demog;
    label age='Age as of Oct 30 2010';
    rename id=studentid;
    /* RUN group */
  append base=TravisCounty data=Austin_ISD;
quit;
```

How to End the PROC DATASETS Procedure

PROC DATASETS executes until it encounters a:

- QUIT statement
- RUN CANCEL statement
- DATA statement; or
- A new PROC statement.

Please see the SAS documentation for PROC DATASETS for more details. There are several features and options available that have not been included here.

References

SAS Institute Inc. (2010), *"SAS Procedures", SAS Version 9 Online Documentation*, Cary, NC: SAS Institute Inc.

Contact Information

Linda Adams
Texas Education Agency
1701 N. Congress Ave.
Austin, TX 78701
Linda.Adams@TEA.state.tx.us