Optimum Clinical Data Structures for Use with SAS/PH-Clinical™ Software
Sandra D. Schlotzhauer, SAS Institute Inc., Cary, NC
Andrew T. Fagan, SAS Institute Inc., Cary, NC

Abstract
SAS/PH-Clinical software is a new SAS Institute software product for increasing the productivity of clinical reviewers within drug or biotechnology research organizations. Although the product is designed to work with a variety of clinical data structures, the internal algorithms handle some structures more efficiently than others. In addition to data set structures, any aspect of the clinical data that requires a view overlay affects the efficiency of the software. For example, if variables must be renamed in the study definition process, SAS/PH-Clinical software overlays a DATA step view on the original clinical data. Before any patient group can be created, the software generates this view and reads it, rather than reading the original data. This paper discusses aspects of clinical data structures that can affect performance. The discussion also provides techniques for defining studies quickly, including using the FASTLOAD utility, creating new variables, and obtaining administrative reports.

Introduction
Before focusing on how clinical data structures affect performance, you need to know two key concepts for SAS/PH-Clinical software and to understand the definitions for the clinical data structures handled by the software.

First, SAS/PH-Clinical software consists of three major building blocks: the Resource Manager, the Patient Group component, and the Functional Components. The Resource Manager contains the meta data for clinical trials, users, security groups, devices, and more. The Patient Group Generator is built on the Resource Manager. It captures information from the Resource Manager and combines that information with input from users to build patient groups. The Functional Components are built on the Patient Group Generator, and provide a menu-driven interface for users to browse the data, create graphs and tables, perform analyses, and more. The diagram in the next column depicts the building blocks.

The second key concept is the Product Administrator, or PA. This person defines trials, users, and devices, and performs installation and maintenance tasks. Your site can have more than one administrator, so you can choose, for example, to identify one administrator for each therapeutic area. The most important task PAs perform is to define clinical trials to the software. One step in defining trials is to identify the data set structures, or levels.

When you define trials, the data can reside in SAS data sets, DBMS tables, or a combination of the two. The first step in identifying the data is to identify their location (librefs, data sets, or view descriptors). The next step is to identify the level for each data set or table. (For brevity, the rest of this paper refers to data sets; the points made also apply to DBMS tables.) In SAS/PH-Clinical software, you can specify one of four levels:

- P, for Patient, used for data sets with one observation per patient. For example, medical history and demographic data are often recorded this way.
- V, for Visit, used for data sets with one observation per patient per visit. Examples often include signs and symptoms data, efficacy data, and lab data.
- PE, for Patient Event, used for data sets with one observation per event for each patient. Examples include Adverse Event (AE or ADR) data, where each adverse experience is entered as one event...
and so, one observation). A key point here is that these events are not tied to visits.

VE, for Visit Event, used for data sets with one observation per event for each patient and visit. Examples include data that are collected more than once for each visit, such as data for the left and right eye in an ophthalmology trial. Sometimes lab data are collected this way, where each observation in the lab data set has variables to identify the patient, visit, lab test, and lab result. VE data sets are the least efficient data set organization for use with SAS/PH-Clinical software.

These structures fall into a hierarchy with the number of observations per patient increasing as you move down the hierarchy. P data sets are at the top, since they contain only one observation per patient. The next level of the hierarchy depends on whether the data are visit-related. One branch contains PE data sets, since they contain multiple observations per patient. The second branch is visit-related, and V data sets appear above VE data sets in the hierarchy. Both V and VE data sets are visit-related, but V data sets contain only one observation per patient per visit, and VE data sets contain more than one observation per patient per visit. The diagram below depicts the hierarchy.

The next several sections discuss how data set levels affect performance when SAS/PH-Clinical software builds a patient group.

### Using Few VE Data Sets

When creating a patient group, SAS/PH-Clinical software uses a combination of Structured Query Language (SQL) views and SAS DATA step views. The end result is the WORK.PHDATA data set. This data set has a very specific structure; so for example, patient-level information such as the patient's age is available on every record.

During the process of building the WORK.PHDATA data set, the Patient Group component uses SQL views to create full joins of VE data sets required for the patient group. Suppose the EYE and REFDOI variables are in the EYEXAM data set, which is a VE data set. And, suppose the BUN and CREATINE variables are in the LABS data set, which is also a VE data set. If users choose these four variables for a patient group, the product builds a view that is the full SQL join of these two VE data sets. If the original data sets are large, this view can consume large amounts of memory and take a considerable amount of time to build. The memory and time requirements increase as the number of VE data sets increases.

The full join ensures that the values for all four variables (EYE, REFDOI, BUN, and CREATINE) are available in one observation in WORK.PHDATA. This structure enables users to query combinations of values in different original data sets. For example, users could find patients with refraction diopters greater than 10 and creatinine greater than 150 (REFDOI>10 and CREATINE>150).

A key point in building WORK.PHDATA is that any WHERE-processing is performed as one of the final steps. Thus, if users choose the By criteria or By Patient Number radio buttons in the Define Patient Group window, the product generates WHERE criteria from the user-specified criteria or patient numbers. These criteria are applied to a view that contains not only the combined VE information but the combined P, V, and PE information as well. The observations that meet the criteria are then saved as WORK.PHDATA.

In terms of performance, the optimum situation is a study with at most one VE data set. Note that SAS/PH-Clinical software can handle studies with multiple VE data sets. As these data sets become large; or as the number increases, you will see performance degradation during the process of building a patient group. Notice that this should not have a major effect on other components; once the patient group is built, the rest of the components work with WORK.PHDATA. One caveat is important here: SAS/PH-Clinical software is an interactive SAS software product, and if WORK.PHDATA is very large, you will see the same performance inside SAS/PH-Clinical software as you would in other SAS software products. Suppose WORK.PHDATA has 500,000 observations. Within SAS/PH-Clinical software, you receive essentially the same response time when working with a patient...
group of this size as you would receive in working interactively with a SAS data set of this size.

Since the optimum may not be possible, one approach for optimizing performance is to minimize the number of VE data sets. To do so, consider which of the VE data sets for a study are appropriate for conversion to V data sets. Suppose a study contains the following VE data sets:

- EYES with 2 observations per patient per visit
- LAB with 28 observations per patient per visit
- CONMED with from 1 to 15 observations per patient per visit.

The EYES and LAB data sets are candidates for conversion to V data sets. Since these two data sets have the same number of observations for each patient for each visit (2 and 28, respectively), converting the data sets results in very few variables with missing values.

For example, suppose EYES contains the PATIENT, VISIT, EYE, REFDIOP, ASTIG, and GLAUCOMA variables. Each patient has an observation for each eye and each visit. The EYE variable identifies the event, and making this a VE data set. The first few observations from the data set are:

```
PATIENT VISIT EYE REFDIOP ASTIG GLAUCOMA
1 1 L 2.5 N N
1 1 R 3.25 N N
1 2 L 2.5 N N
1 2 R 3.5 N N
```

Suppose converting this VE data set to a V data set results in the EYES2 data set with PATIENT, VISIT, REFDIOP, REFDIOPR, ASTIGL, ASTIGR, GLAUCOML, and GLAUCOMR variables. The first few observations in EYES2 (corresponding to those shown above for EYES) are:

```
PATIENT VISIT REFDIOP REFDIOPR ASTIGL ASTIGR
1 1 2.5 3.25 N N
1 2 2.5 3.5 N N
```

If the study has 100 patients with 5 visits, the conversion reduces this from a 1000-observation data set with 5 variables to a 500-observation data set with 8 variables. More importantly, the conversion to a V data set results in the product combining the EYES2 data set with other V data sets using SQL views. Since the V data set has fewer observations per patient, this process uses fewer resources than combining multiple VE data sets.

For the LAB data set, suppose there are again 100 patients and 5 visits. Each patient has 28 observations for each visit, resulting in a data set of 14,000 observations. Suppose LAB contains the variables PATIENT, VISIT, LABTEST, and LABVALUE. The LABTEST variable identifies the event making this a VE data set. Converting this to a V data set, LAB2, results in variables such as PATIENT, VISIT, BUN, CREATINE, GLUCOSE, WBC, RBC, HGB, HCT, and so on. Even if some patients have only 25 lab tests, the LAB2 data set is not sparse; most variables have values for most visits. Suppose LAB2 has 500 observations with 36 variables.

Converting both of these data sets means that the product combines the EYES2 and LAB2 data sets, resulting in a SQL view with 500 observations and 36 variables. Combining the original data sets, EYES and LAB, results in a SQL view with 140,000 observations and 8 variables.

The CONMED data set is not a good candidate for conversion. The Frequency Of Visits report from the HELPDEFS program (provided with SAS/PH-Clinical software) can assist you in making this decision. Suppose the report shows the following for the first visit:

- 35 patients had 1 visit-number-001
- 12 patients had 2 visit-number-001
- 5 patients had 3 visit-number-001
- 2 patients had 4 visit-number-001
- 1 patient had 5 visit-number-001

This report tells you that there are up to 15 concomitant medications for the first visit. Converting this data set to a V data set would result in a data set with large amounts of missing data (which is inefficient), since you need to define 15 variables for concomitant medications. And, for all but 8 patients, the last 13 variables contain missing values. In addition, this is the report for the first visit; perhaps other visits will require adding even more variables for concomitant medications. For example, if the third visit shows a patient with 17 observations, you need to define 17 variables.

Finally, you should think of how users might want to look at the data. Rather than querying to find a given concomitant medication in CONMDR1 through CONMDR15 (in a V data set), they are likely to want to query one variable, CONMDRUG.

Contrast this with the LAB data set. In most cases, users are likely to want to query on a specific lab test. Suppose users want to find patients with a white blood cell count above 8. If the patient group is built using the VE data set (LAB), their query is:

```
LABTEST=WBC AND LABVALUE>8
```
If the patient group is built using the V data set (LAB2), their query is:
\[ \text{DN} > 8 \]

The second query is more intuitive.

The following is a summary of the points contained in the examples above:

- Conversion is more appropriate when the VE data set contains approximately the same number of observations per patient for each visit.
- Conversion is more appropriate when it results in a more intuitive querying process for the users.

### Using Few PE Data Sets

This topic is similar to the previous one in that the software uses SQL views to join PE data sets. If users choose variables in two or more PE data sets, the full join of those data sets is constructed. If the original data sets are large, this view can consume large amounts of memory and take a considerable amount of time to build. Since PE data sets are often smaller in size than VE data sets, the performance issues are less obvious and, in fact, may not be noticeable if the data sets are fairly small and if no VE data sets are involved in the patient group. However, after constructing the full join of PE data sets, the software then constructs the full join of the combined PE view and the combined VE view. Thus, if VE data sets are involved in the patient group, the performance advantages of using few PE data sets are more obvious.

In general, the fewer PE data sets that are defined for a particular study, the faster the product creates a patient group. Solutions include converting PE data sets to P data sets and combining information from two or more PE data sets into one PE data set. Suppose a completed study contains the following PE data sets:
- ADVERS1 with 1 to 5 observations per patient
- ADVDETS with 1 to 5 observations per patient
- HOSP with 0 to 1 observations per patient
- DIARY with 0 to 40 observations per patient

The ADVERS1 data set contains the basic information for adverse experiences, and ADVDETS contains the details for each adverse event. These two data sets are candidates to combine into one PE data set. HOSP contains details on patients who required hospitalization during the study, with one observation per hospitalization per patient. Since the study is complete and no patient had more than one hospitalization, this data set can be converted to a P data set. The DIARY data set is neither a good candidate for combination with another PE data set nor for conversion to a P data set; it must remain as a PE data set.

Consider the ADVERS1 and ADVDETS data sets. Since ADVDETS contains the details for each adverse event reported in ADVERS1, each data set contains the same number of observations for each patient. Suppose ADVERS1 contains the variables PATIENT, ADVCODE, ADVDESC, DTSTART, and DTEND, as follows:

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>ADVCODE</th>
<th>ADVDESC</th>
<th>DTSTART</th>
<th>DTEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>340</td>
<td>Headache</td>
<td>02MAR92</td>
<td>02MAR92</td>
</tr>
<tr>
<td>1</td>
<td>341</td>
<td>Migraine</td>
<td>07MAR92</td>
<td>09MAR92</td>
</tr>
<tr>
<td>15</td>
<td>921</td>
<td>Broken leg</td>
<td>21FEB92</td>
<td></td>
</tr>
</tbody>
</table>

Suppose ADVDETS contains the variables PATIENT, ADVDESC, RELATE, CONTINUE, and REOCCUR as follows:

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>ADVDESC</th>
<th>RELATE</th>
<th>CONTINUE</th>
<th>REOCCUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Headache</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>1</td>
<td>Migraine</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Broken leg</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

To create one PE data set, merge the two data sets by PATIENT and ADVDESC. Since the study is complete, the most efficient approach is to create the new data set, say ADVALL, and use ADVALL in study definition. If the study is ongoing or information in the original data sets may change, you should instead create a DATA step view that combines ADVERS1 and ADVDETS. Then, use this view in study definition. This, however, will not result in as great a reduction in processing time as working against the ADVALL data set.

Suppose you have a slightly different situation, where the details are not always reported in the ADVDETS data set. If, in most cases, the details are reported, you should still combine the two data sets.

For the HOSP data set, you know there is at most one observation per patient. Converting this data set involves no SAS code; simply identify the data set as a P data set when defining or updating the study definition. Then, the HOSP data set isn’t combined with other PE data sets in SQL views. Instead, it is joined with other P data sets in SQL views that are constructed to combine all one-observation-per-patient variables. Since the SQL join first constructs an internal (in memory) table of all possible combinations (including cross-patient combinations), the SQL view for the P data sets requires less memory than that for PE data sets.

The DIARY data set is a PE data set that cannot be easily combined with any other PE data set. In
In addition, converting DIARY to a P data set would involve creating a data set that might have many missing values, which is inefficient. Suppose the variables in DIARY are PATIENT, DATE, and COMMENT, where COMMENT is a 200-character variable that patients use to briefly summarize the day. Converting DIARY to a P data set would involve transposing in order to create the COMENT1 through COMENT40 variables. If the majority of patients entered 40 comments, the resulting data set would have few missing values. If, however, the majority of patients entered only a few comments (10 or fewer, for example), the resulting data set would be very sparse.

In addition, consider how users will want to query the diary data. The most intuitive query is to search for text in the COMMENT variable, for example, searching for the word "sleepy." Since the Subsetting feature in SAS/PH-Clinical software provides at most 13 criteria at a time, users then have to search for COMENT1 through COMENT40 containing the word "sleepy." This is more awkward. Conversion to a P data set results in a very awkward query. Users then have to search for the data set containing the word "sleepy." Since the Subsetting feature in SAS/PH-Clinical software provides at most 13 criteria at a time, users would need to apply the first 13 criteria, save the new patient group, apply the second 13 criteria, save this new patient group, and so on. The process becomes quite cumbersome and could be frustrating for users.

The following is a summary of the points in these examples:

- Combining PE data sets is appropriate when the data sets have the same number (or almost the same number) of observations per patient.
- If a study is complete and a data set that could have contained more than one observation per patient does not, then define this as a P data set.
- Consider how users will want to query the data before transposing a PE data set to a P data set.

### Combining P and V Information

Combining P information into fewer P data sets and combining V information into fewer V data sets can reduce the time required to create a patient group. This is best illustrated by examples. Suppose a study has DEMOG, HISTORY, ENTRY, and EXCLUDE as P data sets, and VITALS, EFFICACY, and LAB as V data sets.

### Combining P Data Sets

Deciding to combine P data sets involves deciding whether you should join these data sets as part of defining the study or let the Patient Group component join the data sets as needed based on users' choices when creating a patient group. If you join the data sets in defining the study, you can either:

- create a new data set that is the match-merge of the original data sets, or
- create a view that is the match-merge of the original data sets.

In both cases, the matching is done using the patient identifier variable. The advantage of a new data set is that the data are immediately available to the Patient Group component; however, this means that as the data in the original data sets change over time, you will need to update the merged P data set periodically for the study to remain up to date. The advantage of a view is that the data are automatically updated since the view is built each time it is accessed; however, the disadvantage is that the view is rebuilt when the patient group is created, which adds a step to the process.

The decision to combine P data sets also involves being aware of what variables users are likely to choose together. This is best explained by example.

Suppose medical reviewers are likely to want to look at demographics and medical history information together, and that this information is contained in the DEMOG and HISTORY data sets. Further, suppose that clinical data monitors are likely to want to look at entry and exclusion criteria together, and that this information is contained in the ENTRY and EXCLUDE data sets. (This might be the case, for example, when the monitors are responsible for ensuring that patients incorrectly included in the study are identified and handled as quickly as possible.)

In this situation, you should combine the DEMOG and HISTORY data sets (into DEMHIST, say), and combine the ENTRY and EXCLUDE data sets (into CRITERIA, say).

The decision about whether to use a view or an actual data set when you combine the data sets depends on your answers to the updating and waiting decisions discussed earlier. In either case, the reason that combining the data sets may be more efficient involves the algorithm for building a patient group. After combining all P data sets with each other (using views), SAS/PH-Clinical software combines all P data sets (again, using views) and then joins the combined PE and P views. If users choose P variables that exist in only one data set, the product can skip the step of combining P data sets. Thus, when the medical reviewers are often likely to choose variables from DEMOG and HISTORY, the algorithm can skip a step...
Combining V Data Sets

The considerations discussed above for combining P data sets also apply to V data sets. One additional point to consider for V data sets is whether the visit identifier variable is computed or generated as a result of a lookup table (discussed in the next section). Both of these cases require SAS/PH-Clinical software to overlay a DATA step view on the V data sets. In either of these cases, joining all V data sets first and then applying the views to define the visit identifier variable is preferable. This reduces the number of data sets that require view overlays.

For example, suppose the VITALS and LAB data sets contain date and time variables that must be concatenated to create the VISITID variable, but the EFFICACY data set already contains a VISITID variable. In this case, the most efficient approach is to create a new data set (say, RESULTS) that joins VITALS and LAB, match-merging on the patient and visit identifier variables. Then generate the VISITID variable for this new data set by concatenating the DATE and TIME variables. When building a patient group, the software overlays a DATA step view on the RESULTS data set.

In contrast, if you use the VITALS and LAB data sets, the software overlays a view on each data set before joining it with other V data sets.

This approach involves the fewest steps when creating a patient group, since it involves the fewest DATA step views and fewest joins of V data sets. If, however, the VISITID variable existed in all three data sets, you would see even greater efficiency by combining all three data sets. The next section discusses issues related to the visit identifier variable.

The following is a summary of the points in this section:

- Combining P data sets eliminates a step from the process of creating a patient group.
- Consider what variables users are likely to choose together when joining P data sets.
- Combining V data sets eliminates a step from the process of creating a patient group.
- Consider what variables users are likely to choose together when joining V data sets.

- When the visit identifier variable is computed or generated from a lookup table, joining the V data sets increases efficiency.

Specifying the Visit Identifier

In the last step of defining a study to SAS/PH-Clinical software, you specify a visit identifier variable. To see optimum performance when creating a patient group, you should specify a variable contained within the data sets instead of a variable obtained through a series of views. In addition, this should be a single variable instead of a computed variable. These two situations are discussed here.

Visit Identifier in the Data Sets

Your site may store visit identifier variables in a master data set and then obtain the specific visit information through a series of views. In essence, your site may use a master lookup table for visits. For example, suppose the relative day for the study is collected in the RELDAY variable in all V and VE data sets. Then, the MASTER data set contains the mapping from RELDAY to the VISITNO variable. If you want to use VISITNO as the visit identifier, you need to construct a DATA step view that maps the RELDAY to VISITNO and then use this view in the study definition. Since using this view (or series of views) requires that the views be built and used when the patient group is created, the process of creating a patient group involves more steps (and thus, more time and more memory). When the visit identifier variable exists in the data sets themselves, the process involves fewer steps.

Visit Identifier Not Computed

In some cases, the visit identifier variable may exist in the data sets instead of in a lookup table as described above. However, the visit identifier variable may not be one unique variable but a combination of variables. For example, the visit identifier may be the concatenation of date and time, where the date and time are stored in two separate variables. This might occur in a Phase I study or other study of short duration. In this situation, you can define a new variable and then assign this new variable as the visit identifier variable. However, using the computed variable as the visit identifier variable requires that at least one view be built each time users create a patient group for the study. Specifically, views are built for each V or VE data set required for the patient group. Once again, this involves more steps and can thus involve more time and more memory.

To summarize the points in this section: for efficiency, use a visit identifier variable that exists in the data sets.
Study Definition Topics

This section discusses issues that affect the performance of SAS/PH-Clinical software when defining a study. Some of these topics may also affect the performance when creating a patient group, in that they may result in fewer steps. Finally, some of these topics address the fact that your time is valuable, and approaches that facilitate using the FASTDUMP and FASTLOAD utilities reduce the amount of time required to define a study.

Assigning Variable Names

When defining a study, you examine the Source field in the Admin: Study Variable List window to determine if variables are unique. For variables where the Source field reads "Datasets," you may find that variables with the same name have different meanings. In this situation, you need to rename the variables. If the original clinical data sets contain unique names across the entire study, renaming won't be necessary. For example, consider the situation where the DATE variable identifies the visit date in some data sets, identifies the date of onset of an adverse experience in another data set, and identifies the start date of concomitant medication use in another data set. In this case, the DATE variable has three distinct meanings, and you need to rename variables to generate, for example, VISDATE, AEDATE, and CONMDATE. If the DATE variables were distinct in the original data sets, you wouldn't need to perform the renaming.

The renaming process can add a step to the process of creating a patient group. If users choose all variables, or if they choose a renamed variable, the Patient Group component must create at least one view before beginning the process of creating a patient group. In the example above, if users choose all variables, the product accesses a view (in the Resource Manager) for all data sets that originally contained the DATE variable. In the view, the DATE variable is renamed as appropriate. If users choose AEDATE as one of the variables of interest (and not the other variables renamed from the original DATE variables), then the product accesses a view for the one data set that contains the variable renamed from DATE to AEDATE. In both situations, accessing the view adds steps (and processing time) to the process of creating a patient group.

In the original clinical data sets, assigning variable names that will be meaningful to users can shorten the time required for you to define a study. Since part of the process of defining variables to the study involves deciding if the labels are meaningful, you can save time in study definition if you already know that the labels are meaningful. Note that this has no effect on the process of creating a patient group. When patient groups are created, the labels are retrieved from the Resource Manager. This is true regardless of whether or not you have changed the labels from those in the original clinical data sets.

Standardizing Variable Names

If your site standardizes variable names across studies, you can use the FASTDUMP and FASTLOAD utilities more efficiently. For example, suppose the studies for a given drug collect similar information. If you have defined one study for the drug, and subsequent studies use the same variable names, you can use the FASTDUMP utility to dump the first study to a file. Then, you can simply modify the librefs and possibly the data set names in the dumped file. Next, you can use the FASTLOAD utility to define the second study for the drug. Contrast this with the situation where the variable names differ substantially across studies. In this situation, you need to perform all the steps described above and also need to change information on variables.

In general, the FASTDUMP and FASTLOAD utilities are most efficiently used when your studies are similar. While you can edit the flat file produced by the FASTDUMP utility, you need to weigh the potential time required to significantly edit the file against the time required to generate a second study using the full-screen process. The ideal situation occurs when the only aspect of the flat file that you need to change is the librefs. The next case occurs when you also need to change the data set names. When you need to change many of the variables, you may find it more efficient to use the full-screen process to define the study.

Deleting Variables

One of the steps in defining a study is to provide information on the variables for the study. In this step, you can update the variable names and labels, rename variables, or delete variables. Optimizing issues for changing variable names and labels and renaming are discussed above. When deleting variables from the study definition, the Resource Manager builds a view of the original clinical data sets, dropping the variables you delete.

If you need to delete only a few variables, you can do so quickly and move to the next step of defining a study. However, if you need to delete many variables, it may be a more efficient use of your time to delete the variables from the original clinical data sets.
Since the Resource Manager overlays views on each data set with deleted variables, the Patient Group component must use these views in two ways. First, if users choose all variables for a study or choose subsetting that involves variables in a data set from which you have deleted other variables, the Patient Group component must build the views before starting to build the patient group itself. Second, if users choose only some of the variables in the study, the Patient Group component accesses the study variable list, and then, if users choose variables from data sets with deleted variables, builds the appropriate views. The main point to consider is that deleting the variables in a view (instead of building a new data set that contains only the necessary variables) may add processing steps and processing time to the process of creating a patient group.

Equally important, if you have standardized variable names and want to delete the same standardized variables from various studies, you can most efficiently use the FASTDUMP and FASTLOAD utilities by deleting the variables for data sets in one study, dumping the definition, and using the dump file for other studies.

**Using Date Variables**

If most users at your site use full dates instead of partial dates (such as only the month and year), and if your site stores both the full-date variable and components of the date, you may want to delete the component variables from the study. For example, suppose your site stores DATE, DAY, MONTH, and YEAR for the visit dates. You may want to delete the DAY, MONTH, and YEAR variables from the study (either by deleting them from the original data sets or from only the study definition; see above). If your site has a number of date variables, this will reduce the number of renames you need to do when defining a study. In addition, it reduces the number of variables users see. For example, suppose a study includes the DATEVIS, DATEDAE, DATEDCONM, DATEDLAB, and DATEHOSP variables, each with corresponding variables for the day, month, and year of the visit. By deleting the components of the date, you can delete 15 variables from the list that users see.

Suppose you want to delete the component variables for day, month, and year but some users require the ability to separate out the date components. You can delete the component variables and still give these users access to the components. Include these users in the VAREDIT security group. Then, these users can choose the Enter SAS Statements Directly choice in the Variable Creation window, which appears when users choose the Redefine data choice in the Other tasks window. The users in the VAREDIT security group can then use SAS functions to obtain the components of the dates. For example, to obtain only the month and year for a date variable, users can create a new variable, called MONYR, that uses the MONTH and YEAR functions and concatenates the returned values from the functions.

If the date variable has missing values because some of the components are missing, then you may still want to keep the components. In this situation, users will not be able to use the SAS functions as described above, since the date value itself is missing.

**Summary**

This paper has discussed aspects of clinical data that can affect performance of SAS/PH-Clinical software. For optimum performance when creating a patient group, techniques include the following:

- minimizing the use of Visit Event (VE) data sets
- minimizing the use of Patient Event (PE) data sets
- for Visit (V) and VE data sets, keeping the visit variable within the data sets instead of in a master data set
- combining Patient (P) and V information into as few data sets as possible.

As a Product Administrator, you can see performance improvements in the study definition process. Although these techniques aren't directly seen by users, they may reduce the time needed for the full-screen process for defining studies or make it easier for you to use the FASTLOAD utility. For optimizing performance in study definition, techniques include:

- assigning unique names to every variable in the study
- assigning meaningful labels to every variable in the study
- standardizing as many variable names as possible across studies
- eliminating variables that are not required
- keeping full date variables, not the components of the date.

SAS and SAS/PH-Clinical are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. * indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.