Chapter 1 DBF and DIF Procedures

DBF Procedure

The DBF procedure translates .DBF files to and from SAS data sets that are compatible with the current release of the SAS System. PROC DBF produces one output file but no printed output. The output file contains the same information as the input file.

The DBF procedure works with DBF files created by all the current versions and releases of dBASE (II, III, III PLUS, IV, and 5.0) and with DBF files that are created by other software products.

Future versions of dBASE files might not be compatible with the current version of the DBF procedure. SAS Institute cannot be responsible for upgrading PROC DBF to support these changes to dBASE software. However, attempts will be made to support new versions of dBASE files with each new version of the SAS System. To use the DBF procedure, you must have a SAS/ACCESS to PC File Formats license.

Specifications

PROC DBF *options*;

The DBF statement is the only statement associated with the DBF procedure.

PROC DBF Statement

PROC DBF *options*;

The following options can appear in the PROC DBF statement. You must specify a .DBF file with the DB*n*= option, where *n* is 2, 3, 4, or 5. However, you can specify only one of these options.

DB2|DB3|DB4|DB5=*fileref*

specifies the fileref of a .DBF file. Unless a FILENAME statement is specified before the procedure statement, the second level of the referenced .DBF file must be DBF, and the file must be located in your current directory. The DB*n*= option that you specify should correspond to the version of dBASE with which the .DBF file is compatible. You must specify a .DBF file with the DB*n*= option, where *n* is 2, 3, 4, or 5. You can specify only one of these options.

DATA=<*libref.*>*member*

names the input SAS data set. Use this option if you are creating a .DBF file from a SAS data set. If you use the DATA= option, do not use the OUT= option. If you omit the DATA= option, the SAS System creates an output SAS data set from the .DBF file.

OUT=<*libref.*>*member*

names the SAS data set created to hold the converted data. Use this option only if you do not specify the DATA= option. If OUT= is omitted, the SAS System creates a temporary data set named _DATA_ in the WORK library. If OUT= is omitted or if you do not specify a two-level name in the OUT= option, the SAS data set created by PROC DBF is not permanently saved.

Converting .DBF Fields to SAS Variables

Numeric fields are stored in character form by .DBF files. These numeric fields become SAS numeric variables when converting from a .DBF file to a SAS data set. Character fields become SAS character variables. Any character field of a length greater than 200 is truncated to 200. Logical fields become SAS character variables with a length of 1. Date fields become SAS date variables. When converting a .DBF file to a SAS data set, fields whose data are stored in auxiliary .DBF files (Memo and General fields) are ignored.

When a dBASE II compatible file is translated into a SAS data set, any colons in dBASE field names are changed to underscores in SAS variable names. Conversely, when a SAS data set is translated into a dBASE II .DBF file, any underscores in SAS variable names are changed to colons in dBASE field names.

Converting SAS Variables to .DBF Fields

Numeric fields are stored in character form in .DBF files. SAS numeric variables become numeric fields with a length of 16 when converting from a SAS data set to a .DBF file. A SAS numeric variable with a decimal value must be stored with a decimal format in order to be converted to a .DBF numeric variable with a decimal value. In other words, unless you associate the SAS numeric variable with an appropriate format in a SAS FORMAT statement, the corresponding .DBF field will not have any value to the right of the decimal point. Several ways you can have a format associated with the variable in the data set are: in the DATA step when the data set is created, using the DATASETS procedure, and from the

Contents window of the ACCESS command.

If the number of digits—including a possible decimal point—exceeds 16, a warning message is issued and the dBASE numeric field is filled with the character ’9’. All SAS character variables become .DBF fields of the same length. When converting from a SAS data set to a .DBF file compatible with dBASE III or later, SAS date variables become .DBF date fields. When converting from a SAS data set to a dBASE II file, SAS date variables become dBASE II character variables of the form YYYYMMDD.

Missing Values

.DBF files do not have a designated missing value. If a SAS numeric value is missing, the corresponding dBASE numeric field is filled with the character ’9’.

Examples

Example 1: Conversion of a dBASE II File to a SAS Data Set

In this example, a dBASE II file named EMPLOYEE.DBF is converted to a SAS data set. Since no FILENAME statement is specified, the last level of the filename is assumed to be DBF, and the file is assumed to be in your current directory.

```
proc dbf db2=employee out=save.employee;
run;
```

Example 2: Conversion of a SAS Data Set to a dBASE 5 File

In this example, a SAS data set is converted to a dBASE 5 file. A FILENAME statement is used to specify a fileref that names the dBASE 5 file. The FILENAME statement must come before the PROC DBF statement.

```
libname mylib ’\my\directory’;
filename employee ’c:\sasdemo\employee.dbf’;
proc dbf db5=employee data=mylib.employee;
run;
```

DIF Procedure

The DIF procedure converts Data Interchange Format (DIF) files to SAS data sets that are compatible with the current release of the SAS System. This procedure can also be used to convert SAS data sets to DIF files. Software Arts, Inc., developed the Data Interchange Format to be used as a common language for data. Originally, DIF was made popular by products such as Lotus 1-2-3 and VisiCalc. Although DIF is not as popular today as it once was, it is still supported by many software products.

PROC DIF produces one output file but no printed output. The output file contains the same information as the input file.

It is important to remember that any DIF file that you plan to convert to a SAS data set should be in a tabular form; that is, all items in a given column should represent the same type of data. When converting from a DIF file to a SAS data set, each row of the DIF file becomes an observation in the SAS data set. Conversely, when converting a SAS data set to a DIF file, each SAS observation becomes a row in the DIF file. To use the DIF procedure, you must have a SAS/ACCESS to PC File Formats license.

Specifications

The DIF statement is the only statement associated with the DIF procedure.

PROC DIF *options*;

PROC DIF Statement

PROC DIF DIF=*fileref options*;

where DIF= specifies the fileref of a DIF file. Unless a FILENAME statement was specified, the second level of the referenced DIF file must be DIF, and the file must be located in your current directory.

Following are the options:

DATA=<*libref.*>*member*
: names the input SAS data set. Use this option if you are creating a DIF file from a SAS data set. If you use this option, omit the OUT= option. If you omit the DATA= option, the SAS System creates an output SAS data set from the DIF file.

OUT=<*libref.*>*member*
: names the SAS data set that holds the converted data. Use this option only if you omit the DATA= option. If OUT= is omitted, the SAS System creates a temporary data set named _DATA_ in the WORK library. If OUT= is omitted, or if you do not specify a two-level name in the OUT= option, the SAS data set created by PROC DIF is not permanently saved.

LABELS
: causes PROC DIF to write the names of the SAS variables as the first row of the DIF file and a row of blanks as the second row of the DIF file. The actual data portion of the DIF file begins in the third row. The LABELS option is allowed only when converting a SAS data set to a DIF file.

PREFIX=*name*
: specifies a prefix to be used in constructing SAS variable names when you convert a DIF file to a SAS data set. For example, if PREFIX=VAR, the new variable names are VAR1, VAR2, . . . ,VAR*n*.

  If you omit the PREFIX= option, PROC DIF assigns the names COL1, COL2, . . . , COL*n* to the variables in the output SAS data set.

SKIP=*n*
: specifies the number of rows, beginning at the top of the DIF file, to be ignored when you convert a DIF file to a SAS data set. For example, suppose the first row of your DIF file contains column headings and the second row of your DIF file is a blank row. The actual data in your DIF file begin in row 3. You specify SKIP=2 so that PROC DIF

ignores the nondata portion of your DIF file. Alternatively, you could delete the first two rows of your DIF file before using PROC DIF.

Converting DIF Variables to SAS Variables

Character variables in a DIF file (sometimes referred to as *string values*) become SAS character variables of length 20. If a DIF character variable's value is longer than 20 characters, it is truncated to a length of 20 in the SAS output data set. The quotation marks that normally enclose character variable values in a DIF file are removed when the value is converted to a SAS character value.

Numeric variables, which can be represented in either integer or scientific notation in a DIF file, become SAS numeric variables when a DIF file is converted to a SAS data set.

Transferring SAS Data Sets to and from Other Software Products Using DIF

DIF files are not generally used as the native file format for a software product's data storage. Therefore, transferring data between the SAS System and another software product is a two-step process when using DIF files.

To send SAS data sets to another software product using DIF files, you must first run PROC DIF to convert your SAS data set to a DIF file. Then, use whatever facility is provided by the target software product to read the DIF file. (This may be provided by an import tool or from a File>Open dialog box.) After the DIF file data have been read by the application, they can be manipulated and saved in the application's native format.

To transfer data in the opposite direction (for example, from a software product to a SAS data set), the process is reversed. First, export the data to a DIF file, and then run PROC DIF to read the DIF file into a SAS data set.

Caution

It is important to remember that any DIF file that you plan to convert to a SAS data set should be in a tabular form; that is, all items in a given column should represent the same type of data. When converting from a DIF file to a SAS data set, each row of the DIF file becomes an observation in the SAS data set. If there are any rows in your DIF file that contain inconsistent data, for example, a row of underscores, dashes, or blanks, delete these rows before converting the DIF file to a SAS data set. If your DIF file contains title or heading lines, you can either delete these rows or use the SKIP= option to make PROC DIF ignore them.

Missing Values

The developers of the Data Interchange Format (DIF) suggest that you treat all numeric values that have a value indicator other than V as missing values. PROC DIF follows this convention. When a DIF file is converted to a SAS data set, any numeric value with a value indicator other than V becomes a SAS missing value.

When a SAS data set that has missing values for some numeric variables is converted to a DIF file, the following assignments are made in the DIF file for the variables with missing values:

- the type indicator field value is set to **`0`**
- the number field value contains a string of 16 blanks
- the value indicator is set to **`NA`**.

Examples

Example 1: Conversion of a DIF File to a SAS Data Set

In this example, a DIF file named EMPLOYEE.DIF is converted to a SAS data set. Since no FILENAME statement is specified, the last level of the filename is assumed to be DIF, and the file is assumed to be in your current directory.

```
proc dif dif=employee out=save.employee;
run;
```

Example 2: Conversion of a SAS Data Set to a DIF File

In this example, a SAS data set is converted to a DIF file. A FILENAME statement is used to specify a fileref that names the DIF file. The FILENAME statement must come before the PROC DIF statement.

```
filename employee 'c:\sasdemo\employee.dif';
proc dif dif=employee data=save.employee;
run;
```

Reference

"Programmer's Guide to the DIF," *Software Arts Technical Notes* (SATN-18).