

## 8. Blocking

### 8.1 Introduction

All VPLX files organize the data into blocks. The CREATE step examples in Chapters 3-7 each create a VPLX file with a single block, and only Chapter 7 makes any more than passing reference to blocks. Examples in Chapter 2 hint at the existence of blocks, however, particularly as the TRANSFORM step adds new variables to a new block that it creates.

Many VPLX applications are possible by 1) creating a single block in the CREATE step, which occurs automatically when the CREATE step specification does not include any BLOCK statements, and 2) allowing the TRANSFORM step automatically to place new variables into blocks. Many VPLX users only use blocking implicitly in this manner.

This chapter serves two purposes:

- to introduce the notion of blocking, which is critical to an understanding of several aspects of the TRANSFORM step; and
- to describe how to take advantage of blocking in the CREATE step.

This chapter assumes only a reading of Chapters 3, 4, and 7. Subsequent chapters on the CREATE step may be understood without a reading of this chapter, although Chapters 10 and 11 mention blocking. Chapter 14 and later chapters on the TRANSFORM step presume a reading of at least Section 8.2.

Version 94.11 implements a new syntax to specify blocks in the CREATE step while continuing to support earlier versions. Experience had made clear that the previous version of the syntax was confusing. This chapter will describe only the new syntax.

Summary of this chapter:

- Section 8.2 discusses the general nature of blocking in both the CREATE and TRANSFORM steps.
- Section 8.3 describes the BLOCK statement in the CREATE step.
- Section 8.4 provides an example of blocking in the CREATE step.

**8.2.1 A General Definition of Blocks.** A block consists of one or more of the following: 1) a weighted count, 2) a set of non-class variables, 3) a set of shared class variables. Blocks have two distinguishing features:

1. All variables in the block, including the weighted count, if present, are cross-classified by the associated class variables, if any.
2. All variables in the block share the same weighted count. For example, this weighted count will be used as the denominator in deriving means for any real variables associated with the block. The CREATE step produces a weighted count for each block, but the TRANSFORM step can form new blocks with or without weighted counts.

Blocks created by the CREATE step have the following features:

- Every block includes an associated weighted count, which is produced automatically.
- A block may have one or more non-class variables, although blocks with simply weighted counts are permitted and sometimes useful.
- The associated class variables, if any, cross-classify the weighted count and non-class variables completely. The VPLX file does not include any margins of the cross-classification, however. Instead, margins are automatically constructed within other steps of the system, including the DISPLAY and TRANSFORM steps.

Chapter 3 described the SELECT statement in the CREATE step. In its global form, that is, in the absence of BLOCK statements, SELECT affects which observations are included. Chapter 7 illustrated CLASS statement in a global form, where each class variable automatically cross-classified the other variables. In a CREATE step without BLOCK statements, VPLX treats all appearances of SELECT and CLASS globally, creating a single block.

The BLOCK statement makes it possible to restrict SELECT and CLASS statements to operate only on subsets of variables. The advantages are evident in large applications. In some applications, using SELECT with blocking is more efficient than employing MISSING if the intention is to restrict the universes of different groups of variables to a few specific domains. In small applications, using CLASS statements globally is effective, but blocking may significantly reduce the size of the outgoing VPLX file if the complete cross-classification by all CLASS variables is not needed.

Blocks created by the TRANSFORM step have the following features:

- New blocks may be created with or without an associated weighted count.
- A block may have one or more non-class variables, although blocks with simply weighted counts are permitted and sometimes useful.
- The associated class variables, if any, cross-classify the weighted count and non-class variables completely. Like the CREATE step, the TRANSFORM step can create blocks without stored marginal totals, for weighted counts and variables types real, real with missing, categorical, etc., whose marginal totals may be constructed by summation. For other types of variables, such as derived, the VPLX file may separately store marginal values. The decision about whether a marginal value is stored may be made for each class variable separately.

### 8.3 The BLOCK Statement

**8.3.1 General Form.** The recommended use of the new syntax is to place all block definitions at the end of the CREATE step. The rest of the CREATE step, leading up to the BLOCK statement, should follow these rules:

- DROP and KEEP statements must not be used. (Their function is replaced by KEEP specifications associated with each BLOCK statement. DROP is not allowed.)
- SELECT statements must not be used. (Their function is replaced by / SELECT statements included within the BLOCK statement.)
- CLASS statements should be used as normal. Class variables are no longer global cross-classifiers, except at the block level for each block in which they appear in a /CLASS statement. Class variables must be included on one or more / CLASS statements to be included on the output VPLX file at all.

The syntax of the BLOCK statement is

```
BLOCK  [ [KEEP]  vlist  ]
        [ / CLASS classvar1 [ * classvar2 [ * classvar3 ...]] ]
        [ / SELECT [IF]  vname1(range1 ) ]
        [ / SELECT [IF]  vname2(range2 ) ]
```

## 8.4

where *vlist* contains a list of one or more variables, the ranges represent values to be assigned to each of the categories, and, optionally, labels of up to 24 characters for each category are provided. The "/"s in the syntax separate the different categorical levels of variable. For example, the following illustrates all aspects of the syntax:

```
block  lf_status_h earnings_h / class age * sex * state /  
      select if ethnicity (1)
```

BLOCK statements may contain just a list of variables:

```
block  rooms persons
```

a list of variables cross-classified by class:

```
block  rooms persons / class tenure
```

a list of variables with select only:

```
block  rooms persons / select if tenure (1)
```

or represent only a cross-classification of the weighted count:

```
block  class tenure * rooms_cl
```

combine this approach with select:

```
block  class tenure * rooms_cl / select if msa_status (1)
```

Some forms are allowed but less useful. For example, simply getting a count under a specific condition:

```
block  select if tenure (1)
```

is more usefully done in other ways.

**Important Note:** With blocking, any variable not assigned to a block is dropped. A variable can only be included in one block, but COPY may be used to include identical information, under different variable names, in different blocks.

## 8.4 An Example of BLOCK.

The following example is based on modest changes to exam20.crd, used in Chapter 7.

```

comment  EXAM21

comment  This example combines features of EXAM16 and EXAM18
         to illustrate the use of CLASS

create  in = exampl11.dat  out = exampl11.vpl

input   rooms persons cluster tenure

         4 variables are specified

format  (4f2.0)

comment  The input data set contains
5 7 1 2
6 8 2 2
5 2 3 1
4 1 4 2
8 4 5 1
8 2 6 1

comment  Make tenure a class variable.  Also form a class variable
         from rooms

class   tenure (2 / 1,3 ) 'Renters' 'Owners and others'

class   rooms into rooms_cl (1-5/6-high) '1-5 rooms' '6 or more rooms'

cat     rooms into rooms_cat (1-5/6-high) '1-5 rooms' '6 or more rooms'

cat     persons into persons_cat (1-3/4-high) '1-3 persons'
         '4 or more persons'

labels  rooms 'Number of rooms' persons 'Persons'
         rooms_cat 'Number of rooms' persons_cat 'Number of persons'
         tenure 'Tenure' rooms_cl 'Rooms' rooms_cat 'Rooms'

block   rooms_cat / class tenure

block   rooms persons_cat / class tenure*rooms_cl

(Simple) jackknife replication assumed

Size of block  1  =           6

Size of block  2  =          16

Total size of tally matrix =     22

Unnamed scratch file opened on unit 13

```

## 8.6

```
Unnamed scratch file opened on unit 14

**** End of CREATE specification/beginning of execution

End of primary input file after obs #      6

display

option  ndecimal=2 totals

list    n(1) rooms rooms_cat persons_cat / class total tenure /
        n(2) rooms persons_cat / class  rooms_cl tenure (1) * rooms_cl
```

Exhibit 8.1 An example illustrating blocking in the CREATE step. Blocking is used to eliminate the redundancy between ROOMS\_CL and ROOMS\_CAT. VPLX reports the sizes of the two blocks, and the total size is somewhat less than in the previous example. The output of the DISPLAY step is omitted for the sake of brevity but agrees with previous results except for omitting the detailed display of ROOMS\_CAT by ROOMS\_CL.

NOTES

---