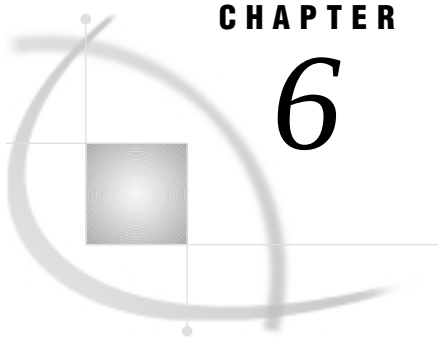


CHAPTER

6

Creating and Loading SYSTEM 2000 Databases

Introduction

The DBLOAD procedure enables you to create and load a SYSTEM 2000 database from a SAS data set. Optionally, you can create the database definition only and do one or more incremental loads later.

The DBLOAD procedure constructs SYSTEM 2000 statements to create the database definition. You can load new logical entries into the database, or you can insert new records into existing logical entries. The data can come from a SAS data file, from a view created with the SQL procedure, or from a SYSTEM 2000 database or another DBMS (using a view created with the ACCESS procedure).

The procedure associates each SAS variable in the input data with a SYSTEM 2000 item and assigns a default item name, number, and type to each item. By default, items are non-key at level 0. You can use the defaults or change the names, key/non-key status, and level numbers as necessary. When you are finished customizing the items, the DBLOAD procedure creates the SYSTEM 2000 database.

The DBLOAD procedure can run in interactive line and batch modes.

This chapter presents examples of creating and loading a database. The examples create a database named BANKING, with input data that resides in the SAS data file TRANS.BANKING, shown in “TRANS.BANKING Data File” on page 160. The data contain information regarding individual checking and savings account transactions. Also, the section “Incremental Load Example” on page 69 shows a sample incremental load for an existing database.

Refer to Chapter 8, “DBLOAD Procedure Reference,” on page 97 for reference information on the DBLOAD procedure.

SYSTEM 2000 BANKING Database Definition

The examples of the DBLOAD procedure given in this chapter create a BANKING database and load data into it. In this new database, each logical entry contains data for one customer.

The ENTRY schema record at level 0 contains two items: the customer name and customer ID. There is one schema record at level 1; it contains information about the customer's checking and savings accounts. The schema record at level 2 contains information about the transactions for each account.

After you create the BANKING database, you can use the QUEST procedure to issue a SYSTEM 2000 DESCRIBE statement, which would produce Output 6.1 on page 64, showing the database definition.

Output 6.1 BANKING Database Definition

```
SYSTEM RELEASE NUMBER  12.1
DATA BASE NAME IS      BANKING
DEFINITION NUMBER             1
DATA BASE CYCLE NUMBER       18
     1*  CUSTNAME (CHAR X(20))
     2*  CUSTID (CHAR X(7))
   100*  RECORD_LEVEL_1 (RECORD)
     101*  ACCOUNT NUMBER (INTEGER NUMBER 9999 IN 100)
     102*  ACCOUNT TYPE (CHAR X IN 100)
     200*  RECORD_LEVEL_2 (RECORD IN 100)
       201*  TRANS TYPE (CHAR X IN 200)
       202*  TRANS AMOUNT (NON-KEY MONEY $9(7).99 IN 200)
       203*  TRANS DATE (DATE IN 200)
```

Notice that the records are stair-stepped down and to the right, reflecting the record relationships. That is, record C200 is related to record C100, which is related to the level 0 record.

SYSTEM 2000 BANKING Data

The input SAS data file for the examples is shown in "TRANS.BANKING Data File" on page 160. If you want to run the examples, be sure that you sort the observations before you use the DBLOAD procedure. Sorting groups the observations by accounts for each customer, which produces data in the sequence required for loading the three-level BANKING database.

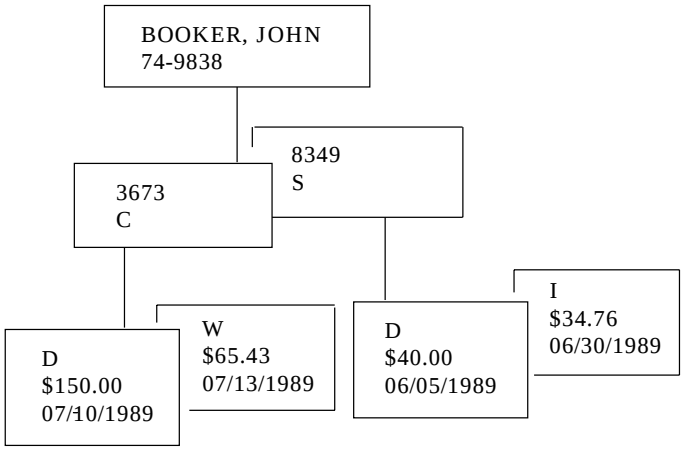
Each observation in the input data file represents one transaction. For example, John Booker has four transactions, two for each of his accounts. His name and account numbers are repeated in each observation as shown in Output 6.2 on page 65.

Output 6.2 Sample SAS TRANS.BANKING Data File

OBS	CUSTNAME	CUSTID	ACCTNUM	ACCTTYP	TRANSTYP	TRANSAMT	TRANSDAT
1	BOOKER, JOHN	74-9838	8349	S	D	$40.00	05JUN89
2	LOPEZ, PAT	38-7274	9896	S	D	$15.67	23JUN89
3	JONES, APRIL	85-4941	4141	C	W	$213.78	29JUN89
4	BOOKER, JOHN	74-9838	8349	S	I	$34.76	30JUN89
5	MILLER, NANCY	07-6163	7890	S	I	$53.98	30JUN89
6	LOPEZ, PAT	38-7274	9896	S	I	$16.43	30JUN89
7	JONES, APRIL	85-4941	4141	C	W	$354.70	30JUN89
8	MILLER, NANCY	07-6163	7890	S	D	$1,245.87	01JUL89
9	JONES, APRIL	85-4941	4141	C	D	$2,298.65	01JUL89
10	MILLER, NANCY	07-6163	3876	C	W	$45.98	08JUL89
11	ROGERS, MIKE	96-5052	4576	C	D	$75.00	10JUL89
12	BOOKER, JOHN	74-9838	3673	C	D	$150.00	10JUL89
13	LOPEZ, PAT	38-7274	9896	S	D	$50.00	10JUL89
14	BOOKER, JOHN	74-9838	3673	C	W	$65.43	13JUL89
15	ROGERS, MIKE	96-5052	4576	C	W	$12.34	13JUL89
16	ROGERS, MIKE	96-5052	4576	C	W	$45.67	13JUL89
17	MILLER, NANCY	07-6163	3876	C	D	$56.79	14JUL89
18	ROGERS, MIKE	96-5052	4576	C	W	$12.16	15JUL89

After you sort the input data file by customer name and account type, the DBLOAD procedure loads each customer as a logical entry in the SYSTEM 2000 database. Redundant data is reduced, thus saving storage space. The logical entry for John Booker would look like Figure 6.1 on page 65.

Figure 6.1 Sample Logical Entry in BANKING Database



After you load the TRANS.BANKING input data, you could run this SYSTEM 2000 LIST statement using the QUEST procedure:

```
list c1, c101, c102, c201, c202;
```

The output from this LIST statement is shown in Output 6.3 on page 66.

Output 6.3 BANKING Database Data from LIST Command

```
* CUSTNAME        ACCOUNT NUMBER   ACCOUNT TYPE   TRANS TYPE   TRANS AMOUNT
***
* BOOKER, JOHN              3673   C              D                 $150.00
*                                                 W                  $65.43
*                           8349   S              D                  $40.00
*                                                 I                  $34.76
* JONES, APRIL              4141   C              W                 $213.78
*                                                 W                 $354.70
*                                                 D               $2,298.65
* LOPEZ, PAT                9896   S              D                  $15.67
*                                                 I                  $16.43
*                                                 D                  $50.00
* MILLER, NANCY             3876   C              W                  $45.98
*                                                 D                  $56.79
*                           7890   S              I                  $53.98
*                                                 D               $1,245.87
* ROGERS, MIKE              4576   C              D                  $75.00
*                                                 W                  $12.34
*                                                 W                  $45.67
*                                                 W                  $12.16
```

Notice, for example, the data values for John Booker - his name appears only once here, but he has two account numbers and four transactions. Since the examples of the DBLOAD procedure given in this chapter rank the data values into levels, you have a clear, logical view of the data.

Effect of the Version 6 Compatibility Engine

You can use both Version 6 and Version 7 data files to create and load SYSTEM 2000 databases with the DBLOAD procedure. However, for Version 7 data files, any variables whose names are longer than 8 characters will have their names truncated to 8 characters in the access and view descriptors created by the DBLOAD procedure. The DBLOAD RENAME statement can be used to rename the variables in the SYSTEM 2000 database, but it will not change the variable names in the access and view descriptors. The shortened names must be used to access the data described by the view descriptors.

Loading the BANKING Database

You create and load a SYSTEM 2000 database by using the DBLOAD procedure with options and statements that describe the SYSTEM 2000 database you want to create and the data you want to load into the database. To load the BANKING database with the DBLOAD procedure, use the following options and statements:

```
JCL statements;
```

```
proc dbload dbms=s2k data=trans.banking;
   s2kpw=mine;
   dbn=banking;
   accdesc=mylib.bank;
   viewdesc=vlib.myview;
   s2kmode=m;
   rename acctnum='account number' 4= 'account type'
         5='trans type' 6='trans amount'
         7='trans date';
   index 1=y 2=y 3=y 4=y transtyp=y 7=y;
   level 3=1 4=1 5=2 6=2 transdat=2;
   list all;
 load;
 run;
```

JCL statements;
submit your statements for execution under the SAS System.

proc dbload dbms=s2k data=trans.banking;
invokes the DBLOAD procedure. The DBMS= option specifies the DBMS into which you want to load data. The DATA= option specifies the SAS data file where the data reside.

s2kpw=mine;
issues the password that will become the master password for the new database.

dbn=banking;
identifies the database you want to create with the DBN= statement, in this case, a SYSTEM 2000 database named BANKING.

accdesc=mylib.bank;
specifies the access descriptor libref and member name. The access descriptor is created automatically by the DBLOAD procedure. In this example, the name specified is MYLIB.BANK. The default access descriptor name would be WORK.BANKING.ACCESS, where BANKING is the name of the database to be created. The access descriptor member name must not already exist when you are creating a new database.

viewdesc=vlib.myview;
specifies the view descriptor libref and member name for the new database. The view descriptor is created automatically by the DBLOAD procedure. In this example, the specified name is VLIB.MYVIEW. The default view descriptor name would be WORK.BANKING.VIEW, where BANKING is the name of the database to be created. The view descriptor member name must not already exist when you are creating a new database.

s2kmode=m;
creates the new database in a Multi-User environment. The default (S) is a single-user job. For a Multi-User session, the new database files can be allocated when the session is initialized or dynamically allocated during execution using the ALLOC command (Release 12.0 and later). For a single-user job, you must allocate the database files in the JCL for the job.

rename acctnum='account number' 4= 'account type' 5='trans type' 6='trans amount' 7='trans date';
changes the names of the last five items with the RENAME statement. You always use a SAS variable on the left-hand side of the equal sign. You can use either the SAS variable name or its positional equivalent as shown in the LIST statement, and you can rename as many items as you want in one RENAME statement.

index 1=y 2=y 3=y 4=y transtyp=y 7=y;
defines items as key (indexed) with the INDEX statement. This statement makes key items of all items except TRANS AMOUNT. TRANS AMOUNT is not listed, so it defaults to non-key. You always use a SAS variable on the left-hand side of the equals sign. You can use either the SAS variable name or its positional equivalent as shown in the LIST output, and you can index as many items as you want in one INDEX statement.

level 3=1 4=1 5=2 6=2 transdat=2;
changes the level number of an item with the LEVEL statement. In this example, ACCOUNT NUMBER and ACCOUNT TYPE become items in a level 1 record; TRANS TYPE, TRANS AMOUNT, and TRANS DATE become items in a level 2 record. The DBLOAD procedure automatically defines the schema record names RECORD_LEVEL_1 and RECORD_LEVEL_2, respectively, and assigns appropriate component numbers.

You always use a SAS variable on the left-hand side of the equals sign. You can use either the SAS variable name or its positional equivalent as shown in the LIST output, and you can change the level number for as many items as you want in one LEVEL statement.

list all;
lists the items, levels, and index settings with the LIST statement.

load; run;
executes the DBLOAD procedure and creates and loads the database.

The output of the LIST statement is shown in Output 6.4 on page 68.

Output 6.4 LIST Statement Output

```
Command ===>

PROC DBLOAD for SYSTEM 2000 - OPTIONS FOLLOW:
   Input data set=        TRANS    BANKING  DATA
   View descriptor=       VLIB     MYVIEW   VIEW
   Access descriptor=     MYLIB    BANK     ACCESS
   Database name=         BANKING
   S2KMODE=               M
   Label option=          N
   Create option=         N
   S2KLOAD=               N
------------SAS NAME---LEVEL---INDEX---COMPONENT NAME----
  1          CUSTNAME            YES     CUSTNAME
  2          CUSTID              YES     CUSTID
  3          ACCTNUM    1        YES     ACCOUNT NUMBER
  4          ACCTTYP    1        YES     ACCOUNT TYPE
  5          TRANSTYP   2        YES     TRANS TYPE
  6          TRANSAMT   2                 TRANS AMOUNT
  7          TRANSDAT   2        YES     TRANS DATE
```

Subsetting Your Input Data

To subset your input data, use the SAS WHERE statement. Creating a subset of the input data is useful if you need to transfer only a portion of your SAS data to a

SYSTEM 2000 database. For example, you might want to include only observations in which the value in a variable is greater than a specified number. Here is an example using the SAS WHERE statement:

```
proc dbload dbms=s2k data=trans.banking;
  s2kpw=mine;
  dbn=banking;
  s2kmode=m;
  where acctnum > 4141;
load;
run;
```

This example subsets the input data to include only those observations in which the SAS variable ACCTNUM has a value greater than 4141. Here, none of the items are renamed or indexed, and they are all at level 0.

Note that you use the SAS variable name in the WHERE statement, not the SYSTEM 2000 item name. For information on the syntax of the SAS WHERE statement, see *SAS Language Reference: Dictionary*.

Incremental Load Example

This section illustrates how to load more logical entries into an existing SYSTEM 2000 database. You invoke the DBLOAD procedure and specify the input data file and the appropriate view descriptor. The view descriptor contains the database name, the component names, levels, and so on. It also contains the password for the database and the access mode (single user or Multi-User). You can use a SAS WHERE clause to limit the input. (If the view descriptor includes a SYSTEM 2000 where-clause, it does not affect an incremental load.)

To perform the incremental load with the DBLOAD procedure, use the following options and statements. In this example, the data file is TRANS.INCLOAD.

```
JCL statements;

proc dbload dbms=s2k data=trans.incload;
  viewdesc=vlib.myview;
load;
run;
```

proc dbload dbms=s2k data=trans.incload;
invokes the DBLOAD procedure. The DBMS= option specifies the DBMS into which you want to load data. The DATA= option specifies the SAS data file where the data reside.

viewdesc=vlib.myview;
specifies the appropriate view descriptor with the VIEWDESC= statement.

load; run;
executes the DBLOAD procedure and loads the database.

Adding New Logical Entries Versus Updating Existing Logical Entries

Incremental loads can add either new logical entries or new records to existing records. Both types of incremental loading are performed the same way, as shown in the above example. How then does the SYSTEM 2000 engine know whether to insert new logical entries or append records to existing logical entries?

One simple way for the engine to tell the difference is whether you have issued the S2KLOAD statement. If so, the input observations are treated as new logical entries. Several observations may be collected to form each logical entry, but they are all new. Your observations must be sorted in order to achieve the correct result.

If you did not issue the S2KLOAD statement, your results depend on the order of your observations and whether your view descriptor contains a BY key. A BY key identifies the placement of inserted data records in an incremental load. See “Using a BY Key to Resolve Ambiguous Inserts” on page 130. When using a BY key, it is best (less ambiguous) if your view descriptor and the BY key begin at level 0, even if you are loading records only at some lower level.

Selecting the Processing Mode for the Load

Two modes of processing are available when you are loading data with the DBLOAD procedure: insert mode and optimized load mode. Optimized load mode is a fast, efficient way to add new logical entries to the database. Insert mode must be used to add data records to existing logical entries. For details about these processing modes, see the S2KLOAD statement discussion in “Procedure Statements” on page 99.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software: Reference, Version 8

ISBN 1–58025–549–3

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999