CHAPTER

2

SYSTEM 2000 Essentials

Introduction

This chapter introduces SYSTEM 2000 data management software, SAS Institute's hierarchical database software for OS/390, to SAS System users. If you are an experienced SYSTEM 2000 user, you can skip this chapter. This chapter focuses on terms and basic concepts that will help you use the SAS/ACCESS interface to SYSTEM 2000 software, including

- SYSTEM 2000 software and SYSTEM 2000 databases
- SYSTEM 2000 item types and indexing
- selecting specific values and ordering data
- SYSTEM 2000 passwords, execution environments, and database files.

For more information on SYSTEM 2000 terms and concepts, see Appendix 1, "Information for the Database Administrator," on page 119 and Appendix 2, "Advanced User Topics," on page 127.

SYSTEM 2000 Software

SYSTEM 2000 software is a hierarchical data management system for mainframe computer systems running under the OS/390 operating system. With SYSTEM 2000 software, you define the types of data to be stored in the database and the relationships between the data. In addition, you can load the database, and you can retrieve and update the data.

The software's hierarchical database structure provides efficient data storage by reducing the amount of redundant data. In addition, you have indexing capabilities for fast and efficient retrievals, data qualification and sorting capabilities, complete data security through passwords, both Multi-User and single-user execution environments, a complete set of diagnostic messages, optional transaction journaling, and rollback recovery from system crashes or other failures.

SYSTEM 2000 Databases

A SYSTEM 2000 database is hierarchical in that you store and access data according to organized relationships between groups of associated data. When a SYSTEM 2000 database is created, a plan is devised, called a database definition. In a database definition

- the database has an assigned name
- the data to be stored are labeled
- the data are arranged into groups
- relationships are established between the groups of data.

Typically, a database is organized according to the types of data and how you want to use the data. You must understand and be familiar with your database's organization in order to retrieve and update information accurately and efficiently. And you must be familiar with the organization and contents of the database to create descriptor files for the SAS/ACCESS interface. Output 2.1 on page 9 shows an excerpt of the EMPLOYEE database definition, which is the output of a SYSTEM 2000 DESCRIBE statement. (For a complete listing of the EMPLOYEE database definition, see Appendix 3, "Example Data," on page 143).

Output 2.1 Database Definition

```
  SYSTEM RELEASE NUMBER  12.1
  DATA BASE NAME IS      EMPLOYEE
  DEFINITION NUMBER             2
  DATA BASE CYCLE NUMBER       25
       1*  EMPLOYEE NUMBER (INTEGER NUMBER 9999)
       2*  LAST NAME (CHAR X(10) WITH  FEW FUTURE OCCURRENCES )
       3*  FORENAME (NON-KEY CHAR X(20))
.
.
.
      16*  ZIP CODE (CHAR X(5) WITH  FEW FUTURE OCCURRENCES )
     100*  POSITION WITHIN COMPANY (RECORD)
       101*  POSITION TITLE (NON-KEY CHAR X(10) IN 100)
       102*  DEPARTMENT (CHAR X(14) IN 100 WITH SOME FUTURE OCCURRENCES )
       103*  MANAGER (CHAR XXX IN 100 WITH  FEW FUTURE OCCURRENCES )
       104*  POSITION TYPE (CHAR X(12) IN 100 WITH SOME FUTURE OCCURRENCES )
       105*  START DATE (DATE IN 100)
       106*  END DATE (NON-KEY DATE IN 100)
       110*  SALARY WITHIN POSITION (RECORD IN 100)
         111*  PAY RATE (MONEY $9999.99 IN 110)
         112*  PAY SCHEDULE (CHAR X(7) IN 110)
         113*  EFFECTIVE DATE (DATE IN 110)
         114*  CURRENT DEDUCTION (NON-KEY MONEY $9999.99 IN 110)
.
.
.
     400*  EDUCATIONAL BACKGROUND (RECORD)
       410*  EDUCATION (RECORD IN 400)
         411*  SCHOOL (CHAR X(15) IN 410)
         412*  DEGREE/CERTIFICATE (CHAR X(7) IN 410 WITH  FEW FUTURE OCCUR
               RENCES )
         413*  DATE COMPLETED (DATE IN 410)
         414*  MAJOR FIELD (NON-KEY CHAR X(16) IN 410)
         415*  MINOR FIELD (NON-KEY CHAR X(12) IN 410)
```

Database Name

The database name is a unique name, from one to 16 characters long, that is assigned to a specific SYSTEM 2000 database definition. Each database also has one or more passwords associated with it.

To create descriptor files for the SAS/ACCESS interface, you must know the name and a password for the SYSTEM 2000 database you want to access.

Labeling Data

A database definition consists of schema records and schema items, which describe a blueprint for the type of data to be stored. For example, in the preceding definition, EMPLOYEE NUMBER is a schema item and POSITION WITHIN COMPANY is a schema record.

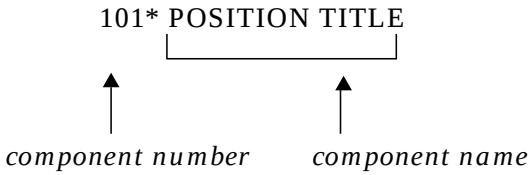
A schema item names and defines the characteristics of a group of values; that is, it has a name, a type, and a picture (length). Each value stored in a SYSTEM 2000

database corresponds to a schema item. For example, the following schema item describes the numbers used as employee identification numbers. The four 9s indicate that each employee number can contain up to four digits.

```
1* EMPLOYEE NUMBER (INTEGER NUMBER 9999)
```

A schema record groups associated schema items; it is explained in the next section.

Schema items and records are referred to as schema components, and each is identified by both a component number and a component name, as shown below. (A component number can also be referred to as a C-number, for example, C101.)



To access data stored in a SYSTEM 2000 database, you must refer to either the component number or the component name. Both are unique in the database definition to avoid ambiguity. Each line in a database definition begins with the component number and the component name.

When you create descriptor files for SYSTEM 2000 databases, the ACCESS procedure creates corresponding SAS variable names from the SYSTEM 2000 item names. You can then use the variable names in SAS procedures.

Grouping Data

In a SYSTEM 2000 database, associated schema items are grouped by schema record. That is, different schema records store different groups of data, and a schema item belongs to only one schema record. Grouping associated schema items into schema records is similar to planning a form; a form is usually divided into sections with one section for each set of related data.

For example, looking at the EMPLOYEE database definition in Output 2.1 on page 9, schema items C1 through C16 pertain to personal information on each employee. These items are grouped in one record, the ENTRY record or C0 record. (The component number and name for the ENTRY record is not listed in a definition unless it has been renamed.) The schema items for information about an employee's position (C101 through C106) are grouped in schema record C100, POSITION WITHIN COMPANY.

Establishing Relationships between Schema Records

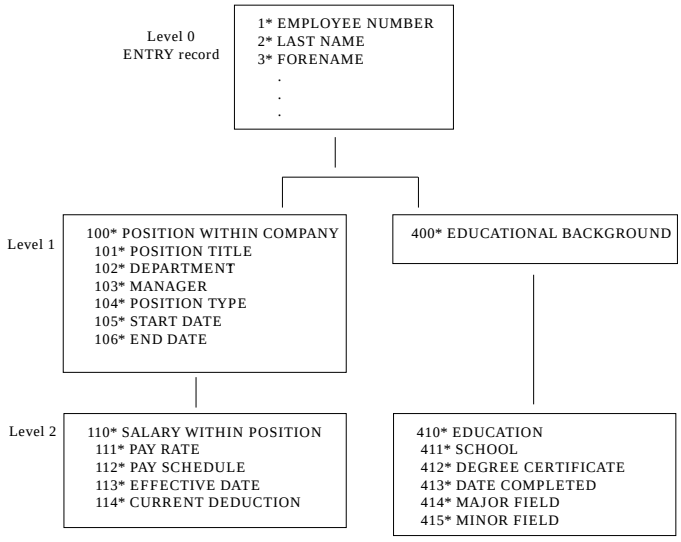
Relationships between schema records are established by arranging the schema records into levels. That is, each schema record is placed at a certain level, which creates a hierarchical structure.

These levels are achieved by ranking schema items with values that occur once per employee (such as an employee's name and address) at a higher level than schema items with multiple values (such as an employee's job titles and salaries). That is, schema items having a one-to-many relationship with other schema items rank higher in the database hierarchy than those other schema items.

For example, using the EMPLOYEE database, notice the levels illustrated below. (In the database definition, shown in Figure 2.1 on page 11, schema items are indented under their parent schema record, and schema records are stair-stepped to the right, reflecting the relationships between the records.)

- Schema items C1 through C16 store values that occur once per employee. These items are grouped as the top level of the database, that is, in the ENTRY record or at *level 0*.
- The ENTRY record has a one-to-many relationship with the POSITION WITHIN COMPANY record (C100). That is, each employee can have more than one position during his or her employment, so position title, department, and so on can have multiple values. Since positions are associated with specific employees, the POSITION WITHIN COMPANY record is related to the ENTRY record. POSITION WITHIN COMPANY is below level 0, that is, at *level 1*.
- Positions have a one-to-many relationship with salary data, because an employee can have more than one salary in a single position. Salary information is grouped in a record named SALARY WITHIN POSITION (C110), which is related to the POSITION WITHIN COMPANY record. SALARY WITHIN POSITION is below level 1, that is, at *level 2*.

Figure 2.1 Levels in Database Definition



The next set of terms refers to the relationships between the levels, which are like relationships in a family.

- The record immediately above a given record is its parent. Every record can have only one parent, and no record is an orphan, except for the ENTRY record, the level 0 record.
- A record that lies on a level above a specified record in the same path is an ancestor record. For example, the ENTRY record is an ancestor of the POSITION WITHIN COMPANY record and the SALARY WITHIN POSITION record. Actually, the ENTRY record is an ancestor of all other records in the database.
- A record is a *descendant* of all its ancestors. For example, C100 is a descendant of the ENTRY record, and C110 is a descendant of the ENTRY record and C100.
- The records immediately below a given record are its *children*. C100 is a child of the ENTRY record; C110 is a child of C100.
- The *path* of a record is the record and all its ancestors. For example, C110, C100, and the ENTRY record make up a path. And C410, C400, and the ENTRY record make up another path.

Schema records are disjoint if their paths are different. For example, records C110 and C410 are disjoint. When you create a view descriptor, you cannot include items that are from disjoint schema records. That is, items from C110 and items from C410 cannot be included in the same view descriptor.

- A *family* consists of a record, all its ancestors, and all its descendants.

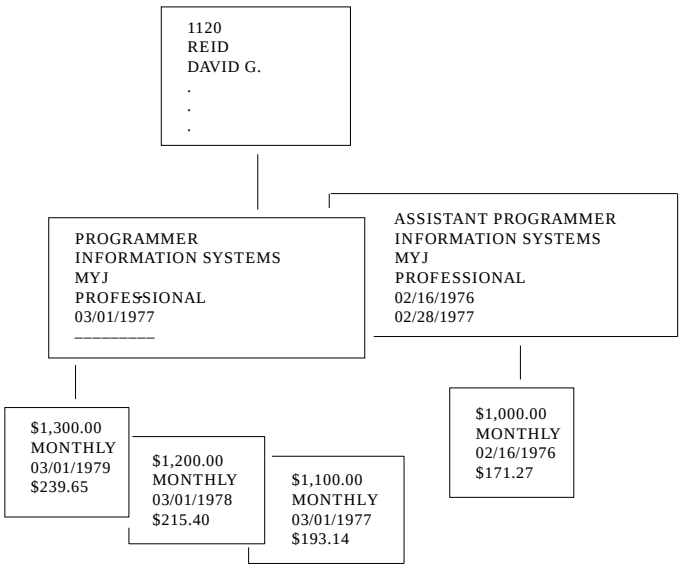
Logical Entries

From your point of view, a database consists of groups of logically related data called logical entries. The database definition serves as a pattern to create logical entries for the database and to interpret them. A logical entry contains groups of related data called data records. A data record is an identifiable set of values treated as a unit and associated with a schema record.

For example, in the EMPLOYEE database, logical entries contain data about employees; that is, all data records pertaining to a single employee make up a single logical entry. Each logical entry has a data record for personal data (such as the employee's name, address, and birthday) and a data record pertaining to the position that the employee holds in the company (such as title, department, manager, and pertinent dates). If the employee has held several positions, there is a data record for each position, as illustrated in Figure 2.2 on page 12.

Using the layout of the database definition, the following figure replaces the schema items with values for one employee. David Reid has held two positions: programmer and assistant programmer. In addition, he has three salary data records for his programmer position.

Figure 2.2 Logical Entry



When you use a view descriptor in a SAS program to access a SYSTEM 2000 database, you must be familiar with the database structure in order to understand how the interface view engine maps a SYSTEM 2000 logical entry into multiple SAS observations and back again. Mapping data between the SAS System and SYSTEM 2000 software is explained next.

Mapping Data between the SAS System and SYSTEM 2000 Software

When you create a view descriptor to access data stored in a SYSTEM 2000 database, you define one path in the database. For example, using the EMPLOYEE database, you could define a view descriptor that includes the items LAST NAME, FORENAME, POSITION TITLE, and PAY RATE. When you access the data using the view descriptor, the interface view engine maps the specified path for each logical entry into multiple observations. For example, the logical entry for David Reid (shown above) would be mapped into the following SAS observations:

Output 2.2 Logical Entry Mapped into SAS Observations

LASTNAME	FORENAME	POSITION	PAYRATE
REID	DAVID G.	ASSISTANT PROGRAMMER	$1,000.00
REID	DAVID G.	PROGRAMMER	$1,100.00
REID	DAVID G.	PROGRAMMER	$1,200.00
REID	DAVID G.	PROGRAMMER	$1,300.00

When browsing SYSTEM 2000 data, such as with the FSVIEW procedure, the results would be similar to Output 2.2 on page 13, with LASTNAME and FORENAME values repeated for each POSITION value and PAYRATE value. Actually, though, the value DAVID G. REID is stored in the database only once. For retrievals, the results are straightforward. When updating, however, you need to keep in mind that values at higher levels in the database generally do not exist as often as they seem to.

Using the FSVIEW procedure, suppose you need to change the spelling of a last name, for example, from Reid to Reed. All you need to do is type 'REED' over one of the values 'REID,' and with a single update operation, the last names are all corrected. On the other hand, using the FSEDIT procedure, suppose you want to delete an observation for David Reid. Each observation for his positions and salary data would display his last name and first name. If you deleted an observation, for example the one for Assistant Programmer, the deletion would not affect his last name and forename values, but the POSITION and PAYRATE values would be physically removed. See Chapter 5, "Browsing and Updating SYSTEM 2000 Data," on page 43 for more information and an example of deleting an observation from a SYSTEM 2000 logical entry.

Null Data – Missing Values

A logical entry does not have to have data at every level of the database definition. A logical entry can contain nulls; that is, missing values or empty records.

- A null item is a schema item that has no value in the data record. For example, in the logical entry shown in Figure 2.2 on page 12, since David Reid still works for the company, he does not have a value for the schema item END DATE for his programmer position. Therefore, that item is a null item.
- A data record consisting entirely of null items is called a null record. A null record can occur when there are data for a given data record but no data for its parent

record. For example, position information exists but an employee hasn't been hired yet. (That is, there is data at level 1 but the ENTRY record is a null record.) Or salary information exists, but position information hasn't been entered. (That is, there is data at level 0 and level 2, but a null record exists at level 1.) In both examples, the null record must be present in the database since a parent record must exist for all records (except the ENTRY record).

- Sometimes a schema record contains no schema items. Such a schema record is called a control node. A control node serves as a control record for descendant records. Looking at the EMPLOYEE database definition in Appendix 3, "Example Data," on page 143, notice that schema record 400 is a control node.

Note: In the SAS System, nulls are referred to as missing values. SYSTEM 2000 software and the SAS System handle nulls (missing values) differently. The interface view engine, however, takes care of the differences in a predictable, useful way. See "Missing Values (Nulls)" on page 132 for a discussion of the differences. △

SYSTEM 2000 Item Types

Every schema item in a SYSTEM 2000 database has a specified item type. The item type tells the software how the values for that item are to be stored and displayed. The manner of storage determines how you can use the values. For example, values consisting exclusively of digits can be stored in a manner suitable for computation. You can specify numeric item types, a date item type, and character item types.

How the values are displayed and stored is also determined by the picture (or length) assigned to an item. For example, a picture for decimal numbers indicates how many digits can be stored and where the decimal point is to appear when the values are displayed or used in computation.

When you create a view descriptor, in addition to assigning SAS variable names from the schema item names, the ACCESS procedure assigns SAS formats, informats, and lengths using the item's picture. See "ACCESS Procedure Data Conversions" on page 96 for the default SAS variable formats and informats for each SYSTEM 2000 item type.

Numeric Item Types

The following items are numeric item types. A numeric item type's picture indicates the number of places required by the longest value expected for an item. A numeric item type's picture is specified using repetitions of the numeral 9. For example, 9999 or 9(4) specifies four places. Values for numeric item types cannot exceed their specified picture; that is, overflow is not allowed for numeric values.

INTEGER
: stores whole numbers.

DECIMAL
: stores numbers with a decimal point.

MONEY
: also stores numbers with a decimal point, but these values include a floating dollar sign at the left and CR at the right (if negative) when displayed.

REAL
: stores fullword (single precision) floating point (or FLOAT) numbers. REAL items do not have a picture. Each REAL value occupies one word (four bytes) in the database.

DOUBLE
stores double word (double precision) floating point numbers. DOUBLE items do not have a picture. Each DOUBLE value occupies two words (eight bytes). You can also use the DOUBLE item type for storing times. (SYSTEM 2000 software does not have a TIME item type.)

Date Item Type

You can specify dates using the DATE item type. A date does not have an assigned picture.

DATE — stores calendar dates in a fixed format. For example, if the date format is MM/DD/YYYY (the default), the value stored must be in the form 07/04/1989 (including the slashes). You cannot store dates prior to the origin of the Gregorian calendar (October 15, 1582).

Note: SYSTEM 2000 software does not have a TIME item type. To store times, use the DOUBLE item type. △

Character Item Types

The following item types are character item types. A character item type's picture corresponds to the number of places that would accommodate most of the values for the item. A character item type's picture is specified using repetitions of the letter X. For example, XXXX or X(4) specifies four places. Values for character item types, except for the UNDEFINED item type, can exceed their picture (up to 250 characters) if the specified picture is at least X(4). That is, CHARACTER and TEXT item types have overflow capabilities.

CHARACTER
stores alphanumeric values with trailing, leading, and extra internal blanks removed. For example, JOHN∅∅∅SMITH is stored and displayed as JOHN∅SMITH.

TEXT
also stores alphanumeric values, but blanks are not removed. For example, ∅∅JOHN∅∅∅SMITH∅∅∅ is stored and displayed as ∅∅JOHN∅∅∅SMITH∅∅∅ .

UNDEFINED
stores binary bit-string data. UNDEFINED items can contain any of the 256 EBCIDIC characters, which are treated like TEXT items except that overflow is not allowed.

Note: When you create a view descriptor, the ACCESS procedure assigns default variable lengths to the corresponding SAS variables using the pictures of the selected items. However, since CHARACTER and TEXT item types have overflow capabilities, there may be values stored in the database that are greater than the default variable length. When you use the view descriptor to select data stored in the database, the larger values will not be recognized.

Therefore, to access values that exceed their item's picture, you must change the length in the view descriptor definition to the largest possible value stored in the database, up to a maximum of 200. △

SYSTEM 2000 Indexing

One of the specifications when defining a schema item is whether SYSTEM 2000 software is to create an index of its values. The software uses the indexes to access the appropriate data records quickly and efficiently.

A schema item for which an index is created is a key item, implying that it provides easy access to data records containing its values. Thus, a key item has an associated index of every distinct value that occurs for the schema item. Note, however, that key values do not have to be unique, and there can be many key items in a database definition or none.

If a schema item is defined as non-key, its values are not indexed, although they can be searched sequentially.

In addition, you can create or delete an index of values by using the SYSTEM 2000 CREATE INDEX and REMOVE INDEX statements. The software automatically changes the specified item to key or non-key. (For information on these statements, see the appropriate SYSTEM 2000 documentation.)

Selecting a Subset of Data

A database wouldn't be very efficient if all logical entries had to be accessed when you needed data from only some of them. Therefore, SYSTEM 2000 software allows you to specify a where-clause to identify those parts of the database that are relevant to your query or update.

A where-clause lets you select a subset of values by specifying conditions that values must meet. A where-clause consists of the keyword WHERE (or WH) and one or more conditions. Typically, a condition consists of a schema item, an operator, and a value or a range of values, as illustrated in these examples:

```
WHERE ACCRUED VACATION EXISTS
WHERE SEX EQ MALE
WHERE BIRTHDAY SPANS 01/01/1949 * 12/31/1949
WHERE STREET ADDRESS CONTAINS /RIM ROCK/
```

You can also combine conditions using connector operators to form expressions, as illustrated in the following example:

```
WHERE SKILL TYPE = COBOL AND
YEARS OF EXPERIENCE = 4
```

For the SAS/ACCESS interface to SYSTEM 2000 software, you can include a SYSTEM 2000 where-clause in a view descriptor to specify selection criteria. In addition to or instead of a where-clause, you can specify selection criteria in a SAS program using a SAS WHERE clause.

Note that the SYSTEM 2000 where-clause and the SAS WHERE clause have some differences. For example, in a SYSTEM 2000 where-clause, the date format (by default) is MM/DD/YYYY, and you do not have to include single quotes around character literals.

For more information on SYSTEM 2000 where-clauses and a description of the syntax, see “SYSTEM 2000 Where-Clause” on page 90. For more information on using a SAS WHERE clause, see “Using a SAS WHERE Clause for Selection Criteria” on page 134 and “Connecting Strings” on page 139.

Sorting Output

In addition to selecting specific data, you can specify the output order for data. This is done by specifying a SYSTEM 2000 ordering-clause, which consists of sort keys, separated by commas. A sort key can be a schema item or a schema record. You can also specify for each sort key whether the output is to be ordered in ascending order or in descending order. For example, the output produced by the following ordering-clause is first sorted by LAST NAME (C2) in ascending order (the default) and then by YEARS OF EXPERIENCE (C203) in descending order (due to the HIGH specification):

```
OB C2, HIGH C203
```

For the SAS/ACCESS interface to SYSTEM 2000 software, you can specify data order by including a SYSTEM 2000 ordering-clause when you create a view descriptor. In addition, you can specify data order in a SAS program using a SAS BY clause. Note, however, that a SAS BY clause overrides a SYSTEM 2000 ordering-clause stored in a view descriptor. For more information on SYSTEM 2000 ordering-clauses, see "SYSTEM 2000 Ordering-clause" on page 94.

SYSTEM 2000 Passwords

SYSTEM 2000 software provides data security with a multi-level password system. Three types of passwords (consisting of one to four alphanumeric characters with no blanks) can be associated with a SYSTEM 2000 database:

- a master password - required
- secondary passwords
- a DBA password.

The master password holder has unqualified access to the database. This is the password under which a database is created. The master password holder can also assign multiple secondary passwords with authorities assigned to individual database components and a DBA password with authorities assigned to individual SYSTEM 2000 statements.

A secondary password holder can have retrieval, update, where-clause, or no access for any combination of database components. An authority is a SYSTEM 2000 code that associates a secondary password with a database component and determines what kind of access to the database the password has. These authorities are designated as R-, U-, W- and N-authorities. For example, a secondary password holder can have retrieval and where-clause authority for all components but no authority to update them.

The DBA password provides a level of security between that of the master password and the secondary passwords. This password allows the DBA to administer databases without being able to access the data stored in them.

For the SAS/ACCESS interface, you must supply a SYSTEM 2000 password in both the access descriptor and the view descriptor to access SYSTEM 2000 data. They can be the same or different passwords; however, the password assigned to the view descriptor must encompass the data described by the access descriptor. The view descriptor password can be stored in the view or you can supply (or override) one with a SAS data set option.

SYSTEM 2000 Execution Environments

When you access a SYSTEM 2000 database, you can work in either a single-user execution environment or a Multi-user execution environment.

In a single-user environment, you are working with your own copy of SYSTEM 2000 software. You usually have exclusive access to the database. However, the single-user environment can be configured so that all users can query the database.

In a Multi-User environment, many users can access a database at the same time, with queries and updates being handled by the Multi-User software simultaneously for all databases being accessed. The Multi-User software automatically ensures data protection during concurrent updates, and it automatically guards a database against conflicting tasks.

Database Files

SYSTEM 2000 software has eight database files associated with each database. The first six are required files.

- File 1 - Master Record and Definition Table
- File 2 - Distinct Values Table
- File 3 - Extended Field Table
- File 4 - Multiple Occurrence Table
- File 5 - Hierarchical Table
- File 6 - Data Table
- File 7 - Update Log (optional)
- File 8 - Rollback Log (optional)

When you are working in a single-user environment, you must allocate the appropriate database files in your SAS session prior to accessing the data. For a Multi-User environment, the database files can be allocated before the Multi-User software is initialized or (Release 12.0 and later) the files can be dynamically allocated during execution using the ALLOC command.

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