Merge Processing and Alternate Table Lookup Techniques

Prepared by

International SAS® Training and Consulting

Destiny Corporation
100 Great Meadow Rd Suite 601
Wethersfield, CT 06109-2379
Phone: (860) 721-1684 1-800-7TRAINING Fax: (860) 721-9784
Email: info@destinycorp.com
Web: www.destinycorp.com

Lookups Using Merges

The values to retrieve during a table lookup operation can be stored in a SAS data set.

A key is a unique value found in the data set and is used to determine the observation obtained by the lookup.

This form of MERGE is called MATCH MERGING. It is the most traditional form of pulling two files together.

Consider the following examples using this data:

VIEWTABLE: saved.managers

	DEPT	MANAGER	TITLE
1	101	B WILLIE	MANAGER
2	201	S SMITH	VICE PRESIDENT
3	301	R OSWALD	ADMINISTRATOR
4	401	J JONES	PRESIDENT

VIEWTABLE: saved.employee

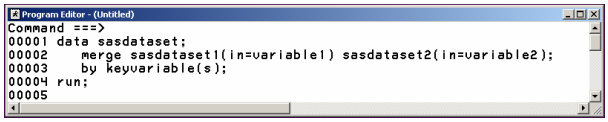
	DEPT	EMPLOYEE
1	301	AL FRANKLIN
2	301	CRAIG MASTERS
3	301	FAT TUESDAY
4	401	JIM DIXON
5	401	JOHN DOE
6	401	JOHN HOWES
7	201	JP JONES
8	301	LARRY SMITH
9	401	PAUL JONES
10	401	PAULIE SURE
11	201	RICHARD NIXON
12	401	SALLY MAY
13	301	STEVE SMITH
14	501	ELIZABETH DOLE

Notes:

- MERGE statements are written in the data step, in conjunction with a BY statement.
- This combination of the MERGE and BY statements reads and combines observations from two or more data sets, based upon their key variable values.
- MATCH MERGING uses a special data set option called (IN=variable). This structures a variable in the program data vector.
- It can be checked for its contribution to the data step. If it has a value of 1, the observation was contributed by that data set; if it has a value of 0, the observation was not.

Syntax

The syntax for data step merging is as follows:



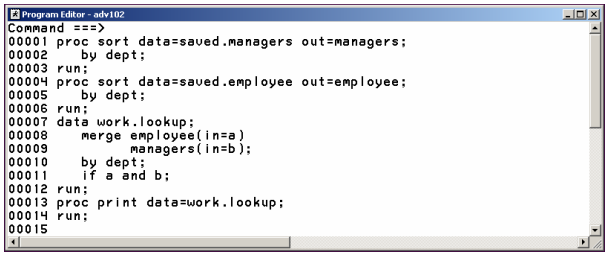
This assumes that the incoming data sets are coming into the step organized by the key variable(s). The organization can be done either by sorting or by indexing.

Guidelines

- The BY variables must be common to all data sets (the same name) and must be the same type in those data sets.
- The BY variables must be the same length in all data sets.
- Data sets must be indexed or sorted by the BY variables.
- The values of the BY variables determine the observations to be MATCH MERGED.
- The value of the BY variable must change in all data sets in order for the variables read from the data sets to no longer be retained.
- Possible selection statements against two data sets with IN= variables called A and B would be:

⇒ IF A;
⇒ IF B=1;
⇒ IF A AND B;
⇒ IF A OR B;
⇒ IF NOT(A AND B);

Example



Obs	DEPT	EMPLOYEE	MANAGER	TITLE
1	201	JP JONES	S SMITH	VICE PRESIDENT
2	201	RICHARD NIXON	S SMITH	VICE PRESIDENT
3	301	AL FRANKLIN	R OSWALD	ADMINISTRATOR
4	301	CRAIG MASTERS	R OSWALD	ADMINISTRATOR
5	301	FAT TUESDAY	R OSWALD	ADMINISTRATOR
6	301	LARRY SMITH	R OSWALD	ADMINISTRATOR
7	301	STEVE SMITH	R OSWALD	ADMINISTRATOR
8	401	JIM DIXON	J JONES	PRESIDENT
9	401	JOHN DOE	J JONES	PRESIDENT
10	401	JOHN HOWES	J JONES	PRESIDENT
11	401	PAUL JONES	J JONES	PRESIDENT
12	401	PAULIE SURE	J JONES	PRESIDENT
13	401	SALLY MAY	J JONES	PRESIDENT

Advantages

- Multiple values can be retrieved.
- The only limit to the size of the data set is disk space.
- Use of multiple BY variables allow for lookups that are dependent upon more than one variable.
- More than one data set can be used to provide access to different tables.

Disadvantages

- BY variables must be used to sort or index data sets.
- Key values must have an exact match.
- BY variables must be present in all data sets.

Lookups Using Indexes

Indexes may also be used to retrieve data from tables.

Indexes are well used in the following circumstances:

- The table is too large to hold in memory (SQL tries to hold the entire table in memory).
- Only a few values need to be retrieved.
- A SET or MODIFY statement contains the key= option.

You may use indexes when performing an SQL join:

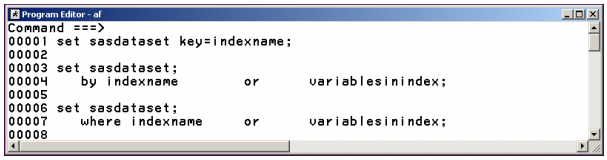
- Using a BY statement variable list starting with a simple or composite primary key

  OR

- Using a WHERE statement referencing a simple or composite primary key.

Syntax

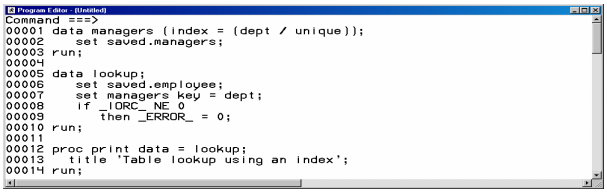
The syntax for index usage is as follows:



Guidelines

- Do not index when the physical size of the data set is small.
- Create an index based on a variable with a large number of distinct values (e.g., SSN would be a good candidate, but gender would not be a good candidate.)
- Indexes should be used to retrieve a small subset of the data set.
- Keep the number of indexes small so disk storage and update costs are minimal.
- Indexes must conform to the assumptions concerning value distributions.
- Sort your data set in order of the most frequently used index.
- Use the MODIFY statement when the master data set is indexed based on the variables used for matching transactions.

Example



Line Number	Description
1	Unique option says that all values of **dept** are unique in the saved.managers data set at the time that the index is created.
7	Key= option says that SAS will use the dept index when searching the managers data set.
8	_IORC_ is an automatic variable that is created when you use the KEY= option. "_IORC_ NE 0" means that there is no match. By querying it and setting _ERROR_ = 0 appropriately, the PDV will not be printed in the log.

Obs	DEPT	EMPLOYEE	MANAGER	TITLE
1	301	AL FRANKLIN	R OSWALD	ADMINISTRATOR
2	301	CRAIG MASTERS	R OSWALD	ADMINISTRATOR
3	301	FAT TUESDAY	R OSWALD	ADMINISTRATOR
4	401	JIM DIXON	J JONES	PRESIDENT
5	401	JOHN DOE	J JONES	PRESIDENT
6	401	JOHN HOWES	J JONES	PRESIDENT
7	201	JP JONES	S SMITH	VICE PRESIDENT
8	301	LARRY SMITH	R OSWALD	ADMINISTRATOR
9	401	PAUL JONES	J JONES	PRESIDENT
10	401	PAULIE SURE	J JONES	PRESIDENT
11	201	RICHARD NIXON	S SMITH	VICE PRESIDENT
12	401	SALLY MAY	J JONES	PRESIDENT
13	301	STEVE SMITH	R OSWALD	ADMINISTRATOR
14	501	ELIZABETH DOLE	R OSWALD	ADMINISTRATOR

Advantages

- Only the observations needed are read from the lookup data set.
- Multiple values are retrieved as a result of the lookup operation.
- The appropriate master observation is directly accessed.
- No additional disk space is required because updates are done in place with the MODIFY statement.

Disadvantages

- Increased resources required to store and maintain the index.
- Key values must have an exact match in order to be found.

Using the KEY= Option

When using the KEY= option, SAS creates an automatic variable called _IORC_. This can be used to determine whether or not the index search returned a value:

- _IORC_ = 0 indicates that SAS found a matching observation

IORC ne 0 this depends on whether or not the UNIQUE option was specified when the index was created:

  - If UNIQUE was specified, _IORC_ ne 0 means that SAS did not find a matching observation.
  - If UNIQUE was not specified, _IORC_ ne 0 may indicate that SAS found multiple matching observations.

Lookups Using Formats

The PROC FORMAT statement allows creation of user-defined formats for data.

When a table lookup is required, pre-defined formats can be used to retrieve the appropriate data needed.

Formats are temporary or permanent and are stored in a Formats catalog in a SAS data library.

Use the PROC FORMAT statement to define the following:

1. Value Formats: Create labels for codes using either numeric or character values.

2. Picture Formats: Create templates for numeric values like 999-99-9999 for social security numbers.

3. Informats: Control the reading and storing of data in data entry applications.

Syntax

NOTE: The cntlin option allows specifying a SAS data set as input; typing of actual values is not necessary.

Per SAS Online Help the syntax for Proc Format is as follows:

Syntax

```
PROC FORMAT <option(s)>;
    EXCLUDE entry(s);

    INVALUE <$>name <(informat-option(s))>
    value-range-set(s);

    PICTURE name <(format-option(s))>
    value-range-set-1 <(picture-1-option(s) )>
    <...value-range-set-n <(picture-n-option(s))>>;

    SELECT entry(s);

    VALUE <$>name <(format-option(s))>
    value-range-set(s);
```

To do this	Use this statement
Exclude catalog entries from processing by the FMTLIB and CNTLOUT= options	EXCLUDE
Create an informat for reading and converting raw data values	INVALUE
Create a template for printing numbers	PICTURE
Select catalog entries from processing by the FMTLIB and CNTLOUT= options	SELECT
Create a format that specifies character strings to use to print variable values	VALUE

To do this	Use this option
Specify a SAS data set from which PROC FORMAT builds an informat or format	CNTLIN=
Create a SAS data set that stores information about informats or formats	CNTLOUT=
Print information about informats or formats	FMTLIB
Specify a SAS catalog that will contain the informats or formats that you are creating in the PROC FORMAT step	LIBRARY=
Specify the number of characters of the informatted or formatted value that appear in PROC FORMAT output	MAXLABLEN=
Specify the number of characters of the start and end values that appear in the PROC FORMAT output	MAXSELEN=
Prevent a new informat or format from replacing an existing one of the same name	NOREPLACE
Print information about each format and informat on a separate page	PAGE

For more information please refer to SAS Online Help.

Guidelines

- Format names may be up to 8 characters long including the $ (dollar) sign.
- Values for numeric formats are not placed in quotation marks (character formats are quoted.)
- Missing values may be formatted.
- Mixed case text is allowed for formatted values. However, formats are case sensitive.
- Character formats work only for character variables.
- Numeric formats work only for character variables.
- SAS format names and user-defined format names cannot be the same.
- User-defined character formats begin with a $ (dollar sign) in the name.
- User-defined formats cannot end in a number.
- All formatted values are quoted.
- A value statement must accompany each format.
- Formatted values can be up to 200 characters long.
- A value cannot be mapped to more than one formatted value. When the format is created, values are sorted by default.

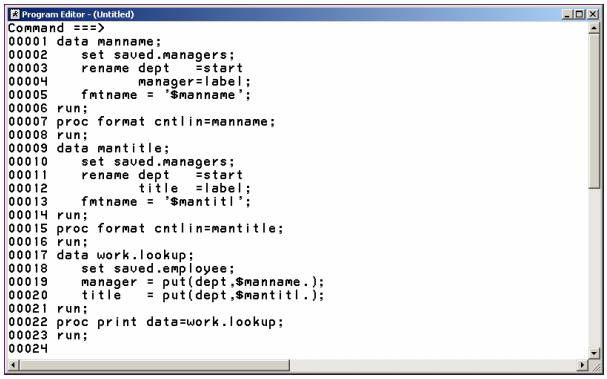
The PUT function is typically used to test on the returned, formatted value of a variable.

The syntax is as follows:

PUT(argument,format.)

The following two data sets SAVED.MANAGERS and SAVED.EMPLOYEES are used in all of the lookup examples.

Example



Line Numbers	Description
3-4, 11-12	The variables are renamed to **start** and **label** to meet the requirements of PROC FORMAT. The variable used for **start** is the key value.
5, 13	A format name is assigned to the required variable name of **fmtname**.
7-8, 15-16	The datasets are written out to the respective formats using the data set names from the previous steps.
2, 10	Notice that the original data set has to be read twice, because of the variable name requirements.
19,20	The new variables in the new dataset are created by the format lookup from the second dataset.
5 & 19, 13 & 20	Notice the matching format names.
1 & 7, 9 & 15	Notice the matching data set names.

Obs	DEPT	EMPLOYEE	manager	title
1	301	AL FRANKLIN	R OSWALD	ADMINISTRATOR
2	301	CRAIG MASTERS	R OSWALD	ADMINISTRATOR
3	301	FAT TUESDAY	R OSWALD	ADMINISTRATOR
4	401	JIM DIXON	J JONES	PRESIDENT
5	401	JOHN DOE	J JONES	PRESIDENT
6	401	JOHN HOWES	J JONES	PRESIDENT
7	201	JP JONES	S SMITH	VICE PRESIDENT
8	301	LARRY SMITH	R OSWALD	ADMINISTRATOR
9	401	PAUL JONES	J JONES	PRESIDENT
10	401	PAULIE SURE	J JONES	PRESIDENT
11	201	RICHARD NIXON	S SMITH	VICE PRESIDENT
12	401	SALLY MAY	J JONES	PRESIDENT
13	301	STEVE SMITH	R OSWALD	ADMINISTRATOR
14	501	ELIZABETH DOLE	501	501

Advantages

- Formats can be placed in PROC statements, eliminating the need to reprocess data from the data step.
- Values which need to be looked up can be discrete, a list, or a range of values.
- Format tables can be easily maintained by using CNTLOUT= and CNTLIN= which specify data set information, instead of hard coded values.
- Values are looked up using a binary search, by default.
- Using the PUT function with a format in the data step requires less CPU time than a MERGE.

Disadvantages

- The entire format must be loaded into memory. This limits the size of the format.
- Only one value will be returned as the result of a format lookup.
- Only one variable may be used to perform the lookup operation.
- Only one format can be applied to each variable at a time.
- Some formats require more disk space than a data set with the same data.

Lookups Using Arrays

Arrays can also be used for table lookups.

Arrays are often used when the data to be retrieved can be identified by position, for example, the 1st item, 2nd item etc., or when the table value to be retrieved is identified by one or more numeric values.

The ARRAY (table) can be made up of values which are either hard coded in the data step or are stored into a data set or external file and then loaded into array variables via a set or input statement.

Use the ARRAY statement in the data step to define a set of variables to be processed in a similar manner.

Syntax

The syntax for ARRAY statement is as follows (SAS Online Help):

```
ARRAY
        ARRAY array-name { subscript } <$> < length > < array-elements >
        < initial-value-list )>;
```

A temporary ARRAY is usually defined to list all potential values with which to compare data.

The ARRAY option used is _temporary_.

Guidelines

- Variables are created if they do not already exist in the program data vector.
- An ARRAY must be defined before the ARRAY name can be referenced.
- The ARRAY must contain either all character elements or all numeric elements.
- ARRAY statements are not executable.
- ARRAY statements cannot be used in compile time statements like DROP, KEEP and FORMAT.
- ARRAY elements do not become part of the output data set. They exist only for the duration of data step processing.
- A SET statement can be used to load an ARRAY.

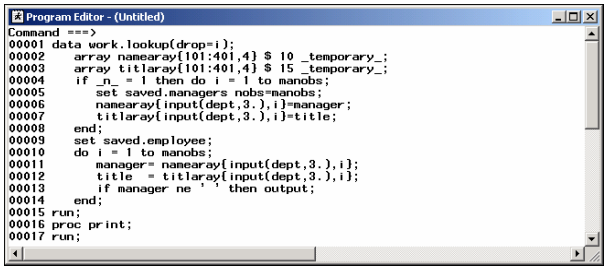
Loading an ARRAY from A SAS Data Set

ARRAY values should be stored in data sets when:

- The master data set contains too many values to easily initialize in the ARRAY statement.

- Values in the master data set change often.
- Many programs use the same values.

Example



```
Program Editor - (Untitled)
Command ===>
00001 data work.lookup(drop=i);
00002    array namearay(101:401,4) $ 10 _temporary_;
00003    array titlaray(101:401,4) $ 15 _temporary_;
00004    if _n_ = 1 then do i = 1 to manobs;
00005       set saved.managers nobs=manobs;
00006       namearay(input(dept,3.),i)=manager;
00007       titlaray(input(dept,3.),i)=title;
00008    end;
00009    set saved.employee;
00010    do i = 1 to manobs;
00011       manager= namearay(input(dept,3.),i);
00012       title  = titlaray(input(dept,3.),i);
00013       if manager ne ' ' then output;
00014    end;
00015 run;
00016 proc print;
00017 run;
```

Line Numbers	Comments
2 & 3	The array dimensions specify a 2-dimensional array. For the first dimension, the lower boundary is 101, and the upper boundary is 401; for the second dimension, the lower boundary is the default of 1, and the upper boundary is 4. The values of dept are 101, 201, 301, 401, hence the boundaries. We already know there are 4 values, hence the second dimension.
2&3	One array is needed for each variable being assigned.
4	Process saved.managers only during the first execution of the data step.
4	The variable manobs is the number of observations in the saved.managers data set; manobs gets its value at compile time.
6 & 7	The input function is used because the dept variable is character, and a numeric value is required for the array index value.
6 & 7	These statements load their respective values into the arrays.
9	Both arrays are completely loaded by the time saved.employee starts to be processed.
13	Both manager and title will be blank on observations until the proper array element is processed. The IF statement prevents observations from being created with blank values.

Obs	DEPT	MANAGER	TITLE	EMPLOYEE
1	301	R OSWALD	ADMINISTRATOR	AL FRANKLIN
2	301	R OSWALD	ADMINISTRATOR	CRAIG MASTERS
3	301	R OSWALD	ADMINISTRATOR	FAT TUESDAY
4	401	J JONES	PRESIDENT	JIM DIXON
5	401	J JONES	PRESIDENT	JOHN DOE
6	401	J JONES	PRESIDENT	JOHN HOWES
7	201	S SMITH	VICE PRESIDENT	JP JONES
8	301	R OSWALD	ADMINISTRATOR	LARRY SMITH
9	401	J JONES	PRESIDENT	PAUL JONES
10	401	J JONES	PRESIDENT	PAULIE SURE
11	201	S SMITH	VICE PRESIDENT	RICHARD NIXON
12	401	J JONES	PRESIDENT	SALLY MAY
13	301	R OSWALD	ADMINISTRATOR	STEVE SMITH
14	401	J JONES	PRESIDENT	ELIZABETH DOLE

Advantages

- The exact position of values can be used.
- Multiple values can be used to determine the ARRAY element to be retrieved.
- Numeric mathematical expressions can be used to determine which element of the ARRAY is to be referenced. An exact match is not necessary.
- The data set being looked up does not have to be indexed or sorted.

Disadvantages

- The ARRAY must fit in memory.
- Numeric values are needed as pointers in ARRAY elements.
- The lookup operation results in only one value being retrieved.

Lookups Using SQL Joins

The PROC SQL statement does not require all data sets to have common variables in order to join them.

Multiple tables may be joined to create a new data set.

Using a join, a subquery or both can eliminate data retrieval problems, in many situations.

Subqueries are used when more than one query is needed to achieve the desired results.

Each subquery provides a subset of the table used in the query.

While joins and subqueries are used in queries, a join is usually the most efficient process.

PROC SQL can also be used to create SQL views and SAS data sets.

Syntax

The syntax used for SQL joins, as displayed in SAS Online Help, is as follows:

```
PROC SQL< option-list>;
    ALTER TABLE alter-statement;
    CREATE create-statement;
    DELETE delete-statement;
    DESCRIBEdescribe-statement;
    DROPdrop-statement;
    INSERTinsert-statement;
    RESET <reset statement option-list>;
    SELECT select-statement;
    UPDATE update-statement;
    VALIDATEvalidate-statement;
```

For additional information please refer to SAS Online Help.

Guidelines

- Joins can be performed in many ways. The different forms of joins available in SAS include an Inner Join, Outer Join, Full Join, Left Join, and a Right Join.
- The default Join is an Inner Join.
- SQL Joins are table lookups that require exact matches across observations.

- SQL can also be used to summarize data during the process.
- SQL can also be used to return data in a sorted order during the process.
- SQL can also be used to return statistics and subsetted data before or after the join process.
- Indexes, when available, are used appropriately.

Example

```
Program Editor - (Untitled)
Command ===>
00001 proc sql;
00002    create table lookup as
00003    select a.dept,a.employee,b.manager,b.title
00004    from saved.employee a, saved.managers b
00005    where a.dept=b.dept;
00006 quit;
00007 proc print data=work.lookup;
00008 run;
00009
```

Obs	DEPT	EMPLOYEE	MANAGER	TITLE
1	301	AL FRANKLIN	R OSWALD	ADMINISTRATOR
2	301	CRAIG MASTERS	R OSWALD	ADMINISTRATOR
3	301	FAT TUESDAY	R OSWALD	ADMINISTRATOR
4	401	JIM DIXON	J JONES	PRESIDENT
5	401	JOHN DOE	J JONES	PRESIDENT
6	401	JOHN HOWES	J JONES	PRESIDENT
7	201	JP JONES	S SMITH	VICE PRESIDENT
8	301	LARRY SMITH	R OSWALD	ADMINISTRATOR
9	401	PAUL JONES	J JONES	PRESIDENT
10	401	PAULIE SURE	J JONES	PRESIDENT
11	201	RICHARD NIXON	S SMITH	VICE PRESIDENT
12	401	SALLY MAY	J JONES	PRESIDENT
13	301	STEVE SMITH	R OSWALD	ADMINISTRATOR

Advantages

- Data sets need not be sorted or indexed, although SQL will use indexes when available.
- Data sets (tables), views, or reports (queries) can be created.
- Many data sets can be used without having common variables in all data sets.

Disadvantages

- Up to thirty-two tables can be joined at one time.
- SQL joins require more resources (memory) than using MERGE statements.

Lookups Using Macros

Macros may also be used to call data from a table.

Values can be placed into macro variables using a CALL SYMPUT function or a %LET statement.

These macro variable values containing table information can then be called using a SYMGET function.

The SYMGET function relates program data vector key variables to macro variables.

Syntax

The syntax used for macro lookups is as follows:

```
Program Editor - (Untitled)
Command ===>
00001 data _null_;
00002    set or input section;
00003    call symput('characterprefix'!!
00004       keyvariable,trim(valuevariable(s));
00005 run;
00006 data result;
00007    set or input section;
00008    newvariable=symget('characterprefix'!!valuevariable(s));
00009 run;
00010
```

Guidelines

- Macro lookups require a key to assign and retrieve information.
- Macro variables load in memory.
- Consider using this method when there are a few different values across a large file and not many unique values matching with many unique values.

Example

```
Program Editor - (Untitled)
Command ===>
00001 data _null_;
00002    set saved.managers;
00003    call symput('dep'!!dept,manager!!title);
00004 run;
00005 data lookup;
00006    set saved.employee;
00007    manager=substr(symget('dep'!!dept),1,20);
00008    title  =substr(symget('dep'!!dept),21,20);
00009 run;
00010 proc print data=work.lookup;
00011 run;
00012
```

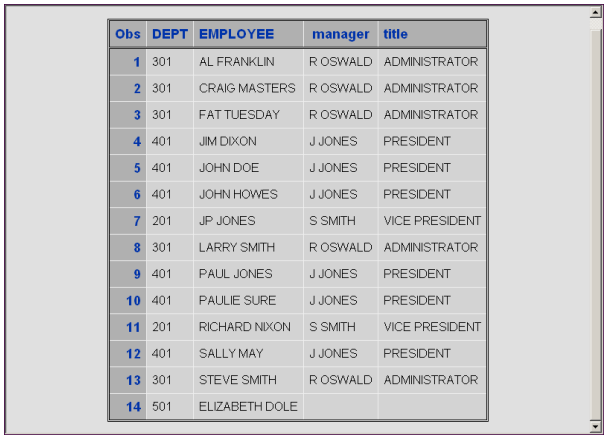
Line Number	Comments
3	The macro variable name is formed as follows: the first 3 characters will be 'dep'; the rest of the variable name will be the value of the key variable, which in this case is dept. The value of the macro variable is the value of manager concatenated with the value of title.
7 & 8	To get the value of manager and title, get the appropriate part of the macro variable. Both manager and title are 20 bytes.

Here is another version of the program that avoids string concatenation and the substr function, which can be slow. This may also be easier to understand and modify.

```
Program Editor - (Untitled)
Command ===>
00001 data _null_;
00002    set saved.managers;
00003    call symput('man' !! dept, manager);
00004    call symput('ttl' !! dept, title);
00005 run;
00006
00007 data lookup2;
00008    set saved.employee;
00009    length manager title $20;
00010    manager = symget('man' !! dept);
00011    title   = symget('ttl' !! dept);
00012 run;
00013
00014 proc print data = lookup2;
00015  title 'Lookup2 using macros';
00016 run;
```

Line Number	Description
3	Create macro variables called 'man101', 'man201', etc. that contain only the manager's name.
4	Create macro variables called 'ttl101', 'ttl201', etc. that contain only the manager's title.
10	Look up the manager's name using the symget function to read the macro variables of 'man101', etc.

11	Look up the manager's title using the symget function to read the macro variables of 'ttl101', etc.



Obs	DEPT	EMPLOYEE	manager	title
1	301	AL FRANKLIN	R OSWALD	ADMINISTRATOR
2	301	CRAIG MASTERS	R OSWALD	ADMINISTRATOR
3	301	FAT TUESDAY	R OSWALD	ADMINISTRATOR
4	401	JIM DIXON	J JONES	PRESIDENT
5	401	JOHN DOE	J JONES	PRESIDENT
6	401	JOHN HOWES	J JONES	PRESIDENT
7	201	JP JONES	S SMITH	VICE PRESIDENT
8	301	LARRY SMITH	R OSWALD	ADMINISTRATOR
9	401	PAUL JONES	J JONES	PRESIDENT
10	401	PAULIE SURE	J JONES	PRESIDENT
11	201	RICHARD NIXON	S SMITH	VICE PRESIDENT
12	401	SALLY MAY	J JONES	PRESIDENT
13	301	STEVE SMITH	R OSWALD	ADMINISTRATOR
14	501	ELIZABETH DOLE		

Advantages

- Data sets do not need to be sorted prior to the lookup.
- Save processing time.
- Match against single key variables or multiple key variables.
- Retrieve single variable data values or multiple variable data values.
- Lookup macro variables do not take up disk space.

Disadvantages

- Requires an exact match.
- Requires a good understanding of SAS macros, programming with functions and strings.
- Since macro variables load in memory, many of them may use up available memory.
- Since the value of the key variable is used as part of the macro variable name, the key values cannot be than the maximum length of a variable name.
- Consider increasing memory for macro variables as displayed:

```
OPTIONS MSYMTABMAX=value;
```