CHAPTER 2  

SAS Names and Support for DBMS Names

Introduction

In Version 7 of SAS software, SAS naming conventions have been enhanced to allow longer names for SAS data sets and SAS variables. The conventions also allow case-sensitive or mixed case names for SAS data sets and variables.

The following SAS language elements can now be up to 32 characters in length:

- members of SAS libraries, including SAS data sets, data views, catalogs, catalog entries, and indexes
- variables in a SAS data set
- macros and macro variables.

The following SAS language elements remain unchanged with a maximum length of 8 characters:

- librefs and filerefs
- SAS engine names and passwords
- names of SAS/ACCESS access descriptors and view descriptors (in order to maintain compatibility with Version 6 names)
- variable names in SAS/ACCESS access descriptors and view descriptors.

See the SAS Language Reference Dictionary for a complete description of the new SAS naming conventions.

The following libref.dataset shows the longer name for the data set:
MYDB.TEMP_EMPLOYEES_QTR4_98. Likewise, a variable name can be longer and defined in mixed case, such as Q4HireDates.

When you specify mixed case or case-sensitive names in SAS code, SAS displays the names as you have specified them. In this example, the SAS variables, Flight and dates, are defined in mixed case:

data TEST;
  input Flight $3. +3 dates date7.;
  format dates date9.;
datalines;
114 01MAR98
When the TEST data set is output, the variable names are stored as they are defined, instead of automatically being displayed in uppercase. However, when SAS processes the names, it can process them as FLIGHT and DATES. Because of the way that SAS processes names, it recognizes variable names regardless of the case in which they were created. For example, if you were to use PROC DATASETS to rename the Flight variable, the procedure would recognize Flight even if you input it as flight or FLIGHT. However, the new variable name is stored as the mixed case name All_flights:

```
proc datasets library=work memtype=data;
  modify test;
  rename flight=All_flights;
run;
```

SAS/ACCESS LIBNAME and PROC SQL Options

Using the SAS/ACCESS LIBNAME statement and PROC SQL options, SAS software can handle table and column names in DBMSs that are case-sensitive or nonstandard for SAS, that is, names that allow special characters or blanks that are not allowed in SAS names. The following list briefly describes these options. See your DBMS chapter for information about how SAS processes your DBMS-specific names.

`PRESERVE_COL_NAMES=YES | NO`

is an option in the SAS/ACCESS LIBNAME statement. This option preserves spaces, special characters, and mixed case in DBMS column names if you specify YES. The default value for this option is DBMS specific. See “SAS/ACCESS LIBNAME Options” on page 26 for more information on this option.

```
libname myoradb oracle user=testuser
  password=testpass
  path='myorapath'
  preserve_col_names=yes;

proc sql;
create table empl_test as
  select IDNUM
       LNAME
  from myoradb.employees;
```
PRESERVE_TAB_NAMES=YES | NO

is an option in the SAS/ACCESS LIBNAME statement. This option preserves blank spaces, special characters, and mixed case in DBMS table names if you specify YES. The default value for this option is DBMS specific. See “SAS/ACCESS LIBNAME Options” on page 26 for more information about this option.

libname myoradb oracle user=testuser
  pass=testpass
  path='mypath'
  preserve_tab_names=yes;

proc sql dquote=ansi;
  create table myoradb.STAFF_test as
    select * from myoradb.staff;
quit;

DQUOTE=ANSI | SAS

is a PROC SQL option. This option specifies whether PROC SQL treats values within double quotes as a character string or as a column name or table name. When you specify DQUOTE=ANSI, your SAS code can refer to DBMS names that contain characters and spaces that are not allowed by SAS naming conventions. The following program creates a SAS data view that contains the variable name ID Num.

proc sql dquote=ansi;
  options validvarname=any;
  create view staffview as
    select IDNUM as 'ID Num'n
    from myoradb.STAFF_test;
quit;

proc print data=work.staffview;
run;


The SAS system option VALIDVARNAME=ANY and SAS name literals for variable names also enable you to retain nonstandard names but with certain restrictions. For more information on the VALIDVARNAME= system option, see Chapter 5, “Macro Variables and System Options,” on page 55.

In the following example, the name literals are quoted and followed by an n. Name literals can be used only in a DATA step or in PROC SQL.

options validvarname=any;

data accounts_99;
  infile acct_projections;
  input 'Amount Budgeted'n 'Amount Spent'n 'Amount Difference'n;
....
run;

See the SAS naming conventions and “VALIDVARNAME” on page 58 for more information.

---

### DBMS to SAS Names

The following rules are applied when you map DBMS column names to SAS variable names:

- Non SAS characters in DBMS column names appear as underscores in SAS variable names. For example, DBMS column name `MY$DEPT` becomes SAS variable name `MY_DEPT`.
- SAS makes DBMS column names unique SAS variable names by appending a number (starting with 0) to the variable name when they are normalized. For example, DBMS column names `MY$DEPT`, `My$Dept`, and `my$dept` become SAS variable names `MY_DEPT`, `MY_Dept0`, and `MY_DEPT1`.

Table 2.1 on page 14 and Table 2.2 on page 15 describe how SAS processes DBMS names when it is retrieving DBMS data. This information applies generally to the DBMS names; see your DBMS chapter for possible exceptions. See “Naming Examples” on page 16 for examples that illustrate the different kinds of naming actions and defaults.

<table>
<thead>
<tr>
<th>If your DBMS column name as input is a...</th>
<th>...and you want this SAS variable name...</th>
<th>...Then use these LIBNAME or PROC SQL Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-sensitive DBMS column name, such as <code>Flight</code></td>
<td>Default SAS variable name (uppercase), such as <code>FLIGHT</code></td>
<td>None</td>
</tr>
<tr>
<td>DBMS column name with characters that are not valid in SAS names, such as <code>MY$Flight</code></td>
<td>Default SAS variable name (uppercase) where an underscore replaces the invalid characters, such as <code>MY_FLIGHT</code></td>
<td>None</td>
</tr>
<tr>
<td>Case-sensitive DBMS column name, such as <code>Flight</code></td>
<td>Case-sensitive SAS variable name, such as <code>Flight</code></td>
<td><code>preserve_col_names=yes</code></td>
</tr>
<tr>
<td>DBMS column name with characters that are not valid in SAS names, such as <code>MY$Flight</code></td>
<td>Case-sensitive SAS variable name where an underscore replaces the invalid characters, such as <code>My_Flight</code></td>
<td><code>preserve_col_names=yes</code></td>
</tr>
<tr>
<td>DBMS column name with characters that are not valid in SAS names, such as <code>MY$Flight</code></td>
<td>Nonstandard, case-sensitive SAS variable name, such as <code>My$Flight</code></td>
<td><code>proc sql dquote=ansi</code> and <code>reserve_col_names=yes</code> or in a DATA or PROC step, use a SAS name literal and <code>preserve_col_names=yes</code> <code>validvarname=any</code></td>
</tr>
</tbody>
</table>

These options may be required. Default values for these options are DBMS specific.
Table 2.2  DBMS Table to SAS Data Set Names

<table>
<thead>
<tr>
<th>If your DBMS table name as input is...</th>
<th>...And you want this SAS data set name...</th>
<th>...Then use these LIBNAME or PROC SQL Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-sensitive DBMS table name, such as <code>Staff</code></td>
<td>Default SAS data set or member name (uppercase), such as <code>STAFF</code></td>
<td><code>preserve_tab_names=no</code></td>
</tr>
<tr>
<td>DBMS table name with characters that are not valid in SAS names, such as <code>All$Staff</code></td>
<td>Default SAS data set name (uppercase) where an underscore replaces the invalid characters, such as <code>ALL_STAFF</code></td>
<td><code>preserve_tab_names=no</code></td>
</tr>
<tr>
<td>Case-sensitive DBMS table name, such as <code>Staff</code></td>
<td>Case-sensitive SAS data set, such as <code>Staff</code></td>
<td><code>preserve_tab_names=yes</code></td>
</tr>
<tr>
<td>DBMS table name with characters that are not valid in SAS names, such as <code>All$Staff</code></td>
<td>Case-sensitive SAS data set name where an underscore replaces the invalid characters, such as <code>All_Staff</code></td>
<td><code>preserve_tab_names=yes</code></td>
</tr>
<tr>
<td>DBMS table name with characters that are not valid in SAS names, such as <code>All$Staff</code></td>
<td>Nonstandard, case-sensitive SAS data set name, such as <code>All$Staff</code></td>
<td><code>proc sql dquote=ansi preserve_tab_names=yes</code></td>
</tr>
</tbody>
</table>

These options may be required. Default values for these options are DBMS specific.

SAS to DBMS Names

Table 2.3 on page 16 and Table 2.4 on page 16 describe how SAS names are used when you use SAS/ACCESS software to create DBMS data. This information applies generally; see your DBMS chapter for possible exceptions. See “Naming Examples” on page 16 for examples that illustrate the different kinds of naming actions and defaults.
### Naming Examples

In this example, the SAS/ACCESS LIBNAME option `PRESERVE_COL_NAMES=` enables you to preserve the case of the column names in the ORACLE data that is accessed through `mydblib.staff`.

```
libname mydblib oracle user=ng password=luck
  path=myoradb
```
SAS Names and Support for DBMS Names

Naming Examples

```sas
preserve_col_names=yes;

proc sql;
  select Idnum, Lname, Fname
  from mydblib.staff
  where City="STAMFORD";

DBMS column and table names that contain characters or blanks that are not valid in SAS cannot be specified directly in a SAS DATA step or procedure, except if you are using:
  - the DQUOTE=ANSI option in PROC SQL
  - SAS name literals written as
    `string'`n

Therefore, you must first specify these kinds of DBMS names as standard SAS names in a PROC SQL view and then reference that view in a DATA step or procedure. In the following example, notice that the LIBNAME options are embedded in the PROC SQL view:

libname mysaslib 'SAS-data-library';

proc sql dquote=ansi;
create view mysaslib.birthdays as
  select first.work_id,
      first."@lastname" as lastname,
      second."birth date" as birthdate
  from dbms1.employees1 as first,
       dbms1.employees2 as second
  where first.work_id=second.work_id
using libname dbms1 odbc user=joshua
    password=freude datasrc='Jo’s Data'
    preserve_col_names=yes
    preserve_tab_names=yes;

proc print data=mysaslib.birthdays;
run;

See the SQL Procedure chapter in the SAS Procedures Guide for more information about embedded libnames in PROC SQL views.

In this example, you use PROC SQL to create a new ORACLE table based on data from other ORACLE tables. By using PRESERVE_COL_NAMES=YES, you preserve the case-sensitivity of the aliased column names.

libname myoralib oracle user=testuser
    password=testpass
    path='myorapath'
    preserve_col_names=yes;

proc sql;
create table myoralib.gtforty as
  select lname as Lastname,
      fname as Firstname,
      salary as Salary
  format=dollar10.2
```
from myoralib.staff a,
myoralib.payroll b
where (a.idnum eq b.idnum) and
(salary gt 40000);

proc print noobs;
title 'Employees with Salaries over $40,000';
run;

Output 2.2  Updating DBMS Data

<table>
<thead>
<tr>
<th>Lastname</th>
<th>Firstname</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAREFOOT</td>
<td>JOSEPH</td>
<td>$43,025.00</td>
</tr>
<tr>
<td>BANADYGA</td>
<td>JUSTIN</td>
<td>$88,606.00</td>
</tr>
<tr>
<td>BRANCACCIO</td>
<td>JOSEPH</td>
<td>$66,517.00</td>
</tr>
<tr>
<td>BRADY</td>
<td>CHRISTINE</td>
<td>$68,767.00</td>
</tr>
<tr>
<td>COHEN</td>
<td>LEE</td>
<td>$91,376.00</td>
</tr>
<tr>
<td>CARTER-COHEN</td>
<td>KAREN</td>
<td>$40,260.00</td>
</tr>
<tr>
<td>CASTON</td>
<td>FRANKLIN</td>
<td>$41,690.00</td>
</tr>
<tr>
<td>FERNANDEZ</td>
<td>KATRINA</td>
<td>$81,081.00</td>
</tr>
<tr>
<td>GRAHAM</td>
<td>ALVIN</td>
<td>$65,111.00</td>
</tr>
<tr>
<td>GREGORSKI</td>
<td>DANIEL</td>
<td>$68,096.00</td>
</tr>
<tr>
<td>HARRIS</td>
<td>CHARLES</td>
<td>$84,685.00</td>
</tr>
<tr>
<td>HASENHAUER</td>
<td>CHRISTINA</td>
<td>$70,736.00</td>
</tr>
<tr>
<td>HAVELKA</td>
<td>RAYMOND</td>
<td>$41,551.00</td>
</tr>
<tr>
<td>HERRERO</td>
<td>CLYDE</td>
<td>$66,130.00</td>
</tr>
<tr>
<td>KIMANI</td>
<td>ANNE</td>
<td>$40,899.00</td>
</tr>
<tr>
<td>HARSHBURN</td>
<td>JASPER</td>
<td>$89,632.00</td>
</tr>
<tr>
<td>MORGAN</td>
<td>ALFRED</td>
<td>$42,264.00</td>
</tr>
<tr>
<td>NEWKIRK</td>
<td>SANDRA</td>
<td>$84,536.00</td>
</tr>
<tr>
<td>NEWKIRK</td>
<td>WILLIAM</td>
<td>$52,270.00</td>
</tr>
<tr>
<td>NEWTON</td>
<td>JAMES</td>
<td>$84,203.00</td>
</tr>
<tr>
<td>NORRIS</td>
<td>DIANE</td>
<td>$43,433.00</td>
</tr>
<tr>
<td>O’NEAL</td>
<td>BRYAN</td>
<td>$40,079.00</td>
</tr>
<tr>
<td>PARKER</td>
<td>JAY</td>
<td>$41,526.00</td>
</tr>
<tr>
<td>PENNINGTON</td>
<td>MICHAEL</td>
<td>$71,349.00</td>
</tr>
<tr>
<td>RAYNOR</td>
<td>MILTON</td>
<td>$43,900.00</td>
</tr>
<tr>
<td>RHOES</td>
<td>JEREMY</td>
<td>$40,586.00</td>
</tr>
<tr>
<td>RIVERS</td>
<td>SIMON</td>
<td>$53,798.00</td>
</tr>
<tr>
<td>ROUSE</td>
<td>JEREMY</td>
<td>$43,071.00</td>
</tr>
<tr>
<td>STEPHENSON</td>
<td>ADAM</td>
<td>$42,178.00</td>
</tr>
<tr>
<td>STEPHENSON</td>
<td>ROBERT</td>
<td>$91,908.00</td>
</tr>
<tr>
<td>THOMPSON</td>
<td>WAYNE</td>
<td>$89,977.00</td>
</tr>
<tr>
<td>TRIPP</td>
<td>KATHY</td>
<td>$84,471.00</td>
</tr>
<tr>
<td>TUCKER</td>
<td>ALAN</td>
<td>$41,538.00</td>
</tr>
<tr>
<td>UPCHURCH</td>
<td>LARRY</td>
<td>$89,858.00</td>
</tr>
<tr>
<td>VENTER</td>
<td>RANDALL</td>
<td>$66,558.00</td>
</tr>
<tr>
<td>WELCH</td>
<td>DARIUS</td>
<td>$40,858.00</td>
</tr>
<tr>
<td>WELLS</td>
<td>AGNES</td>
<td>$42,274.00</td>
</tr>
</tbody>
</table>

In the next example, you create a temporary SAS data set that has case-sensitive names. Then you define your LIBNAME statement and use a SAS DATA step to create the new DBMS table, MYDB.College-Hires-1998. Because you are using a DATA step to create the DBMS table, you must specify the table name as a name literal and use the PRESERVE_TAB_NAMES= and PRESERVE_COL_NAME= options.

data College_Hires_1998;
  input IDnum $4. +3 Lastname $11. +2
    Firstname $10. +2 City $15. +2
    State $2.
  datalines;
3413 Schwartz Robert New Canaan CT
3523 Janssen Heike Stamford CT
3565 Gomez Luis Darien CT

;

libname myoralib oracle user=testuser
password=testpass
path='myorapath'
preserve_tab_names=yes preserve_col_names=yes;

data myoralib.'College-Hires-1998'n;
  set College_Hires_1998;
run;