

Integrating SAS with DBMSs

The ACCESS and DBLOADER procedures

The ACCESS procedure: general info

- It was introduced in SAS Version 6 (a legacy method)
- It is pretty cumbersome to use.
- It has two steps:
 - Create an **ACCESS descriptor** to describe the data in a single DBMS table
 - Create a second descriptor, call **VIEW descriptor** to define a subset of DBMS data described by the ACCESS descriptor.

Statement sequence for accomplishing tasks in the ACCESS procedure

Task	Statements and Options to Use
Create an access descriptor	PROC ACCESS <i>statement-options</i> ; CREATE <i>libref.member-name.ACCESS</i> ; <i>database-connection-statements</i> ; <i>editing-statements</i> ; RUN ;
Create an access descriptor and a view descriptor	PROC ACCESS <i>statement-options</i> ; CREATE <i>libref.member-name.ACCESS</i> ; <i>database-connection-statements</i> ; <i>editing-statements</i> ; CREATE <i>libref.member-name.VIEW</i> ; SELECT <i>column-list</i> ; <i>editing-statements</i> ; RUN ;
Create a view descriptor from an existing access descriptor	PROC ACCESS <i>statement-options</i> , including ACCDESC= <i>libref.access-descriptor</i> ; CREATE <i>libref.member-name.VIEW</i> ; SELECT <i>column-list</i> ; <i>editing-statements</i> ; RUN ;

The ACCESS procedure: types of statements

The ACCESS procedure has several types of statements:

- *Database connection statements:* used to connect to your DBMS.
- *Creating and updating statements:* CREATE and UPDATE.
- *Table and editing statements:*
 - ASSIGN
 - DROP
 - FORMAT
 - LIST
 - QUIT
 - RENAME
 - RESET
 - SELECT
 - SUBSET
 - TABLE
 - UNIQUE

The ACCESS Procedure: syntax

```

PROC ACCESS <options>;
    database-connection-statements;
CREATE libref.member-name.ACCESS | VIEW <password-option>;
UPDATE libref.member-name.ACCESS | VIEW <password-option>;
TABLE= <'>table-name<'>;
ASSIGN <=> YES | NO | Y | N;
DROP <'>column-identifier-1 <'> <...<'>column-identifier-n<'>>;
FORMAT <'>column-identifier-1<'> <=> SAS-format-name-1
<...<'>column-identifier-n<'><=>SAS-format-name-n>;
LIST <ALL | VIEW | <'>column-identifier<'>>;
QUIT;
RENAME <'>column-identifier-1<'> <=> SAS-variable-name-1
<...<'>column-identifier-n<'><=>SAS-variable-name-n>;
RESET ALL | <'>column-identifier-1<'> <...<'>column-identifier-n<'>>;
SELECT ALL | <'>column-identifier-1<'> <...<'>column-identifier-n <'>>;
SUBSET selection-criteria;
UNIQUE <=> YES | NO | Y | N;

RUN;

```

The tasks of the statements

Statement	Task
PROC ACCESS Statement	Access relational DBMS data
Database Connection Statement	Provide DBMS-specific connection information
ASSIGN Statement	Indicate whether SAS variable names and formats are generated
CREATE Statement	Create a SAS/ACCESS descriptor file
DROP Statement	Drop a column so that it cannot be selected in a view descriptor
FORMAT Statement	Change a SAS format for a DBMS column
LIST Statement	List columns in the descriptor and give information about them
QUIT Statement	Terminate the procedure
RENAME Statement	Modify the SAS variable name
RESET Statement	Reset DBMS columns to their default settings
SELECT Statement	Select DBMS columns for the view descriptor
SUBSET Statement	Add or modify selection criteria for a view descriptor
TABLE= Statement	Identify the DBMS table on which the access descriptor is based
UNIQUE Statement	Generate SAS variable names based on DBMS column names
UPDATE Statement	Update a SAS/ACCESS descriptor file

The CREATE statement

CREATE *libref.member-name*.ACCESS | VIEW <password-option>;

The **CREATE** statement is required. It names the access descriptor or view descriptor that you are creating. Use a three-level name:

- The first level identifies the libref of the SAS library where you want to store the descriptor,
- The second level is the descriptor name,
- The third level specifies the type of SAS file (specify **ACCESS** for an access descriptor or **VIEW** for a view descriptor).

Required Arguments

- ***libref.member-name***: identifies the libref of the SAS library where you want to store the descriptor and identifies the descriptor name.
- **ACCESS**: specifies an access descriptor.
- **VIEW**: specifies a view descriptor.

The ASSIGN statement

Indicates whether unique SAS variable names and formats are generated

Syntax: **ASSIGN** <=> YES | NO | Y | N;

Default: NO

Interaction: FORMAT, RENAME, RESET, UNIQUE

Required Arguments:

YES: generates unique SAS variable names from the first eight characters of the DBMS column names. If you specify **YES**, you cannot specify the RENAME, FORMAT, RESET, or UNIQUE statements when you create *view descriptors* that are based on the access descriptor.

When you specify **YES**, SAS generates names according to these rules:

- You can change the SAS variable names only in the access descriptor.
- SAS variable names that are saved in an access descriptor are *always* used when view descriptors are created from the access descriptor. You cannot change them in the view descriptors.
- The ACCESS procedure **allows names only up to eight characters**.

NO: lets you modify SAS variable names and formats when you create an access descriptor and when you create view descriptors that are based on this access descriptor.

Each time the SAS/ACCESS interface encounters a CREATE statement to create an access descriptor, the ASSIGN statement is reset to the default **NO** value.

The RENAME statement

Function: Modifies the SAS variable name

Interaction: ASSIGN, RESET

Syntax: **RENAME** <'>column-identifier-1<'> <=> SAS-variable-name-1 <...<'> column-identifier-n<'><=>SAS-variable-name-n>;

Required Arguments:

- **column-identifier:** specifies the DBMS column name or the positional equivalent from the LIST statement, which is the number that represents the column's place in the descriptor. The equal sign (=) is optional.
- **SAS-variable-name:** specifies a SAS variable name.

The RENAME statement sets or modifies the SAS variable name that is associated with a DBMS column. Two factors affect the use of the RENAME statement: whether you specify the ASSIGN statement when you are creating an access descriptor, and the type of descriptor that you are creating.

- If you omit the ASSIGN statement or specify it with a **NO** value, the renamed SAS variable names that you specify in the access descriptor are retained when an ACCESS procedure executes.
- If you specify the **YES** value in the ASSIGN statement, you can use the RENAME statement to change SAS variable names only while creating an access descriptor. SAS variable names and formats that are saved in an access descriptor are always used when creating view descriptors that are based on the access descriptor.

Example:

```
Rename employee_address=emp_addr;
Rename employee_address emp_addr;
```

The LIST statement

Lists columns in the descriptor and gives information about them

Default: ALL

Syntax: LIST <ALL | VIEW | <'>*column-identifier*<'>>;

Optional Arguments

ALL: lists all DBMS columns in the table, positional equivalents, SAS variable names, and SAS variable formats that are available for a descriptor.

VIEW: lists all DBMS columns that are selected for a view descriptor, their positional equivalents, their SAS names and formats, and any subsetting clauses.

column-identifier: lists information about a specified DBMS column, including its name, positional equivalent, SAS variable name and format, and whether it has been selected. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks.

The SUBSET statement

Adds or modifies selection criteria for a view descriptor

Syntax: **SUBSET** *selection-criteria*;

Required Argument

selection-criteria: one or more DBMS-specific SQL expressions that are accepted by your DBMS, such as WHERE, ORDER BY, HAVING, and GROUP BY. Use DBMS column names, not SAS variable names, in your selection criteria.

You can use the SUBSET statement to specify **selection criteria** when you create a view descriptor. This statement is optional. If you omit it, the view retrieves all data (rows) in the DBMS table. The SUBSET statement is case sensitive. The SQL statement is sent to the DBMS exactly as you type it. Therefore, you must use the correct case for any DBMS object names. If you specify more than one SUBSET statement per view descriptor, the last SUBSET overwrites the earlier SUBSETs.

For example, you could submit the following SUBSET statement for a view descriptor that retrieves rows from a DBMS table:

subset *where firstorder is not null;*

If you have multiple selection criteria, enter them all in one SUBSET statement, as shown in this example:

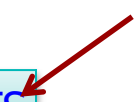
subset ***where*** *firstorder is not null*
and *country = 'USA'*
order by *country;*

Example: Create access and view descriptors

```

proc access dbms=oracle;
  /* create access descriptor */
  create work.pvfc.access;
  user='xy';
  password='xxxxx';
  table=product_t;
  assign=no;
  list all;
  /* create view descriptor */
  create work.products.view;
  select productid productdescription
  productfinish productstandardprice;
  subset where productstandardprice < 500 ;
run;
proc print data=products;
run;

```



Obs	PRODUCTI	PRODUCTD	PRODUCTF	PRODUCTS
1	2	Coffee Table	Natural Ash	200.00
2	3	Computer Desk	Natural Ash	375.00
3	8	Computer Desk	Walnut	250.00

The DBLOAD procedure: general info

The DBLOAD procedure works in the opposite direction. It is used to copy data into a DBMS from a SAS dataset. It enables you to create and load a DBMS table, append rows to an existing table, and submit non-query DBMS-specific SQL statements to the DBMS for processing.

The procedure constructs DBMS-specific SQL statements to create and load, or append, to a DBMS table by using one of these items:

- a SAS data file
- an SQL view or DATA step view
- a view descriptor that was created with the SAS/ACCESS interface to your DBMS
- another DBMS table referenced by a SAS libref that was created with the SAS/ACCESS LIBNAME statement.

The procedure associates each SAS variable with a DBMS column and assigns a default name and data type to each column. It also specifies whether each column accepts NULL values.

Statement sequence for accomplishing tasks in the BDLOAD procedure

Task	Statements and Options to Use
Create and load a DBMS table	PROC DBLOAD <i>statement-options;</i> <i>database-connection-options;</i> TABLE= <'> <i>table-name</i> <'>; LOAD; RUN;
Submit a dynamic, non-query DBMS-SQL statement to DBMS (without creating a table)	PROC DBLOAD <i>statement-options;</i> <i>database-connection-options;</i> SQL <i>DBMS-specific-SQL-statements;</i> RUN;

The BDLOAD Procedure: syntax

PROC DBLOAD *<options>;*

database connection statements;

TABLE= *<'>table-name <'>;*

ACDESC= *<libref.>access-descriptor;*

COMMIT= *commit-frequency;*

DELETE *variable-identifier-1 <...variable-identifier-n>;*

ERRLIMIT= *error-limit;*

LABEL;

LIMIT= *load-limit;*

LIST *<ALL | COLUMN | variable-identifier>;*

NULLS *variable-identifier-1 = Y | N <...variable-identifier-n = Y | N>;*

QUIT;

RENAME *variable-identifier-1 = <'>column-name-1<'> <...variable-identifier-n = <'>column-name-n<'>>;*

RESET *ALL | variable-identifier-1<...variable-identifier-n>;*

SQL *DBMS-specific SQL-statement;*

TYPE *variable-identifier-1 = 'column-type-1' <...variable-identifier-n = 'column-type-n'>;*

WHERE *SAS-where-expression;*

LOAD;

RUN;

The tasks of the statements

Statement	Task
PROC DBLOAD	Send data from SAS to a DBMS
Database Connection	Provide DBMS connection information
ACCDESC=	Create an access descriptor based on the new DBMS table
COMMIT=	Issue a commit or save rows after a specified number of inserts
DELETE	Specify variables that should not be loaded into the new table
ERRLIMIT=	Stop the loading of data after a specified number of errors
LABEL	Cause DBMS column names to default to SAS labels
LIMIT=	Limit the number of observations that are loaded
LIST	List information about the variables to be loaded
LOAD	Create and load the new DBMS table
NULLS	Specify whether DBMS columns accept NULL values
QUIT	Terminate the procedure
RENAME	Rename DBMS columns
RESET	Reset column names and data types to their default values
SQL	Submit a DBMS-specific SQL statement to the DBMS
TABLE=	Name the DBMS table to be created and loaded
TYPE	Change default DBMS data types in the new table
WHERE	Load a subset of data into the new table

The PROC DBLOAD statement

(Sends data from SAS to a DBMS)

Syntax: PROC DBLOAD <options>;

Arguments

DBMS=database-management-system: specifies which database management system you want to access. This DBMS-specific option is required. See the DBMS-specific reference for details.

DATA=<libref.>SAS-data-set: specifies the input data set. You can retrieve input data from a SAS data file, an SQL view, a DATA step view, a SAS/ACCESS view descriptor, or another DBMS table to which a SAS/ACCESS libref points. If the SAS data set is permanent, you must use its two-level name, *libref.SAS-data-set*. If you omit the DATA= option, the default is *the last SAS data set that was created*.

- **APPEND:** appends data to an existing DBMS table that you identify by using the TABLE= statement. When you specify APPEND, the input data specified with the DATA= option is inserted into the existing DBMS table. Your input data can be in the form of a SAS data set, SQL view, or SAS/ACCESS view (view descriptor).

(*CAUTION:* When you use APPEND, you must ensure that your input data corresponds exactly to the columns in the DBMS table.)

The ACCDESC= statement

- Function: Creates an access descriptor based on the new DBMS table
- Syntax: **ACCDESC=**<libref.>access-descriptor;

The ACCDESC= statement creates an access descriptor based on the DBMS table that you are creating and loading. If you specify ACCDESC=, the access descriptor is automatically created after the new table is created and loaded.

The NULLS statement

- Function: Specifies whether DBMS columns accept NULL values
- Default: Y
- Syntax: **NULLS** *variable-identifier-1* = Y|N <...*variable-identifier-n* = Y| N>;
- Y: accept NULL values, N: reject NULL values.
- If you specify N for a numeric column, no observations that contain missing values in the corresponding SAS variable are loaded into the table. A message is written to the SAS log, and the current error count increases by one for each observation that is not loaded.
- If a character column contains blanks (the SAS missing value) and you have specified N for the DBMS column, blanks are inserted. If you specify Y, NULL values are inserted.
- The *variable-identifier* argument can be either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents the variable's place in the data set. For example, if you want the column that is associated with the third SAS variable to accept NULL values, submit this statement:
`nulls 3=y;`
- You can list as many variables as you want in one NULLS statement. If you have previously defined a column as NULLS=N, you can use the NULLS statement to redefine it to accept NULL values.

Example: The PROC BDLOAD porcedure

```
proc dbload dbms=oracle
data=sasuser.year2000;
  user='xy';
  password='xxxxx';
  table=DBL_t;
  nulls 1=n;
  list;
  load;
run;
```

Obs	CRGOREV1	CRGOREV2	CRGOREV3	CRGOREV4	CRGOREV5	CRGOREV6	DATE
1	3280638	561692	2128545	1817984	223134	.	01JAN2000:00:00:00
2	3275164	534184	1878010	1860242	214236	969241	02JAN2000:00:00:00
3	3258884	552088	2123491	1840034	213864	942459	03JAN2000:00:00:00
4	3330580	552294	2357934	1812278	226276	958295	04JAN2000:00:00:00
5	3301534	564340	2145639	1819898	227258	982329	05JAN2000:00:00:00
6	3326138	576790	2131639	1833088	220462	1163366	06JAN2000:00:00:00
7	3246350	533278	2377156	1844822	222436	.	07JAN2000:00:00:00
8	3298714	547340	2156809	1820122	220312	950041	08JAN2000:00:00:00
9	3285232	544116	1952562	1807724	218156	976357	09JAN2000:00:00:00
10	3280508	555252	2142039	1804266	217604	983877	10JAN2000:00:00:00
11	3302822	555976	2345564	1754802	217096	976635	11JAN2000:00:00:00
12	3296096	538238	2179523	1853246	233602	992769	12JAN2000:00:00:00
13	3277750	545314	2138169	1829864	233022	1194362	13JAN2000:00:00:00
14	3251002	575892	2394990	1804538	212506	1002203	14JAN2000:00:00:00
15	3174884	531338	2133343	1819038	216170	968085	15JAN2000:00:00:00
16	3288652	544672	1919772	1844646	222704	975887	16JAN2000:00:00:00
17	3260738	550964	2145919	1812130	222432	991983	17JAN2000:00:00:00
...

Practice

- Use the ACCESS procedure to associate with the Oracle database table, Employee_payroll_t, you created before (using your own user name and password). You are required to create an access descriptor called myaccdsct in the SASUSER library. Then you create a view descriptor in SASUSER to extract all the information of employees who have a salary greater than \$100,000 and create a SAS dataset called highsal. Use the proc print procedure to display the contents of highsal. (Hint: you need to use the RENAME statement to change the column names in the DBMS table.)
- Use the BDLOAD procedure to load a SAS dataset called COMPANY in SASUSER library into your Oracle database and name the table as company_to_dbms_t. Then, use the LIBNAME procedure to associate SAS to your database and display the company_to_dbms_t with the PROC PRIN procedure.

The code and result for bullet 1



Obs	EMP_ID	EMP_GND	SALARY
1	121141	M	194885
2	121142	M	156065
3	120101	M	163040
4	120102	M	108255
5	120259	M	433800
6	120260	F	207885
7	120261	M	243190
8	120262	M	268455
9	120659	M	161290

The code and result for bullet 1

```

proc access dbms=oracle;
  /* create access descriptor */
  create sasuser.myaccdscpt.access;
  user='xy';
  password='xxxxxxx';
  table=employee_payroll_t;
  assign=yes;
  rename employee_gender=emp_gnd;
  rename employee_id=emp_id;
  list all;
  /* create view descriptor */
  create sasuser.highsal.view;
  select employee_id employee_gender salary;
  subset where salary > 100000;
run;
proc print data=sasuser.highsal;
run;

```

Obs	EMP_ID	EMP_GND	SALARY
1	121141	M	194885
2	121142	M	156065
3	120101	M	163040
4	120102	M	108255
5	120259	M	433800
6	120260	F	207885
7	120261	M	243190
8	120262	M	268455
9	120659	M	161290

The code and result for bullet 2

Obs	NAME	AGE	SEX	SSN
1	Morrison, Michael	32	M	
2	Rudelich, Herbert	39	M	029-46-9261
3	Vincent, Martina	34	F	074-53-9892
4	Benito, Gisela	32	F	228-88-9649
5	Sirignano, Emily	12	F	442-21-8075
6	Harbinger, Nicholas	36	M	446-93-2122
7	Phillipon, Marie-Odile	28	F	776-84-5391
8	Gunter, Thomas	27	M	929-75-0218

The code and result for bullet 2

```
proc dbload dbms=oracle
  data=sasuser.company;
  user='xy';
  password='xxxxxxx';
  table=company_to_dbms_t;
  list;
  load;
run;
libname myora oracle
  user='xy'
  password='xxxxxxx';
run;
proc print
  data=myora.company_to_dbms_t;
run;
```

Obs	NAME	AGE	SEX	SSN
1	Morrison, Michael	32	M	
2	Rudelich, Herbert	39	M	029-46-9261
3	Vincent, Martina	34	F	074-53-9892
4	Benito, Gisela	32	F	228-88-9649
5	Sirignano, Emily	12	F	442-21-8075
6	Harbinger, Nicholas	36	M	446-93-2122
7	Phillipon, Marie-Odile	28	F	776-84-5391
8	Gunter, Thomas	27	M	929-75-0218