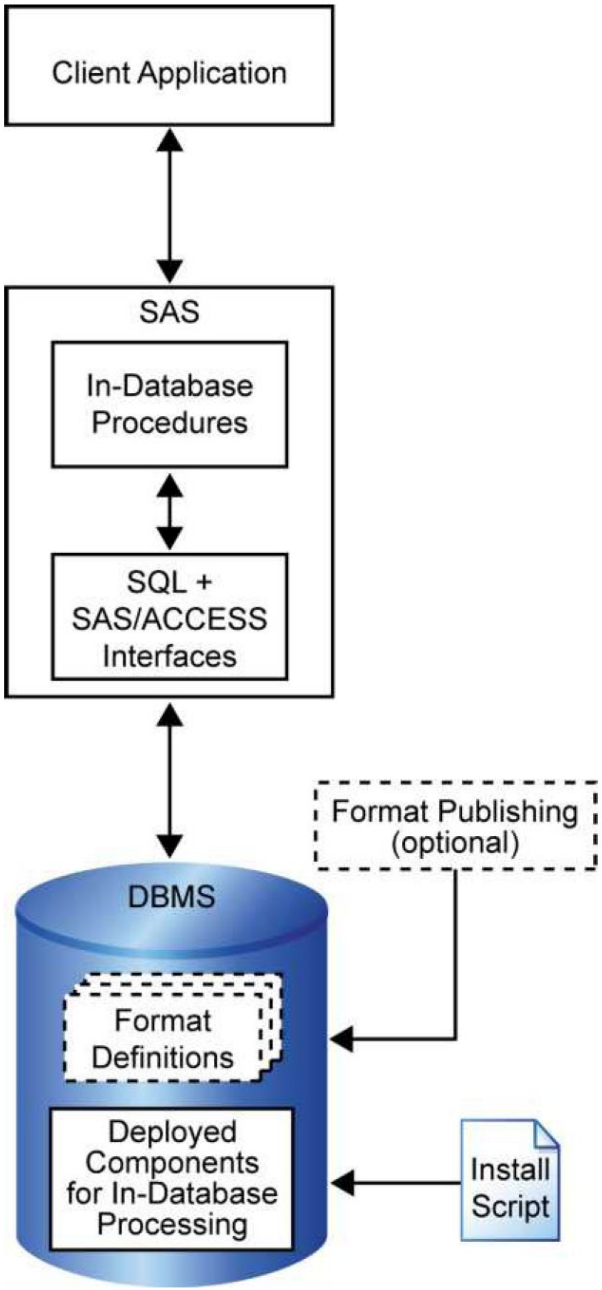
SAS In-Database Processing

Introduction to In-Database Processing

- Using conventional processing, a SAS procedure (by means of the SAS/ACCESS engine) receives all the rows of the table from the database. All processing is done by the SAS procedure. Large tables mean that a significant amount of data must be transferred.
- Using the new in-database technology, the procedures that are enabled for processing inside the database generate more sophisticated queries. These queries allow the **aggregations** and **analytics** to be run inside the database.
  - Some generate SQL procedure syntax and use **implicit pass-through** to generate the native SQL.
  - Other generate native SQL and use **explicit pass-through**.
- The queries submitted by SAS in-database procedures reference DBMS SQL functions and, in some cases, the special SAS functions that are deployed inside the database. For example,
  - the SAS_PUT( ) function, which enables you to execute PUT function calls inside the database,
  - SAS functions for computing sum-of-squares-and-crossproducts (SSCP) matrices.
- A much smaller result set is returned for the remaining analysis that is required to produce the final output. As a result,
  - more work is done inside the database,
  - less data movement can occur,
  - significant performance improvements.

A diagram Illustrating the In-Database Procedure Process



In-Database Processing Overview

In-Database Feature	Software Required	DBMSs Supported
format publishing and the SAS_PUT() function	• Base SAS • SAS/ACCESS Interface to the DBMS	DB2 under UNIX Netezza Teradata Aster
scoring models	• Base SAS (and SAS EM) • SAS/ACCESS Interface to the DBMS • SAS Scoring Accelerator • SAS Model Manager (optional)	Aster *n*Cluster DB2 under UNIX Greenplum Netezza Teradata Oracle, Hadoop, Aster
Base SAS procedures: FREQ RANK REPORT SORT SUMMARY/MEANS TABULATE	• Base SAS • SAS/ACCESS Interface to the DBMS	DB2 under UNIX and PC Hosts Oracle Netezza Teradata Aster, Hadoop
SAS/STAT procedures: CORR CANCORR DMDB DMINE DMREG FACTOR PRINCOMP REG SCORE TIMESERIES VARCLUS	• Base SAS (for CORR) • SAS/ACCESS Interface to Teradata • SAS/STAT (for CANCORR, FACTOR,PRINCOMP, REG, SCORE, VARCLUS) • SAS/ETS (for TIMESERIES) • SAS Enterprise Miner (for DMDB, DMINE, DMREG) • SAS Analytics Accelerator	Teradata

In-Database Base SAS Procedures and Supported BDMS

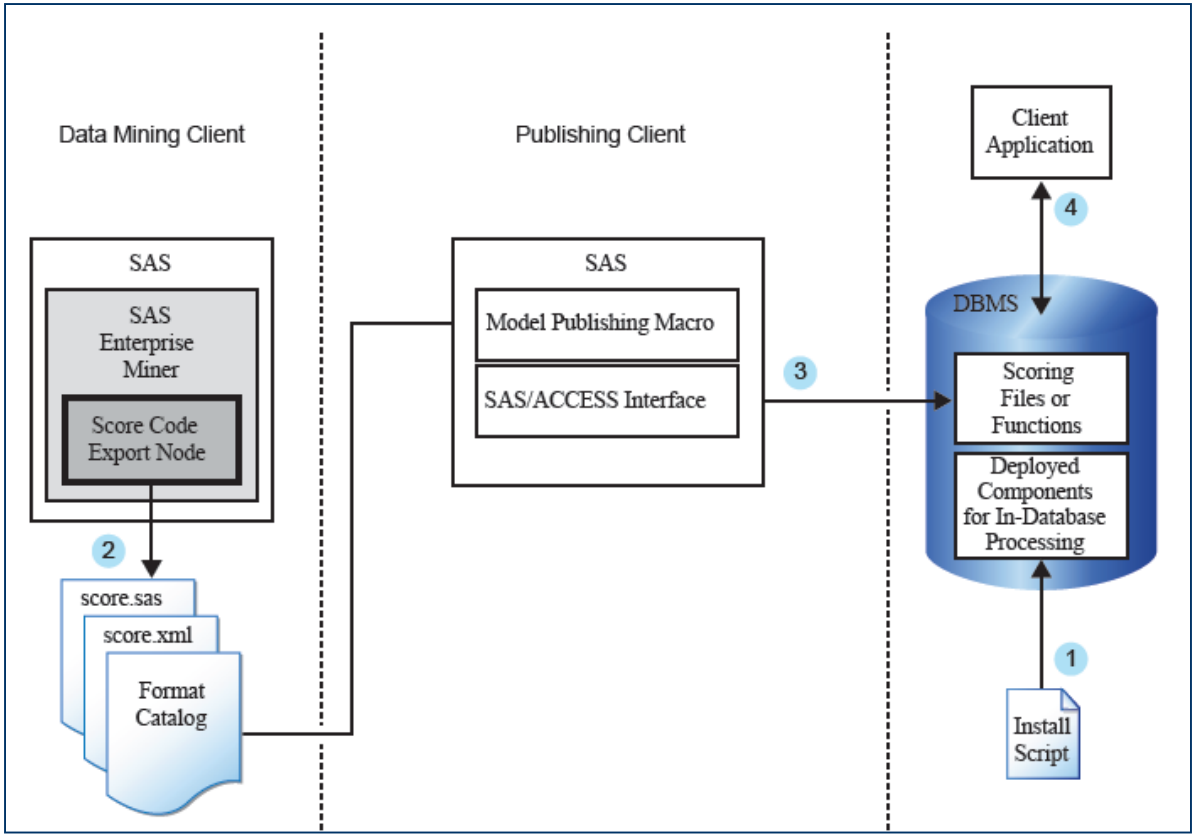
The following Base SAS procedures have been enhanced for in-database processing inside Aster *n*Cluster, DB2 under UNIX and PC Hosts, Greenplum, Netezza, Teradata, and **Oracle**.

- FREQ
- RANK
- REPORT
- SORT
- SUMMARY/MEANS
- TABULATE

An Overview of the SAS Scoring Accelerator

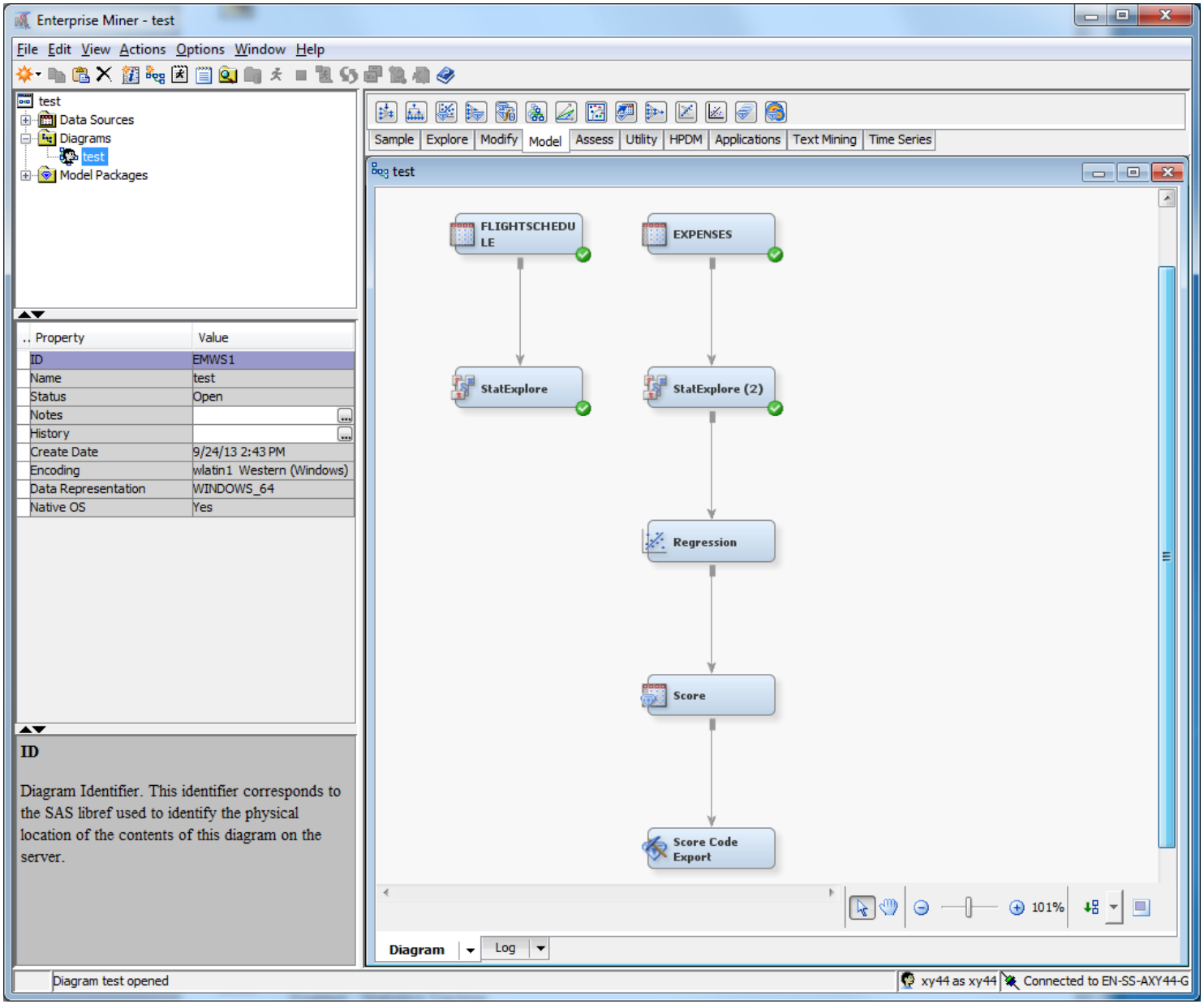
- When using conventional processing to access data inside a DBMS, SAS (Enterprise Miner) asks the SAS/ACCESS engine for all rows of the DB table. The SAS/ACCESS engine generates an SQL SELECT * statement and passes it the DB, which fetches all the rows in the table. Then the SAS/ACCESS engine returns them to SAS Enterprise Miner. This is time consuming.
- The SAS Scoring Accelerator embeds the SAS Enterprise Miner scoring models directly in a DB, so the scoring process is done inside the database and thus does not require the transfer of data.
- The SAS Scoring Accelerator takes the models developed by SAS Enterprise Miner and translates them into scoring files or **functions** that can be deployed inside the database. After the scoring functions are published, the functions extend the database's SQL language and can be used in SQL statements like other database functions. After that, they are used by the SAS Embedded Process to run the scoring model.
- The SAS Scoring Accelerator consists of two components:
  1. The Score Code Export node in SAS Enterprise Miner. This extension exports the model scoring logic (including metadata about the required input and output variables) from SAS Enterprise Miner.
  2. The publishing client that includes a scoring publishing macro. This macro translates the scoring model into files that are used inside the database to run the scoring model. The publishing client then uses the SAS/ACCESS Interface to the database to publish the files to the database.

Scoring Accelerator: Process Flow Diagram



1. Install the components that are necessary for in-database processing.
2. Use SAS Enterprise Miner to create a scoring model. Use the Score Code Export node to export files that are used to create the scoring files or functions to a score output directory.
3. Start SAS 9.4 and run the SAS publishing macros. This creates the files that are needed to build the scoring files or functions and publish those files to the database.
4. After the scoring files or functions are created, you can run your scoring model.

An Example: SAS EM and Scoring Accelerator



User-Defined Functions and the SAS Embedded Process

There are two methods by which formats and scoring models are processed inside the database:

- **User-defined functions (UDFs)**
  - Formats and scoring models are converted by the publishing macros into scoring and format **functions** that are similar to any user-defined functions in the database.
  - In-database processing of formats and scoring models by means of user-defined functions is currently supported by DB2 under UNIX, Greenplum, Netezza, Teradata and **Oracle** (newest version).

- **SAS Embedded Process**
  - The SAS Embedded Process is a SAS server process that is installed and runs inside the database to read and write data from the database. The advantage of using the SAS Embedded Process is that a single function or a stored procedure is used instead of multiple, user-defined functions.
  - The SAS Embedded Process is currently supported for Aster *n*Cluster format publishing and scoring models, and for DB2 and Teradata scoring models.

Running SAS In-Database Procedures

- The **SQLGENERATION** system option or the SQLGENERATION LIBNAME option must be set to DBMS or DBMS='*database-name*'.
- The SQLGENERATION system option or LIBNAME statement option controls whether and how in-database procedures are run inside the database. By default, the SQLGENERATION system option is set to NONE DBMS='ASTER DB2 GREENPLUM NETEZZA ORACLE TERADATA'.
- Conventional SAS processing is also used when specific procedure statements and options do not support in-database processing.
- The LIBNAME statement must point to a valid version of the DBMS:
  - Aster *n*Cluster 5.0 or higher
  - DB2 UDB9.5 Fixpack 7 running only on AIX or LINUX x64
  - Greenplum
  - Netezza 5.0 or higher
  - Teradata server running version 12 or higher for Linux
  - Oracle 9i or higher

The SQLGENERATION= System Option

- **Functionality:** Specifies whether and when SAS procedures generate SQL for in-database processing of source data.
- **Valid in:** configuration file, OPTIONS statement, SAS System Options window
- **Syntax:**

  SQLGENERATION=<(>NONE | DBMS | NONE DBMS='*engine1... enginen*
  <EXCLUDEDB=*engine* | '*engine1... enginen*'>
  <EXCLUDEPROC="*engine*='*proc1... procn*' *enginen*='*proc1... procn*' "><)>

  SQLGENERATION=" "
- **Default:** NONE DBMS='ASTER DB2 GREENPLM NETEZZA ORACLE TERADATA'
- **Restriction:** For DBMS= and EXCLUDEDB= values, the maximum length of an engine name is 8 characters. For the EXCLUDEPROC= values, the maximum length of a procedure name is 16 characters. An engine can appear only once, and a procedure can appear only once for a given engine.
- **Data source:** Aster *n*Cluster, DB2 under UNIX and PC Hosts, Greenplum, Oracle, and Teradata.

Syntax Description

- **NONE:** prevents those SAS procedures that are enabled for in-database processing from generating SQL for in-database processing. This is a common state.
- **DBMS:** allows SAS procedures that are enabled for in-database processing to generate SQL for in-database processing of DBMS tables through supported SAS/ACCESS engines. This is a comman state.
- **DBMS='*engine1 engine2 ... enginen*':** specifies one or more SAS/ACCESS engines. It modifies the primary state.
- **Restriction:** The maximum length of an engine name is 8 characters.
- **EXCLUDEDB=*engine* | '*engine1 engine2 ... enginen*':** prevents SAS procedures from generating SQL for in-database processing for one or more specified SAS/ACCESS engines.
- **EXCLUDEPROC="*engine*='*proc1 proc2 ... procn*' *enginen*='*proc1 proc2 ... procn*' ":** identifies engine-specific SAS procedures that do not support in-database processing.
- **“ ”:** resets the value to the default that was shipped.

Examples

- The default that is shipped with the product:

```
options sqlgeneration='';
proc options option=sqlgeneration;
run;
```

- SAS procedures generate SQL for in-database processing for all databases except DB2 in this example.

```
options sqlgeneration='';
options sqlgeneration=(DBMS EXCLUDEDB='DB2');
proc options option=sqlgeneration;
run;
```

Examples

- In this example, in-database processing occurs only for Teradata. SAS procedures that are run on other databases do not generate SQL for in-database processing.

```
options sqlgeneration='';
options SQLGENERATION = (NONE DBMS='Teradata');
proc options option=sqlgeneration;
run;
```

- In this example, SAS procedures generate SQL for Teradata and Oracle in-database processing. However, no SQL is generated for PROC1 and PROC2 in Oracle.

```
options sqlgeneration='';
options SQLGENERATION = (NONE DBMS='Teradata Oracle'
EXCLUDEPROC="oracle='proc1 proc2'");
proc options option=sqlgeneration;
run;
```

SQLGENERATION= LIBNAME Option

Function: Specifies whether and when SAS procedures generate SQL for in-database processing of source data.
Valid in: SAS/ACCESS LIBNAME statement.
Default: NONE DBMS='Teradata'
Restriction: You must specify NONE and DBMS=, for the primary state.
Data source: Aster *n*Cluster, DB2 under UNIX and PC Hosts, Greenplum, Oracle, and Teradata, etc.

Example:

```
libname myora oracle direct_exe=delete sqlgeneration=(none
dbms='oracle') datasrc=server-name user=user-id
password=password schema=schema;
```

Example: In-database computing with PROC FREQ

```
options msglevel=i;
options sqlgeneration='';
options SQLGENERATION = (NONE DBMS='Oracle');
proc options option=sqlgeneration;
run;

libname myora oracle
user='xy' password='xxxxxxx';
run;

proc freq data=myora.employee_payroll_t;
tables empsex*salary/chisq;
run;
```

Example: In-database computing with PROC FREQ, con't

```
libname myora oracle
user='xy' password=XXXXXXXXXX;
NOTE: Libref MYORA was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name:
   run;
   proc freq data=myora.employee_payroll_t;
   tables empsex*salary/chisq;
   run;
WARNING: In-database formatting is not available on the database. In-database processing will proceed without it.
NOTE: SQL generation will be used to construct frequency and crosstabulation tables.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           1.00 seconds
      cpu time            0.62 seconds
```

Example: In-database computing with PROC FREQ, con't

The FREQ Procedure

Frequency Percent Row Pct Col Pct	EMPLOYEE_GENDER (EMPLOYEE_GENDER)	22710	24015	24025	24100	24390	24515	24990	25020	25080	25110	25125	25130	25180	25185	25195	25205	252
	F	0	1	1	0	0	0	1	1	1	0	0	1	0	2	0	0	
		0.00	0.24	0.24	0.00	0.00	0.00	0.24	0.24	0.24	0.00	0.00	0.24	0.00	0.47	0.00	0.00	0.
		0.00	0.52	0.52	0.00	0.00	0.00	0.52	0.52	0.52	0.00	0.00	0.52	0.00	1.05	0.00	0.00	0.
		0.00	100.00	100.00	0.00	0.00	0.00	100.00	33.33	100.00	0.00	0.00	100.00	0.00	100.00	0.00	0.00	0.
	M	1	0	0	1	1	1	0	2	0	1	1	0	1	0	1	1	
		0.24	0.00	0.00	0.24	0.24	0.24	0.00	0.47	0.00	0.24	0.24	0.00	0.24	0.00	0.24	0.24	0.
		0.43	0.00	0.00	0.43	0.43	0.43	0.00	0.86	0.00	0.43	0.43	0.00	0.43	0.00	0.43	0.43	0.
		100.00	0.00	0.00	100.00	100.00	100.00	0.00	66.67	0.00	100.00	100.00	0.00	100.00	0.00	100.00	100.00	100.
	Total	1	1	1	1	1	1	1	3	1	1	1	1	1	2	1	1	
		0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.71	0.24	0.24	0.24	0.24	0.24	0.47	0.24	0.24	0.

Statistics for Table of EMPLOYEE_GENDER by SALARY

Statistic	DF	Value	Prob
Chi-Square	390	385.6234	0.5531
Likelihood Ratio Chi-Square	390	530.5755	<.0001
Mantel-Haenszel Chi-Square	1	2.0711	0.1501
Phi Coefficient		0.9537	
Contingency Coefficient		0.6901	
Cramer's V		0.9537	

WARNING: 100% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 424

In-database Processing Limitations

- Row order:
  - DBMS tables have no inherent order for the rows. Therefore, the BY statement with the OBS option and the FIRSTOBS option prevents in-database processing.
  - If you specify the ORDER=DATA option for input data, the procedure might produce different results for separate runs of the same analysis.
  - The order of rows written to a database table from a SAS procedure is not likely to be preserved because the DBMS has no obligation to maintain row order.
  - However, you can print a table using the SQL procedure with an ORDER BY clause to get consistent row order.
- By-groups: only the DESCENDING option is supported.
- The following libname statement options and settings prevent in-database processing:
  - DBMSTEMP=YES
  - DBCONINIT
  - DBCONTERM
  - DBGEN_NAME=SAS
- Some dataset-related options and settings also prevent in-database processing:
  - RENAME= on a dataset.
  - DATA= and OUT= datasets are the same DBMS table.
  - DATA= and OUT= options may affect in-database processing, and refer to the documentation of a specific SAS procedure.
- Other items:
  - DBMSes do not support SAS passwords.
  - SAS encryptions that requires passwords are not supported.