SAS® GLOBAL FORUM 2020 #SASGF

Paper SAS4266-2020

How to Access and Manage Microsoft Azure Cloud Data Using SAS®

Kumar Thangamuthu, SAS Institute Inc.

ABSTRACT

The popularity of cloud data storage has grown exponentially over the last decade as more and more organizations are transitioning from on-premises to cloud data storage and data management. Microsoft Azure is one of the big players accelerating the move to cloud. In this paper, we cover the following topics:

- Overview of SAS® to access and manage data in Azure Storage.
- SAS best practices and options to work with big data and relational databases in Azure Storage.

This paper contains data access examples and use cases to explore Azure cloud data using SAS.

INTRODUCTION

As organizations start to realize the impact of digital transformation, they are moving storage to the cloud as they move their computing to the cloud. Data Storage in the cloud is elastic and responds to demand while only paying for what you use, similar to compute in the cloud. Organizations must consider data storage options, efficient cloud platform and services, and migrating SAS applications to the cloud.

SAS provides efficient SAS Data Connectors and SAS In-Database Technologies support to Azure database variants. A data storage running in Azure cloud is much like an on-premise database, but instead Microsoft manages the software and hardware. Azure services can take care of the scalability and high availability of the database with Database as a Service (DBaaS) offerings and minimal user input. SAS integrates with Azure cloud databases whether SAS is running on-premise or in the cloud.

AZURE STORAGE AND DATABASES

Many common data platforms already in use are being refactored and delivered as service offerings to Azure cloud customers. Azure offers database service technologies that are familiar to many organizations. It is important to understand the terminology and the different database services to best meet the demands of your business use case or application. Benefits to organizations are reducing hardware and software footprint to manage. Databases that scale automatically to meet business demand and software that optimizes and creates backups means organizations can spend more time deriving insights from their data and less time managing infrastructure.

Organizations can connect from the SAS platform and access data from the various Azure data storage offerings. Whether it be Azure Blob storage file system or Azure HDInsight supporting elastic Hadoop data lake or Relational databases such as SQL Server, MySQL, MariaDB, or PostgreSQL, SAS/ACCESS engines and data connectors have these covered with optimized data handling abilities to empower organizations going through digital transformation journey.

In this paper we will look at code samples to connect to some of the Azure storage and databases from the SAS platform, as well as the ability of access data along with log files to understand the execution behind the scene.

SAS AND AZURE DATA LAKE STORAGE

SAS Viya can read and write ORC and CSV data files to Azure Data Lake Storage Generation 2(ADLS2). There is a new data connector called SAS ORC Data Connector to facilitate the data transfer between CAS and ADLS2. The SAS ORC Data Connector enables you to load data from an Apache Optimized Row Columnar table into CAS. This data connector can be used with a path or Azure ADLS2 CASLIB.

SAS LIBRARY TO ADLS

```
/*create a CAS session */

cas casauto;

caslib "Azure Data Lake Storage Gen 2" datasource=( srctype="adls"

      accountname=”sasdemo”
      filesystem="data"
      dnsSuffix=dfs.core.windows.net
      timeout=50000 tenantid=<Azure Application Tenant ID UUID>
      applicationId=<Azure registered application UUID>

)
path="/" subdirs global

/* creates library reference for SAS compute */

libref=AzureDL;

caslib _all_ assign;
```

Here is an explanation of the parameters that are used to create a caslib:

- CASLIB – A library reference. The caslib is the space holder for the specified data access. The Azure Data Lake Storage Gen 2 CAS library is used to specify the ADLS data source.
- SRCTYPE – Source type is ADLS which corresponds to Azure Data Lake Storage connection.
- ACCOUNTNAME – Azure Data Lake Storage account name.
- FILESYSTEM – ADLS container file system name.
- TENANTID & APPLICATIONID – Available from the Azure Registered Application page for your organization or individual use.
- PATH – Points to the directory structure where the file system resides.
- LIBREF – Creates a SAS library reference along with a CAS library.

```
1 %studio_hide_wrapper;
82 cas casauto;
NOTE: The session CASAUTO connected successfully to Cloud Analytic Services
sas.viya.com using port
5570. The UUID is xxxxxx-xxxxx-xxxx-xxxxxxxxx. The user is demo and the
active caslib is CASUSERHDFS(demo).
NOTE: The SAS option SESSREF was updated with the value CASAUTO.
NOTE: The SAS macro _SESSREF_ was updated with the value CASAUTO.
NOTE: The session is using 2 workers.
83
84 caslib "Azure Data Lake Storage Gen 2" datasource=(
85 srctype="adls"
86 accountname='sasdemo'
87 filesystem="data"
88 dnsSuffix=dfs.core.windows.net
89 timeout=50000
90 tenantid="xxxxxxx-xxxxxx-xxxxx-xxxxxxxxxxx"
91 applicationId="xxxxxx-xxxx-xxxxx-xxxxxxxxxxx"
92 )
93 path="/"
94 subdirs
95 global
96 libref=AzureDL
97 ;
NOTE: 'Azure Data Lake Storage Gen 2' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'Azure Data Lake Storage Gen
2'.
NOTE: CASLIB Azure Data Lake Storage Gen 2 for session CASAUTO will be
mapped to SAS Library AZUREDL.
NOTE: Action to ADD caslib Azure Data Lake Storage Gen 2 completed for
session CASAUTO.
98
99 caslib _all_ assign;
```

SAS Viya uses the available CAS session CASAUTO to create the CAS library reference to ADLS. In this example, it uses a CAS clustered environment with 3 nodes, including 2 worker nodes. To make the CAS library available to all the users, global parameters can be used. ORC or CSV can be loaded from the ADLS blob container file system "data" to a CAS in-memory cluster or saved to ADLS from CAS.

LOAD AND SAVE ORC DATA TO AZURE STORAGE FROM SAS VIYA

Let's look at an example to load a SAS dataset to a CAS in-memory server. Once the data is in CAS, it can be used for any distributed data processing, report, analytics, or modeling. The final CAS in-memory data output is saved as an ORC file to Azure Data Lake Storage.

```
proc casutil;
load data=sashelp.class casout="class" outcaslib="adls" replace;
save casdata="class" casout="class.orc" replace;
quit;
```

Important parameters:

- PROC CASUTIL – CASUTIL procedure that works with CAS tables including data transfer, table & file information, and drop & delete tables.
- LOAD – CAS action set to load data to CAS in-memory servers

- DATA – SAS traditional data set used as input
- CASOUT – Output CAS in-memory distributed table
- OUTCASLIB – Output CAS Library reference
- SAVE – CAS action set to save CAS in-memory table to ADLS
- CASDATA – Input CAS table
- CASOUT – ORC output file saved to ADLS

```
1 %studio_hide_wrapper;
82 proc casutil;
NOTE: The UUID '09107ec2-6ce1-7242-905d-59e03ec9e015' is connected using
session CASAUTO.
83 load data=sashelp.class casout="class" outcaslib="adls" replace;
NOTE: SASHELP.CLASS was successfully added to the "adls" caslib as "CLASS".
84
85 save casdata="class" casout="class.orc" replace;
NOTE: Cloud Analytic Services saved the file class.orc in caslib adls.
NOTE: The Cloud Analytic Services server processed the request in 0.954582
seconds.
86 quit;
NOTE: PROCEDURE CASUTIL used (Total process time):
real time 1.24 seconds
cpu time 0.06 seconds
87
88 %studio_hide_wrapper;
```

SAS AND AZURE HDINSIGHT HADOOP DATA LAKE

Traditionally, the Hadoop platform has been deployed to solve an organization’s Data Lake business needs on-premise. With more and more incoming data along with deploying analytical applications such as SAS Viya in the cloud, organizations are moving some of their Data Lake needs to the cloud along with the data. Azure HDInsight is one such Hadoop service that provides a cloud native Hadoop platform along with elasticity to scale up or down the Data Lake. The SAS platform can leverage Azure HDInsight Hadoop service to access, load, or save data using the SAS Access to Hadoop engine.

LIBNAME STATEMENT TO CONNECT AND ACCESS DATA

```
option set = SAS_HADOOP_CONFIG_PATH = "/opt/hdinsight/conf";
option set = SAS_HADOOP_JAR_PATH = "/opt/hdinsight/jars";

options sastrace = ',,,sd' sastraceloc = saslog no$stsuffix fullstimer;

libname hdilib HADOOP
uri='jdbc:hive2://sgfdemo.azurehdinsight.net:443/default;ssl=true?hive.server
2.transport.mode=http;hive.server2.thrift.http.path=hive2'
server='sgfdemo.azurehdinsight.net' user="admin" pw="demopassword"
schema="default";
```

As a one-time configuration setup, before creating a SAS library reference and connecting to an Azure HDInsight cluster, the Hadoop administrator would execute a HadoopTracer Python script available from SAS to extract all the necessary JAR and config files. The SAS administrator can further use these extracted files to connect to an Azure HDInsight cluster by setting two option variables. This step is performed regardless of the type of Hadoop distribution or cloud platform.

Some of the important parameters:

- SAS_HADOOP_CONFIG_PATH - Specifies the directory path for the Azure HDInsight cluster configuration files.
- SAS_HADOOP_JAR_PATH - Specifies the directory path for the Azure HDInsight JAR files.
- URI – Azure HDInsight JDBC URI to connect to Hive server 2. Once a similar JDBC URI is retrieved from Azure HDInsight documentation, just modify the HDInsight server name.

```
1 %studio_hide_wrapper;
83
84 option set = SAS_HADOOP_CONFIG_PATH = "/opt/hdinsight/conf";
85 option set = SAS_HADOOP_JAR_PATH = "/opt/hdinsight/jars";
86
87
88 options sastrace = ',,,sd' sastraceloc = saslog no$stsuffix fullstimer;
89 libname hdilib HADOOP
90
uri='jdbc:hive2://sgfdemo.azurehdinsight.net:443/default;ssl=true?hive.serv
er2.transport.mode=http;
90 ! hive.server2.thrift.http.path=hive2'
91 server='sgfdemo.azurehdinsight.net' user="admin" pw=XXXXXXXXXXXXXXXXXXX
schema="default" ;
NOTE: Libref HDILIB was successfully assigned as follows:
Engine: HADOOP
Physical Name:
jdbc:hive2://sgfdemo.azurehdinsight.net:443/default;ssl=true?hive.server2.t
ransport.mode=http;hive.server2.thrift.http.path=hive2
```

The JDBC URI contains some of the necessary parameters that are enabled and assigned values by default. SSL is set to true by default, and REST transport mode is set to HTTP. Data can be loaded from an Azure HDInsight cluster to the SAS platform or saved to the cloud. SAS PROCs such as Proc Append, Sort, Summary, Means, Rank, Freq, and Transpose are supported on Azure HDInsight cluster. Furthermore, the DATA step and PROC SQL data preparation with bulk load are handled to save data efficient on the cloud.

SAS AND AZURE SQL DATABASE

Organizations can connect and access data from an Azure SQL Database using SAS/Access to Microsoft SQL Server from SAS. All the features from SAS/Access to SQL Server running on-premise would be available in the cloud as well. Running the SQL database in the cloud gives organizations elasticity to scale the database. Let's look at code samples to connect and access data from using SAS library and CAS library.

LIBNAME STATEMENT TO CONNECT TO AZURE SQL DB

Azure SQL Database connection information typically specified in the odbc.ini file on the SAS servers, along with a data source name (DSN). Specify the DSN name in the libname connection parameter.

```
libname mydblib sqlsvr noprompt="uid=sgfdemo;
pwd=demopassword; dsn=Azure SQL Database;" stringdates=yes;
```

The SQLSVR connection parameter enables the LIBNAME statement to use SAS/Access to SQL Server to connect to Azure SQL Database.

```
1 %studio_hide_wrapper;
SQLSRV: AUTOCOMMIT is NO for connection 0
82 libname mydblib sqlsvr noprompt=XXXXXXXXX
83 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX stringdates=yes;
NOTE: Libref MYDBLIB was successfully assigned as follows:
Engine: SQLSVR
Physical Name: Azure SQL Database
84
85 %studio_hide_wrapper;
```

Load Data from SAS to Azure SQL DB

We can use the same SAS library, mydblib, created above to load the SAS data set to an Azure SQL Database.

```
data mydblib.cars;
set sashelp.cars;
run;
```

```
1 %studio_hide_wrapper;
82 data mydblib.cars;
83 set sashelp.cars;
84 run;
SQLSRV_5: Prepared: on connection 1
SELECT * FROM cars WHERE 0=1
Summary Statistics for SQLSVR are:
Total SQL prepare seconds were: 0.000185
Total seconds used by the SQLSVR ACCESS engine were 0.018767
SQLSRV: AUTOCOMMIT is NO for connection 2
NOTE: SAS variable labels, formats, and lengths are not written to DBMS
tables.
SQLSRV_6: Executed: on connection 2
CREATE TABLE cars ( Make varchar(13), Model varchar(40), Type varchar(8),
Origin varchar(6), DriveTrain varchar(5), MSRP
money, Invoice money, EngineSize float, Cylinders float, Horsepower float,
MPG_City float, MPG_Highway float, Weight float,
Wheelbase float, Length float)
SQLSRV: COMMIT performed on connection 2.
SQLSRV_7: Prepared: on connection 2
INSERT INTO cars ( Make , Model , Type , Origin , DriveTrain , MSRP ,
Invoice , EngineSize , Cylinders , Horsepower , MPG_City ,
MPG_Highway , Weight , Wheelbase , Length ) VALUES ( ? , ? , ? , ? , ? , ?
, ? , ? , ? , ? , ? , ? , ? , ? , ? )
…
.
.
SQLSRV_435: Executed: on connection 2
Prepared statement SQLSRV_7
NOTE: There were 428 observations read from the data set SASHELP.CARS.
SQLSRV: COMMIT performed on connection 2.
NOTE: The data set MYDBLIB.cars has 428 observations and 15 variables.
SQLSRV: COMMIT performed on connection 2.
Summary Statistics for SQLSVR are:
Total SQL execution seconds were: 5.032152
Total SQL prepare seconds were: 0.000536
Total seconds used by the SQLSVR ACCESS engine were 5.340702
SQLSRV: COMMIT performed on connection 2.
NOTE: DATA statement used (Total process time):
real time 5.89 seconds
user cpu time 1.34 seconds
system cpu time 0.73 seconds
memory 903.68k
OS Memory 38416.00k
Timestamp 02/19/2020 05:01:59 AM
85
86 %studio_hide_wrapper;
```

From the log we can see that the SQLSVR Access engine is used to process and load the data to the Azure SQL Database. Before and after the DATA step execution, summary statistics are performed to help provide additional SQL execution statistics.

SAS VIYA CAS LIBRARY TO CONNECT TO AZURE SQL DB

Similarly, SAS/Access to SQL Server can be used to connect to an Azure SQL Database using a data connector. This approach is helpful to organizations with streamlined data movement when SAS Viya is running in the cloud to load and save data between Azure SQL Database and SAS Viya in-memory servers.

```
cas casauto;
caslib azsqldb desc='Microsoft SQL Server Caslib'
dataSource=(srctype='sqlserver'
username='sas'
password='demopassword'
sqlserver_dsn="Azure SQL Database"
catalog='*');
```

Azure SQL Database connection information has to be added to the odbc.ini files in all the CAS cluster nodes. This facilitates optimized data movement between multiple CAS worker nodes and Azure SQL Database.

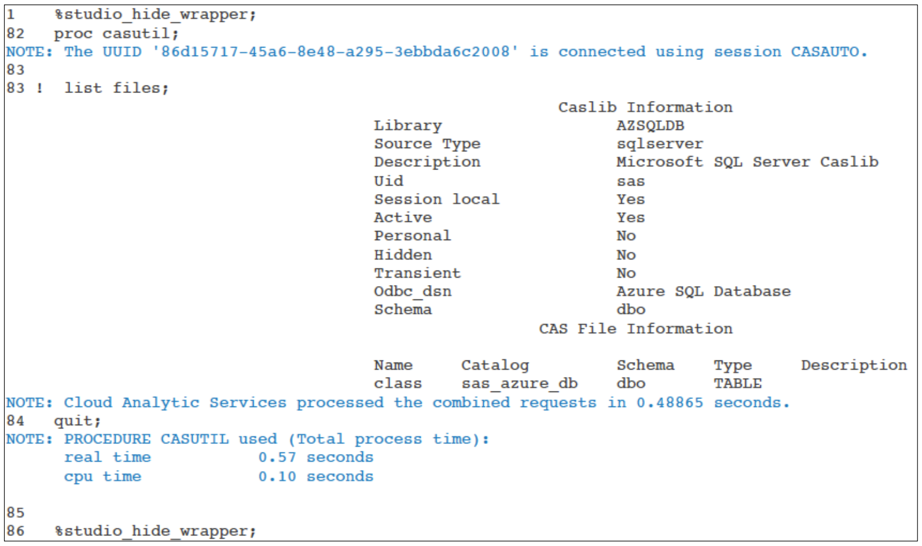
- SRCTYPE – Source type SQLSERVER is the key parameter to connect to an Azure SQL Database using SAS Viya Data Connector to Microsoft SQL Server.

```
1 %studio_hide_wrapper;
82 cas casauto;
WARNING: A session with the name CASAUTO already exists.
83
84 caslib azsqldb desc='Microsoft SQL Server Caslib'
85 dataSource=(srctype='sqlserver'
86 username='sas'
87 password=XXXXXXXXXX
88 sqlserver_dsn="Azure SQL Database"
89 catalog='*')
90 ;
NOTE: 'AZSQLDB' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'AZSQLDB'.
NOTE: Action to ADD caslib AZSQLDB completed for session CASAUTO.
91
92 %studio_hide_wrapper;
```

List Table Files from Azure SQL DB

Before loading data to the cloud database, we can look at the contents of the database using the LIST FILES action from the PROC CASUTIL CAS procedure.

```
proc casutil;
      list files;
quit;
```



```
1    %studio_hide_wrapper;
82   proc casutil;
NOTE: The UUID '86d15717-45a6-8e48-a295-3ebbda6c2008' is connected using session CASAUTO.
83
83 !  list files;
                                                      Caslib Information
                                      Library                  AZSQLDB
                                      Source Type              sqlserver
                                      Description              Microsoft SQL Server Caslib
                                      Uid                      sas
                                      Session local            Yes
                                      Active                   Yes
                                      Personal                 No
                                      Hidden                   No
                                      Transient                No
                                      Odbc_dsn                 Azure SQL Database
                                      Schema                   dbo
                                                    CAS File Information

                                      Name     Catalog         Schema    Type      Description
                                      class    sas_azure_db    dbo       TABLE
NOTE: Cloud Analytic Services processed the combined requests in 0.48865 seconds.
84   quit;
NOTE: PROCEDURE CASUTIL used (Total process time):
      real time           0.57 seconds
      cpu time            0.10 seconds

85
86   %studio_hide_wrapper;
```

From the log we can see that there is one table named class inside the Azure SQL Database sas_azure_db.

Load and Save SAS Dataset to Azure SQL DB

With one table in the Azure SQL Database, let's load another table from a SAS Viya in-memory cluster to the Azure SQL Database. In the following code sample, first we are loading a SAS data set named cars to a CAS in-memory cluster. Then the distributed CAS in-memory table is saved to the Azure SQL Database.

```
proc casutil;
      load data=sashelp.cars outcaslib="azsqldb"
      casout="cars" replace;
      save casdata="cars" casout="cars" replace;
quit;
```

One parameter to mention here, OUTCASLIB, contains the name of the CAS library that we created earlier to access the Azure SQL Database. Information about other parameters has been furnished in the ADLS section above.

```
1 %studio_hide_wrapper;
82 proc casutil;
NOTE: The UUID '86d15717-45a6-8e48-a295-3ebbda6c2008' is connected
using session CASAUTO.
83 load data=sashelp.cars outcaslib="azsqldb"
84 casout="cars" replace;
NOTE: SASHELP.CARS was successfully added to the "AZSQLDB" caslib as
"CARS".
85 save casdata="cars" casout="cars" replace;
NOTE: Performing serial SaveTable action using SAS Data Connector to
SQL Server.
NOTE: Cloud Analytic Services saved the file cars in caslib AZSQLDB.
NOTE: The Cloud Analytic Services server processed the request in
10.168781 seconds.
86
87 quit;
NOTE: PROCEDURE CASUTIL used (Total process time):
real time 10.45 seconds
cpu time 0.06 seconds
88
89 %studio hide wrapper;
```

From the log SAS Viya uses SAS Data Connector to SQL Server to save the CAS in-memory table named cars to the Azure SQL Database. The Save CAS action, which is part of the TABLES CAS action set has been called to perform a serial save in this case.

SAS VIYA AND AZURE MYSQL DATABASE

MySQL is another widely used relational database management system that is accessible from SAS Viya as an Azure MySQL database with elastic server configurations to support various data volumes. SAS Data Connector to MySQL can be used to access, load, and save data between CAS in-memory cluster nodes and an Azure MySQL Database.

Other flavors of MySQL databases such as Azure MariaDB Database can also be accessed using SAS Data Connector to MySQL. Steps are exactly the same as MySQL, just modify the server host to Azure MariaDB, along with the appropriate credentials.

```
cas casauto;
caslib azurems desc='MySQL Caslib'
dataSource=(srctype='mysql'
host='sgfdemo.mysql.database.azure.com'
username='sas@sgfdemo'
password='demopassword'
database="demo")
/* global */
;
```

The MySQL client must be installed on all the CAS cluster nodes. The MySQL client library path must be added to the CAS configuration settings file on the nodes. Refer to MYSQL Data Connector in the SAS Cloud Analytics Services 3.5: User's Guide for additional information.

SRCTYPE – Source type MYSQL is the key parameter to connect to an Azure MySQL Database using SAS/Access to MySQL.

```
1 %studio_hide_wrapper;
82 caslib azurems desc='MySQL Caslib'
83 dataSource=(srctype='mysql'
84 host='sgfdemo.mysql.database.azure.com'
85 username='sas@sgfdemo'
86 password=XXXXXXXXXX
87 database="demo")
88
89 /* global */
90 ;
NOTE: 'AZUREMS' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'AZUREMS'.
NOTE: Action to ADD caslib AZUREMS completed for session CASAUTO.
91
92 %studio_hide_wrapper;
```

LOAD AND SAVE SAS DATASET TO AZURE MYSQL DATABASE

```
proc casutil;
load data=sashelp.cars outcaslib="azurems"
casout="cars" replace;
save casdata="cars" casout="cars" replace;
list files;
quit;
```

```
1 %studio_hide_wrapper;
82 proc casutil;
NOTE: The UUID '1a39b02d-9a7c-c342-8491-8b9f2a8c40f8' is connected
using session CASAUTO.
83 load data=sashelp.cars outcaslib="azurems"
84 casout="cars" replace;
NOTE: SASHELP.CARS was successfully added to the "AZUREMS" caslib as
"CARS".
85 save casdata="cars" casout="cars" replace;
NOTE: Performing serial SaveTable action using SAS Data Connector to
MySQL.
NOTE: Cloud Analytic Services saved the file cars in caslib AZUREMS.
NOTE: The Cloud Analytic Services server processed the request in
67.365248 seconds.
```

From the log SAS Viya uses SAS Data Connector to MySQL to save a CAS in-memory table named cars to an Azure MySQL Database. The Save CAS action, which is part of the TABLES CAS action set has been called to perform a serial save in this case.

SAS VIYA AND AZURE POSTGRESQL DATABASE

The SAS platform can access data from another popular relational database management system(RDBMS), PostgreSQL. Azure PostgreSQL Database is a Database as a Service (DBaaS) offering to support data movement as needed through various server configurations based on data volume and demand.

SAS LIBRARY TO LOAD SAS DATASET

```
libname azurepg postgres server="sgfdemo.postgres.database.azure.com"
port=5432
user=sas@sgfdemo password='demopassword' database=postgres;

/*Load a SAS dataset to Azure PostgreSQL database*/
data azurepg.cars;
set sashelp.cars;
run;
```

```
1 %studio_hide_wrapper;
82 libname azurepg postgres
server="sgfdemo.postgres.database.azure.com" port=5432
83 user=sas@sgfdemo password=XXXXXXXXXX database=postgres;
NOTE: Libref AZUREPG was successfully assigned as follows:
Engine: POSTGRES
Physical Name: sgfdemo.postgres.database.azure.com
84
85 data azurepg.cars;
86 set sashelp.cars;
87 run;
NOTE: SAS variable labels, formats, and lengths are not written to
DBMS tables.
NOTE: There were 428 observations read from the data set
SASHELP.CARS.
NOTE: The data set AZUREPG.CARS has 428 observations and 15
variables.
NOTE: DATA statement used (Total process time):
real time 7.52 seconds
cpu time 0.27 seconds
88
```

The LIBNAME statement uses the POSTGRES keyword to utilize SAS/Access Engine to PostgreSQL to connect to an Azure PostgreSQL Database. In this example, a SAS dataset named cars is loaded to the destination cloud PostgreSQL database.

CAS LIBRARY TO AZURE POSTGRESQL

```
cas casauto;

caslib postgrescaslib desc='PostgreSQL Caslib'
dataSource=(srctype='postgres'
server='sgfdemo.postgres.database.azure.com'
username='sas@sgfdemo'
password='demopassword'
database="postgres");
```

```
1 %studio_hide_wrapper;
82 caslib postgrescaslib desc='PostgreSQL Caslib'
83 dataSource=(srctype='postgres'
84 server='sgfdemo.postgres.database.azure.com'
85 username='sas@sgfdemo'
86 password=XXXXXXXXXX
87 database="postgres");
NOTE: 'POSTGRESCASLIB' is now the active caslib.
NOTE: Cloud Analytic Services added the caslib 'POSTGRESCASLIB'.
NOTE: Action to ADD caslib POSTGRESCASLIB completed for session
CASAUTO.
88
89 %studio_hide_wrapper;
```

The CAS library postgrescaslib is created by connecting to the destination Azure PostgreSQL Database. Source type POSTGRES is the keyword that utilizes SAS Data Connector to PostgreSQL.

Load and Save SAS Dataset to Azure PostgreSQL DB

```
proc casutil;
```

```
load data=sashelp.class outcaslib="postgrescaslib"
casout="class" replace;
save casdata="class" casout="class" replace;
list files;
quit;
```

```
1 %studio_hide_wrapper;
82 proc casutil;
NOTE: The UUID '1a39b02d-9a7c-c342-8491-8b9f2a8c40f8' is connected using
session CASAUTO.
83 load data=sashelp.class outcaslib="postgrescaslib"
84 casout="class" replace;
NOTE: SASHELP.CLASS was successfully added to the "POSTGRESCASLIB" caslib as
"CLASS".
85 save casdata="class" casout="class" replace;
NOTE: Performing serial SaveTable action using SAS Data Connector to
PostgreSQL.
NOTE: Cloud Analytic Services saved the file class in caslib POSTGRESCASLIB.
NOTE: The Cloud Analytic Services server processed the request in 3.696087
seconds.
Name                          Catalog            Schema             Type

TRANSACTIONS_VENDORS          POSTGRESCASLIB     public             TABLE
cars                          POSTGRESCASLIB     public             TABLE
class                         POSTGRESCASLIB     public             TABLE
pg_buffercache                POSTGRESCASLIB     public             VIEW
pg_stat_statements            POSTGRESCASLIB     public             VIEW
NOTE: Cloud Analytic Services processed the combined requests in 0.952014
seconds.
87 quit;
NOTE: PROCEDURE CASUTIL used (Total process time):
real time 5.02 seconds
cpu time 0.13 seconds
88
89 %studio_hide_wrapper;
```

From the log SAS Viya uses SAS Data Connector to PostgreSQL to save the CAS in-memory table named cars to the Azure PostgreSQL Database. The Save CAS action, which is part of the TABLES CAS action set has been called to perform a serial save in this case.
The LIST FILES action in the CASUTIL procedure show the cars table has been saved to the database in the cloud.

CONCLUSION

You have looked at mostly data processing SAS code samples in this paper, however once the data is loaded to CAS in-memory servers for distributed parallel processing or SAS compute server, advanced analytics and model methods can be applied to get relevant insight out of the prepared data. Whether the final decision is to keep data on-premise or in the cloud or a hybrid approach, SAS Data Connector supports virtually any data sources to access, load, move, model data efficiently, or navigate through the analytic life cycle. It is important to think about the use case for your database and the type of data you plan to store before you select an Azure database service.

REFERENCES

SAS Viya 3.5, 2019. "SAS Access to ADLS." Available at https://go.documentation.sas.com/?docsetId=casref&docsetTarget=n1ogaeli0qbctqn1e3fx8gz70lkq.htm&docsetVersion=3.5&locale=en.

SAS Viya 3.5, 2019. "Support for Databases & Azure HDInsight" Available at https://support.sas.com/en/documentation/third-party-software-reference/viya/35/support-for-databases.html.

SAS 9.4, 2019. "Support for Azure HDInsight" Available at https://support.sas.com/en/documentation/third-party-software-reference/9-4/support-for-hadoop.html

SAS 9.4, 2019. "Support for Cloud and Database Variants" Available at https://support.sas.com/en/documentation/third-party-software-reference/9-4/support-for-database.html#hadoop

ACKNOWLEDGMENTS

I thank Jeff Stander for his support in accessing Azure Data Lake Storage from SAS platform.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kumar Thangamuthu
SAS Institute Inc.
kumar.thangamuthu@sas.com
www.sas.com