Paper SAS5200-2016

It's raining data! Harnessing the Cloud with Amazon Redshift and SAS/ACCESS®

Chris DeHart and Jeff Bailey, SAS Institute Inc., Cary, NC

ABSTRACT

Every day, companies all over the world are moving their data into the Cloud. While there are many options available, much of this data will wind up in Amazon Redshift. As a SAS® user, you are probably wondering, "What is the best way to access this data using SAS?" This paper discusses the many ways that you can use SAS/ACCESS® to get to Amazon Redshift. We compare and contrast the various approaches and help you decide which is best for you. Topics that are discussed are building a connection, moving data into Amazon Redshift, and implementing performance best practices.

INTRODUCTION

These days the cloud and cloud computing are becoming popular words in the computer industry worldwide. The new architecture brings better agility, flexibility, scalability, and cost savings than traditional services. Not only do more companies plan to adopt this new technology in their IT architecture, many businesses now depend on it. Amazon Redshift provides users with a new option for data warehousing. SAS users can run statistics, analysis, and prediction work in the data warehouse. Indeed, you can access Amazon Redshift in many ways, such as through the PostgreSQL command utility or other GUI tools. This paper introduces SAS/ACCESS Interface to Amazon Redshift and describes the capabilities of the engine and its optimal use with Amazon Redshift.

This paper covers these topics:

- main features of AWS, Amazon Redshift, and SAS/ACCESS Interface to Amazon Redshift
- building a connection to Amazon Redshift
- moving data into Amazon Redshift
- performance comparisons

BACKGROUND

CLOUD COMPUTING PLATFORM

Amazon Web Services (AWS) is an IT infrastructure platform that originally offered services to businesses by way of web services. It is now commonly known as cloud computing and provides highly reliable, scalable, low-cost infrastructure.

Users might want to access the services in the cloud via AWS from the SAS environment. Doing this requires that you have an available AWS account. To learn how to create an AWS account, get help, and find more detailed information, go to http://aws.amazon.com.

CLOUD DATA WAREHOUSE

Amazon Redshift is a fast, powerful, fully managed, petabyte-scale data warehouse service in AWS. It offers you fast query performance, and it makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. One of the most significant advantages is that it has a similar SQL interface to PostgreSQL. Users can access the Amazon Redshift cluster through common PostgreSQL client tools, such as standard ODBC and JDBC drivers and the PSQL utility.

Unlike most other relational database management systems (RDBMSs), Amazon Redshift uses the column-oriented storage principle. It lets users handle analysis workloads on huge-scale data sets. Another advantage is its query operations for data warehousing. It also uses a data distribution mode whereby stored rows can be distributed to node slices according to a distribution key that is defined for a table when data is loaded into the database service. An appropriate distribution key can allow the database to load data and execute queries in parallel and, therefore, be more efficient.

To learn more about launching an Amazon Redshift cluster, go to http://docs.aws.amazon.com/redshift/latest/gsg/rsgsg-launch-sample-cluster.html.

SAS/ACCESS INTERFACE TO AMAZON REDSHIFT

SAS/ACCESS to Amazon Redshift is the latest SAS/ACCESS LIBNAME engine, and it is tailor-made for accessing Amazon Redshift. The engine is derived from the ODBC and PostgreSQL SAS/ACCESS engines, which gives the engine the best of the ODBC and PostgreSQL functionalities. SAS/ACCESS to Amazon Redshift combines the general interface of the ODBC engine along with better SQL support afforded by the PostgreSQL engine. The engine also comes bundled with a SAS branded version of the Progress DataDirect Amazon Redshift ODBC driver.

While SAS/ACCESS to Amazon Redshift borrows some technology from SAS/ACCESS Interface to PostgreSQL, we do not recommend using SAS/ACCESS to PostgreSQL to connect to Amazon Redshift. Amazon Redshift is based on a modified version of PostgreSQL 8.0.2 which has some important differences compared with the industry standard PostgreSQL 9.x. You can learn more about these differences in the Amazon Redshift documentation:

http://docs.aws.amazon.com/redshift/latest/dg/c_redshift-and-postgres-sql.html

WHY SAS/ACCESS TO AMAZON REDSHIFT INSTEAD OF SAS/ACCESS TO ODBC?

One of the biggest questions you might be asking is why you should choose SAS/ACCESS to Amazon Redshift instead of SAS/ACCESS Interface to ODBC? In the following sections we illustrate several key areas where SAS/ACCESS to Amazon Redshift can save you time and money.

SIMPLIFIED INSTALLATION AND CONFIGURATION

By including the DataDirect Amazon Redshift driver, SAS/ACCESS to Amazon Redshift dramatically simplifies the installation and configuration tasks normally associated with SAS/ACCESS to ODBC. This is especially true on UNIX platforms, where in some cases not only do you have to download the associated ODBC driver and a compatible ODBC Driver Manager, but you also must build the ODBC Driver Manager from a source distribution. The process of building an ODBC Driver Manager can be fraught with complications, and even at the end of the process you still cannot be sure the Driver Manager will work with SAS/ACCESS to ODBC and the driver that you have installed until you try it. With the bundled DataDirect Amazon Redshift driver you get not only the necessary ODBC driver to connect to Amazon Redshift but also the fully compatible ODBC Driver Manager in one complete package.

SIMPLIFIED CONNECTION METHOD

The SAS/ACCESS to Amazon Redshift engine has improved connection processing that removes the need to setup data sources ahead of time in an ODBC configuration file like odbc.ini. The LIBNAME engine syntax for SAS/ACCESS to Amazon Redshift allows you to specify the server and database of the Amazon Redshift cluster directly on the LIBNAME statement itself, as shown in this example:

```
LIBNAME myredshift sasiorst SERVER="myredshift.redshift.amazonaws.com"
   DB=mydb USER=myuser PWD=mypwd;
```

The above syntax is all that is required to connect to the Amazon Redshift database from the client side. Whereas with SAS/ACCESS to ODBC you might require an administrator to modify the odbc.ini for you (and wait until they had time to do so), with SAS/ACCESS to Amazon Redshift you can specify all the parameters in SAS itself and start running much faster.

BETTER SQL FUNCTION SUPPORT

SAS/ACCESS to Amazon Redshift also has better support for passing down SQL functions when using PROC SQL to connect to Amazon Redshift. Due to the generic nature of ODBC, the SAS/ACCESS to ODBC engine can only pass down certain functions that are well defined in the ODBC specification and that are also flagged as supported by the underlying ODBC driver. Because SAS/ACCESS to Amazon Redshift is targeted only at Amazon Redshift, the engine can pass down a wider array of SQL functions. As far as actual numbers go, in our internal tests we noted that when connected via SAS/ACCESS to ODBC to Amazon Redshift we saw a total of 26 SQL functions that were available to be passed down to the database. In SAS/ACCESS to Amazon Redshift, that number increases to 44 SQL functions! The extra functions that are passed down include aggregate functions like STDEV and VAR for standard deviation and variance, respectively, as well as date functions like HOUR, MINUTE, and SECOND and YEAR, MONTH, and DAY. By passing down more SQL functions, your code can execute much faster in Amazon Redshift when you use those functions in PROC SQL.

BETTER WRITE PERFORMANCE WITH DATADIRECT DRIVER

By including the DataDirect Amazon Redshift ODBC driver, SAS/ACCESS to Amazon Redshift is also able to provide substantial increases in write performance over the Amazon-supplied ODBC driver and SAS/ACCESS to ODBC. We will delve deeper into write performance in the next section.

LOADING DATA TO AMAZON REDSHIFT

Because Amazon Redshift is optimized for query performance, writing to the database can be a resource-intensive process that can seem slower than the equivalent process on other database systems. Fortunately, there are a couple of methods you can use to help speed up the process of writing data to Amazon Redshift: ODBC driver bulk processing and Amazon Simple Storage Service (Amazon S3) loads.

DATADIRECT DRIVER BULK PROCESSING

One of the great features of the DataDirect Amazon Redshift ODBC driver is the driver's optimization of parameter arrays. What are parameter arrays you say? Parameter arrays are just a fancy way to describe how an ODBC application like SAS/ACCESS to Amazon Redshift sends multiple rows of data to insert into table at one time. In ODBC-based SAS/ACCESS engines, parameter arrays are activated by the INSERTBUFF data set option. In SAS/ACCESS to Amazon Redshift, the INSERTBUFF option is set automatically by the engine. What does that mean to you? You see better performance out of the box.

In one scenario we tested, SAS/ACCESS to ODBC (without any options) loaded our test data set in 28 minutes. That same data set in SAS/ACCESS to Amazon Redshift (again, without any options) loaded the data in 54 seconds! The results here were using the DataDirect Amazon Redshift ODBC driver to upload a 35MB SAS data set. If we switch to the Amazon-supplied ODBC driver, the load of the same data via SAS/ACCESS to ODBC took 34 minutes. If we add the same INSERTBUFF value that SAS/ACCESS to Amazon Redshift uses to SAS/ACCESS to ODBC to make this more of an apples-to-apples comparison, the load using the Amazon-supplied driver still takes 28 minutes. This simple example shows the power of the DataDirect driver's parameter array optimization. The chart in Figure 1 groups all these numbers together to show you the difference in performance.

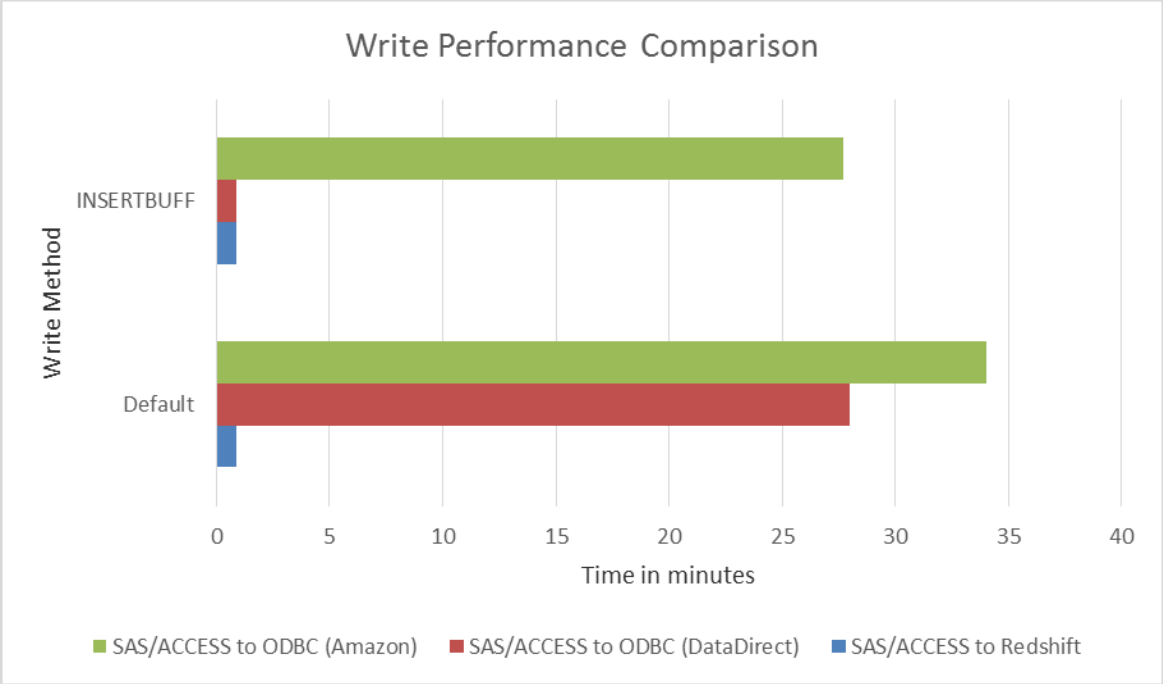

Figure 1. Write Performance Comparison between SAS/ACCESS Engines

During our testing, we found one other performance tip when using SAS/ACCESS to Amazon Redshift to insert data: use the AUTOCOMMIT=NO LIBNAME option if you know your data will insert without error. Executing the INSERT statements inside a transaction (with AUTOCOMMIT=NO) allows the statements to execute more quickly, but if an error occurs then all the rows inserted in that transaction are lost. Amazon Redshift automatically ends any transaction in which an error occurs, so to provide maximum data integrity SAS/ACCESS to Amazon Redshift defaults to committing every block of rows.

COPY COMMAND WITH AMAZON S3

For large amounts of data, Amazon recommends using the COPY command instead of relying on the standard INSERT method. Although SAS/ACCESS to Amazon Redshift doesn't currently support bulk loading via the Amazon S3 command automatically, there are steps you can take inside SAS to accomplish an Amazon S3 load to Amazon Redshift. In particular, if the data you wish to load to Amazon Redshift is in the form of a SAS data set, you can use the following steps to execute the Amazon S3 load:

1. Export a data set to the DLM format.
2. Upload the DLM file to Amazon S3 by calling the AWS command line interface (CLI) through the SAS X command.
3. Execute the COPY command by using a PROC SQL pass-through connection.

Please note that this example uses some advanced techniques, and all users might not have the necessary access to the information required. In particular, installing and using the AWS CLI and obtaining AWS access keys might prove difficult for some users.

We'll take a closer look at each of these three steps.

Step 1. Export the Data Set Using PROC EXPORT:

```
proc export data=sashelp.zipcode outfile="/tmp/zipcode.dlm"
   DBMS=DLM replace;
   delimiter='07'x;
   putnames=no;
run;
```

Note that we're overriding the default delimiter in PROC EXPORT to separate our column data with the unprintable ASCII character '0x07' in case our data contains embedded commas.

This example uses the Zipcode data set in the SASHELP library, which contains 21 columns and just over 41,000 rows.

Step 2: Upload the Data File to Amazon S3

To upload our new DLM file containing the Zipcode data to Amazon S3, we're going to use the AWS CLI. You can download this utility for free at Amazon's website: https://aws.amazon.com/cli/. The website also explains the necessary steps to configure the AWS CLI with your AWS access keys. After installing and configuring the AWS CLI, we can now upload our data file to Amazon S3 so that the data is available to the COPY command:

```
x "aws s3 cp /tmp/zipcode.dlm s3://mybucket/tmp/zipcode.dlm";
```

This code uses the SAS X command to allow you to execute the AWS CLI command from within SAS without having to open a separate shell. The AWS CLI `s3` command allows you to interact with Amazon S3 and the `cp` command uploads our data file into Amazon S3 into a bucket named mybucket. You can also use the AWS CLI `s3` command to create a bucket if you don't already have one.

Step 3: Execute the COPY Command

Now that we have our data file in Amazon S3, we can tell Amazon Redshift to perform a COPY command to load that data into our target Amazon Redshift table:

```
libname myred sasiorst server="myredshift.redshift.amazonaws.com" db=mydb
      user=myuser pwd=mypwd;
proc sql;
   connect using myred;
   execute (COPY ziptable FROM 's3://mybucket/tmp/zipcode.dlm' credentials
      'aws_access_key_id=<access key>;aws_secret_access_key=<secret key>'
      region 'us-east-1' DELIMITER '\007') by myred;
quit;
```

First, we create a connection to Amazon Redshift using the LIBNAME statement to the SAS/ACCESS to Amazon Redshift engine. In PROC SQL, we specify the CONNECT USING statement to tell PROC SQL to use the existing LIBNAME connection instead of creating a separate connection for the pass-through statement. Next, we can call the Amazon Redshift COPY command by using the EXECUTE statement along with the BY myred identifier. Because we are re-using the LIBNAME connection for our PROC SQL statement, there is no need to call the DISCONNECT statement as you normally would for a connection established with the CONNECT TO statement.

Let's go over some of the items in that COPY command. In our example, we're going to be inserting our data into the Amazon Redshift table Ziptable, and this table is the target of the COPY command. Next, we have the Amazon S3 specification point to the data file that we uploaded to Amazon S3 in step 2 above. The CREDENTIALS clause specifies the same AWS access keys that you used earlier to configure the AWS CLI command. At the end, we've specified the delimiter that we used in PROC EXPORT so that the COPY command knows how our data is separated. Note that SAS syntax uses a hexadecimal value for DELIMITER, while the COPY command uses an octal representation. It happens that for the character we've chosen, the representation in hexadecimal ('07'x) is the same number as it is in octal ('\007').

After the COPY command completes, you'll see a WARNING in the SAS log similar to Output 1. Warning in SAS Log after Successful Amazon S3 Load

```
WARNING: During execute: [SAS][ODBC Redshift Wire Protocol
driver][Redshift]INFO: Load into table 'ziptable' completed, 41232
record(s) loaded successfully.(File
/home/awsrsqa/padb/src/pg/src/backend/commands/commands_copy.c; Line 2213;
Routine DoCopy; )
```

Output 1. Warning in SAS Log after Successful Amazon S3 Load

This message indicates that the Amazon S3 load was successful, and it reports how many rows were loaded into Ziptable.

After all those steps to get the data into Amazon Redshift, how long did the entire process take? It took 12.5 seconds! That's right, even with all the manual steps to export the data, use the AWS CLI command, and finally execute the COPY command, the combined time was only 12.5 seconds. For a comparison, we'll include our chart from earlier (Figure 1) and add the numbers from using Amazon S3 load to give you Figure 2.

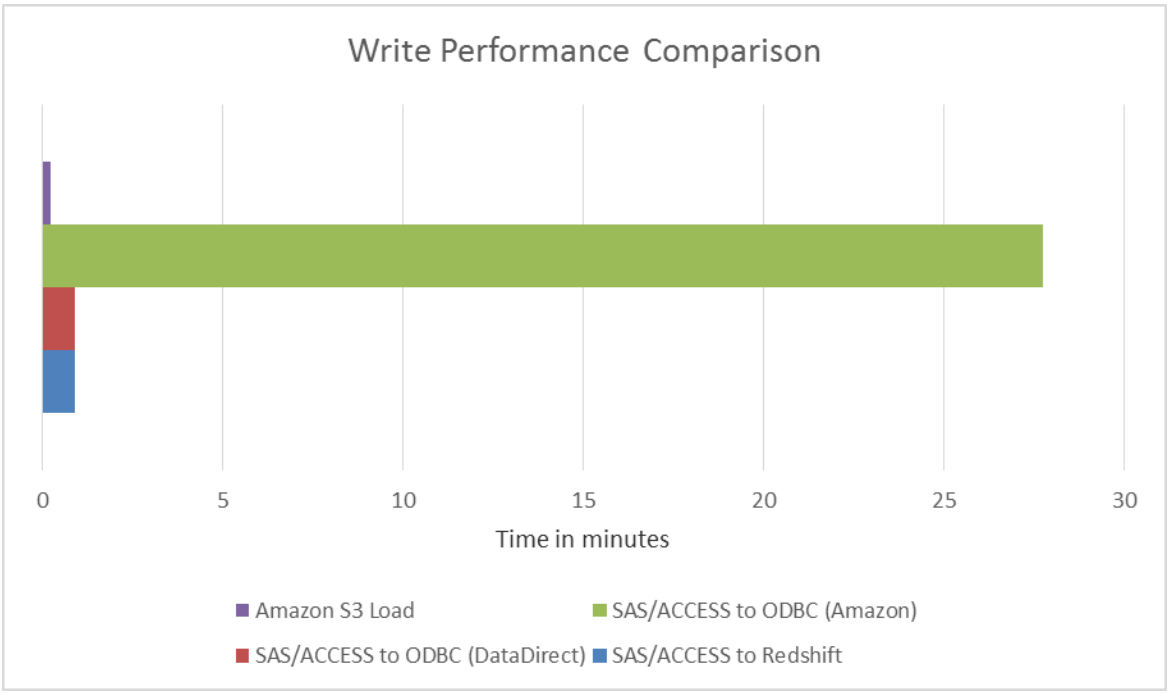

Figure 2. Performance Comparison Including Amazon S3 Load

This chart uses the best numbers from each of the write methods we discussed earlier along with the number for the Amazon S3 load scenario. As you can see, the difference is quite dramatic. If you are willing to go through a few extra steps, you can cut your data loading time considerably.

There is a great tutorial on Amazon's website that walks you through a complete data loading scenario involving Amazon S3: http://docs.aws.amazon.com/redshift/latest/dg/tutorial-loading-data.html

Any commands that need to be executed against Amazon Redshift as part of this tutorial can be done through the explicit pass-through connection, as seen above in our simple COPY example.

FUTURE FEATURES

BULK LOADING VIA AMAZON S3

In a future release of SAS/ACCESS to Amazon Redshift, we will include the capability to upload data directly into Amazon S3 storage and execute a COPY command. This functionality will be triggered by the use of a BULKLOAD=YES data set option, and it can be used from both a DATA step and in procedures like PROC SQL that insert data. This new functionality will eliminate much of the hassle described in the previous section, and instead it will perform all three steps of an Amazon S3 load internally in SAS/ACCESS to Amazon Redshift. It will also eliminate the need to install the AWS CLI utility (or a similar tool) to upload the data file into Amazon S3. This new process will allow you to take advantage of the significant performance gains of using Amazon S3 loads while simplifying the steps needed to accomplish the load.

SAS IN-DATABASE PROCEDURE PUSHDOWN

Future releases of SAS/ACCESS to Amazon Redshift will also support the in-database processing of Base SAS procedures, which include RANK, FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE. This functionality will allow those procedures to pass down more SQL to the database other than just `SELECT * FROM table`. The benefit of this additional SQL being passed to the database is a huge performance boost, especially for larger tables. By performing much of the initial aggregations on the database, the in-database processing of Base SAS procedures limits the amount of data that has to be passed back to SAS, and therefore greatly increases the performance of those Base SAS procedures.

LIBNAME NICKNAME SUPPORT

As you may have noticed in the above examples, the SAS/ACCESS to Amazon Redshift LIBNAME syntax currently uses the engine name, `sasiorst`, as the engine nickname (identifier) instead of a friendly name. The next release of the SAS/ACCESS to Amazon Redshift product will add the nickname `redshift` so that your LIBNAME statement will look like:

```
libname myred redshift server="myredshift.redshift.amazonaws.com" db=mydb
      user=myuser pwd=mypwd;
```

Note that you can update the nickname in the first release of SAS/ACCESS to Amazon Redshift to use the friendly name with the following code:

```
proc nickname cat=sashelp.core engine;
   add nickname=redshift module=sasiorst
   desc="SAS/ACCESS to Amazon Redshift"
   preferred eng;
quit;
```

The above code updates the CORE catalog in your SAS installation to associate the nickname `redshift` with the `sasiorst` engine name to allow you to use an easy to remember name in your LIBNAME statements to SAS/ACCESS to Amazon Redshift.

CONCLUSION

Amazon Redshift provides a powerful, scalable data warehousing service, and SAS/ACCESS Interface to Amazon Redshift provides an optimized interface for harnessing the power of Amazon Redshift. By bundling the DataDirect Amazon Redshift ODBC driver, SAS/ACCESS to Amazon Redshift is easier to install and configure than SAS/ACCESS to ODBC. The DataDirect Amazon Redshift ODBC driver also provides increased performance in writing data to Amazon Redshift through SAS/ACCESS to Amazon Redshift. Future releases of SAS/ACCESS to Amazon Redshift will provide even more performance benefits through integrated Amazon S3 bulk loads and SAS In-Database Procedure pushdown. With all these performance-enhancing optimizations, SAS/ACCESS to Amazon Redshift provides a powerful interface to Amazon Redshift that saves you time and money when harnessing the data downpour from the cloud.

REFERENCES

Amazon Redshift Database Developer Guide. Available at http://docs.aws.amazon.com/redshift/latest/dg/welcome.html.

SAS/ACCESS 9.4 for Relational Databases: Reference, Eighth Edition. Available at http://support.sas.com/documentation/cdl/en/acreldb/69039/PDF/default/acreldb.pdf.

Wang, James Ke. 2015. “Step into the Cloud: Ways to Connect to Amazon Redshift with SAS/ACCESS®.” Proceedings of the SAS Global Forum 2015 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings15/SAS1789-2015.pdf.

ACKNOWLEDGMENTS

The authors extend their heartfelt thanks and gratitude to these individuals:

Salman Maher, SAS Institute Inc., Cary, NC
James Ke Wang, SAS Institute Inc., Beijing, China
Erwan Granger, SAS Institute Inc., Montreal, Canada
Meredyth Bass, SAS Institute Inc., Cary, NC
Keith Handlon, SAS Institute Inc., Cary, NC
Bill Oliver, SAS Institute Inc., Cary, NC
Pan She, SAS Institute Inc., Beijing, China
Peng Li, SAS Institute Inc., Beijing, China

RECOMMENDED READING

- *SAS/ACCESS® 9.4 for Relational Databases: Reference, Eighth Edition.* Available at http://support.sas.com/documentation/cdl/en/acreldb/69039/PDF/default/acreldb.pdf.

CONTACT INFORMATION

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

Chris DeHart
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Chris.DeHart@sas.com
http://www.sas.com

Jeff Bailey
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Jeff.Bailey@sas.com
www.linkedin.com/in/jeffreydbailey