PAPER SAS0245-2017

Accessing DBMS with the GROOVY Procedure and a JDBC Connection

Lilyanne Zhang, SAS Institute Inc.

ABSTRACT

SAS/ACCESS® software grants access to data in third-party database management systems (DBMS), but how do you access data in DBMS not supported by SAS/ACCESS products? The introduction of the GROOVY procedure in SAS® 9.3 lets you retrieve this formerly inaccessible data through a JDBC connection. Groovy is an object-oriented, dynamic programming language executed on the Java Virtual Machine (JVM). Using Microsoft Azure HDInsight as an example, this paper demonstrates how to access and read data into a SAS data set using PROC GROOVY and a JDBC connection.

INTRODUCTION

Groovy is an object-oriented, dynamic language that compiles on the Java virtual machine (JVM). While most Java code can be executed alongside Groovy code, Groovy has a flexible nature that simplifies the strict syntax in Java code. Within SAS, PROC GROOVY can run Groovy code along with specified files, parse Groovy statements into Groovy class objects, and make them available to other PROC GROOVY statements or Java DATA step objects. In addition, the procedure can update CLASSPATH environment variables.

COMPARING JAVA TO GROOVY

Groovy simplifies the Java JDBC API by creating new instances of the SQL class. As you can see, Groovy reduces the number of lines Java requires a significant amount. Below is a snippet of Java code taken from the IBM developerWorks documentation. Here, we see the declaration of a connection and queries to a database using the standard Java JDBC API.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class JDBCExample1 {
  public static void main(String[] args) {
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    try{
      Class.forName("com.mysql.jdbc.Driver");
      con=DriverManager.getConnection("jdbc:mysql://localhost:3306/words",
        "words", "words");
      stmt = con.createStatement();
      rs = stmt.executeQuery("select * from word");
      while (rs.next()) {
        System.out.println("word id: " + rs.getLong(1) +
            " spelling: " + rs.getString(2) +
            " part of speech: " + rs.getString(3));
      }
    }catch(SQLException e){
      e.printStackTrace();
    }catch(ClassNotFoundException e){
      e.printStackTrace();
    }finally{
      try{rs.close();}catch(Exception e){}
      try{stmt.close();}catch(Exception e){}
```

```
         try{con.close();}catch(Exception e){}
        }
      }
   }
```

The same Java code above can be executed in significantly fewer lines using Groovy. Groovy does not require the usage of ResultSet classes along with the many try catch blocks for catching exceptions.

```
import groovy.sql.Sql
class GroovySqlExample1{
  static void main(String[] args) {
    def sql = Sql.newInstance("jdbc:mysql://localhost:3306/words", "words",
           "words", "com.mysql.jdbc.Driver")
    sql.eachRow("select * from word"){ row ->
       println row.word_id + " " + row.spelling + " " + row.part_of_speech
    }
  }
}
```

PROC GROOVY SYNTAX

PROC GROOVY executes Groovy code between SUBMIT and ENDSUBMIT statements. Prior to the SUBMIT statement, the CLASSPATH value can be updated to include any necessary JAR files. To clear a CLASSPATH, use the CLEAR option after the ENDSUBMIT (or SUBMIT) statement. A RUN statement is not necessary because the code will execute following the ENDSUBMIT statement:

```
proc groovy;
submit;
    println('hello world!')
endsubmit;
```

NEED FOR GROOVY TO CONNECT TO OTHER DBMS

With MongoDB, Azure HDInsight, and other NoSQL databases gaining in popularity, there is a need to read this data into SAS. Current SAS/ACCESS products do not support these management systems, but establishing a JDBC connection through PROC GROOVY will enable users to read their corresponding data into SAS.

READING IN DATA USING PROC GROOVY

ADDING AND REMOVING A CLASSPATH

To begin, download the respective JDBC driver. This should be easily found from the documentation section of the DBMS. Access this URL (https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-connect-hive-jdbc-driver) for HDInsight instructions on getting the Hive JDBC driver. Add the complete qualifying pathname in quotation marks as the first statement within the declared GROOVY procedure. Remove classpaths by using the CLEAR statement:

```
proc groovy;
   add classpath=cp;
submit;
    //insert groovy code here
endsubmit;
    clear;
```

ESTABLISHING A CONNECTION

To read in data, we must first connect to the database. Using Groovy's simplified syntax, we can define a new instance of an SQL object and add corresponding parameters. Moreover, we can read, write, and update our database using the execute method:

```
proc groovy;
   add classpath=cp;
submit;
    import groovy.sql.Sql
    import java.util.List
    def db = [url:
    'jdbc:hive2:///default;',user:'user',password:'password',
    driver:'org.apache.hive.jdbc.HiveDriver']
    def sql = Sql.newInstance(db.url, db.user, db.password, db.driver)

    /* Create a new table */
    sql.execute("CREATE TABLE Author (id INT, firstname VARCHAR(64)")
    /* Insert observation */
    sql.execute("INSERT INTO Author(id, firstname) VALUES(1, 'Lily')")
    /* Put values into a List */
    List id = sql.rows('select id from Author')

endsubmit;
clear;
```

READING DATA INTO SAS

We can read the data into SAS in two ways. For Method 1, use the EXPORTS keyword to extract individual column values, and then use a series of PROC procedures to cleanse the data. Unfortunately, this method is limited to the length of a valid SAS macro variable. To maximize efficiency, consider another method – write values to a CSV file and use the SAS/ACCESS Interface to PC Files engine to import the data into SAS. This paper will not discuss this second method due to length restrictions, but it is very doable using Groovy documentation.

Using the EXPORTS Keyword

The EXPORTS keyword is a special PROC GROOVY variable that enables you to use a SAS macro variable to read the data into a SAS data set. SAS macro variables can have a length of up to 65K characters, which depending on the type of data value, might or might not be enough.

Here, we are exporting the ID values into a SAS macro variable and applying basic PROC steps to get our desired data set:

```
proc groovy;
   add classpath=cp;
submit;
    import groovy.sql.Sql
    import java.util.List
    def db = [url:
    'jdbc:hive2:///default;',user:'user',password:'password',
    driver:'org.apache.hive.jdbc.HiveDriver']
    def sql = Sql.newInstance(db.url, db.user, db.password, db.driver)

    /* Put values into a List and export to macro variable*/
    List id = sql.rows('select id from Author')
    exports.putAt('ID', id);
endsubmit;
```

Unfortunately, the macro variable is type is defined as a string and extra effort must be done to convert the data type back into an integer. The macro must also be cleansed to get rid of the extra brackets that come along with exporting the variable. We will now read the macro into a DATA step using the SYMGET function and begin the cleansing process:

```
data temp;
   length id $32;
   id = symget('id);
   id = tranwrd(id, '}, {', '},{');
   do i=1 to 1e6;
      text = scan(value, i, '{},[]');
      if text eq '  ' then leave;
      text = scan(text, 2, '=');
      output;
    end;
    keep name i text;
run;
```

Next, we sort the data by I and transpose the data to get our desired data set.

```
proc sort data=test;
   by i;
run;

proc transpose data=test out=test1(drop=I _NAME_);
    id name;
    var text;
    by i;
run;

proc print data=test1; run;
```

CONCLUSION

The rise of DBMS that are not supported by SAS/ACCESS products requires methods for reading the data into SAS. Using PROC GROOVY and a JDBC connection, you can read that data into SAS through the exported variable along with a few data manipulation techniques. This method proves most effective if you are querying specific data that would otherwise require extra work outside of SAS.

REFERENCES

- SAS PROC GROOVY Documentation. Available at

  http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#p0pq7kp0misx44n14vvub91rzihh.htm.

- Groovy-SQL Documentation. Available at

  http://docs.groovy-lang.org/latest/html/api/groovy/sql/Sql.html.

- Hive JDBC Documentation. Available at

  https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-connect-hive-jdbc-driver.

ACKNOWLEDGMENTS

Special thanks to Salman Maher, Bill Oliver, Jeff Bailey, Sue Her, and the rest of the SAS/ACCESS Research and Development team.

CONTACT INFORMATION

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

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