PharmaSUG 2016 – Paper AD04

SAS® integration with NoSQL data

Kevin Lee, Clindata Insight, Moraga, CA

ABSTRACT

We are living in the world of abundant data, so called “big data”. The term “big data” is closely associated with unstructured data. They are called “unstructured” or NoSQL data because they do not fit neatly in a traditional row-column relational database. A NoSQL (Not only SQL or Non-relational SQL) database is the type of database that can handle unstructured data. For example, a NoSQL database can store unstructured data such as XML (Extensible Markup Language), JSON (JavaScript Object Notation) or RDF (Resource Description Framework) files.

If an enterprise is able to extract unstructured data from NoSQL databases and transfer it to the SAS environment for analysis, this will produce tremendous value, especially from a big data solutions standpoint. This paper will show how unstructured data is stored in the NoSQL databases and ways to transfer it to the SAS environment for analysis. First, the paper will introduce the NoSQL database. For example, NoSQL databases can store unstructured data such as XML, JSON or RDF files. Secondly, the paper will show how the SAS system connects to NoSQL databases using REST (Representational State Transfer) API (Application Programming Interface). For example, SAS programmers can use the PROC HTTP option to extract XML or JSON files through REST API from the NoSQL database. Finally, the paper will show how SAS programmers can convert XML and JSON files to SAS datasets for analysis. For example, SAS programmers can create XMLMap files using XMLV2 LIBNAME engine and convert the extracted XML files to SAS datasets.

INTRODUCTION OF UNSTRUCTURED DATA

Unstructured data is data that either does not have a pre-defined data model or is not organized in a pre-defined way. Usually, unstructured data is text-heavy and has irregularities that make it difficult for programmers to understand and analyze as compared to structured data which has rows and columns. Unstructured data may include books, journals, documents, metadata, health records, audio, video, analog data, images, files, and unstructured text such as the body of an e-mail message, web page, or a word-processed document.

STRUCTURED DATA vs UNSTRUCTURED DATA

So, how is structured data different from unstructured data? Structured data is data that follows a certain schema such as rows and columns.

For example, a DM SAS dataset is structured data as shown in Table 1.

STUDYID	USUBJID	DOMAIN	SUBJID	FRSTDTC	RFENDTC	SITEID	AGE	AGEU	SEX
CDISCPILOT01	01-701-1015	DM	1015	2014-01-02	2014-07-02	701	63	YEARS	F

Table 1: DM SAS dataset

Unstructured data does not follow any particular schema, so programmers can add columns or fields, or modify the structure without fear that the modified data will not fit into the database. In the clinical trial environment, unstructured data can be protocol documents, SAP documents, TFL results and CSR reports.

For example, a SAP document in Code 1 below is an example of unstructured data. As seen in <objectives> element, the programmer can add as many <secondary> elements as needed. In a structured and relational data, this requires normalization and additional tables. Later in the paper, it is shown how unstructured data in xml format is converted into structured SAS datasets.

```
<?xml version="1.0" encoding="UTF-8"?>
<SAP>
        <introduction>The purpose of SAP is to provide details of statistical analyses for the study
        CDISCPILOT01</introduction>
        <objectives>
                <primary/>To evaluate overall survival for study drug
                <secondary/>To evaluate progression free survival
                <secondary/>To evaluate time to progression
                <secondary/>To evaluate relapse incidents
```

```
        </objectives>
        <study_overview>
                <study_design>This is phase 3 study where at least 400 subjects with advanced solid tumor will
                receive weekly 10 mg/kg doses of study drug. </study_design>
        </study_overview>
        ……
</SAP>
```

Code 1: protocol document (CDISCPILOT01-protocol.xml)

INTRODUCTION OF NoSQL DATABASE

NoSQL, which stands for "Not Only SQL", not "NO SQL", can store structured, unstructured or semi-structured data. In a nutshell, NoSQL database can store any type of data. This is good news for DBA and programmers because the data does not need to be formatted in any way to fit it into the database. They can simply load the data into the database without transformation or manipulation. This structure-agnostic data model of the NoSQL database attracts many data-driven enterprises such as Google, LinkedIn, Facebook, Twitter, Netflix and the New York Times. These are early adopters of the NoSQL databases.

There are four types of NoSQL databases.
1. Key Value
2. Document
3. Column-family
4. Graph

This paper will focus on the NoSQL Document database. There are different ways to retrieve data from the NoSQL database. This paper will show how to retrieve data from the NoSQL database using REST API.

INTRODUCTION OF REST

Representational State Transfer (REST) is a simple data exchange format which is platform-, system- and language-independent and communicates through the internet. It uses HTTP and the response files come ready to be used. REST does not use XML to send and receive data through web services. The following examples will show how request and response files from REST can be extracted from the NoSQL database.

HOW TO OBTAIN DATA TO SAS ENVIRONMENT USING REST API

In Code 2 below, SAS programmers will call Yahoo Finance REST web service to receive the data of Apple stock prices from 11/29/2004 to 01/01/2005. The SAS statement FILENAME and its option URL can set up REST communication. The URL up to the question mark, (http://ichart.finance.yahoo.com/table.csv), is the path to REST API and the URL after the question mark, (s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=2005&g=d&ignore=.csv), are the parameters and their inputs to the web service. The URL will call daily quotes of Apple stocks from 11-29-2004 to 01-01-2005 and SAS dataset, price, extracts the information.

```
*** FILENAME and URL will call REST;
filename price url
'http://ichart.finance.yahoo.com/table.csv?s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=2005&g=d&ignore=.csv'
debug;
*** SAS dataset of price will contain daily stock quotes of Apple stock;
data price;
    infile price length=len;
    input record $varying200. len;
run;
```

Code 2: SAS filename and url codes using Yahoo Finance REST API

SAS programmers can also use PROC HTTP to receive Apple daily quotes in CSV file format. The SAS codes in Code 3 will show how to call REST using PROC HTTP and save CSV response files to the local drive.

```
*** Create csv file in local drive ;
filename out "&local_url.\resp-stock-appl.csv";  ** will get response output;
*** URL option in PROC HTTP send a GET request to Yahoo Finance REST;
proc http
   out=out
   url="http://ichart.finance.yahoo.com/table.csv?s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=2005&g=d&ignore=.csv"
```

```
   method="GET";
run;
```
Code 3: SAS proc http codes using Yahoo Finance REST API

Code 4 below shows how SAS programmers can also receive CDASH Control Terminology from the NCI website.

```
*** Codes will receive CDASH CT from NCI URL to local drive;
filename cdash "G:\KL\web connection\URL connection\CDASH Terminology.xls";
**** This gives the exact same results;
proc http out=cdash
          url="http://evs.nci.nih.gov/ftp1/CDISC/SDTM/CDASH%20Terminology.xls"
          method="get" ;
run;
```
Code 4: SAS codes to receive CDASH Control Terminology

HOW TO OBTAIN DATA FROM NOSQL DATABASE TO SAS ENVIRONMENT USING REST API

The New York Times currently uses MongoDB as its NoSQL database solution for its content publishing. Code 5 shows how SAS programmers can receive xml documents from the New York Time. Using REST API, SAS programmers can get the data or documents from the NoSQL database.

There are many APIs available for the New York Times. The paper uses "Books API: Book Reviews" to show how programmers can retrieve the New York Times book reviews. REST API URL of the database is "http://api.nytimes.com/svc/books/v3/reviews.xml". There are two required parameters: ISBN for the specific book, "isbn=9780062409850" and "api-key=xxxxx" for the REST API Key, which a programmer needs to obtain by registering himself or herself as a developer to the New York Times. Here, this paper does not include the exact api-key since it is confidential for each programmer. SAS programmers can find more information on how to use the New York Times book review API in http://developer.nytimes.com/docs/books_api/Books_API_Book_Reviews.

```
*** file name that will receive document;
filename review "C:\KL\BookReview\isbn-9780062409850.xml";
**** call resp api using proc http;
proc http
       out=review
       url="http://api.nytimes.com/svc/books/v3/reviews.xml?isbn=9780062409850&api-key=xxxxx"
       method="GET" ;
run;
```
Code 5: SAS codes using the New York Times REST API

Code 6 is the xml file that a SAS programmer receives from New York Times REST API.

```
<?xml version="1.0" encoding="UTF-8"?>
<result_set>
   <status>OK</status>
   <copyright>Copyright (c) 2016 The New York Times Company.  All Rights Reserved.</copyright>
   <num_results>1</num_results>
   <results>
      <result>
         <url>http://www.nytimes.com/2015/07/14/books/review/harper-lees-go-set-a-watchman.html</url>
         <publication_dt>2015-07-14</publication_dt>
         <byline>RANDALL KENNEDY</byline>
         <book_title>Go Set a Watchman</book_title>
         <book_author>Harper Lee</book_author>
         <summary>"Go Set a Watchman" demands that its readers abandon the immature sentimentality ingrained by
middle school and the film adaptation of "To Kill a Mockingbird."</summary>
         <isbn13>
            <isbn13_item>9780062409850</isbn13_item>
            <isbn13_item>9780062409874</isbn13_item>
            <isbn13_item>9780062409881</isbn13_item>
            <isbn13_item>9780062409904</isbn13_item>
            <isbn13_item>9780062454812</isbn13_item>
            <isbn13_item>9781473535404</isbn13_item>
         </isbn13>
      </result>
   </results>
```

```
</result_set>
```
Code 6: isbn-9780062409850.xml

HOW TO CONVERT XML DATA TO SAS DATA

Data from NoSQL document database is usually in either XML or JSON format. SAS programmers can convert XML or JSON files to SAS datasets. Code 6 is a xml file that a SAS programmer receives from the New York Times. SAS programmers can convert this xml file to SAS datasets for further analysis or transformation.

SAS programmers can use SAS XML mapper to map xml file to SAS datasets. In SAS 9.3, SAS programmers create XMLMap file using XMLV2 LIBNAME engine. XMLMap file can create SAS datasets from a hierarchical XML file. Code 7 will show how a SAS programmer can create XMLMap file (See the appendix 1 for actual response.map) and use XML map file to create SAS datasets in Table 2 to 6.

```
*******************************************************************************
*  Create XMLMap file and create SAS datasets from XML file
*******************************************************************************;
** response xml files;
filename  resp "  C:\KL\BookReview\isbn-9780062409850.xml ";

**** Create response xml map file;
filename  respmap " C:\KL\BookReview\response.map ";
libname resp xmlv2 xmlmap=respmap automap=replace;

**** Convert response xml files to SAS temporary dataset in work area;
proc copy in=resp out=work;
run;
```
Code 7: SAS codes that can create XMLMap file and also convert XML file to SAS datasets using XMLMap file.

In the SAS working library, there are two 5 SAS datasets created –isbn13, isbn13_item, result, result_set and results.

Result_ORDINAL	Isbn13_ORDINAL
1	1

Table 2: SAS dataset of isbn13

Isbn13_ORDINAL	Isbn13_item_ORDINAL	Isbn_item
1	1	9780062409850
1	2	9780062409874
1	3	9780062409881
1	4	9780062409904
1	5	9780062454812
1	6	9781473535404

Table 3: SAS dataset of isbn13_item

results_ORDINAL	result_ORDINAL	url	Publication_dt	byline	Book_title	Book_author	summary
1	1	http://www.nytimes.com/2015/07/14/books/review/harper-lees-go-set-a-watchman.html	2015-07-14	RANDALL KENNEDY	Go Set a Watchman	Harper Lee	“Go Set a Watchman” demands that its readers abandon the immature sentimentality ingrained by middle school and the film adaptation of “To Kill a Mockingbird.”

Table 4: SAS dataset of result

Result_set_ORDINAL	Status	Copyright	Num_results
1	OK	Copyright (c) 2016 The New York Times Company. All Rights Reserved	1

Table 5: SAS dataset of result_set

Result_set_ORDINAL	results_ORDINAL

1	1

Table 6: SAS dataset of results

Now, a SAS programmer can use these data for further analysis.

HIGH LEVEL DESIGN OF SYSTEM ARCHITECTURE BETWEEN NOSQL DATABASE AND SAS

Figure 1 shows the high level design of the system architecture between SAS and NoSQL database over the internet using REST API. SAS programmers can send PROC HTTP request to the NoSQL database using RESP API. SAS programmers can receive response files in XML and convert XML files to SAS datasets for further analysis and reporting.

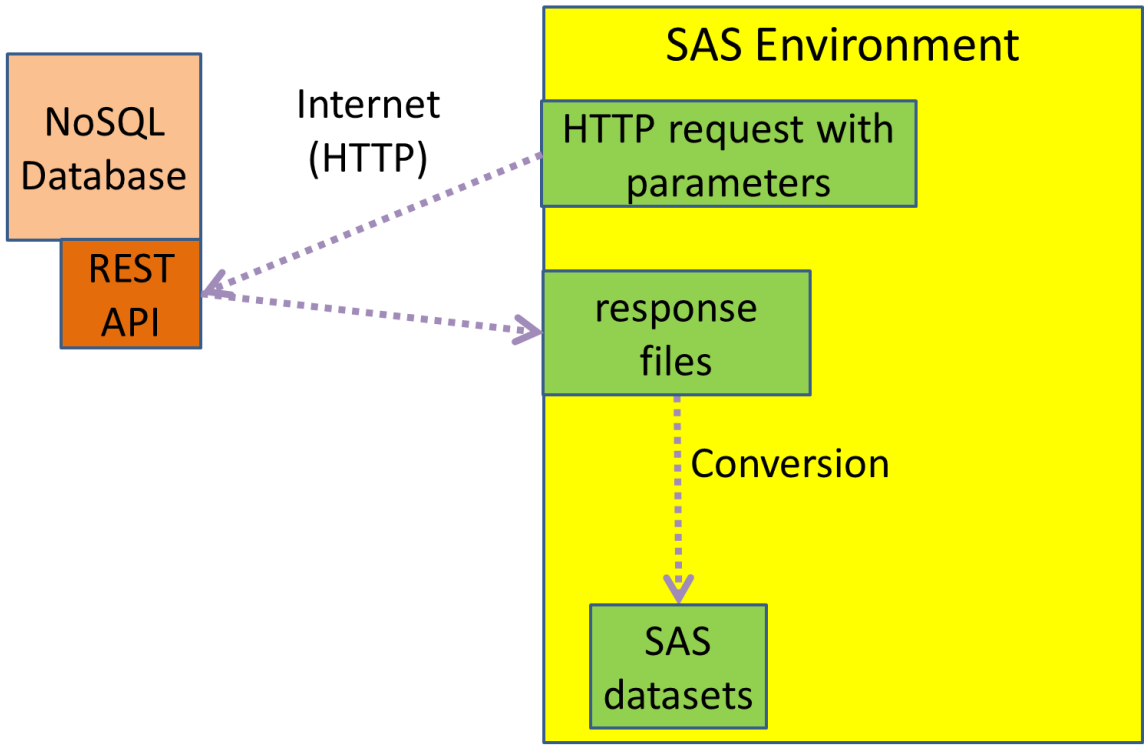

Figure 1: High Level Design of System Architecture between NoSQL database and SAS

CONCLUSION

The ability to retrieve unstructured and structured data to the SAS environment from NoSQL database is extremely useful and powerful. With this, SAS programmers can use SAS analytic power to combine unstructured, structured and semi-structured data and analyze them together. For example, SAS programmers can extract real-time data from Twitter or Google (NoSQL database) into the SAS environment for real-time reporting and analysis. In the drug development process, if the protocol documents are generated and stored in a NoSQL database, SAS programmers can extract the study protocol documents using REST API and create SDTM trial design domains.

REFERENCES

- Introduction of NoSQL database, http://nosql-database.org
- NoSQL – Wikipedia, https://en.wikipedia.org/wiki/NoSQL
- The SAS programmer's guide to XML and Web Services, Chirstopher W. Schacherer.
- The Ins and Outs of Web-Based Data with SAS, Bill McNeill
- A simple way of importing from A REST Web Service into SAS in Three Lines of Code, Philip Busby
- Extreme Web Access: What to do when FILENAME URL is not enough, Garth Helf.
- Using Base SAS to talk to the outside world: consuming SOAP and REST Web Services using SAS 9.1 and the new features of SAS 9.2, Curtis E. Mack.
- SAS XML LIBNAME Engine: User's guide
- SAS PROC HTTP: User's guide

ACKNOWLEDGEMENT

I would like to acknowledge and extend my heartfelt gratitude to Clindata Insight team for their inputs and insights, especially Mylihn Paolieri, Richelle Serrano and Peng Yang.

CONTACT INFORMATION

Your comments and questions are valued and welcomed. Please contact the author at

Kevin Lee
Director of Data Science
Clindata Insight
klee@clindatainsight.com
215-738-0350

TRADEMARKS

Appendix 1 – response.map

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ############################################################### -->
<!-- 2016-03-27T22:08:41 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 904300.0.0.20150204190000_v940m3 -->
<!-- ############################################################### -->
<!-- ###  Validation report                                  ### -->
<!-- ############################################################### -->
<!-- XMLMap validation completed successfully. -->
<!-- ############################################################### -->
<SXLEMAP name="AUTO_GEN" version="2.1">

   <NAMESPACES count="0"/>

   <!-- ############################################################### -->
   <TABLE description="result_set" name="result_set">
      <TABLE-PATH syntax="XPath">/result_set</TABLE-PATH>

      <COLUMN class="ORDINAL" name="result_set_ORDINAL">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="status">
         <PATH syntax="XPath">/result_set/status</PATH>
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
         <LENGTH>2</LENGTH>
      </COLUMN>

      <COLUMN name="copyright">
         <PATH syntax="XPath">/result_set/copyright</PATH>
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
         <LENGTH>68</LENGTH>
      </COLUMN>

      <COLUMN name="num_results">
         <PATH syntax="XPath">/result_set/num_results</PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

   </TABLE>

   <!-- ############################################################### -->
   <TABLE description="results" name="results">
      <TABLE-PATH syntax="XPath">/result_set/results</TABLE-PATH>

      <COLUMN class="ORDINAL" name="result_set_ORDINAL">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN class="ORDINAL" name="results_ORDINAL">
         <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results</INCREMENT-PATH>
         <TYPE>numeric</TYPE>
         <DATATYPE>integer</DATATYPE>
      </COLUMN>

   </TABLE>
```

```
<!-- ################################################################## -->
<TABLE description="result" name="result">
   <TABLE-PATH syntax="XPath">/result_set/results/result</TABLE-PATH>

   <COLUMN class="ORDINAL" name="results_ORDINAL">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
   </COLUMN>

   <COLUMN class="ORDINAL" name="result_ORDINAL">
      <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results/result</INCREMENT-PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>integer</DATATYPE>
   </COLUMN>

   <COLUMN name="url">
      <PATH syntax="XPath">/result_set/results/result/url</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>81</LENGTH>
   </COLUMN>

   <COLUMN name="publication_dt">
      <PATH syntax="XPath">/result_set/results/result/publication_dt</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>date</DATATYPE>
      <FORMAT width="10">IS8601DA</FORMAT>
      <INFORMAT width="10">IS8601DA</INFORMAT>
   </COLUMN>

   <COLUMN name="byline">
      <PATH syntax="XPath">/result_set/results/result/byline</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>15</LENGTH>
   </COLUMN>

   <COLUMN name="book_title">
      <PATH syntax="XPath">/result_set/results/result/book_title</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>17</LENGTH>
   </COLUMN>

   <COLUMN name="book_author">
      <PATH syntax="XPath">/result_set/results/result/book_author</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>10</LENGTH>
   </COLUMN>

   <COLUMN name="summary">
      <PATH syntax="XPath">/result_set/results/result/summary</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>159</LENGTH>
   </COLUMN>

</TABLE>

<!-- ################################################################## -->
<TABLE description="isbn13" name="isbn13">
   <TABLE-PATH syntax="XPath">/result_set/results/result/isbn13</TABLE-PATH>

   <COLUMN class="ORDINAL" name="result_ORDINAL">
```

```
        <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results/result</INCREMENT-PATH>
        <TYPE>numeric</TYPE>
        <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN class="ORDINAL" name="isbn13_ORDINAL">
        <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results/result/isbn13</INCREMENT-
PATH>
        <TYPE>numeric</TYPE>
        <DATATYPE>integer</DATATYPE>
      </COLUMN>

   </TABLE>

   <!-- ################################################################ -->
   <TABLE description="isbn13_item" name="isbn13_item">
      <TABLE-PATH syntax="XPath">/result_set/results/result/isbn13/isbn13_item</TABLE-PATH>

      <COLUMN class="ORDINAL" name="isbn13_ORDINAL">
        <INCREMENT-PATH beginend="BEGIN" syntax="XPath">/result_set/results/result/isbn13</INCREMENT-
PATH>
        <TYPE>numeric</TYPE>
        <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN class="ORDINAL" name="isbn13_item_ORDINAL">
        <INCREMENT-PATH beginend="BEGIN"
syntax="XPath">/result_set/results/result/isbn13/isbn13_item</INCREMENT-PATH>
        <TYPE>numeric</TYPE>
        <DATATYPE>integer</DATATYPE>
      </COLUMN>

      <COLUMN name="isbn13_item">
        <PATH syntax="XPath">/result_set/results/result/isbn13/isbn13_item</PATH>
        <TYPE>character</TYPE>
        <DATATYPE>string</DATATYPE>
        <LENGTH>13</LENGTH>
      </COLUMN>

   </TABLE>

</SXLEMAP>
```