Introducing Mario, an Internal Audit specific ETL and Data Mart Solution

Andrew Clark

andrewtaylorclark@gmail.com

ABSTRACT

Mario is an audit specific data mart which implements the AICPA Audit Data Standards (AICPA 2015). Built using exclusively open source software, Mario provides a flexible environment that is designed for Internal and External Auditor Journal Entry requirements in disparate system environments.

Kewords: audit data standard, data warehouse, open source, python, etl

INTRODUCTION

In decentralized organizations with disparate systems, financial information may exist in multiple locations and only be consolidated, in aggregate, monthly or quarterly. For details of transactional movements between accounts, data acquisition, either by personalized request or SQL querying, is the primary method of acquiring the detail needed for analysis. With different accounting processes, i.e. batch vs. real-time processing, different chart of account mappings, etc., the process of acquiring and analyzing the journal entry detail can be time-consuming. To ease the burden on Internal Auditors and External Auditors, a flexible open source software package, dubbed “Mario”, has been developed based on the schema of the AICPA Audit Data Standards (AICPA 2015).

System Design Considerations

For ease of use, its open source heritage, analytic prowess and strong integration capabilities, Python 2.7 (Python Software Foundation) was selected as the implementation language. SQL (Chamberlin and Boyce 1974), the language of data, was used in conjunction with ODBC (Microsoft Corporation) connections (utilizing either Windows Authentication or SQL authentication), to extract general ledger data from n number of relational databases. To service a variable number of database queries and handle the inevitable failure, while providing logs with success and failure notifications of each database query, a robust ETL framework was needed.

ETL Process

For handling the Transform part of the Extract, Transform and Load process (ETL), two main choices presented themselves:

1. Perform data transformation within the native database during the extract process
2. Perform data transformation within local memory after extracting from the native database in its original form.

Either approach is viable, with option 2 more effective for large datasets, however for the use case as presented, option 1 was chosen since the system load on the individual databases is trivial. However, refactoring would be relatively simple if the need would arise to utilize local transformation.

Data Testing

Data definition of the transactional general ledger tables is of paramount importance to ensure accurate data. Testing of the data source by creating a Trial Balance for a given period and

comparing to a standard system generated Trial Balance for the period is a necessity before continuing further with the pipeline setup. It is recommended multiple trial balances periods are tested with care given to the dimensions for the invoice statement accounts.

Component Choices

Luigi, an ETL framework developed by Spotify (Bernhardsson and Freider 2012), provides fault tolerance, run memory, (meaning that if a job failure occurs in the workflow, Luigi will remember the state of the other jobs and when the workflow is rerun, will not restart previously successful job parts). For production uses, Luigi provides a centralized scheduler which controls worker nodes in a master-slave relationship. Built primarily for Hadoop (The Apache Software Foundation) jobs, Luigi works well with relational databases. Luigi does have one major drawback: its inability to initiate jobs. However, Cron (Mellor 2003) or equivalent can provide a trivial remediation to this issue.

Keeping with the Open Source focus, MySQL (MySQL), a relational database system, was chosen as the Data Mart. As the schema would be relatively constant in sample use case, with ACID (Atomicity, Consistency, Isolation and Durability) being extremely import in a financial context, and with a manageable data size on a single machine, an SQL vs. a NoSQL (Strauch 2011) system was an easy decision.

Conversion Steps

The AICPA Audit Data Standards (ADS) Base and General Ledger standards (AICPA 2015) form the basis of the Mario system. SQL tables were created off the flat file specifications. Below is the Chart of Accounts table, shown as an example:

```
CREATE TABLE Chart_Of_Accounts (
  ID INT PRIMARY KEY AUTO_INCREMENT,
  GL_Account_Number VARCHAR(100) NOT NULL,
  GL_Account_Name VARCHAR(100),
  Account_Type VARCHAR(25),
  Account_Subtype VARCHAR(25),
  FS_Caption VARCHAR(100),
  GL_Account_Description" VARCHAR(256),
  Business_Unit_Code VARCHAR(25) NOT NULL,
  Parent_GL_Account_Number" VARCHAR(100),
  Segment01 VARCHAR(25),
  Segment02 VARCHAR(25),
  Segment03 VARCHAR(25),
  Segment04 VARCHAR(25),
  Segment05 VARCHAR(25)
);
```

After the tables are set up, a Luigi pipeline is constructed with a class for each subsidiary or unit that has a separate database.

A crucial step of a worthwhile implementation of the Mario framework requires a detailed mapping of each subsidiary general ledger account to each consolidated, or corporate, ledger account. All the subsidiary chart of accounts will be loaded into the Chart of Accounts table with the Corporate GL account for the given subsidiary account given in the Parent GL Account

Number column. A basic query in the completed system would be along the lines of the following (the "?" are placeholders for the year, 2017, month, 5, etc.):

```
SELECT DISTINCT
c.Business_Unit_Code,
a.Journal_ID,
a.Journal_ID_Line_Number,
a.JE_Header_Description,
a.Effective_Date,
a.Fiscal_Year,
a.Period,
b.GL_Account_Number,
b.GL_Account_Name,
a.Amount,
a.Amount_Credit_Debit_Indicator,
a.Reporting_Amount
FROM
GL_Detail AS a
LEFT JOIN Chart_Of_Accounts AS b
ON b.GL_Account_Number = a.GL_Account_Number
AND a.Segment01 = b.Segment01
AND a.Segment02 = b.Segment02
AND a.Segment03 = b.Segment03
AND a.Segment04 = b.Segment04
```

```
AND a.Segment05 = b.Segment05
LEFT JOIN Business_Unit_Listing AS c
ON c.id = a.Business_Unit_Code
WHERE
a.Fiscal_Year =? AND a.Period =?
AND b.GL_Account_Number =?
AND c.Business_Unit_Code =?
ORDER BY
a.Effective_Date DESC;
```

Which combines the Chart_Of_Accounts table with the Business Unit Listing (which has a listing of all the reporting subsidiaries with a "Business Unit Code" between 1 and n), with the primary GL Detail. For more information and parts of the source code, contact the author.

Implementation

The Mario system uses a separate class for each database pull, with the resulting data exported to a time stamped CSV file. The extraction GL queries include a parameterized date value that is handed in from the Luigi framework. For ease of use and abstraction, a separate library called, rather ingeniously, ODBC library has been created to house all the individual connection strings to subsidiary databases.

The next step in the pipeline is the Combine step, which, after waiting for a target file from each database files to be returned, continues by combining the variable number of output CSVs from successful database pulls using the Python glob library:

```
# Previously imported libraries provided here
# for reference
import pandas as pd
import glob

path = `IntermdiateStoragePath/'
location = path + `*_GL_\textit{Mario}_*%s.csv' % self.date
allFiles = glob.glob(location)
frame = pd.DataFrame()
list_ = []
for file_ in allFiles:
  df = pd.read_csv(file_)
  list_.append(df)
frame = pd.concat(list_)
```

All files in the intermediate data path containing the ending of:

`*_GL_Mario_*%s.csv' % self.date

are imported and combined to create a consolidated data frame. This combined data file is then exported, as a csv, with the current date. An error email specifying which companies were not downloaded could be inserted. Data is exported in each step as a checkpoint if a catastrophic failure occurs and the process must be restarted. An email specifying which companies did not have any data to download is provided.

An Insert class is implemented next that takes the intermediate combined file, removes any additional formatting columns, (other cleaning procedures, such as striping white space from GL accounts could be set here as well), and pushes GL detail to the MySQL database. A success email and an intermediate directory purge function are present for a clean execution.

Merlin Graphical User Interface (GUI)

Merlin is a Graphical User Interface (GUI) that allows the user, if they are not comfortable with SQL querying, to access the Mario system. The current options with the Merlin interface are:

1. Output Location
2. File name
3. Fiscal year
4. GL Account Number (based off the consolidated standard)
5. Company (or reporting subsidiary)

The user interface is constructed with the Python Gooey (Kiehl 2014) library. More options, including a date range instead of static month/year periods will be added in the future.

Module Additions

With inspiration from the current micro services revolution, Mario has been constructed with extensibility and modularity in mind. Analytical attachments can be added to the Luigi pipeline, or added through the MySQL database layer. By keeping with the defined AICPA ADS

standards (AICPA 2015), a static schema in the primary tables can be expected. Anticipated modules include:

1. Semi-supervised machine learning journal entry anomaly detection tool. The module will be implemented by creating 'clusters' off multiple years of data, since the nature of financial journal entries is very cyclical in nature.

2. Supervised machine learning algorithm for implementing the "Auditor's sense" (Nelson 2009) on what sorts of transactions warrant closer scrutiny but are hard to quantify using traditional threshold analytic tests.

3. Leveraging the Vendor and Customer definitions provided in the AICPA data standards (AICPA 2015), Mario will be expanded to include consolidated Vendor and Customer tables that will be compared to the US Treasury Department's Office of Foreign Assets Control list of sanctioned firms. Additional lists can be trivially added to the scanning.

4. Standard, traditional audit analytics tests can be built off Mario, including meta data tables which record how many items were surveyed, investigated, etc., at any given run time, whether that be daily, hourly, monthly or quarterly.

5. A LAN accessible web interface that authenticates with LDAP.

Conclusion

Mario is a practical implementation of modern ETL and Data Engineering techniques to solve the problem of unconsolidated financial reporting subsidiaries. By providing a way for Internal and External Auditors (and corporate financial employees) to view the full journal entry detail in the consolidated general ledger chart of accounts, Mario can provide quicker access to data

without waiting on subsidiaries' accountants to run GL detail for requesters. Additionally, with extensibility for groundbreaking machine learning and analytic procedures, Mario provides the bedrock for a world-class audit analytics program.

References

AICPA Assurance Services Executive Committee's Emerging Assurance Technologies Task Force. 2015. *Audit Data Standards.* Available at: https://www.aicpa.org/InterestAreas/FRC/AssuranceAdvisoryServices/Pages/AuditDataStandards.aspx

The Apache Software Foundation. *Apache Hadoop 2.7.4* . Available at: http://hadoop.apache.org/docs/stable/

Bernhardsson, E., E. Freider, 2012. *Luigi.* Available at: https://github.com/spotify/luigi

Chamberlin, D., R. Boyce. 1974. SEQUEL: A Structured English Query Language. *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control*. Association for Computing Machinery: 249–64

Kiehl, C. 2014. *Gooey.* Available at: https://github.com/chriskiehl/Gooey

Mellor, Dale. 2003. *GNU Mcron*. Available at: https://www.gnu.org/software/mcron/design.html

MySQL. *MySQL 5.7 Reference Manual.* Available at: https://dev.mysql.com/doc/refman/5.7/en/

Nelson, Mark W. 2009. A Model and Literature Review of Professional Skepticism in Auditing. *Auditing: A Journal of Practice & Theory* 28 (2): 1 to 34.

Microsoft Corporation. *ODBC Programmer's Reference*. Available at: https://docs.microsoft.com/en-us/sql/odbc/reference/odbc-programmer-s-reference.

Python Software Foundation. *Python Language Reference, version 2.7*. Available at http://www.python.org

Strauch, C., Kriha, W. 2011. *NoSQL databases*. Lecture Notes, Stuttgart Media University, 20.

U.S. Department of the Treasury. Office of Foreign Assets Control (OFAC). Available at: https://www.treasury.gov/about/organizational-structure/offices/Pages/Office-of-Foreign-Assets-Control.aspx