Emilija Dimikj

May 24, 2020

BIDD 220 B Sp 20: Data Migration Techniques (ETL Processing)

Assignment06

ETL process for MongoDB

# Introduction

In this ETL process, data is extracted from Excel and txt files, transform and loaded into MongoDB. Whole process is done using Python script. To apply separation of the ETL process steps, whole code is separated is three scrips: **ExtractScript.py**, **TransformScript.py** and **LoadScript.py**. Additional scripts as **CommonFuncScript.py** contains common functions that can be used by any of these three scripts and **ReportScript.py** contains functions for report generation.

**Functions**  
CommonFuncScript.py

**Reports**  
ReportScript.py

To execute the whole ETL process just run **LoadScript.py**

# Extract Source Data

Process of extracting data is written in Python **ExtractScript.py**. Each file extraction is done in its own function and returns table of extracted data.

Source data is extracted from two files with different structure:

* **Customer.xlsx** – list of Customers, contains 57 columns but in this process, just 10 columns are used for further processing. Extraction is done in **CustomersXLSFile()** function using **pyexcel** library that provides programming interface to read, manipulate and write data in various excel formats .
* **AccountBalances.txt** – list of Accounts with their balances on a specific date. This is a fixed-length file. Each content row is 642 bytes excluding line endings and is ANSI encoded. Line endings are Windows-style CarriageReturn + LineFeed (\r\n, or 0x0D0A). Extraction is done in **AccountBalancesFile()** function *(Figure 1).*

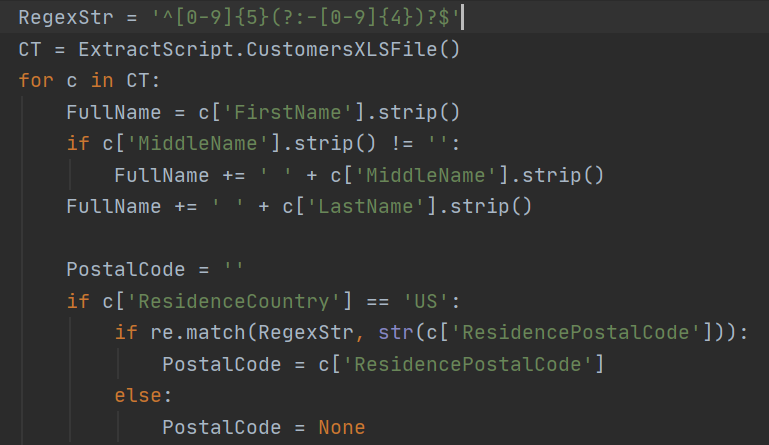


*Figure 1: AccountBalancesFile function*

# Transform Data

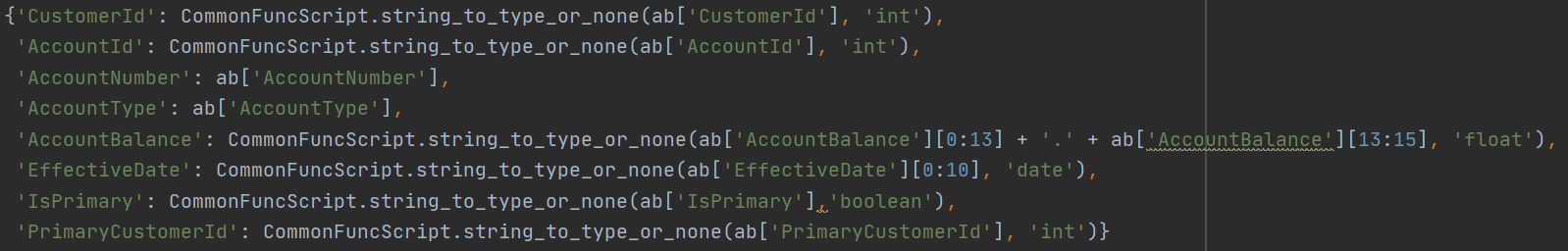
Process of extracting data is written in Python **ExtractScript.py**. Each file’s extracted data is transformed in its own function and returns table of transformed data.

* **TransformCustomers()** gets data from ExtractScript.CustomersXLSFile() function and then apply following transformations:
  + Form Customer’s Full Name as concatenation of FirstName, Middle Name and Last Name *(Figure 2)*.
  + Check if Customer’s Postal code is valid postal code using Regex matching if customer’s country of residence is US *(Figure 2)*.
  + Convert columns into correct data type.



*Figure 2: Transformations on Customers data*

* **TransformAccountBalances()** gets data from ExtractScript.AccountBalancesFile() function and then apply following transformations:
  + Convert columns into correct data type *(Figure 3)*.

*Figure 3: Transformations on AccountBalances data*

* **JoinCustomerAndAccount()** gets transformed data from TransformCustomers() and TransformAccountBalances() and joins these two sets of data *(Figure 4)*



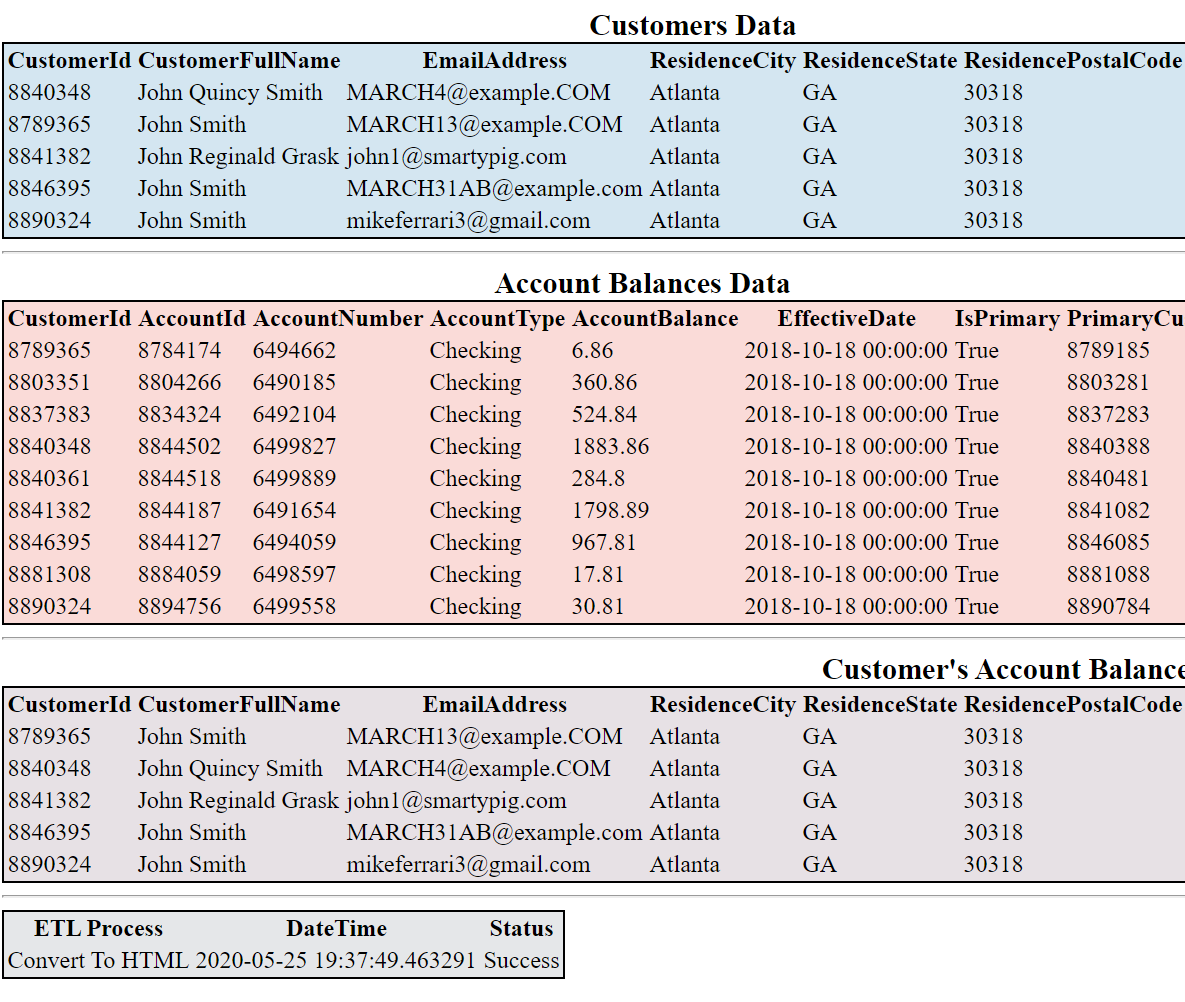
*Figure 4: Relation between Customers and AccountBalances data used for joining the data*

# Load Destination Data

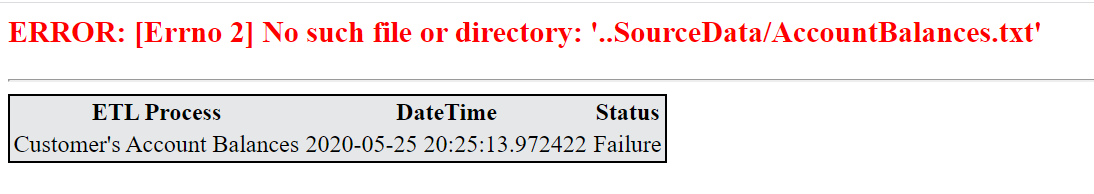
Process of loading data is written in Python **LoadScript.py**. Connection string for mongoDB is wrapped in **getMongoConnectionStr()** function so that maintenance will be easier.

Loading to MongoDB is done in **Load2Mongo()** function using **pymongo** library. This library contains tools for working with MongoDB and is the recommended way to work with MongoDB from Python. Destination Collection is **ClientAccountBalances.ClientBalancesData**. This function returns number of successfully inserted rows or error description in case of failure.

Function **startETLprocess()** is calling Load2Mongo() and checks the type of returning value. To execute the whole ETL process just run **LoadScript.py** In case of success generates HTML report with transformed and inserted data *(Figure 5a)*. In case of failure generates HTML report with error description *(Figure 5b)*.

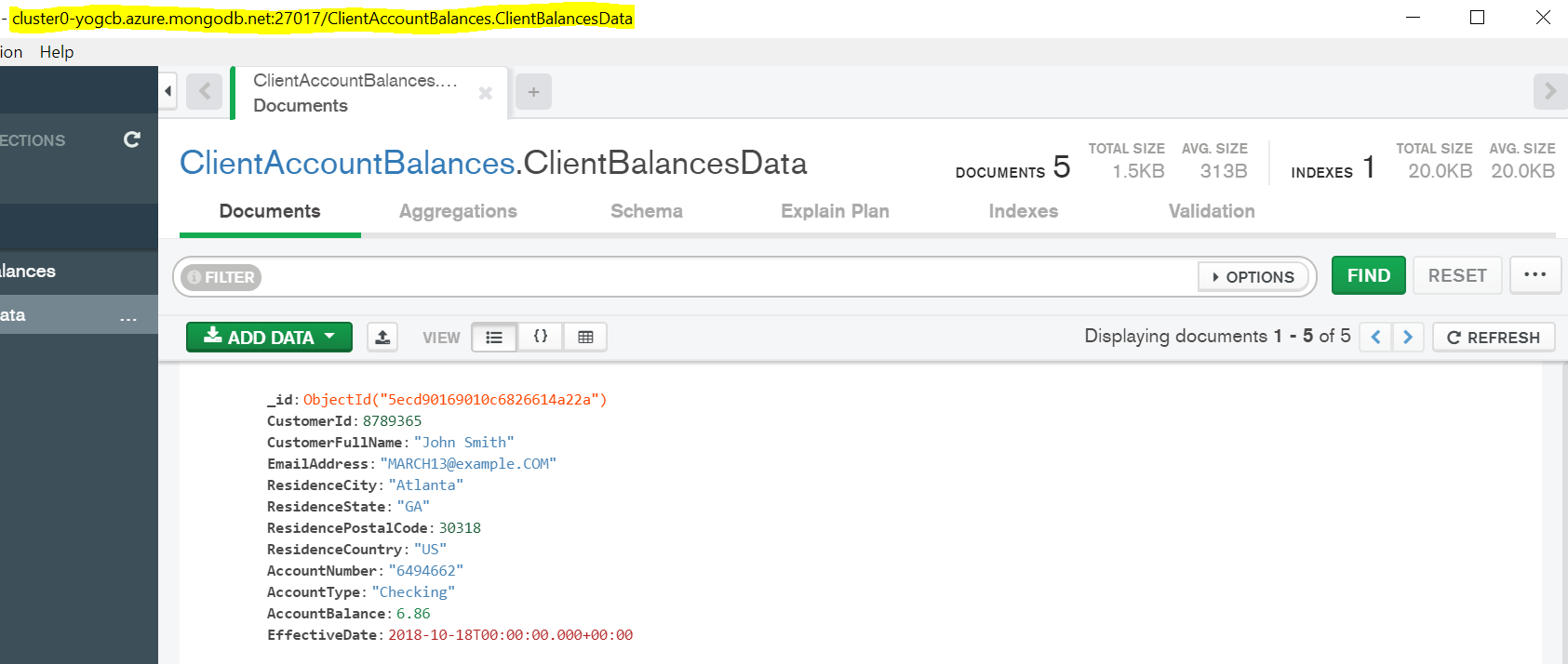
**

*Figure 5a: Report in case of successful execution*



*Figure 5a: Report in case of error*

After successful execution of the scripts, we can see is **ClientAccountBalances.ClientBalancesData** collection is created in MongoDB *(Figure 6)*. By comparing the generated report and documents in the MongoDB we can check if everything is right.



*Figure 6: Created collection and loaded data into MongoDB*

# Summary

Python is an elegant, versatile language with lots of powerful modules and code libraries. Writing Python for ETL starts with knowledge of the relevant frameworks and libraries, such as libraries for accessing and extracting data, connecting to destination database. Python has ability to connect to MongoDB using the standard connection string URI format, can read and write whether single node, replica set or shared cluster.

Without all available third-party python libraries, it would take a lot more time and effort to create an enterprise level application for the No-SQL ETL process.