National College of Ireland

Monitoring and Maintenance of Datasets Integrity of a Cloud Database instance using Machine Learning

MSc Research Project
Cloud Computing

Samuel Das
Student ID: x17119642

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Samuel Das
Student ID:	x17119642
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	20/12/2018
Project Title:	Monitoring and Maintenance of Datasets Integrity of a Cloud Database instance using Machine Learning
Word Count:	9862
Page Count:	32

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	27th January 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Monitoring and Maintenance of Datasets Integrity of a Cloud Database instance using Machine Learning

Samuel Das
x17119642
Master of Science in Cloud Computing

27th January 2019

Abstract

The database systems associated with the cloud computing platform is experiencing exponential growth rate at the cost of increasing data redundancy and data integrity issues. Various data mining tools can operate on these issues for huge datasets for learning of various issues and later other systems can act upon the learnings and stabilise the datasets. In this research study a novel approach is used to help maintain the dataset integrity based on using standard data mining algorithms. So, an automation of monitoring and maintenance of datasets on large scale distributed cloud-based database systems.

Contents

1 INTRODUCTION

The motivation for this research came from the various articles and books which highlighted the issues of database integrity on the cloud database architecture. The database systems growth in many ways have been driven by main sources such as the internet, complex applications and hardware advances. Bernstein et al. (1998)
Hence monitoring for a database service is important especially which are scaling up very fast due to the above three drivers. As stated by Kai Hwang (Hwang et al.; 2013) that "Data consistence checking in SAN-connected data center is a major challenge in cloud computing." (Hwang et al.; 2013) The fact that the cloud database is growing fast in volume and services warrants an effective monitoring and maintenance system for the cloud database architecture which will protect its dataset from being getting corrupt and maligned. Cloud monitoring as understood in the industry, is a very broad term which encompasses many aspects of services from VM monitoring to monitoring of multiple dependent services and systems. Toosi et al. (2014).
The cloud platform adds to the complexity of the data Database management system (DBMS) by using data locations. So, the database schema integrity constraint (IC) becomes questionable and hence this research is the need of the hour.
So, to answer the existing data integrity issues this research endeavours to find the solution to this problem which is currently existing in the real-world database storage, monitoring, maintenance and access environment on the cloud.

The research answers to the question of how to maintain database integrity on a cloud platform using data mining techniques. The research question deals with the important issue which are present in the present world of cloud database architecture and are very important to address because these issues are directly related to the database operations

and dataset integrity. Hence the research question tries to answer how a database integrity could coexist dynamically for an ever-increasing volume of datasets and will it be possible to dynamically address the issues of lack in dataset integrity by dynamic detection and correction of the inconsistent datasets.

To solve the research problem, it is important to have various views and perspectives. The different views will show and highlight root cause of various types of corruption of the tuples in the database. Hence considering the various root causes analysis from various incidents and the understanding of the issue it can be summarised as follows.
The current cloud database architecture promotes large set or number of database services across many different geographically separated data locations.
A foremost observed challenge is in the fact that most cloud-based databases always rely on the replicated data locations across many geographical locations which introduces more challenges to find and sync all the copies of the dataset. Xu et al. (2017)
The existing DBMS in the industry have datasets normalized in them and so the datasets definition and its values were in or out of the dataset very clearly, but presently the is a growing number of cloud database architecture design which considers incomplete datasets, and this contrasts with the traditional DBMS systems. (Widom; 2004) Due to the nature of the cloud the databases are bound to be distributed and hence, problem is bound to happen which is as stated by Terry et al. (2013) that data co-location and data replication across data locations increases complexity with the "trade-offs between performance and consistency are accentuated due to the high communication latencies between datacentres." Terry et al. (2013).

Secondly referring to the work of Chu et. al. (2013) we understand that "Data cleaning is an important problem and data quality rules are the most promising way to face it with a declarative approach. Previous work has focused on specific formalisms, such as functional dependencies (FDs), conditional functional dependencies (CFDs), and matching dependencies (MDs), and those have always been studied in isolation." Chu et al. (2013)

Thirdly referring to the work done by Chaudhuri et. al. (2005) in software systems, the applications or databases, replicates or moves the data from its initial position in the database system many times, also it gets transformed and possibly copied or moved to a destination which might be a separate schema or table in the same database or separate database. Chaudhuri et al. (2005).

Fourthly, as the data moves from a source to a destination schema it may more likely take several steps and movement is done by every CRUD database operation. According to the work of Cui et. al. (2003) those datasets which are used by applications are exposed to lot of translations and transformations operations whose characteristics will "vary from simple algebraic operations or aggregations to complex procedural code." Cui and Widom (2003).
As stated by Shin et al. (2016) "The ongoing explosion in the diversity of memory and storage technology has made hardware heterogeneity a fact of life for modern cloud storage servers. Current storage system designs typically use a mix of multi-device idioms such as caching, tiering, striping, mirroring, etc. to spread data across a range of devices, including hard disks, DRAM, NAND-based solid-state drives (SSDs), and byte address-

able NVRAM or Phase Change Memory (PCM). Each such storage medium exhibits vastly different throughput and latency characteristics; access latencies to data can vary considerably" Shin et al. (2016), so the data movement leaves the datasets vulnerable to corruption.
Hence as stated by Shin et al. (2016) "creates an opportunity for trading off consistency or staleness for performance."

Finally, considering the cloud-based database architecture the database services when executing on various database instances in a virtual environment have a good chance to allow attacks from malware, trojan and viruses. Hence there is a need to protect the database from these incorrect updates that takes place because of multiple user client access and compromised database security.
Hence consistency and performance are both challenges as suggested by Lang et al. (2015) who states that "increasing the operational efficiency of the service generally means increasing the utilization levels of the service by co-locating users together onto fewer physical servers, while high, stable performance is generally achieved by isolating a user onto their own server (potentially very wasteful)".Lang et al. (2015)

So, from above mentioned issues it is understood that the issues are all related to the integrity of tuples. Hence it is quite well understandable that datasets have a chance of getting corrupt in due course of time due to CRUD operations on them.
So, from the above stated points it is very clear that the problem of corrupt datasets is a real threat to database and computational integrity.

This research answers the problem very clear that there is a greater need to monitor and maintain the database and computing integrity for the cloud services and cloud environment.
Hence from the above discussion the research question comes out as, how can dataset or tuples integrity be monitored and maintained of a large-scale database on cloud environment?

The document is divided into sections which are listed as follows.

- "Introduction" section gives a brief idea about the research study.
- "Research Question" section states the research question and the justification for it.
- "Literature Review" section states the architecture and design of the solution.
- "Methodology" section states the architecture and design of the solution.
- "Implementation" section describes the methodology in terms of software tools and low-level design specifications.
- "Evaluation" section describes a systematic analysis of experimental activities and outcomes of the proposal so that decisions can be made about the proposal regarding the improvement and effectiveness.

- "Conclusion and Future Work" section describes the research outcomes and discusses of the implications and the possibility of new research in the future based on this research outcome.

2 Literature Review

2.1 Matching Dependencies methodology for correcting corrupt dataset

The matching dependencies methodology was developed by Bertossi et al. (2013)
The matching dependencies methodology searches and finds the incorrect tuples from the schema of the same use case or business logic. The methodology then repairs by using the integrity constraints as defined earlier in the schema.
As stated by Bertossi et. al. (2013) the methodology "introduced matching dependencies as declarative rules for data cleaning and entity resolution." Bertossi et al. (2013)
The methodology does a matching under forceful circumstances of the dependencies, on the same schema instances under investigation before correcting them. To put it in a easy way around for understanding it can be said that the data correction action can take place for a tuple, only in the case of the tuples belonging to the same business scenario and where these tuples have their attributes very closely matching. Mandros et al. (2017) mentions clearly the difficulties of this methodology as stated here "Given a database and a target attribute of interest, how can we tell whether there exists a functional, or approximately functional dependence of the target on any set of other attributes in the data? How can we reliably, without bias to sample size or dimensionality, measure the strength of such a dependence?" Mandros et al. (2017).
From the above discussion the methodology lacks with two limitations which are user dependency and use case dependency. Hence there is a need of a better methodology which should not be constrained to user and user business scenario.

2.2 Cell Group methodology for dataset recovery

Geerts et al. (2013) came out with this methodology of grouping cells in tuples for dataset recovery.

The cell group methodology main task is to remove inconsistencies and consequently clean the cells or tuples. The methodology requires to learn the database schema constraint definitions, to perform both the task of modification of the incorrect tuples and also replacing with a new value similar to the original value.
The cell group methodology has a separate schema tuples cleaning process. This cleaning process can be used only when there is a set of two schemas of belonging to the same database. The schema is known as source schema and target schema in this process. Both these schemas are defined along with the group or set of related constraints. The function of the source schema is to contain the master dataset which is the clean datasets. This source dataset is an input to the correction process. The source schema also considers the related schema constraints to analyse the master and target dataset for inconsistencies. The other dataset is referred to as the target schema which are having a set of incorrect datasets in it that needs to be corrected. To execute the process of corrections of the dirty datasets the following steps must be done. The with steps of evaluating and comparing relative values of the dirty tuples and track the changes in the relationships of

the source and target datasets that are being changed the final correction of the dataset are completed in this methodology.
The cons of this methodology can be found out from the two facts as follows. The most important fact about this methodology is that there is an initial requirement of the details of the internal schema definitions of the datastores which cannot be directly accessed. Secondly, the datasets are replicated many times across the data locations to add to the initial problem. Zellag and Kemme (2012).
Hence as stated by Geerts et. al. (2013) the uphill activity is to create a process which can correct the incorrect tuples without any user involvement, and hence "identify and repair data errors in a dependable manner." Geerts et al. (2013).
As a conclusion it can be well understood that having user intervention for correction must be reduced to minimum and that is what is required for better user experience. It can be said that by using data analytics along with the data mining algorithms, the user interface can be reduced to the minimum.

2.3 Database integrity thru Fuzzy Logic technique

A very popular method used by Microsoft SQL services was developed using fuzzy logics by Arasu et al. (2011).
The methodology searches for the incorrect tuples by using a look up logic based on fuzzy logic. There are assumptions that needs to be made for the search algorithm to work which is the limitation of this methodology. Hence to find the corrupt data using the fuzzy logic method there must be a matching making of the datasets at first and then upon the results of the match making process the tuples are identified as incorrect. Then upon the tuples must be segregated so that the values of the cells of the tuples will be exposed to run a segregation logic on them so that the values which are incorrect can be categorised. For this categorisation of the cell values, the cell data are tokenised. The token generation logic is weighted and hence requires the tokenizing techniques like whitespace-based tokenisation and the various transformation rules. Arasu et al. (2011)
Hence as stated above, the methodology still has the challenging task for the user where the user must define the tokenisation model. The tokenisation model is built on the transformation rules which also intern has to be defined as well. Now the user has to have a detail view of the dataset under the review only then can the tokenisation and the declaration of the transformation rule can take place. Hence there is the challenge for defining different transformation rules for different types for data.
The main task of developing the transformational rules as per the data integrity violations requires different analysis for different data types and datasets creates a challenge. Secondly the violations of the data integrity constraints do not hold true and valid for many reasons because the data integrity depends on the business model of the database. Assadi et al. (2018).
Ultimately the user takes the decision to configure the tokenisation and further analysis operations of the dataset, this creates a grey area for the entire process of correction of the dataset. This may also not lead to the correct output of the dataset even after the correction is performed at the initial steps and hence may take more complex steps to reach the final correct data values of the tuples. So, though the methodology depends on the fuzzy logic techniques, but still user has to be in control for the configurations. The fuzzy logic techniques establish the rules for the lookup using various schema reference tables. But the main process of lookup and picking up the correct tuples for correction

must be done over multiple steps by establishing a threshold parameter for the continuing the lookup process and then setting up an index over the reference tables. The aggregation and segregation of the intermediate results also adds the complexity of the steps that need to be followed until the result is achieved.
There is lot of dependencies as well on the interim new schema setup for the analysis and the structuring of the reference dataset to enable the extraction query string. Hence Arasu et al. (2011) methodology demands for a highly skilled user for arriving at the correct results.

As discussed above in three sections the three different methodology has been successful. But still the challenges of the requirements still stand tall because of the ever-increasing datasets size and volume and the dynamics of the accessibility of the datasets especially in the cloud database architecture where the database architecture has to deal with the cross platform solution for federated cloud environment which is clearly stated by Moustafa et al. (2015) "as monitoring in federated cloud environments needs a flexible and efficient framework that combines a number of essential properties such as interoperability between different clouds, SLA orientation, and service benchmarking." Moustafa et al. (2015)

The current progress in the cloud services related to database architecture indicates that the there is more room for automation in the process of the data integrity checks and monitoring, hence there is better chances of using latest technology like data mining techniques to process the dataset in the transactional database. Makris and Markovits (2018)
The latest technology of data mining helps to analyse the dataset under inspection in a user-friendly manner. Even if the heterogeneous dataset is considered which are presently the normal of the business model across different cloud platform and heterogeneous computing infrastructure. Coady et al. (2015)
Also, the modern technology of data mining is a tool for predicting the close approximation for a given set of similar values. Using the popular Kmeans algorithm "to impute missing confirmation bias metrics values" and then by using this values there is a possibility to get the right value too. Calikli and Bener (2013)
Hence the Kmeans algorithm can be of good help to go thru the automated iterative process and then get a series of outputs in a scientific way.

3 Methodology

This research is related to the cloud database instances which are holding huge datasets and are also at the same time vulnerable to external and internal attacks or hacking which in many cases alter the dataset values as registered or updated by the main application with which the database is associated. Hence the dataset integrity is lost in these kinds of attacks and hacking so to reinstate again the correct values for those corrupt tuples or dataset we have introduced a novel approach to monitor and correct the dataset continuously with the use of data mining algorithm and techniques and machine learning advantages.

The experiment done has been explained here in a structure, which are described in

each separate sub sections under different headings.
Various UML diagrams have been used here to describe the designs, components, data flow, data set, deployment and execution strategy.
The below sub sections contain the details of the implementation of the proposed design, proposed data flow, experiment steps and the experiment results.
The experiment ends with the test scenarios, test cases and the test results which are part the experiment execution with the objective to prove the research theory.

The solution architecture layout for the project encompasses cloud database instances across different data locations this is achieved using AWS DynamoDB instances created across multiple AWS regional server locations, an web application to processes the data to and from the cloud database and also apply the data mining and data learning algorithms to process the dataset for the detection of the incorrect tuples and also generates the most probable outcome of the in-putted incorrect values.

Here in this proposed architecture we are doing this experiment using two separate use cases and which also have two separate real-world datasets and database instances and set-up in the AWS DynamoDB cloud servers. The user will be allowed to run the datasets for detection of the incorrect values in the datasets and then be prompted for correcting the data values as per the data mining and prediction algorithm outcome. In this current experiment we are trying to prove a working model which will execute the requirement specifications of this research project.
The following is the architecture diagram:

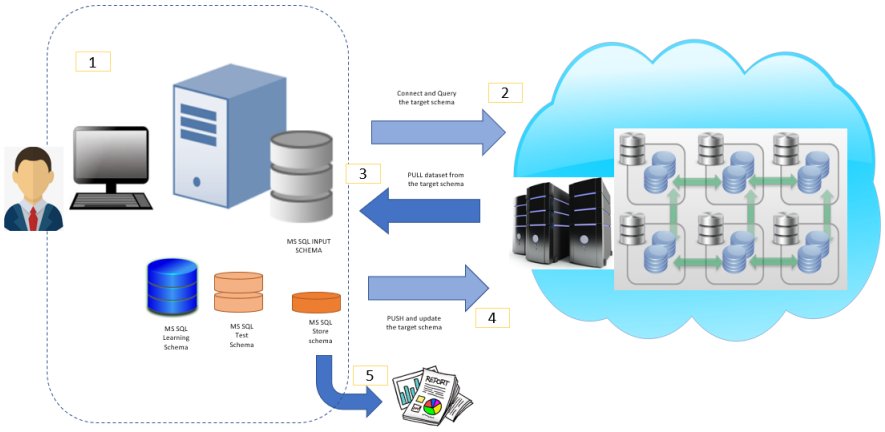


Figure 1: The Component diagram

The figure 1 diagram displays the overall architecture layout for the proposed solution. The entire research experiment is composed of 4 steps as per the data journey thru the system and its components. They are as follows:

- **Step 1:** This is the first process of the experiment. The user connects to the cloud database which is considered as the target database for the project. The target database which is on AWS DynamoDB domain and server has to be understood in detail firstly. The table and schema credentials and the schema layout along with the table specifications must be available to the user. The user notes the schema table layout and connects to the AWS database instance. After successfully

connecting to the AWS database instance the dataset which is targeted to have the corrupt data in it is extracted by the user. The dataset which is extracted is a batch of 5000 tuples. This set of 5000 tuples are required to train the data mining algorithm for finding the deviations in the values of those incorrect tuples and hence identify the tuples also at the same time. The snap shot of the dataset will include the incorrect dataset in it. Also, the dataset snapshot will have only the related and affected attributes of the dataset.

- **Step 2:** The second process of step is to populate the temporary local database for holding the dataset separately away from the source database for analysis. Here the dataset is not the exact replication of the dataset but is the set of those attributes which are identified as the qualifier for the attribute which is under scrutiny. So only those dependable attributes must be considered for the analysis to identify the incorrect tuple. The dataset record or tuple are observed for the OK and NOT OK attribute values that will be required for the data mining to understand and learn the dataset characteristics.

- **Step 3:** The third step in the entire process runs the data mining algorithm on the dataset. The execution of the data mining algorithm will run thru the entire dataset. The algorithm will then note the deviation from the range of values that the same attribute possesses. Any deviation from the range of the values of the target attribute is identified as a potential error in the value and the tuple is identified as an error tuple which is present in the dataset. The identified errors are displayed to the user for a view and better understanding of the content and context of the dataset and its values that need amendment.

- **Step 4:** The fourth step in this process refer to the prediction of the correct values for the incorrect values. The data mining algorithm will use another algorithm called k-nearest neighbour algorithm in this case. This algorithm will take into input the incorrect dataset and run through them. After analysis of the incorrect dataset this algorithm will predict the correct values for the attribute. The predicted values may be 100 percent accurate or may be less than 100 percent accurate.

- **Step 5:** This fifth step is responsible for storing the corrected tuple in the system. This step is very much required for making the system more intelligent and at the same time to keep a record of this correction activity. The library maintains the copy of the original record and the copy of the corrected record separately in itself.

- **Step 6:** The sixth step in the process is to update the source database schema on the cloud with the correct values.

- **Step 7:** The final step in the process is to generate the report of the dataset.

From the above section it is clear now that the dataset employed in the research project is related to different use cases or business scenarios. This difference enables to understand that the proposed solution is not limited to one business or use case only. The proposed solution can be used for any different business case and will be equally effective to solve the problem this research wants to achieve. The following section describes the dataset used in this research project.

3.1 Design and Data

The research project design is done with the minimum complexity and minimum component interfaces so that the data operations and data movement remain the crucial activity for the user to control the various steps in this process. The research project solution deals with the relational dataset. This is because of the less complexity to deal with in the execution of the program and project due to limited time frame. This section highlights the data flow and the design which will be implemented in this research project.

3.1.1 Design

The low-level design and internal architecture are well explained by the diagram below. The Unified Modelling Language (UML) is the industry standard for describing the software engineering model. UML can be used to describe the low-level design which consists of components and processes. The low-level design is based on the high-level architecture diagram. The low-level design describes more of the software model, which can will be used for the software development, engineering and testing. Hence the below diagram uses UML, for identifying the different processes, module and data transfers as per the architecture component diagram presented in figure 1.
The following is the Low-Level Design diagram:

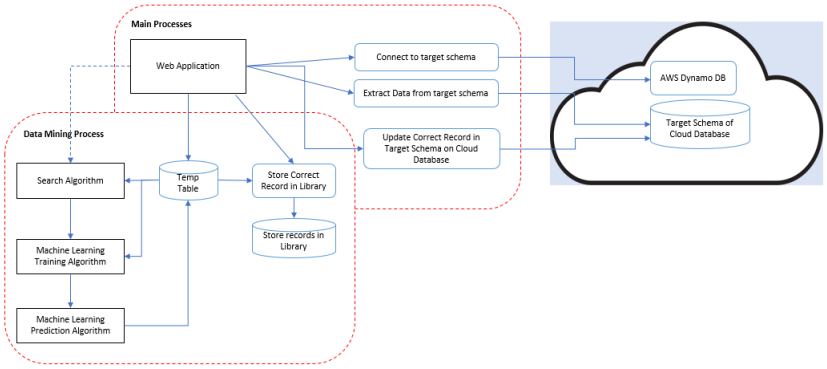


Figure 2: The LOW-LEVEL DESIGN Diagram

Here in our system we provide a solution which is user friendly. We have developed the solution around the data movement in various phases of the entire process under the user commands.
The solution targets to correct the incorrect tuples in the dataset, hence the user connects to the database after getting the details of the database from the AWS console. The database is spread across different data locations. The data location instances are also received from the AWS console.
We have used a web application for user control of the sub-processes. The web application has a dashboard and menu options to show the various activities of the system.
The data mining sub-process takes the dataset for analysis. After the analysis the defective tuples are highlighted in result of the analysis process. The resultant tuples are shown on the dashboard to the user.
The user has the dashboard where the user can select and deselect the tuples for corrections to be performed.
The user then goes to the next step of sending the incorrect tuples for analysis so that the

correct values are suggested by the data mining algorithm. After the dataset has been passed thru the data mining algorithms then the corrected tuple is copied into a library database for future references.
The design gives the user the control on the data movement selectively.

3.1.2 Data

The sample dataset which we have considered is of a simple real-life use case. A simplistic dataset will help to have a better understanding of the research objective and outcomes with complete clarity and no complexity in the outcomes of the research. This research is based on the dataset primarily, hence the data source, data transfers, data CRUD operations and data archiving is of importance with respect to the relational dataset which are from the real-life scenario. The dataset is of a credit card transaction of various customers. The credit card dataset has attributes with numerical and alphanumerical type values in them. We have considered the numerical values in the attributes to be the direct inputs to the data mining algorithm. The user can select other attributes as well like of type alphanumerical. The data noise which generated and introduced in the dataset is hand-crafted and not real.
The following is the Dataset attribute details:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2
1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102
2	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725
3	90000	2	2	2	34	0	0	0	0	0	0	29239	14027
4	50000	2	2	1	37	0	0	0	0	0	0	46990	48233
5	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670
6	50000	1	1	2	37	0	0	0	0	0	0	64400	57069
7	500000	1	1	2	29	0	0	0	0	0	0	367965	412023
8	100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380
9	140000	2	3	1	28	0	0	2	0	0	0	11285	14096
10	20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0

Figure 3: The Dataset attributes

For the data mining algorithm to work we have to take help of the data model or the classifications of the data. The classification of the data has been into two classes which are OK and NOT OK.

3.1.3 Assumptions

This research is based on the issues that are faced by large distributed databases present in the cloud environment. The few assumptions in the methodology are as follows:

1. *From the considered dataset for this experiment, the dad or corrupt tuples are expected to be about 1 tuple per 100000 tuples.*

2. *Only relational schemas and databases are considered to minimise complexity of the logical operations*

3. *Low Cardinality of the dataset in the database so that the search is simple.*

4 Implementation

We present our solution by implementing data mining algorithm and techniques for dataset analysis and predictions. The solution implements AWS cloud platform for hosting the datasets in the database instances which are spread across 3 different data locations.

We selected the target database to be hosted on the AWS Dynamo DB platform as its being the popular cloud database platform. We also implement various AWS public cloud environment-based database services to apply the extraction and various CRUD operation on the target database on the cloud.
For better user control and understanding we have divided the entire process into two sub processes which are a) the discovery process and secondly b) the correction process.

4.1 Web application and Dashboard

We have implemented a web application Java Spring hibernate framework here for the user. The spring framework provides the open source Java based web applications related APIs and database connection APIs. The choice for Java spring framework is because of it light weight and as it's an open source for development of Java applications and web applications. The hibernate provides the database APIs for MS SQL databases which are required to process the dataset in the local database tables.
We have implemented a Java based web application for the user to be able to have a user access to configuration and control of the multiple processes working in the system. The user is also able to execute the processes and modules as a part of the manual steps in the system. The web application has user friendly menus for triggering different functions and a dashboard for the system control, error messages and execution.

4.2 Detection of error tuples from the target cloud database

We have implemented this detection process as the first function or activity of the system. This is done to collect the dataset from the target database on the cloud and initialise the system for the data mining algorithms to run on them. Hence the target database is identified on cloud platform. Here we have selected AWS dynamo DB as a cloud platform. On the AWS dynamo DB, we have installed a target database It is also assured that the cloud database has the dataset across the multiple data locations or regions.
The application connects to the AWS database instance using a TCP/IP connection. The database credentials are passed to the connection Java API for the database connection. Once the connection to the cloud database is successful then the dataset is extracted from the targeted datasets or the targeted tuples from the target cloud database. This extraction of the dataset from the target database is done for large set of tuples and stored in a file having a csv format.
The dataset selection process checks the user definition of the target dataset. Then the database related IC rules, de-duplication rules are considered to extract the information related to each tuple. So, after getting the normalisation dependencies we pool all incorrect tuples. The following logical steps are performed in the detection algorithm.

- Extract tuples as per the target dataset definition from database nodes in different data locations.
- Check for inconsistent IC.
- Save to a temporary learning repository in a separate local storage.

The large quantity of the tuples or dataset includes both incorrect tuples and most correct tuples, each of the category having a ratio of 1 percent and 99 percent respectively. Hence the bulk of the dataset is correct dataset. This set of correct datasets is required

for the training the data mining algorithms to identify which are the correct values in the tuple and with what range the values of the attribute varies. Hence the data mining algorithm can detect the patterns and the ranges of the values of the target attribute of the target dataset.
The extraction of the target database also includes the information on the primary keys and foreign keys defined in the database. The primary key and foreign keys of the different target table set helps to identify the relationship of the data attributes and their uniqueness and hence we can check the integrity constraints (IC) of the tables and their attributes. These IC checks are done with the help of the primary key and foreign keys for all the tuples in the extracted dataset.
This reverse IC checking identifies the OK tuples and the NOT OK tuples. This set of the most probable OK and NOT OK classification of tuples are fed into the system for the initial training set. The training set is required for the identification, categorization and learning by the algorithm. This training set of tuples are stored in the database separately as shown below in the diagrams.

The following is the Learning model details:

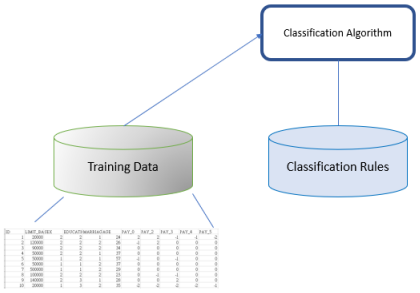


Figure 4: The learning model

The learning model give rise to the learning phase in the system. In the learning phase mentioned here the classification algorithm is responsible classify the data into OK or NOT OK data sets or groups. After the learning phase the classification rules are applied on the tuples which are scrutinized for errors in them and stored separately in the local database table.

The following is the selection model details:

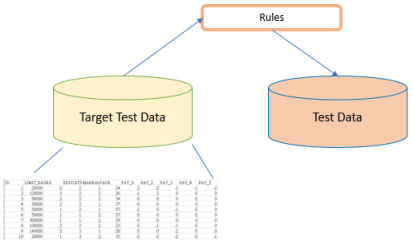


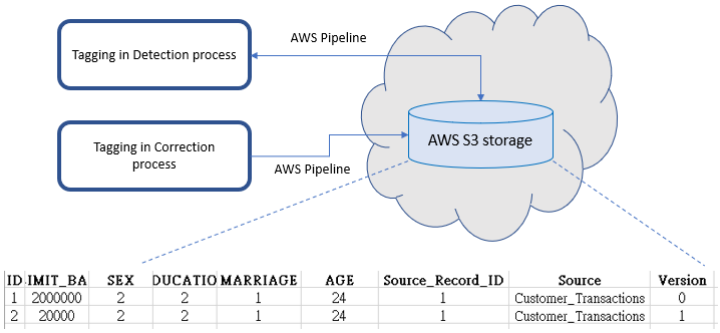
Figure 5: Tuple selection process

This selection rule accuracy is judged as per the selection dataset outputted on the

dashboard. The user then decides which all dataset can be taken forward out of the NOT OK test dataset. Hence the clustering of the test dataset is done to isolate the corrupted tuples. This results in a pool of corrupted tuples which are stored separately on the local database table.

4.3 The library for Records

The error tuples which need correction will be stored in a separate table which is a Library-of-Records (LoR). This LoR storage has been created to keep a track of the tuples which are getting identified for the correction process later on.



ID	IMIT_BA	SEX	DUCATIO	MARRIAGE	AGE	Source_Record_ID	Source	Version
1	2000000	2	2	1	24	1	Customer_Transactions	0
2	20000	2	2	1	24	1	Customer_Transactions	1

Figure 6: The Data Flow details for LoR

The tuples are also kept here with a motive to refer them in the future for tracking and log purpose as well.
In order to store the tuples here in this library or repository we have devised a mechanism to uniquely identify the tuples as well. Hence each tuple's information is also stored along with the tuple data. we have captured the tuple source table information like table name plus the record details together in a single new tuple. Each tuple is store also with a version tag. This tag is a numerical value which starts from 0 and increases by 1 for every single action done on it.
So, the first selected tuple which contains the error in it, will be tagged as version number 0. When the correction is done on it, then after the tag value will be changed to 1 value or version number 1. Both the versions of the record will be available in the LoR table. This is because to trace the original record and also to trace the various versions of the record that has undergone changes or corrections thru this process.

4.3.1 Pseudo-code for LoR

The LoR stores the records with all the versions as its clear from the discussion.

Data: The identified record from the target database on cloud.
Result: Different versions of the record undergone correction

```
   /* Initialize the counter keeping variables                    */
1  versionId = 0;
   def get(dataSet, recordNumber)
   /* Store Initial base version of the record                    */
2  iterations = getVersion(recordNumber);
   while not getRecordSet(dataSet, recordNumber) do
3  |  iterations ++ ;
   |  storeVersion(record,recordNumber, versionId);
   end
4  return records
```

Algorithm 1: Algorithm to store records with versioning.

4.4 The correction of error values for the attributes under investigation

We have done this solution to get the correct value of the incorrect data value or a data point of a target attribute by predicting the correct value using K-Means data mining algorithm. We have chosen K-means over Expectation-Maximisation (EM) algorithm due to the simplicity and linear nature of the iterations having a linear complexity O(n).
The correction process involves two steps. The first step is to select the seed value for grouping the data points and the second step an automatic iteration process using the k-means algorithm which runs until the new data points are discovered or the group center is found out. These new data points are the predicted and correct values. But then out of the three group center or new data point results, only one value is chosen by the measuring the average of the three new group centres.
The assumptions for the correction process to run are

1. The seed value is random within the range of the data points collected.
2. The K value or the number of clusters is set to be constant value of 4 for this research purpose. The more this k value is the more accurate results can be obtained and also more data points can be considered for the iterations. But here K values is always less than the number of data points (P) i.e. K less than P.
3. The data points are the actual values from the targeted table and the dataset extracted for this correction process is 100 data points. Out of the 100 data points it assumed that 99 data points are correct data and only 1 data point is incorrect.
4. The data points which are a set of numerical values of the same attribute hence they are similar to each other. So, no labels are required for the K-means algorithm to do the clustering or grouping.

The clustering of the data points will be based on the training data sets $X^{(1)}$.
The given data point $x^{(1)} \epsilon R^n$. The solution algorithm follows with the following steps for an unsupervised learning to get a k centroid value.

1. Seeding the cluster centroids with values α_1 , α_2, α_3 and $\alpha_4 \epsilon R^n$ randomly.

2. K-Means algorithm Iteration over the $X^{(1)}$ data points till centroid is found:
   $\left\{$
   For every data point $x^{(i)}$, set $centroid^{(i)}$:= ARG MIN || $x^{(i)}$ - α_j $||^2$

   For every j in α_j ,
   set $\alpha_j := \frac{\sum_{i=1}^{n} 1(centroid^{(i)}=j)x^{(i)}}{\sum_{i=1}^{n} 1(centroid^{(i)}=j)}$.
   $\right\}$

4.4.1 Correction Pseudo-code

The K-means algorithm functional calls are described as follows:

Data: 100 data points extracted from the target database on cloud.
Result: 4 different centroid data points

```
1  def kmeans(dataSet, k)
   /* Initialize centroids randomly                                    */
2  numFeatures = dataSet.getNumFeatures();
3  centroids = getRandomCentroids(numFeatures, k);
   /* Initialize the counter keeping variables                         */
4  iterations = 0;
5  old-Centroids = None;
   /* Run the main k-means algorithm                                   */
6  while not stopIterations(old-Centroids, centroids, iterations) do
7  |   old-Centroids = centroids;
8  |   iterations ++ ;
9  |   labels = getLabels(dataSet, centroids)
10 |   centroids = getCentroidValue(dataSet, labels, k)
   end
11 getLabels(dataSet, centroids);
12 return centroids
```

Algorithm 2: Algorithm to find correct values.

Below is the called function definition for getting the cluster details and also the

function for getting the centroid values respectively.

Data: Testing Data set for the clustering $X^{(1)}$
Result: The centroid values $centroid^{(i)}$

```
1 def stopIterations(oldCentroids, centroids, iterations)
  /* Function:  Put a stop to Iterations for clustering          */
  /* Returns True or False when the process of finding k-means
     iteration is complete.                                      */
  /* K-means loop will terminate on the condition that if it has run
     a maximum number of iterations OR the centroids stop changing.
     */
  /* When number of iterations is greater than MAXIMUM number of
     itertions then exit loop                                    */
2 while iterations < MAXIMUM − NUM − itertions do
3 |   calculate centroid value from the cluster of data points....
4 |   oldCentroids == centroids
5 |   return true
  end
```

Algorithm 3: Function definition for clustering Iterations.

Below is the called function definition for getting the cluster details and also the function for getting the centroid values respectively.

Data: Testing Data set for the clustering $X^{(1)}$
Result: Returns a label for each data point in the testing dataset

```
  /* For each data point or element in the testing dataset, find the
     closest centroid then assign that centroid the data point's
     label.                                                      */
1 def getLabels(dataSet, centroids)
```

Algorithm 4: Function to get cluster.

Below is the called function definition for getting the cluster details and also the function for getting the centroid values respectively.

Data: Testing Data set for the clustering $X^{(1)}$
Result: The random K centroid values $centroid^{(i)}$

```
  /* Every centroid value found is a geometric mean of the data
     points that have assigned centroid label.                   */
1 def getCentroidValue(dataSet, labels, k)
```

Algorithm 5: Function to get centroid values.

4.5 Sequence Diagram

The sequence calls of the entire process is captured on the sequence diagram. This diagram helps the user to understand the user case and the related processes which are triggered in the system sequentially for the completion of the process.

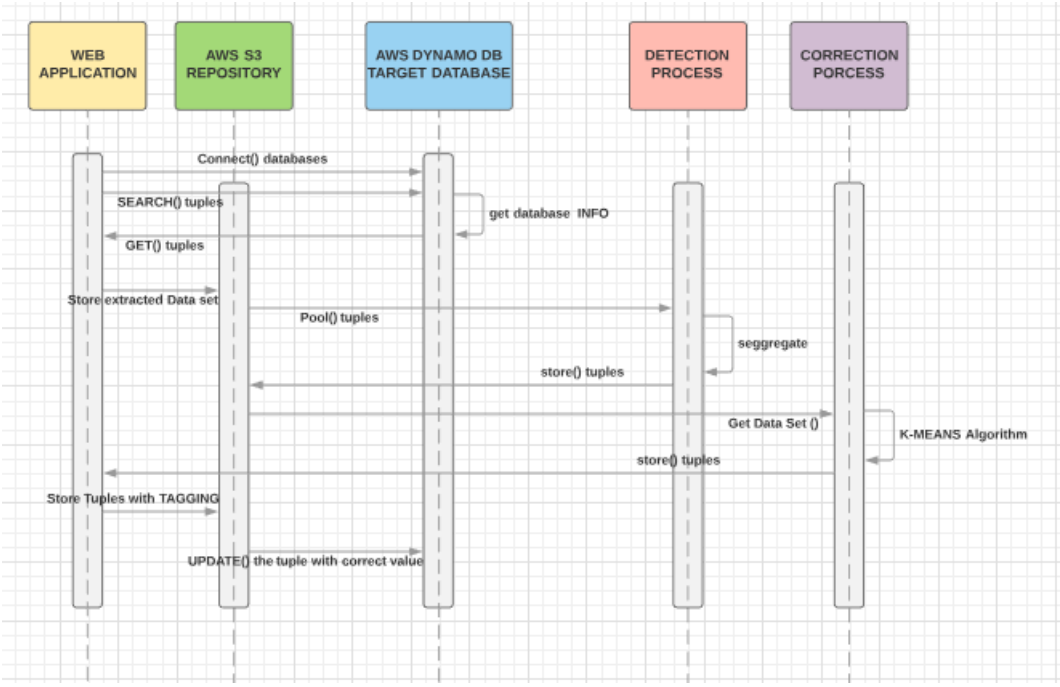


Figure 7: The Sequence diagram

4.6 Data Flow Diagram

We have captured the objects and their relationships in the diagram below. The object diagram here shows the relationships of the functional component of the system.

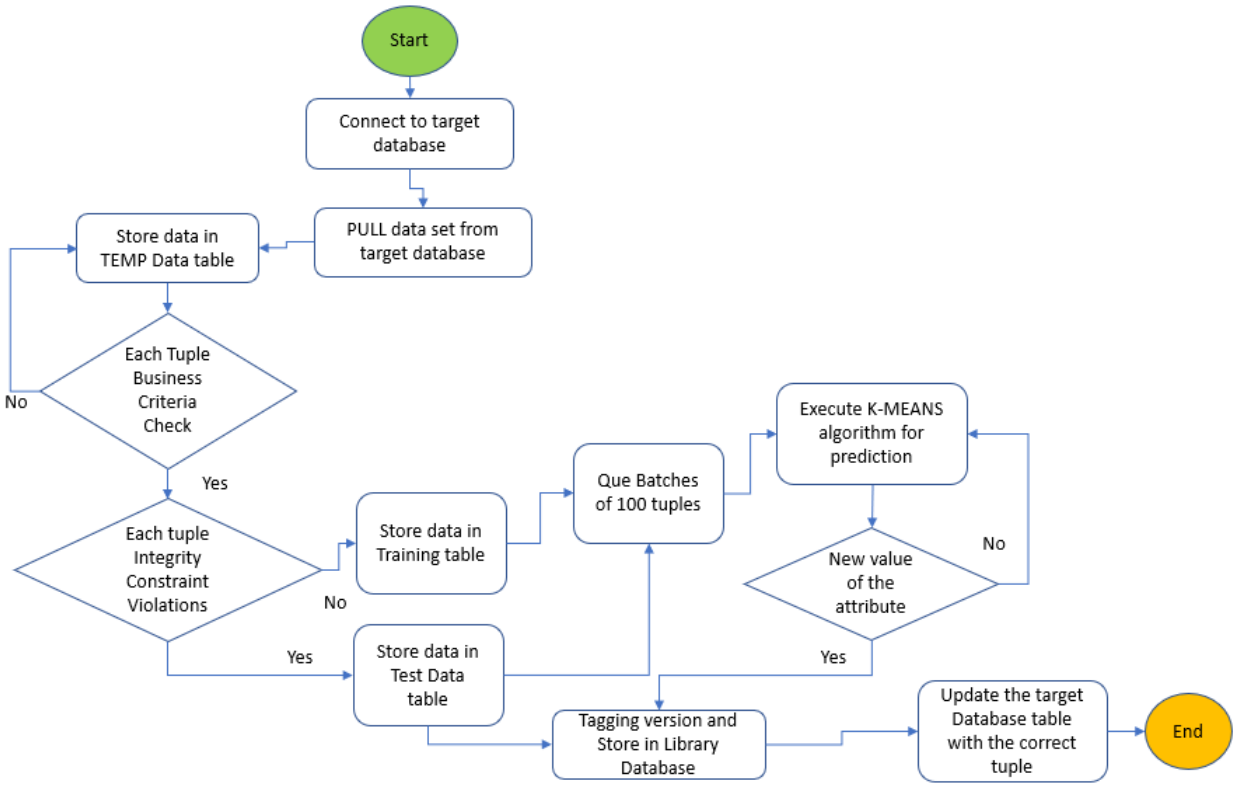


Figure 8: The Data flow diagram

4.7 Object Diagram

We have captured the objects and their relationships in the diagram below. The object diagram here shows the relationships of the functional component of the system.

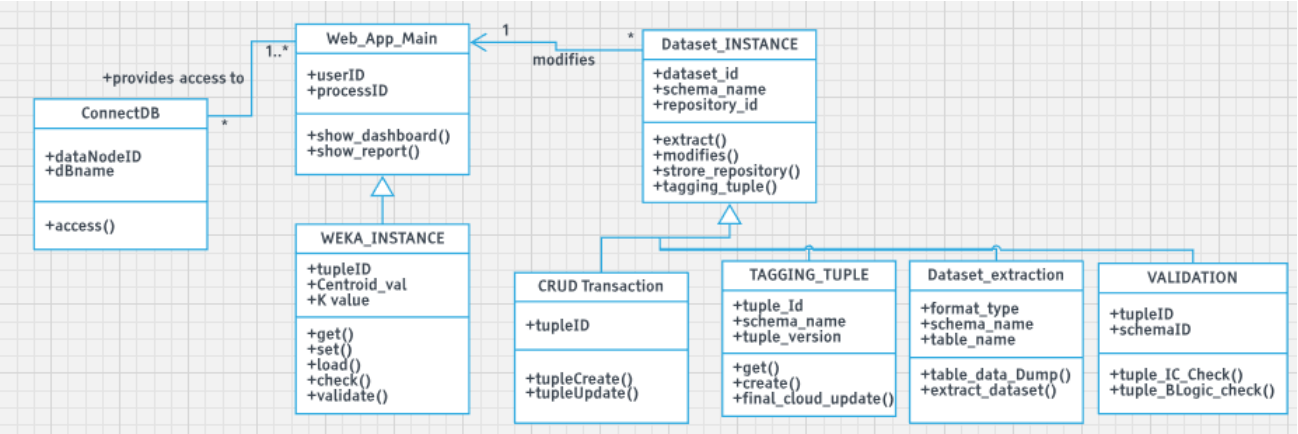


Figure 9: The object diagram

4.8 Configurations and Execution

The detection and correction algorithms work together different on each dataset which are stored separately in the AWS cloud database. So, we can execute multiple instances of the dataset for testing the accuracy of this new suggested solution.
For testing and execution of the suggested solution 3 set of separate datasets have been utilized. The web application launches the execution of the detection and correction algorithms in sequence for a selected group of datasets. The data set is extracted from the source database or the target database with the help of the web application which connects to the AWS Dynamo DB instances on the cloud for multiple regions.

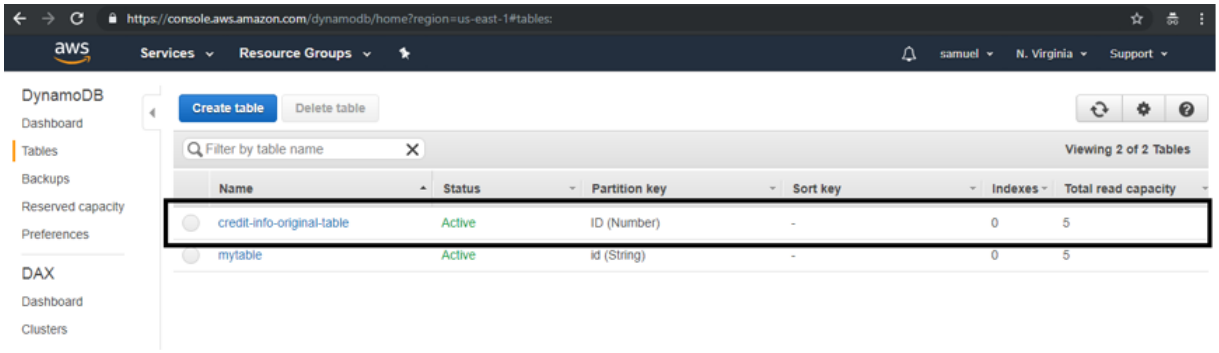


Figure 10: The Target database on AWS Dynamo DB

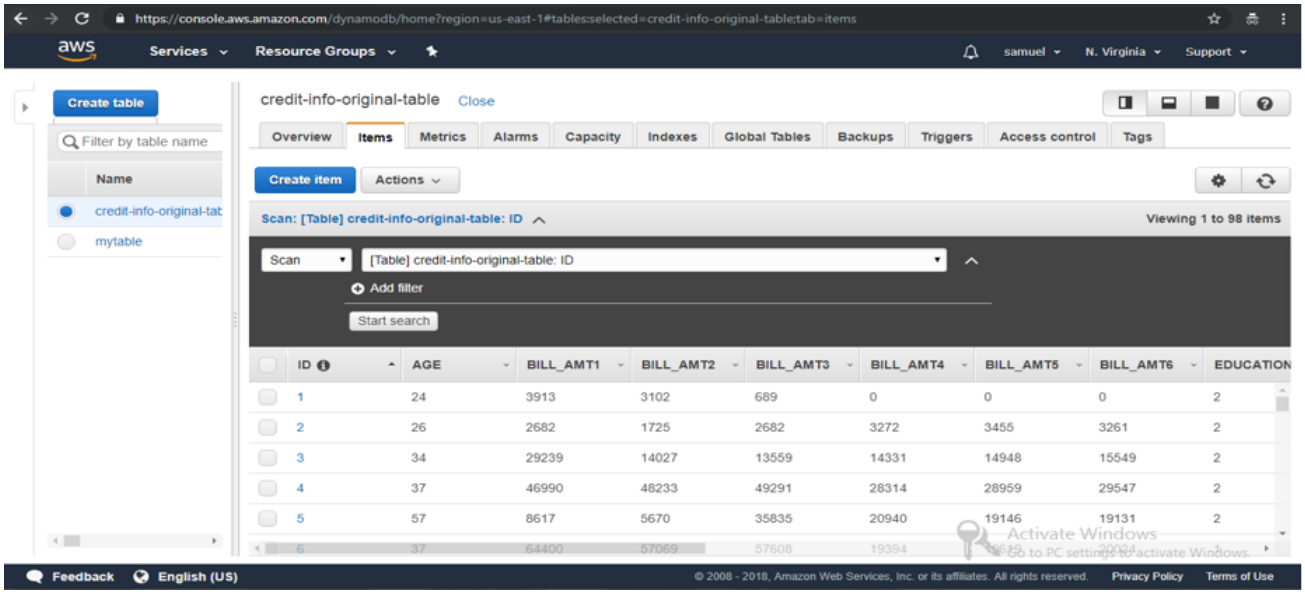
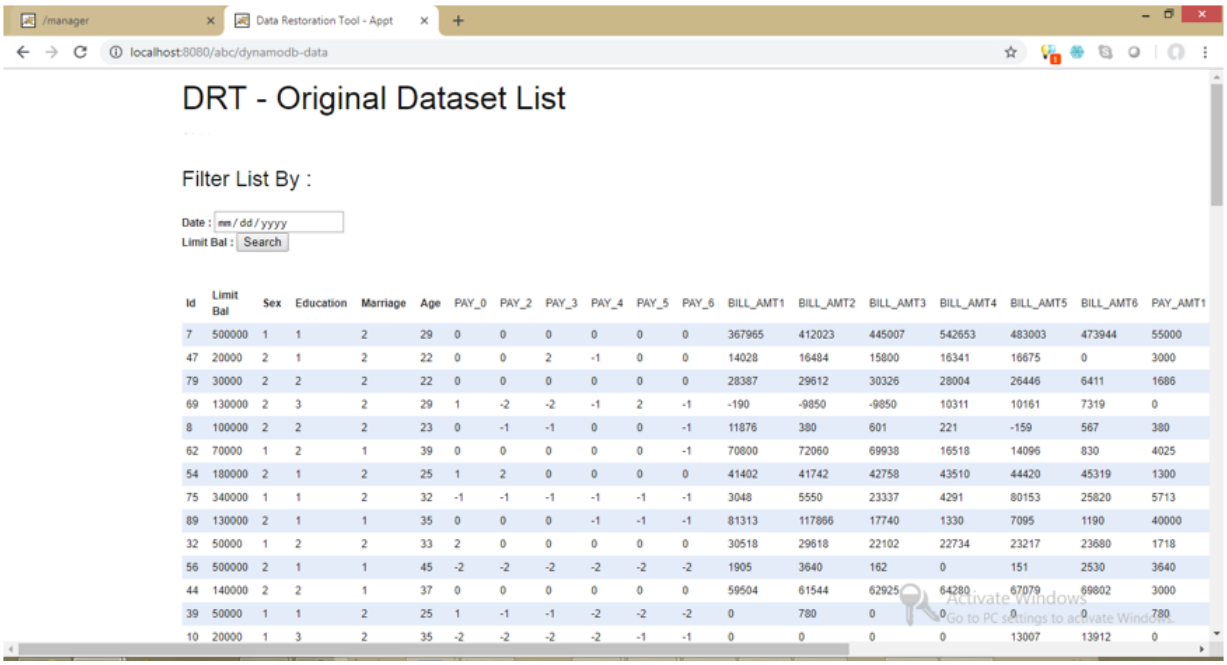


Figure 11: The Target database on AWS Dynamo DB

From the cloud database the web application extracts the information in a JSON format for loading onto the local detection process. The detection process reads these datasets one by one and then detects the integrity constrains issues with the records. This is done with the help of the reference tables for the target table of a particular database or schema.

```
|-------------------------------------------------------------------------|
| SELECT DISTINCT LIMIT_BAL, AGE, columnA, columnB, columnC, columnD_FK    |
| FROM sourceDataSet                                                      |
| INNER JOIN EDUCATION on EDUCATION.ID_EDUCATION = SourceDataSet.EDUCATION_FK|
| INNER JOIN AGE ON AGE.YEAR = EDUCATION.AGE_FK                           |
| WHERE AGE.YEARS != NULL;                                                |
|-------------------------------------------------------------------------|
```

The result of the SQL query gives the final dataset of unique tuples which are incorrect with respect to the data integrity constraint defined in the table or schema.



Id	Limit Bal	Sex	Education	Marriage	Age	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1
7	500000	1	1	2	29	0	0	0	0	0	0	367965	412023	445007	542653	483003	473944	55000
47	20000	2	1	2	22	0	0	2	-1	0	0	14028	16484	15800	16341	16675	0	3000
79	30000	2	2	2	22	0	0	0	0	0	0	28387	29612	30326	28004	26446	6411	1686
69	130000	2	3	2	29	1	-2	-2	-1	2	-1	-190	-9850	-9850	10311	10161	7319	0
8	100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380	601	221	-159	567	380
62	70000	1	2	1	39	0	0	0	0	0	-1	70800	72060	69938	16518	14096	830	4025
54	180000	2	1	2	25	1	2	0	0	0	0	41402	41742	42758	43510	44420	45319	1300
75	340000	1	1	2	32	-1	-1	-1	-1	-1	-1	3048	5550	23337	4291	80153	25820	5713
89	130000	2	1	1	35	0	0	0	-1	-1	-1	81313	117866	17740	1330	7095	1190	40000
32	50000	1	2	2	33	2	0	0	0	0	0	30518	29618	22102	22734	23217	23680	1718
56	500000	2	1	1	45	-2	-2	-2	-2	-2	-2	1905	3640	162	0	151	2530	3640
44	140000	2	2	1	37	0	0	0	0	0	0	59504	61544	62925	64280	67079	69802	3000
39	50000	1	1	2	25	1	-1	-1	-2	-2	-2	0	780	0	0	0	0	780
10	20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0	0	0	13007	13912	0

Figure 12: The Web Application dashboard showing data extracted from AWS Dynamo DB

The dataset goes thru the detection process and then the incorrect tuples are identified and stored on the repository with an initial version which equals to 0. The repository is set-up in the AWS Dynamo DB cloud also for scaling the library and also due to easy access to any application interface or systems in the future. The original version of the incorrect tuple or record is stored here for future references. Hence the first record or tuple is always the incorrect record. After the record is corrected by the correction process, then again, the new correct record is formed here, and a new version is added to it.

```
package com.cloudspokes.dynamodb;

public interface Constraints {
    /*
    AWS MySQL Credentials
    */
    String mySQLDB="data_restoration_db";
    String mySQL_Username="shobhan_das";
    String mySQL_Password="shobhan_das";
    String mysql_Host="samuel-db-instance.cqshpwkbnzpv.us-east-1.rds.amazonaws.com";
    String mysql_Port="3306" ;
    String mysql_driver="com.mysql.jdbc.Driver";


    /*
    AWS Access and Secret Key
    */
    public static final String ACCESSKEY = "AKIAI66YU5FOQ5SPRMJQ";
    public static final String SECRETKEY = "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";
}
```

Figure 13: The Library table parameters as set on cloud AWS Dynamo DB

The prediction process then starts with the cluster group values set to a constant value of 4. Hence in this solution there will be always four cluster that will give four centroid values for 4 different clustered datasets.

```
-------------------------------------------------------------
|     SimpleKMeans kMeansInstance = new SimpleKMeans();      |
```

```
|     int numberOfClusters = 4;                                        |
|     int[] centroids = null                                           |
|                                                                      |
|  //Assumed seed value is as per the data points grouped              |
|  //as per desired grouping.                                          |
|   kMeansInstance.setSeed(10000);                                     |
|   ...                                                                |
|    try {                                                             |
|        kMeansInstance.setNumberOfClusters(numberOfClusters);         |
|        kMeansInstance.buildClusterer(groupOne);                      |
|        //The array returns the centroid value for                    |
|        //each group of data points                                   |
|        centroids = kMeansInstance.getAssignments();                  |
|    } catch (Exception e) {                                           |
|        e.printStackTrace();                                          |
|    }                                                                 |
|----------------------------------------------------------------------|
```

The four clusters generate four centroids which we consider for the new correct value. The average of the four centroids gives the final correct value for the error attribute value.

5 Evaluation

This section evaluates the experiment results for different factors like accuracy, effectiveness, processing speed and user friendliness of our solution. A systematic empirical and statistical analysis of information from the outcomes of execution results helps us to understand the above factors. The data set used for the experiment is real world example and an analysis of the experiment results will show us a near to real picture of the efficiency and effectiveness our solution. Our solution is made up of two distinct processes which are detection process and prediction process. The prediction process is dependent on the detection process with respect to the data which is collected from the target table existing on the cloud. The prediction process is the crux of our solution hence the efficiency and effectiveness of our solution is more dependent on the prediction process and its results.

5.1 Accuracy in Prediction

The detection and correction processes executed 10,000 tuples or records of each three different dataset for 10 days. Each dataset has 10 attributes or columns of data and every dataset has 3 reference tables for referencing for the integrity checks which adds to the complexity.
It is observed that after executing a total 30,0000 records by the detection process, the integrity check SQL query command becomes complex and takes more time as the data quantity increases and the query couldn't be indexed. But as long as the dataset can be targeted very specifically in batches of 1000 records the detection process completes each process in few seconds. Hence it is noted here that the user should have a specific target to focus on. If the user wants to run the detection process on all the existing dataset then the duration of the detection process will consume lot of time and will not be an

effective and best practise to run the detection process.

Hence **limitation** of our solution lies in the fact that a smaller batch of records have to processed in batches to keep the algorithm execute fast. Secondly, the understanding of the business process is required to isolate the dataset and the possible affected tuples is the best way.

The most important process of out solution is the prediction process. But the prediction process also depends on the batches of records which are not more than 1000 in quantity. The learning data set is a must for the prediction algorithm to work. Hence the small batch of the dataset comes in use here for making a dataset for learning.

Similarly, for the prediction algorithm, a large volume of dataset was passed thru the K-means algorithm in batches of 5000 data points at a time. This did not allow the system to execute the results fast enough and at times would halt the execution. Secondly the results set were not accurate due to cluster groups were large to include very large number of data points. Hence it was deemed to bring down the test data set to a manageable count of 1000 and 500 data points to be processed at a time.

Test Cases for Measuring Accuracy							
Test Runs	Dataset Attribute:CREDIT LIMIT	Tuple Volume	Batch Size: Data points	Clusters of Data points	Centroid Avg	Expected Result	Approx Difference in percentage
100	Male	10000	2000	2	26251.65	30000	15
150	Female	50000	2000	2	17457.85	21000	15
50	Less than 30 yrs	50000	3000	5	46551.55	50000	10
50	Between 20 yrs to 40 yrs	50000	2000	5	51532.42	55000	8
50	Between 30 to 40 yrs	50000	1000	4	56572.44	60000	8
50	Between 40 to 50 yrs	50000	1000	4	43568.35	45000	8
50	Between 30 to 40 yrs	50000	500	4	56301.68	60000	8
50	Between 40 to 50 yrs	50000	500	4	43853.84	45000	8

Table 1: Test Cases and Results showing accuracy

As shown in the figure plot below that the accuracy increases when the test data point count was at 1000 and 500.

We tested the data sets with various volume sizes repeatedly to see the prediction capability and its accuracy. The results were known before in hand in-order to compare the predicted values using the KMeans algorithm. The below table shows the nature of the test runs that were executed to find the accuracy. The graph below depicts the pattern

observed in the test runs with the test data sets.

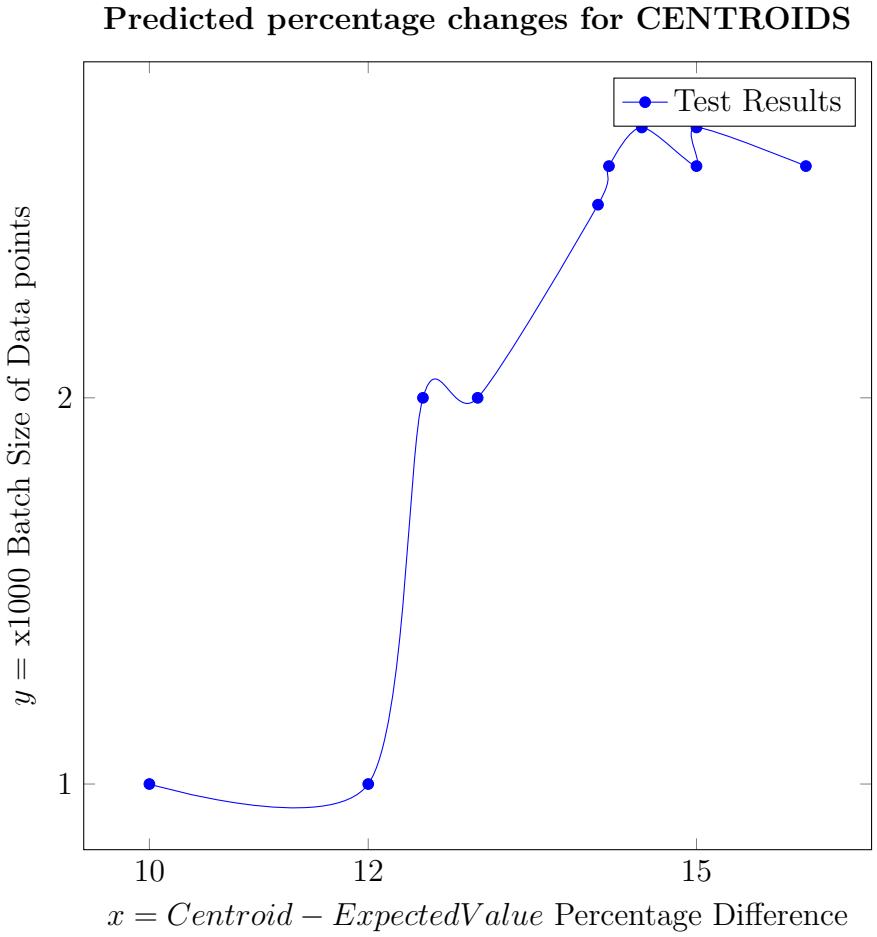


Figure 14: Graph Report for accuracy of centroids predicted by K-means algorithm.

Hence, we can suggest this solution to be fairly accurate to 10 percent on average of the original values which were corrupted. So, the new correct values are approximate as appears from the experimental results. We found also that the Kmeans++ algorithm can deliver better results on the same set of data points. The Kmeans employs geometric circular clusters which exclude related data points and hence the decrease in the effective prediction.
There is better algorithm in data analytics like Expectation-Maximisation (EM) algorithm which can deliver much better results due to its ecliptic clustering characteristic, but we could not employ the Expectation-Maximisation (EM) algorithm due the time consuming factors for the data preparations required for EM algorithm at present for this research duration.

6 Conclusion and Future work

We have proposed a framework that uses the advantages of data mining techniques and delivers an efficient and robust system which is suitable for cleaning the database of malformed values in the attribute of a database tuples.
This solution framework is reliable and with more modifications can deliver very accurate

results.
The proposed framework can be enhanced with more features and functions to support various database instances which are either on public cloud, hybrid cloud and private cloud. With more features and functions the proposed framework can be extended to various attributes with different data types like alphanumerical as well.
The core functioning of the system is presented as a web application which enables easy access to the system over the internet. This feature and functional control of the functions can be segregated into various libraries and APIs which can be called and used in a Java program directly as third-party source libraries. We have collected the statistics of the output showing the efficiency of the system in its functions.
In the future this framework can be extended to process non-structured data from the database. Secondly the framework can expose external APIs which can be called in any Java based application where database entries and updates are taking place from multiple users and multiple external systems. This framework can function as a check gate to detect anomalies in the data that is getting uploaded or added into the structured relational database.

7 Acknowledgements

I would like to extend my sincerest thanks to my supervisor Mr. Manuel Tova-Izquierdo for his support and continuous motivation during this research. His interest and guidance in my research proposal boosted me up in striving for new innovative ideas until the final stage of shaping up the solution. His timely advice and guidance have benefited me in producing the thesis with all the questions answered in well-organized manner.

References

Arasu, A., Chaudhuri, S., Chen, Z., Ganjam, K., Kaushik, R. and Narasayya, V. (2011). Towards a domain independent platform for data cleaning, *Data Engineering Bulletin* .

Assadi, A., Milo, T. and Novgorodov, S. (2018). Cleaning data with constraints and experts, *Proceedings of the 21st International Workshop on the Web and Databases*, ACM, p. 1.

Bernstein, P., Brodie, M., Ceri, S., DeWitt, D., Franklin, M., Garcia-Molina, H., Gray, J., Held, J., Hellerstein, J., Jagadish, H. et al. (1998). The asilomar report on database research, *ACM Sigmod record* **27**(4): 74–80.

Bertossi, L., Kolahi, S. and Lakshmanan, L. V. (2013). Data cleaning and query answering with matching dependencies and matching functions, *Theory of Computing Systems* **52**(3): 441–482.

Calikli, G. and Bener, A. (2013). An algorithmic approach to missing data problem in modeling human aspects in software development, *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, ACM, p. 10.

Chaudhuri, S., Ganjam, K., Ganti, V., Kapoor, R., Narasayya, V. and Vassilakis, T. (2005). Data cleaning in microsoft sql server 2005, *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ACM, pp. 918–920.

Chu, X., Ilyas, I. F. and Papotti, P. (2013). Holistic data cleaning: Putting violations into context, *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, IEEE, pp. 458–469.

Coady, Y., Hohlfeld, O., Kempf, J., McGeer, R. and Schmid, S. (2015). Distributed cloud computing: Applications, status quo, and challenges, *ACM SIGCOMM Computer Communication Review* **45**(2): 38–43.

Cui, Y. and Widom, J. (2003). Lineage tracing for general data warehouse transformations, *The VLDB JournalThe International Journal on Very Large Data Bases* **12**(1): 41–58.

Geerts, F., Mecca, G., Papotti, P. and Santoro, D. (2013). The llunatic data-cleaning framework, *Proceedings of the VLDB Endowment* **6**(9): 625–636.

Hwang, K., Dongarra, J. and Fox, G. C. (2013). *Distributed and cloud computing: from parallel processing to the internet of things*, Morgan Kaufmann.

Lang, W., Bertsch, F., DeWitt, D. J. and Ellis, N. (2015). Microsoft azure sql database telemetry, *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ACM, pp. 189–194.

Makris, C. and Markovits, P. (2018). Evaluation of sensitive data hiding techniques for transaction databases, *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, ACM, p. 11.

Mandros, P., Boley, M. and Vreeken, J. (2017). Discovering reliable approximate functional dependencies, *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 355–363.

Moustafa, S., Elgazzar, K., Martin, P. and Elsayed, M. (2015). Slam: Sla monitoring framework for federated cloud services, *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*, IEEE, pp. 506–511.

Shin, J.-Y., Balakrishnan, M., Marian, T., Szefer, J. and Weatherspoon, H. (2016). Towards weakly consistent local storage systems, *Proceedings of the Seventh ACM Symposium on Cloud Computing*, ACM, pp. 294–306.

Terry, D. B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M. K. and Abu-Libdeh, H. (2013). Consistency-based service level agreements for cloud storage, *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ACM, pp. 309–324.

Toosi, A. N., Calheiros, R. N. and Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey, *ACM Computing Surveys (CSUR)* **47**(1): 7.

Widom, J. (2004). Trio: A system for integrated management of data, accuracy, and lineage, *Technical report*, Stanford InfoLab.

Xu, L., Pavlo, A., Sengupta, S. and Ganger, G. R. (2017). Online deduplication for databases, *Proceedings of the 2017 ACM International Conference on Management of Data*, ACM, pp. 1355–1368.

Zellag, K. and Kemme, B. (2012). How consistent is your cloud application?, *Proceedings of the Third ACM Symposium on Cloud Computing*, ACM, p. 6.

Appendices

Appendix A

List of Software

The implemented using the open source tools and applications. Below are the following proposed.

SOFTWARE	Package	version
Java	Maven	2.5
Java Framework	Spring	3.0.6
Apache	Tomcat	8.0.3
AWS	Java SDK	1.5.0
Java	JDK	1.6
IDE	Eclipse	Mars
IDE	Netbeans	IDE 8.0
AWS	MySQL	5.1.6
Test	Junit	4.7
Javax	Servlet API	2.5

Figure 15: Software List Used

1. AWS services: AWS is a major cloud service provider. The entire project will be hosted on AWS cloud platform.

2. AWS Dynamo DB: The research requires multiple instances of the database instances and data location which will be created using Dynamo DB.

3. R open source language: Artificial Intelligence language for data mining -The research uses data mining technique called expectation-maximization (EM) which is used for knowledge discovery.

4. Java MVC model to create the main program which is the web application. Web application is developed in Java Spring MVC framework.

5. Java Weka API are used for the using the data analytic APIs.

6. Weka Tool is used for analysing the results and testing the data points for confirmation of the test data quality.

Appendix B

Configuration Manual

The software components installation and configuration parameters are described for user to follow the set-up instructions for the system and the data that is required to execute the experiment.

Appendix B.A AWS Cloud Database connection set-up

The AWS cloud database is identified with the following steps for the extraction of the data from the AWS cloud database.

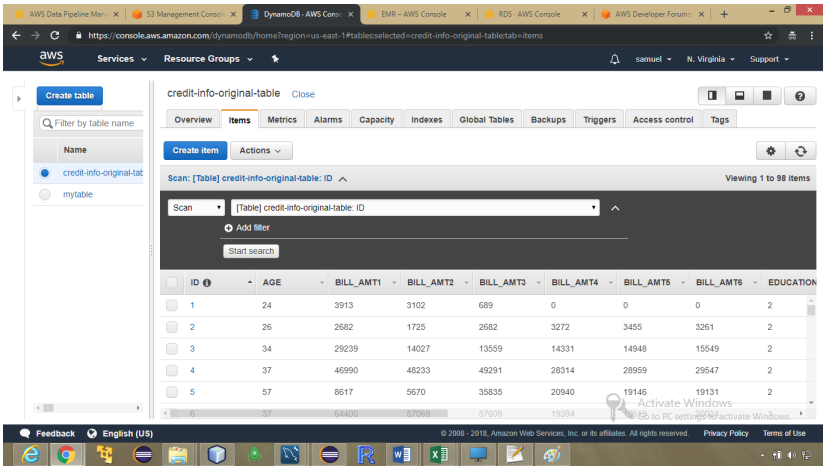


Figure 16: AWS database Identification

We then go ahead to connect to the AWS cloud database with the following steps as shown in the screen shot.

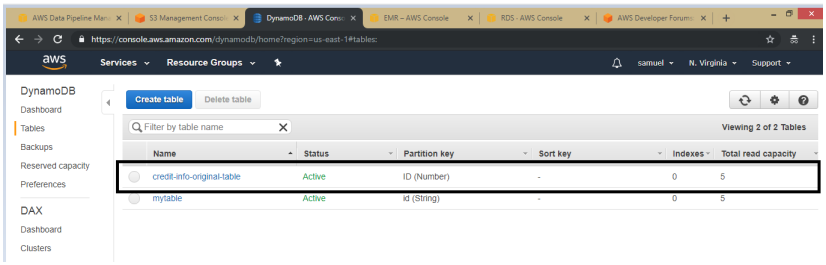


Figure 17: AWS database Instance connection.

Then we create a AWS data pipeline for data transfer process as below.

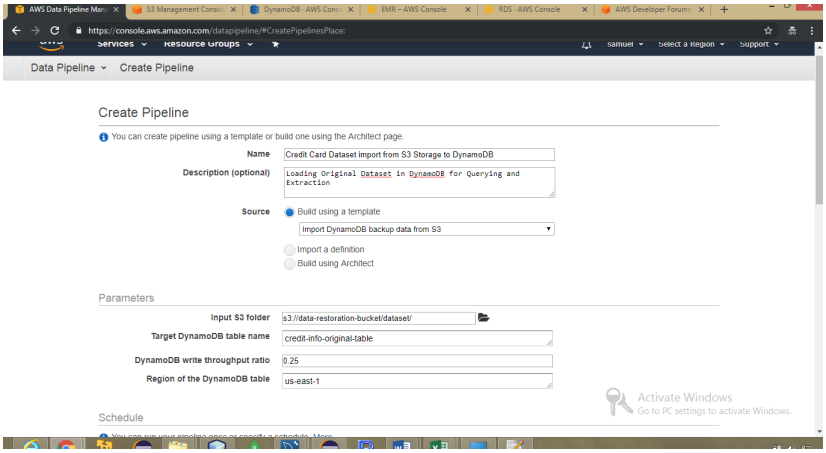


Figure 18: AWS Data Pipeline set-up

After the data pipeline is created then we activate the data pipeline.

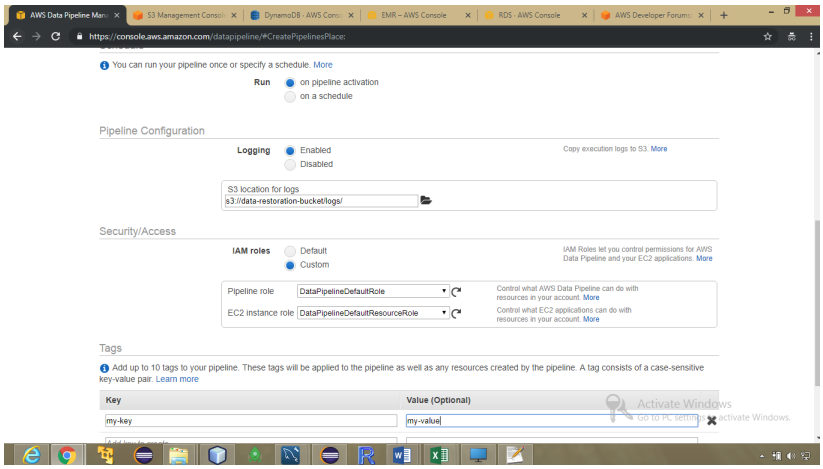


Figure 19: Activate the AWS Pipeline for data transfer

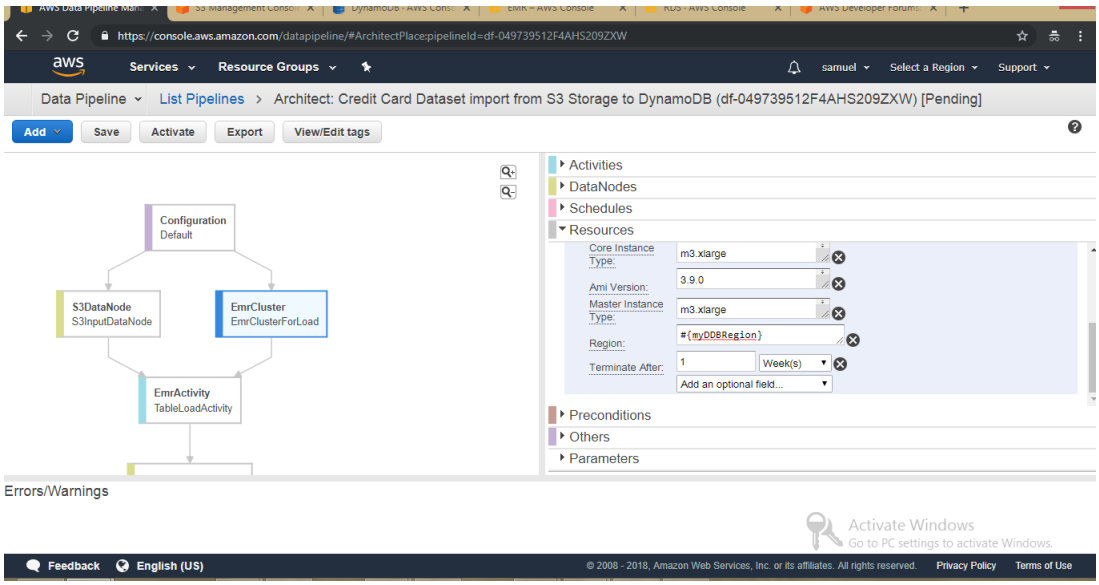


Figure 20: AWS Pipeline for data transfer Running state

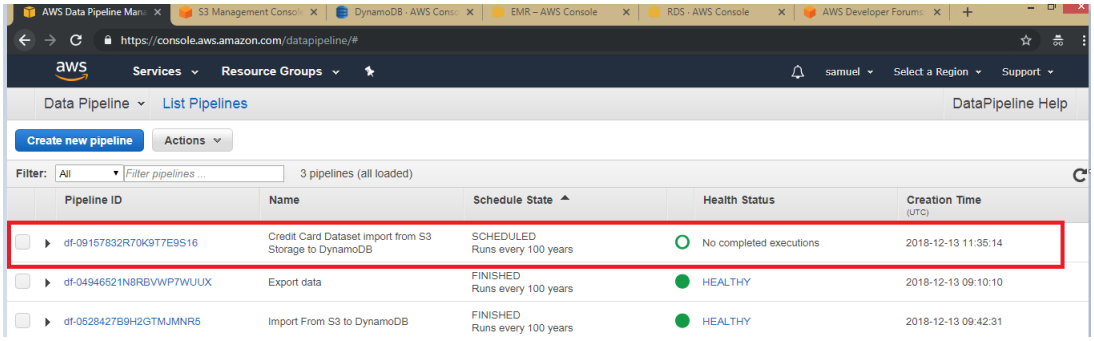


Figure 21: AWS Pipeline Running state

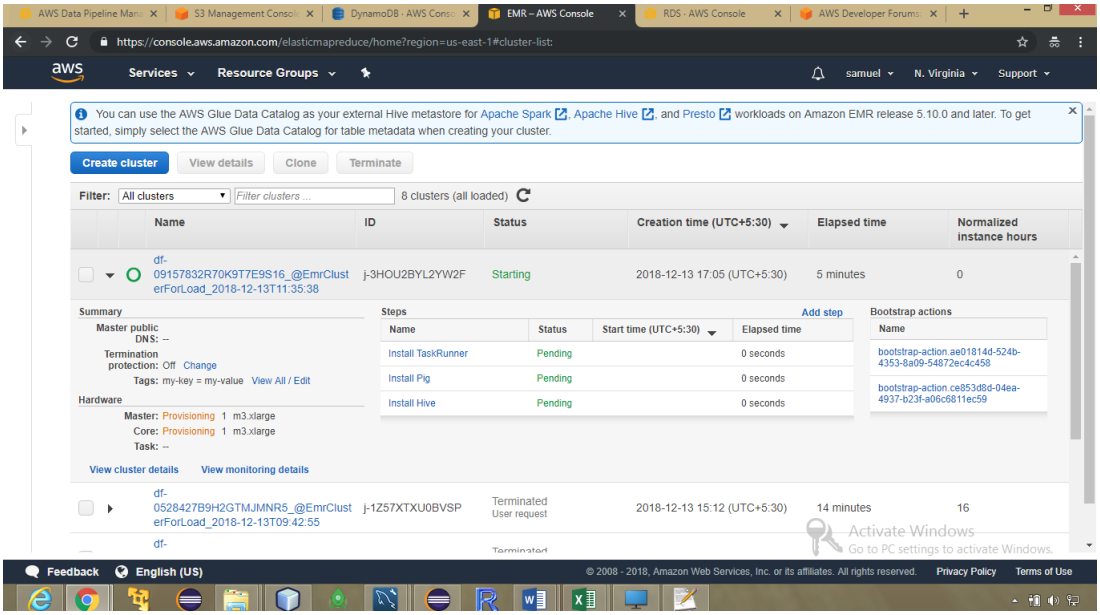


Figure 22: AWS Pipeline Running state

Appendix B.B Web Application Set-up

We configure the web application code for the data extraction. Hence this web application for data extraction and listing records of the table and also for data pull and push through GUI. we deploy the .war file through Tomcat Manager which will be running on localhost. After this Tomcat server is started the we start the AWS beanstalk service for the web application to talk to the AWS database. The Apache can be installed with the following steps.

1. After downloading the installation file, double-click to run installer.
2. Then follow instructions to install.
3. When prompted for the input of "Server Information", enter "localhost".
4. Allow the installer to install to the default folder.
5. Finally click the "Install" button to set up Apache. Then click the "Finish" button.

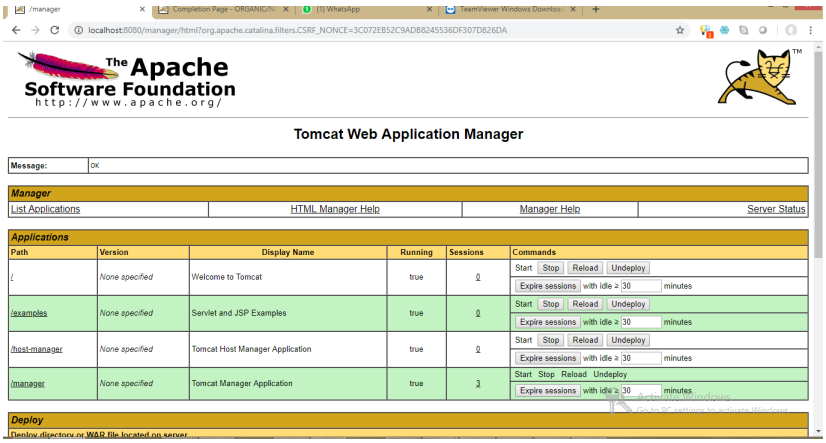


Figure 23: Tomcat Apache Configuration

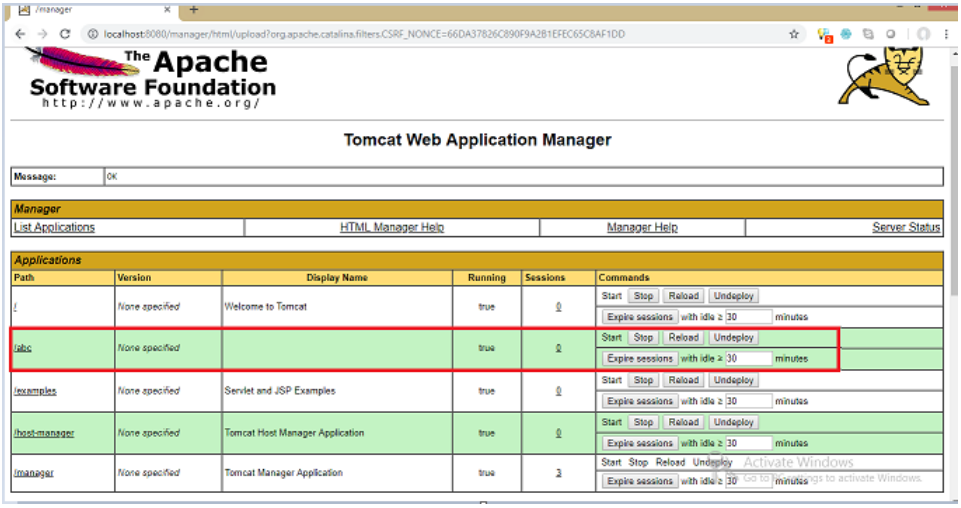


Figure 24: Our Web application Deployed on Tomcat

Appendix B.C Test Data Set-up

For our use case to find the customers of the credit card bank, who have incorrect credit limit, i.e. the limit balance for credit assigned to them the transaction table of the cloud database is the target table of the database. The data extraction is done with the following steps.

1. The code to parse CSV to JSON and JSON to CSV format is done so that the data is understood by DynamoDB since DynamoDB generates JSON in below format as shown in Figure 25.

2. Click on the "Generate Test Data" Button to run the program which will parse the input dataset into specific arff format which is required later for the data learning of the Kmeans algorithm.

3. Observe a file with dot arff extension is generated in the project folder local directory. As shown in figure 26. The same data is also present in the local MySQL database instance , as shown in figure 28.

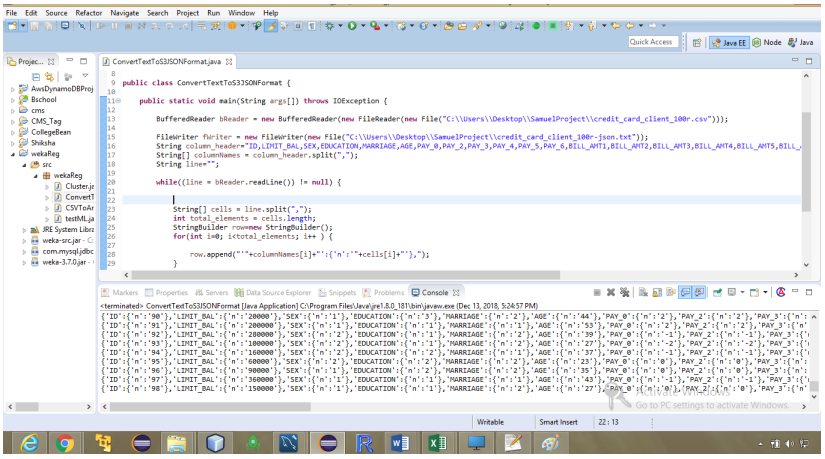


Figure 25: The datasets

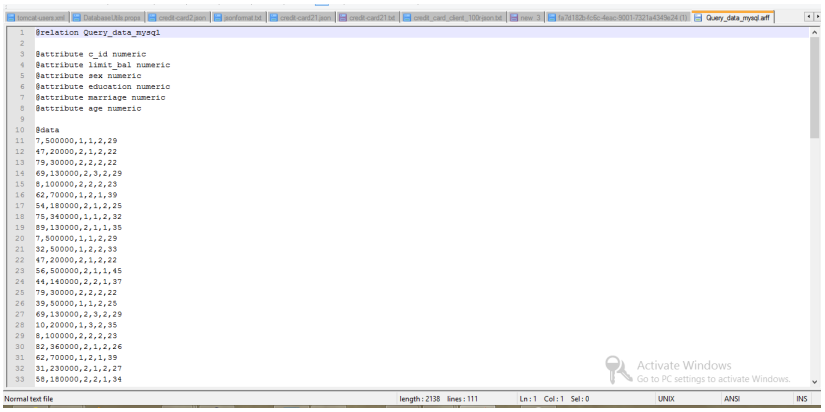


Figure 26: The training datasets

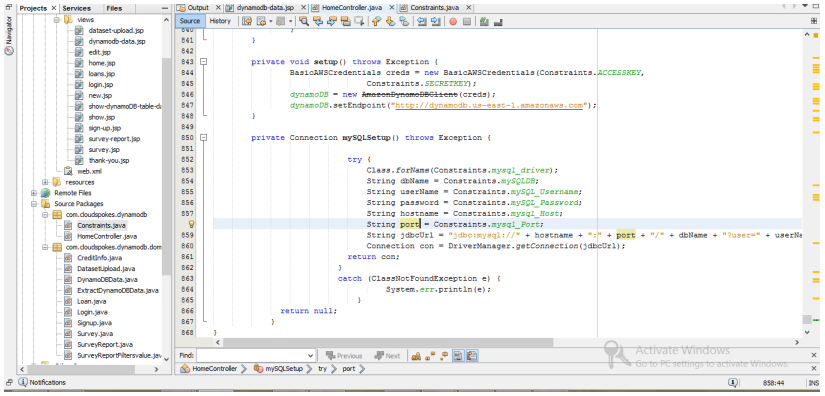


Figure 27: The training datasets