Real-time Automated, Scalable Data Integration Platform for Big Data Analytics

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Technology

in

Computer Science & Engineering

with specialization in Cloud Computing

by

Theophilus Rakesh

15MCC1036

School of Computing Science & Engineering,

VIT University Chennai,

Vandalur-Kelambakkam Road,

Chennai - 600127, India.

May 2017

Declaration

I hereby declare that the dissertation ***Real-time Automated, Scalable Data Integration Platform for Big Data Analytics*** submitted by me to the School of Computing Science and Engineering, VIT University Chennai, 600 127 in partial fulfillment of the requirements for the award of **Master of Technology** in **Computer Science & Engineering with specialization in Cloud Computing** is a bona-fide record of the work carried out by me under the supervision of ***Pradeep KV***.
I further declare that the work reported in this dissertation, has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or University.

Sign: _____

Name & Reg. No.: _____

Date: _____

School of Computing Science & Engineering

Certificate

This is to certify that the dissertation entitled ***Real-time Automated, Scalable Data Integration Platform for Big Data Analytics*** submitted by ***Theophilus Rakesh*** **(Reg. No. 15MCC1036)** to VIT University Chennai, in partial fulfullment of the requirement for the award of the degree of **Master of Technology** in **Computer Science & Engineering with specialization in Cloud Computing** is a bona-fide work carried out under my supervision. The dissertation fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this dissertation have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Supervisor

Signature: .....

Name: .....

Date:

Program Chair

Signature: .....

Name: .....

Date:

Examiner

Signature: .....

Name: .....

Date:

(Seal of the School)

Abstract

Handling huge volumes of data is always tricky in an enterprise environment especially when there are multiple data sources. Integrating data from these data sources to make some logical inference has always been a challenge. Integrating these data in a lot of way will reduce human intervention and eliminate the use of various tools to accomplish similar task.

Aggregation of data becomes complex when the size of the data increases exponentially every day. The complexity further increases when the data has to be captured, analyzed and projected in real time. Existing ETL method of data extraction and mining works perfectly with historical records records that are either pre-stored or updated over longer intervals of time. When analysis is required to be performed on near real-time data, this method does not suit our need. This is mainly because; we try to transform the data intermediality - which increases the loading time. Instead, in this paper we propose a near real-time analysis platform that will load the data as and when available at a much higher rate which is later used for analytics/transformation.

Acknowledgements

I wish to express my sincere thanks to Dr.G.Viswanathan, Chancellor, Mr. Sankar Viswanathan, Vice President, Ms. Kadhambari S. Viswanathan, Assistant Vice President, Dr. Anand A. Samuel, Vice Chancellor and Dr. P. Gunasekaran, Pro-Vice Chancellor for providing me an excellent academic environment and facilities for pursuing M.Tech. program. I am grateful to Dr. Vaidehi Vijayakumar, Dean of School of Computing Science and Engineering, VIT University, Chennai and to Dr. V. Vijayakumar, Associate Dean. I wish to express my sincere gratitude to Dr. B. Rakesh Kanna, Program chair of M.Tech Cloud Computing for providing me an opportunity to do my project work. I would like to express my gratitude to my internal guide Prof.Pradeep K.V and my external guide Mr. Shamsheer Khan who inspite of their buy schedule guided me in the correct path. I am thankful to Honeywell Technology Solutions Lab Pvt. Ltd., Bengaluru for giving me an opportunity to work on my project and helped me gain knowledge. I thank my family and friends who motivated me during the course of the project work.

Contents

List of Figures

The fear of the Lord is the beginning of wisdom ...
Dedicated to the source of absolute wisdom.

Chapter 1

Introduction

Cloud based SaaS applications are used in enterprises commonly these days. This leads to a lot of abstraction and confusion from data analytics and integration point-of-view. Because there are multiple sources for data and we cannot run our analytics or the business logic directly on these data points, we are forced to pool our data at one point and infer from this data point instead. The major challenge with this is, these data are unstructured and Big (increasing in size exponentially in real time). Traditionally, these data are 'transformed' into some sort of structured or semi-structured data format/set and loaded into a Relational Database. The analytics is performed on the aggregated data set. This is efficient. But, when you have lots of data sources to gather data from, this becomes a time consuming process. This is also not efficient for real-time data integration and analytics. As, the 'transformation' usually takes longer time.

In the proposed idea, you query the data using web services according to set business logic. The returned semi-structured or unstructured data set is fed into an RDB. This may lead to data duplication - but, the analytics bot runs on the data store later to ensure only the essential data stays in the RDB. The bot also ensures that the data is available for the user in a specific format. Finally, a presentation layer is used to provide graphical inference to the user.

Chapter 2

Literature Survey

2.1 Real-Time Data ETL Framework for Big Real-Time Data Analysis

This research paper was hugely helpful in understanding the possibility of near real-time data analytics for Big Data. This paper points the challenge in just reducing the refresh time for OLAP (On-Line Analysis Processing) systems to achieve aggregate real-time data - because, the OLTP (On-Line Transaction Processing) systems cannot simply keep-up with the volumes of data that is aggregated by the OLAP system. It also points out to the fact that ETL(Extract-Transform-Load) framework is not compatible for near real-time data analytics as the transformation of data to a desired format requires additional time.
Hence, they propose an unique method to cache/save the OLAP-OLTP transactions in an external storage area called the DSA (Dynamic Storage Area). This however, is not used or implemented in the proposed architectural framework because the CDC (Change Data Capture) is simply too much for any OLTP system to keep-up with.

2.2 Data Integration using Webservices

This is an article published by MIT Sloan School of Management. This article was useful in understanding some of the existing webservices architecture and how data is pooled in them.
Programming standards associated with webservices was understood from this journal. And, some of the examples on how webservices can be used to unlock heterogeneous business systems and aggregate business data were useful. It provides an introduction

to the problems and research issues encountered when applying Webservices to data integration. It shows that webservices will make the development of systems for aggregation both faster and less expensive to develop. A system architecture for webservices based aggregation is presented that is representative of products available from software vendors today.

2.3 Data Integration as a Service for Applications Deployment on the SaaS Platform

This research paper was helpful in understanding various data integration techniques for SaaS applications. As listed in the paper, here are they: *Direct Coding*, *Middleware*, *As-a-service*. A simple architecture for SaaS data integration was understood from this paper.

2.4 Mining Unstructured Data in Software Repositories: Current and Future Trends

This research paper was useful in understanding the various possible data mining techniques for unstructured data. Though the discussion from this paper is out-of-scope of this proposed architecture, it was interesting and useful to understand how unstructured data in the Internet as a whole can be mined.
It was also interesting to read about how even software artefacts such as README files and other installation documents too come under unstructured data and can be very well mined as well. Lastly, it was also interesting to note the author's view points on various possible future trends for Mining Unstructured Data.

2.5 Challenges of Data Integration and Interoperability in Big Data

This research paper was instrumental in understanding, the various challenges in accumulating data from different data sources. And, how ETL is used in Big Data analytics. Various InterOperability challenges are discussed in this paper and it was useful in understanding them. How the scope of the data and its inconsistency pose serious challenge in data integration. It also focuses on how the architecture will run into scalability problems if the data volume is not taken into account.

2.6 Analysis and Optimization of Big-Data Stream Processing

This paper has deeper explanation and mathematical relations which are really beyond the scope of this project work. However, the paper in-detail explains some of the task execution optimization techniques.
The paper proposes some of the parallelizing techniques for Big Data processing/analytics. The Parallel AND and OR techniques can be used if an algorithm is to be built for Big Data analytics for a specific type of requirement - which, here, is not required.

2.7 An Overview and Implementation of Extraction-Transformation-Loading (ETL) Process in Data Warehouse

This research paper was useful in understanding the ETL (Extract-Transform-Load) process in a Data Warehouse. It also briefly discusses a Data Warehouse architecture and the various EIS (Enterprise Information System) involved in this process. It also focuses on CDC (Change Data Capture) is important in the Extraction phase of the ETL process. It also briefly discusses about possible Loading techniques in a Dimensional table - the paper discusses 3 strategies for this.
This paper was also useful in structuring/organizing the contents of this research work.

Chapter 3

Experimental Design & Setup

3.1 Data Warehouse

Definition

Traditionally, a Data Warehouse is a database that stores/aggregates historical data for business analytics. We are now living in times were data generated in an organization increases exponentially by the hour. This poses a very large threat, which is: are all the generated data important and necessary? If it is, how can we effectively manage it? If not, how to process them? This again leads to several other challenges such as: accuracy, near real-time availability, reporting. Having said these, it is also imperative for a Data Warehouse to maintain **ACID** property. A data warehouse houses aggregated data from various data sources to perform efficient data analysis for business needs.

3.1.1 Data Warehouse Architecture

In a data warehouse, data are pooled from various data sources. Commonly, Enterprise Information Systems (EIS) are used to perform analysis with set business logic. Unlike a traditional database, where transactional applications perform read/write operation, this data warehouse architecture houses web service bots and an EIS-like analysis platform. Fig. 1. pictorially explains the DWH architecture. Some of the salient features of a DWH is listed below:

1. The data input to this system is mainly from various external resources. However, accommodations are made to read from internal resources too.

2. The data warehouse houses various web service bots that queries data from various data sources.

3. An EIS-like system could reside inside or outside the DWH for analysis purposes.

4. Finally, a presentation layer on top of all this that could provide visual representation of the inferred data.

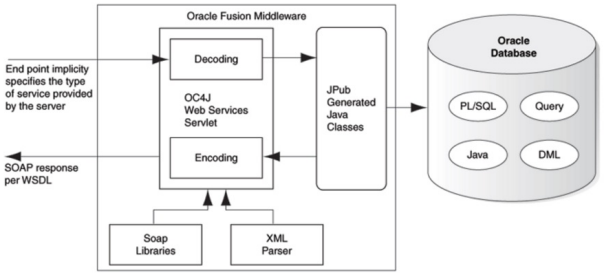

FIGURE 3.1: A DWH with Web service call-out

3.1.2 Scalable DWH Architecture

As this whole platform has to achieve scalability, what better way is left than to deploy/build the entire infrastructure on a self-scalable Cloud platform? This whole system is built inside of Honeywell Technology Solutions Lab Pvt Ltd's in-house VCloud suite.

Here, note that we have slightly redefined the traditional way of defining a Data Warehouse in subsection 2.1.1. The reason for this is, in reality, a DWH not only houses historical data but also houses some of the near real-time data and hence, here, the proposed definition holds good as the data aggregated into the DWH is at near real-time.

3.2 Process Work Flow

A Relational Data Base is used as the data point to run the analytics. Simple Object Access Protocol (SOAP) is used for communication between the web services and the RDB. Universal Description, Discovery, and Integration (UDDI) is used to discover and query the different web services we are connecting to. A simple work flow diagram is shown in Fig. 3.3.

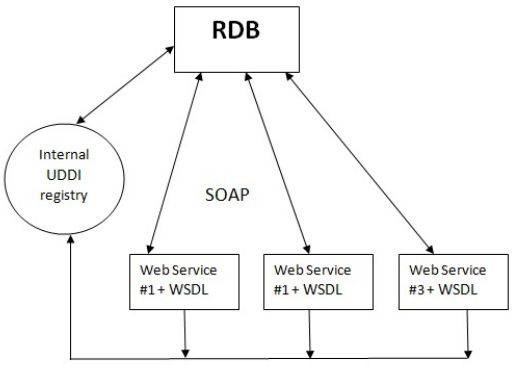

FIGURE 3.2: A simple work flow diagram

3.2.1 Data Aggregator Architecture

Here the data aggregator is the RDB (Relational Data Base). The RDB accumulates data from various data sources as represented pictorially in Fig. 3.3 SOAP is used to standardize the communication between the RDB and the web service. The data is enclosed in XML and binded over HTTP(s).

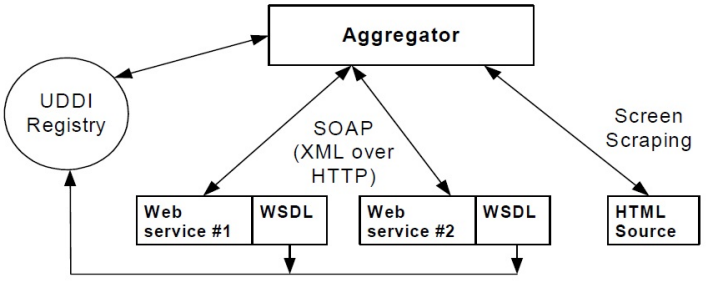

FIGURE 3.3: Data Aggregator architecture

3.2.2 Proposed Data Integration and Analytics architecture

The proposed data analytics architecture as shown in Fig. 3.5 explains that the analytics comes at a very later stage after the data aggregation. Once the data is aggregated, the analytics bot runs on the entire database with pre-set values to ensure a standardized way of analyzing the data.

We do not follow the traditional ETL (Extract-Transform-Load) method of data integration. Instead, here we follow ELT (Extract-Load-Transform) methodology. We perform the analogy and the business logic only after the data is aggregated (loaded) in the RDB.

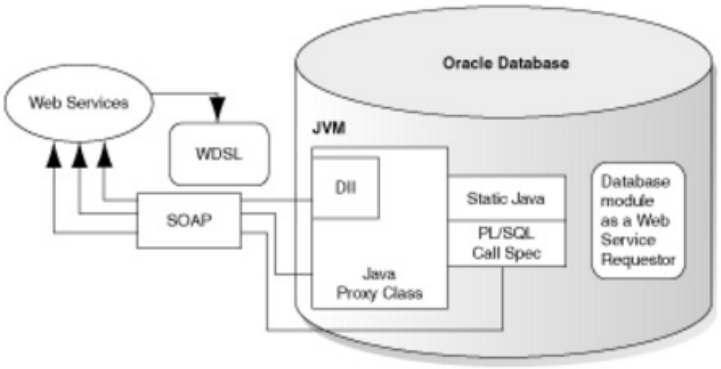

FIGURE 3.4: Proposed DI & A architecture

3.2.3 Database WebSerivces

Oracle 11g database offers web services to be called in or out from the database itself - as shown in Fig. 3.4. Oracle 11g is flexible to maintain un-structured data gathered by the web service bots along with its regular support to PL/SQL. To access the data aggregated from various data sources, a virtual table is created . This table allows you to query a set of returned rows as though it were a table.

3.3 ELT (Extraction, Loading and Transformation)

During the ELT process, data from various sources are extracted and aggregated in the DWH. Extraction is the process to identify all relevant data from various data sources. Loading is a process of dumping data from various sources into the DWH. Transformation is a process of deriving an inference from the data set according to a predefined business logic.

3.3.1 ELT concept

It is imperative to define the scope of the ELT before defining the architectural process. It is also necessary to understand the behavior and the functionality of each of the virtual table, its sources and the kind of business processes it depends on.

1. *Extraction*
   The webservice call out (the bot) identifies various data sources (using UDDI) and extracts the data from these sources as efficiently as possible. The call searches

through files, databases and various other possible data sources (depending the call out code) in selecting the data. Then, this data is transferred to the DWH via XML binded within HTTP(s).
Change Data Capture (CDC) is a very important aspect here. Almost every transaction has a timestamp associated with it. Because the data here is dimensional, it is difficult to implement CDC using dynamic mirror storage proposed by Xiaofang Li and Yingchi Mao in [8]. So, here, we just focus on building a scalable infrastructure that can support webservice call outs for dynamic CDCs at all data points.

2. *Loading*
   Because the web services are called out from within the database, the loading happens instantaneously. The data is loaded per the business logic applied in the web service to query the data from the data sources. As the Oracle 11g allows you to create a virtual table for all the imported records from the different data sources, you can query the data as if you're querying a structured table.

3. *Transformation*
   Transformation allows the user to manipulate data present in the database according to their need. This allows the user to apply any logic/inference they would want on the aggregated data.

4. *Presentation*
   This is a layer of abstraction that hides all the logic and implementation details. Any non technical user can simply use the features in the presentation layer to connect to the DWH and extract any data they would want to view. Here, the presentation layer can be compared to software like Tableau.

Fig. 5.1 shows the data flow for the proposed ELT process

Legends for Fig. 5.1: SFDC : Sales force dot com, SAP: SAP tool, OW: Internal , Avaya: Internal, IEX: Internal, HD-CRM: Internal, HC DB - Internal, D: Dimensions.

3.4 Data Connectors

Data connectors play an important role in connecting to various data sources to extract the data. Because the data is hugely unstructured and we follow an unique ELT method

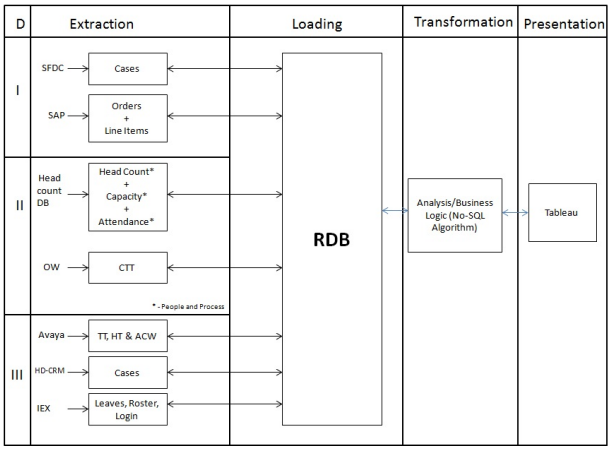

FIGURE 3.5: ELT data flow

of loading data into the DB, we do not need any transformation or gatekeeper to check the data loaded into the DB. We just load them as is and the analytics is performed later as shown in Fig. 3.5.

3.4.1 Data Connector as a driver

Data connectors can be built as a driver and installed in the server that pulls data from various data sources. This can be built in such a way that presentation layer software like Tableau can establish connection with the in-house DB.

3.4.2 Web Data Connector

A WDC is a web-based data connector that connects to the Data point and pulls all the data to the target machine based on a particular set of parameters. This runs a bot script and tries to extract the predefined data source from one or more data sources.

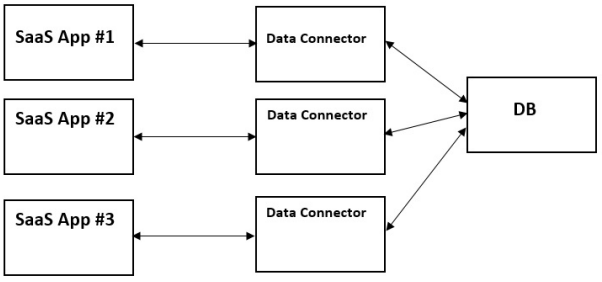

FIGURE 3.6: A simple Data Connector architecture

3.4.3 Web Connector from a Database

As shown in Fig. 3.4 , web connectors can exist within the data base that allows that data base to query the data sources on a regular interval. Fig. 3.4 explains the DC architecture and Fig. 3.6. explains how each data connector connects with different external data source.
This can otherwise be called as Webservice call-out from a Database. This is precisely what we are going to use in our project.

3.5 Definition and Description of Technologies used

1. For Database/Data Warehouse we use Oracle 11g and MySQL Server 2008 R2.
2. For webservice call-outs we use SOAP, RESTful, **JPublisher** and JAX-RPC.
3. For JAX-RPC, Ant Application Server is used (Apache Jakarta server module).

   NOTE: Not all of the technologies listed here were neither fully utilized or implemented during the implementation of the project or for description in this Thesis.

   Here, in this project, we predominantantly use JPublisher and JavaScript for querying from the DataBase.

3.6 JPublisher

Definition

JPublisher is an utility that enables you to translate and represent PL/SQL database objects in a JAVA class for invoking external webservice.

3.6.1 Requirements

Transaltor.jar

This requires a specific set-up in Oracle 11g. It also requires a translator jar file to be installed at the *ORACLE_HOME directory/sqlj/lib* directory.

Webservice classes

Webservices classes are included in dbwsa.jar and dbwsclient.jar which are also found in *ORACLE_HOME directory/sqlj/lib*.

JDBC drivers

Most of the JDBC drivers are pre-installed while Oracle 11g installation. But, specific versions of JDBC drivers such as ojdbc7*.jar for JDK 1.7 and so forth can be manually installed at *ORACLE_HOME directory/jdbc/lib*.

3.6.2 Overview of JPublisher generation

A JPublisher connects can receive WSDL file from a Webservice provider and can create the following:

- Static Java class
- Proxy class or
- PL/SQL call

Based on the requirement and the webservice class that is being used, we can choose any of the above.

3.6.3 JPublisher support for Webservice call-out

In order to generate the classes and the SQL calls mentioned above the a WSDL file has to be given as an input. The publisher key for WSDL should be inputed as this: *-proxywsdl=url*. Figure shown below shows the webservice call-out stub generation.

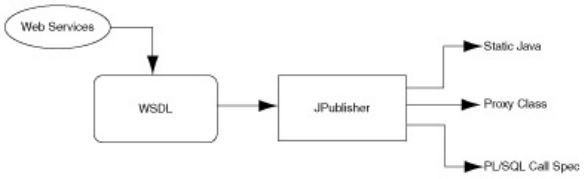

FIGURE 3.7: A simple Data Connector architecture

3.6.4 Sample Java Static Proxy Class Load and JPublisher commands

The requirements mentioned above along with Oracle 11g is necessary. Once the dbwsa.jar is added to the system's environment variable and dbwsclient.jar is loaded into SYS schema, any JavaScript file for a webservice can be invoked in the following fashion using the 'loadjava' command and the .jar files:

```
loadjava -u theo -r -v -f -noverify -genmissing dbwsclient.jar
Password: password
```

A simple JPublisher command will look something like the one shown below. Which, also will include commands such as -httpproxy (which is normally used to bypass firewall), -endpoint (Webservice endpoint URL), -sysuser (Webservice username and password), -dir (directory for temporary operations), -proxyopts (for anyother proxy options).

```
jpub -user theo -sysuser sys/sys_password -
proxywsdl=sample/javacallout.wsdl
   -endpoint=http://login.salesforce.com/…-dir=tmp

Enter theo password: password
```

3.7 Sample webservice invocation

The JAVA code shown below is a sample one that allows one to query any table we need to query from any data source. The code is not unique and hence can be easily used for any other data sources as well. We simply have to replace the column names and the ids of the table contents and replace the webservice endpoint URL.

There are separate functions for multi-table selects and other additional invocations.

3.8 Multi-Table invocation

This can simply be done by adding another function to call for table contents from a different table. As shown in the code below, we need a new *tableschema* function so that multiple tables can be invoked from a single webservice call.

```
myConnector.getSchema = function(schemaCallback) {
        var cols = [{
            id: "id_of_the-Coumns",
            dataType: datatype_of_the_column
        }...
                ];

        var tableSchema = {
            id: "schema_id",
            alias: "Name_of_the_table",
            columns: cols
        };

        schemaCallback([tableSchema]);
    };

  myConnector.getData = function(table, doneCallback) {
          $.getJSON("http://login.salesforce.com/...", function
              var feat = resp.features,
                  tableData = [];

              // Iterate over the JSON object
              for (var i = 0, len = feat.length; i < len; i++)
                  tableData.push({
                      "id": feat[i].id,
                      // Other Column ids
                  });
              }

              table.appendRows(tableData);
              doneCallback();
          });
      };
```

3.9 Incremental Refresh

CDC here is implemented as Incremental Refresh by invoking the same webservice script with the following set of code that allows one to query data whenever it is available.

```
for (var i = 0; i < max_iterations; i++) {
            data.push({
                "id": id,
                //other column ids
            });
```

Chapter 4

Results and Implementation work

Objective

Here we are simply trying to perform Big Data analytics in a near real time. This involves a lot of challenges. Aggregating, OLAP and OLTP transactions have to be near real time - to accommodate this, the infrastructure should be powerful enough to query such a large amount of data and transact such sheer volumes at near real time. CDC (Change Data Capture) as mentioned earlier is the biggest challenge, but, in order to achieve this, here, we are simply using Webservice. The major challenge this method is the sheer transactional load on the OLTP and OLAP systems. In order to cope with the given load of data, we simply build the whole infrastructure in a cloud (auto scalable) platform. Per [9] and [10], Dynamic Mirror Storage technique is not suited, as each and every transaction is unique and has its own validity and hence, it is imperative to capture every transaction and store it as is.

The results shown here are simple comparison of the size of data used by HTSL over the years for report generation by manual extraction by reporting analysts.

4.1 Results and Comparison

Below shown results are for 7 distinct SaaS applications. Most of them external applications and some of them internal. The 7 data sources are listed below as follows:

- SFDC
- SAP

- OW (Onewindow - Internal)
- IEX (Workforce Management)
- HD-CRM (Customer Relationship Management - Internal)
- Avaya
- AMS (Attendance Management System - Internal)

They are not necessarily represented in the same order in the following charts. The data size referenced (or used) for result generating purposes are sizes **per extraction**. The actual size of the data set extracted from each of these data sources are huge and cannot be comprehensively used for result analysis purposes. As, the extraction and report generation time of the implemented (proposed) method simply outwits the traditional performance of manual extraction and report generation strategies.
Hence, for the sake of simplicity and easy understanding of the staggering improvements of the proposed method, we consider the data size of a single extraction from the aggregated data source - which is our warehouse. The size of this data varies based on the type of the report requested by the management. So, here we take the reports that must be necessarily available for the management everyday for business strategies.
As data is unique from each of these data sources, the sizes of data extracted from each of these differs by a large scale and sometimes does not. The results shown are based average data sizes required to build a comprehensive report from these data sources.

4.1.1 Size vs Extraction time - Traditional

The bar chart shown below shows the time taken to extract data from these 7 sources manually by reporting analysts over a period of time.

4.1.2 Size vs Generation time - Traditional

The bar chart shows the time taken by reporting analysts to generate report over a period of time.

4.1.3 Size vs Extraction - New

As shown in Fig. 4.1, the extraction time for the data is huge and takes longer time to manually extract at any particular time. This becomes even more challenging when this has to be done by a person manually. The bar chart below shows that the extraction time

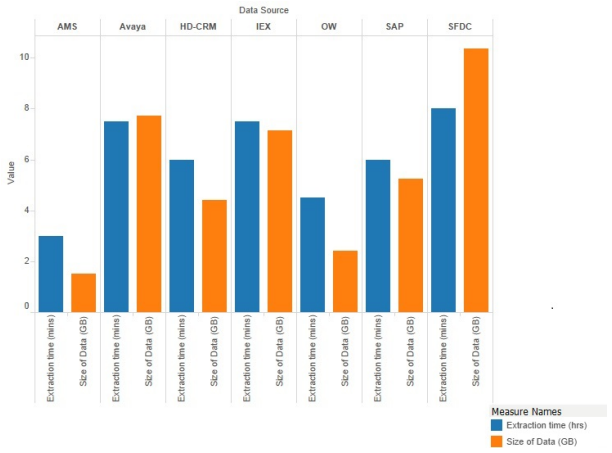

FIGURE 4.1: Size of Data vs Extraction Time - Traditional

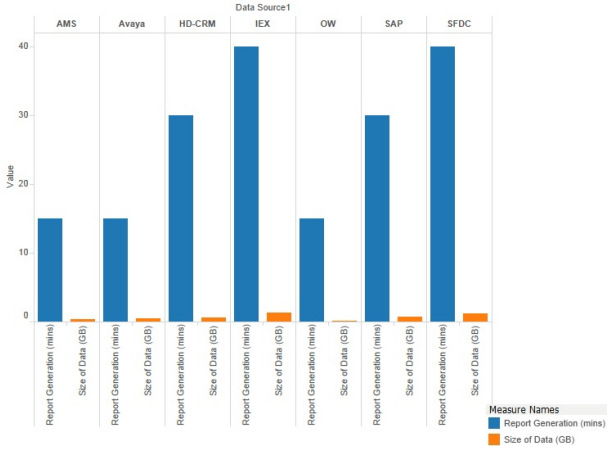

FIGURE 4.2: Size of Data vs Report Generation Time - Traditional

for our project is flat out and is available for all reporting analysts and the management at any given instance as the extraction time is drastically reduced and which is usually scheduled before the start of a working day.

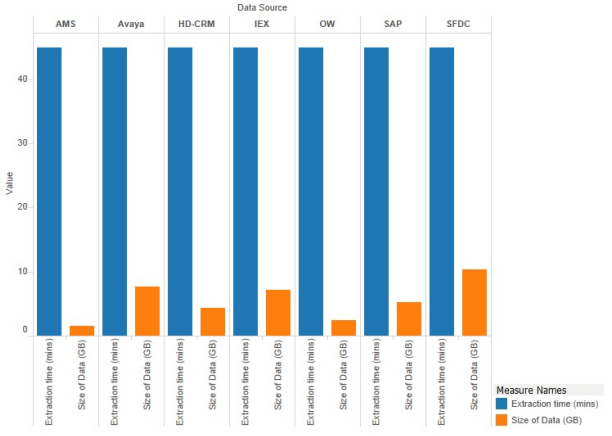

FIGURE 4.3: Size of Data vs Extraction Time - New

4.1.4 Extraction vs Report Generation - a comparison between traditional and the new method

The Box plot below shows the variance in the time taken for extraction in the traditional method versus the proposed method and also the report generation time for the same. As in the proposed method, the reporting is done through tableau, as and when the data is available, reporting facility is also available.

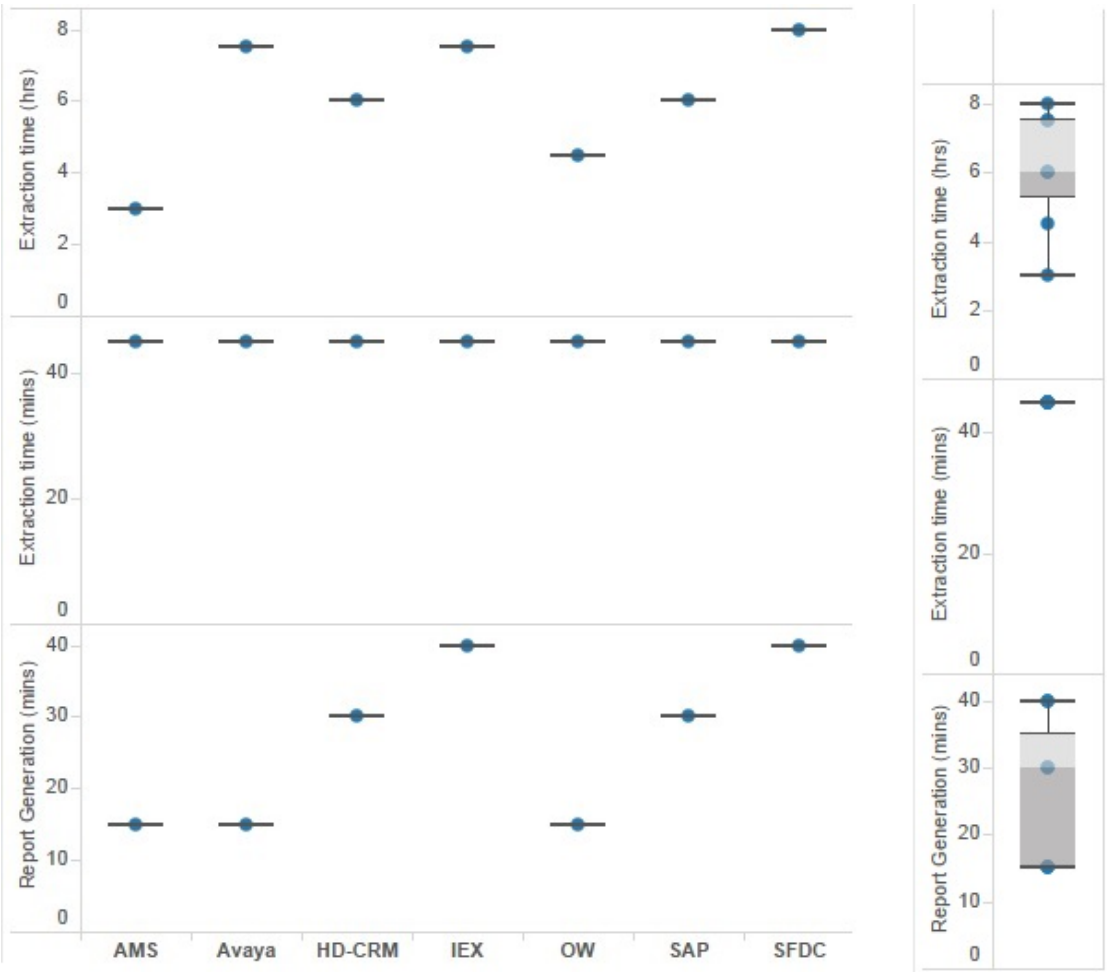

FIGURE 4.4: Extraction Time Old vs New vs Report Generation Time

Chapter 5

Conclusions

This system is currently being built at Honeywell Technology Solutions Lab (HTSL) Pvt Ltd, Benguluru, India and is partially functional as of today. This system is dynamically scalable and any changes to the system's functionality will not affect the logic. The infrastructure is robust and can adopt to any changes.

The ELT process for data warehousing at HTSL is automated. It does not use any gatekeeper application in the middle to perform any intermediary transformation before loading the data gathered into the DWH. This results in faster ELT process. The extraction process scouts for any incremental data values in all of the data sources and will auto update its data repository. The ELT process is scheduled to run when the system is not busy with read/write transactions. Because, the transformation process is done at a later stage, data duplication is a concern. But, most of the duplicated data in the DWH has its own validity and cannot be simply removed or redacted. The virtual table feature provided by Oracle 11g allows the presentation layer to query rows or sets of rows from the DWH as and when required by the user.

The results clearly shows that the proposed technique clearly outwits any traditional manual extraction of data. This also proves that the proposed technique given the infrastructure capability, can be used in enterprises where data is piling exponentially without having to use any complex algorithms for analytics.

Bibliography

[1] Mark Hansen, Stuart Madnick and Michael Siegel. 'Data Integration using Web Services.' MIT Sloan School of Management : MIT Sloan, 2002.

[2] David Ostrowski, Nestor Rychtyckyj, Perry MacNeill. 'Integration of Big Data Using Semantic Web Technologies.' eFord Motor Company. Mira Kim Dept. of EECS, University of CA, Irvine.: IEEE Tenth International Conference on Semantic Computing (ICSC), 2016.

[3] Bo Zhou. *D*ata Integration as a Service for Applications Deployment on the SaaS Platform. Hechi University, Yizhou City, Guangxi, China : 6th International Conference on Biomedical Engineering and Informatics (BMEI), 2013.

[4] Decheng Qiu, Junning Liu and Guoying Zhao. 'Design and Application of Data Integration Platform Based on Web Services and XML.' Hexi University & Fudan University, Artillery Training Base, PLA 371Hospital, China. : 6th International Conference on Electronics Information and Emergency Communication (ICEIEC), 2016.

[5] Gabriele Bavota. 'Mining Unstructured Data in Software Repositories: Current and Future Trends.' Free University of Bozen-Bolzano, Bolzano, Italy : IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016.

[6] Tulika Das, Sheryl Maring, Rick Sapir, Michael Wiesenberg. 'Oracle Database Java Developer's Guide 11g Release 1(11.1) B31225-05.' Oracle Corporation, USA. December 2009.

[7] Rahmadi Wijaya. Bambang Pudjoatmodjo2. 'An Overview and Implementation of Extraction-Transformation-Loading (ETL) Process in Data Warehouse.' Telkom University, Bandung, Indonesia : 3rd International Conference on Information and Communication Technology (ICoICT), 2015.

[8] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. 'Data Mining with Big Data.' Fellows IEEE : IEEE Transactions on Knowledge and Data Engineering, VOL. 26, NO. 1, 2014.

[9] Yingchi Mao, Wei Min, Jiulong Wang, Bicong Jia, Qing Jie. 'Dynamic Mirror Based Real-Time Query ContentionSolution for Support Big Real-Time Data Analysis.' College of Computer and Information Engineering, Hohai University, Nanjing, China. : 2nd International Conference on Information Technology and Electronic Commerce (ICITEC) 2014.

[10] Xiaofang Li, Yingchi Mao. 'Real-Time Data ETL Framework for Big Real-Time Data Analysis.' Changzhou Institute of Technology, China. Hohai University, China : IEEE International Conference on Information and Automation, 2015.

[11] Shahin Vakilinia, Xinyao Zhang and Dongyu Qiu. 'Analysis and Optimization of Big-Data Stream Processing.' Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada. : IEEE Global Communications Conference (GLOBECOM), 2016.

[12] Libina Rose Sebastian, Sheeba Babu, and Dr. Jubilant J Kizhakkethottam. 'Challenges with Big Data Mining: A Review.' Dept. of Computer Science and Engineering, St. Josephs College of Engineering and Technology, Palai, Kerala, India : International Conference on Soft-Computing and Network Security (ICSNS), 2015.