CHAPTER 3

REAL TIME DATA INTEGRATION ARCHITECTURE

(RTDIA)

This chapter defines the design and components of the proposed architecture for the fulfillment of requirements to attain a real time data integration environment. The later part of the chapter deals with the algorithm developed for the identification and integration of data from remote data bases into the central data warehouse and the software developed for testing of the algorithm.

3.1. REAL TIME DATA INTEGRATION ARCHITECTURE.

The basic idea behind designing a “Real Time Integration Architecture” is to provide flawless conversion of data to information and that back into action. In the current E&P data integration and processing sector two major bottle-necks have a slowing down effect on the system which creates a barrier for achieving the goal. Contemporary architecture suffers primarily from improper data identification and then an inefficient data integration system in the E&P sector which leads to a huge loss of time, skilled manpower and sometimes to loss of precious natural resources. Although the E&P industry has highly skilled human resource available for the analysis part but the above stated problems of data integration are a constant source of hindrance.
Figure 3. exemplifies the current scenario of the E&P industry’s information management system. RTDIA’s basic objective is to diminish resistance and enable data integration between operational, tactical and strategic levels of process control. Fundamentally the two objective of the architecture would be:

I. Capability to deliver value addition.

In the current scenario companies are not only hindered by less resource, extremely constrained time line but also has to deliver higher throughputs. To dispense enhanced outcome faster, G&G and engineering departments requires deliberating on their domain works and not on finding and processing of data. Highly skilled workforces are costly to come by, so it is vital that their work hour is properly utilized.

II. View Integration.

To understand the proper and correct working scenario requirement of an integrated view which divulges pertinent information at a single view is overriding. e.g. in an exploration situation, the comprehensive outlook must comprise of tornado analog well production histories, charts that measure risk, rock properties, rig schedules, log files, and other variables relating to the prospect in question.

The concept of this research is to implement recent technological enhancement to device an architecture which would be able to eliminate manual intervention existing in current systems and automate both the flow of information from
operational to tactical to strategic layer, representing data to the information stage, and the actions necessary to translate strategic objectives back to operational drivers to effect strategic decisions in real time.
Figure 3.1. Architecture I [Upper Layer].
The expanded form of the lower level of the architecture is described in an expanded form in figure 3.6 in this chapter.

The above given figure describes a real-time data integration architectural overview. The more prevalent integration tools are basically inert in nature and their sole purpose is to accommodate reports and dashboard. Current BI tools are mainly passive and their objective is to cater for the information consumer by providing reports or at most dashboard-like monitoring of various business processes. With a real-time architecture in the information management system which allows effective and seamless integration throughout heterogeneous systems, the process of data organization and extraction of operational information for decision support systems would become an effortless procedure and proper attention could be given to functioning of control processes. The architecture needs a smart algorithm to converge the data flow from the physically distributed system. The algorithm needs to be able to efficiently identify the requirement from intelligent metadata repository as well as a rollback schema as a failsafe backup.
Figure 3.2. Expansion of “Lower Layer” Architecture
3.2. THE LOWER LEVEL ARCHITECTURE (EXPANDED).

The best solution for any problem is the simplest one. In the above depicted architecture the base platform of all real time data generation like SCADA system is based over a cloud platform, which in turn will provide a highly virtualized environment to enable unimpeded communication among the different heterogeneous data sources. With cloud storage platform the most inconvenient problem which has dogged illustrate E&P industry till date, of heterogeneous source databases for integration would invariably reduce the time and effort for preprocessing of data required before diverse database integration.

There are two types of logical data marts designed in the architecture above

I. Dynamic Logical Data Marts. (*Part-I of the figure)*:

As the name illustrates, the data marts/ specific sector data warehouses which varies with time is defined here as dynamic logical data marts. The time variant databases are the real time data generations processes where the type and definitions of data remains constant over time but the value changes with time which is ultimately required for any further processing for Business Intelligence. These databases contain just the day to day values generated by the different operational processes which transpire everyday in course of exploration and production of hydrocarbon. e.g. exploration data marts, production data marts, drill data marts.
II. **Static Logical Data Marts. (Part-III of the figure)**

The specific databases which remains constant over time like the definition/schema of a database is defined as static logical data marts. They basically consists of two parts the

a. Meta Data Repository

b. Archival Data Repository.

Although these databases are not static over time in the real sense but the time span before which anything changes in these databases occurs is quite big, so these databases can safely be affirmed as static.

a. **Meta Data Repository:**

It contains critical information about the data units which are stored in the dynamic data marts which works in real-time. The basic definition of every data units has to be well defined without any ambiguity as this could lead into data identification problem which will further cause significant loss in time for data integration which is the main aim of the research.

b. **Archival Data Repository**

It is separate database created for storage of historical data for reference purpose, or for different trend analysis. The storage could be in different media thus the identification and extraction of these data could vary from seconds to hours.
depending upon the category of information as well as the type of media it is stored in. The storage media normally could be detachable magnetic cartridge, online hard drive or robotic tape drives. The data is segregated according to the requirement and importance and divided into these different storage media.

3.2.1. DATA INTEGRATION LAYER (*Part-II of the figure*).

The main conception of this architecture is the smart data integration layer. Customarily this amount of data is handled through ETL processes which are an extraction, transformation and loading tool which wheedle out data from different data marts and load into the central data warehouse for any type of analysis purpose. But ETL tools have their own pros and cons as discussed in chapter 2 of this research report. The designed algorithm as stated later in this chapter works in simple steps by identifying the data vertical by its asset-id, then brings in all the related data to the HANA database and creates a table in the In-Memory. From the In-Memory database table different slicing and dicing of data is performed according to the requirement.

I. Benefits:

1. **Unified Data Layer:**

A collective meta-data repository configuration amalgamates data access by establishing a virtual warehouse view of every required distributed data so
that users, in spite of their departments, have entrée to the same values and sources.

2. **Streamlined Processing Cycle:**

Logical data integration through a source description generator brings a simple streamline structure to the flow of data from the heterogeneous data sources to the data warehouse. The link created between a unified meta data repository and the data source makes it absolutely seamless processing and integration system.

In this designing attempt effort was to design an optimum architecture which would be able to handle the humongous pressure created in an E&P company and still be able to provide data in a real time scenario. The heart of this architecture is the data identification and integration algorithm.

### 3.2.2. HANA DATABASE.

Semiconductor memory is by far the fastest type of memory available today and with the increasingly depreciating price of semiconductor main memory (RAM) it is now financially feasible to have large arrays of RAM for high speed processing of colossal volumes of data. HANA Database enables the user to utilize the high speed of semiconductor drives as well as multi core processor to give ultimate proficiency in data analysis and transaction. The SAP HANA supports relational data, graph and text including semi and unstructured data as well contained in the same system making it the most effective option for the current problem. SAP
HANA is also cloud enabled which makes it easier to incorporate in the above described architecture. HANA being 100% ACID (Atomicity, Consistency, Isolation, Durability) compliant making it one of the most reliable Database. [46,116]

In the below given figure 3.3 and figure 3.4 the HANA architecture is depicted in two stages. The figure 3.3 describes the base architecture of HANA database. In it basic form, it shows three basic component of In-memory computing. The first part consists of the distributed databases from where the data is retrieved and brings it into HANA. The third component is the BA/BI tools which communicate to the high speed memory to for further processing of data.

Figure 3.4 depicts the different components which and how it communicates with the HANA engine. Basically the distributed databases are accessed by the developed real-time algorithm as well as the metadata repository through high definition file servers. The extracted data is then brought onto the HANA engine through a Sybase replication engine without making any changes in the primary data source and creates a table which is accessed by any integrated BI tools, making it highly agile and effective tool for real-time data operations.
Figure 3.3. Basic Data Flow Diagram of HANA

Figure 3.4. Component Diagram of HANA Engine
3.3. IDENTIFICATION & INTEGRATION ALGORITHM.

3.3.1. INTRODUCTION.

The required algorithm should not only be able to identify the proper data faster and extract it to the HANA database to make the system real-time compliant. The algorithm stated below tries to find an optimum solution for the present problem by approaching a three dimensional table concept and bring in the required datasets into the HANA database uniquely identified by their Asset-ID. The algorithm then applies different association rules to be able to find the required output in real-time scenario.

This enables a continuous real time data extraction and integration of E&P data into the in-memory HANA Database for further processing and analytics. Through this process the basic problem identified in the beginning of the research of manually locating and extracting data for different analysis would be converted to automatic, convenient and highly accurate.

3.3.2. DATA MODEL.

When we are handling petabytes of data the problem is the huge amount of time consumed in serial processing which normally may run in hours or even days. To handle the problem the algorithm should be able to implement parallel processing as the amount of data to be handled runs into petabytes normally.
To enumerate the other properties of the algorithm which are required to enhance the throughput of the system are given below.

1. Requirement of extremely high read/write operations
2. Economical scanning of data sets and subsets
3. Proficient large scale joining of datasets through one-to-one and one-to-many relationship.
4. Keep record of change of datasets over time through “time-stamping”.

3.3.3. STORAGE SYSTEM.

The storage system could be described as a light, multi-level, distributed sorted map. The map in question is indexed three ways to make an efficient and fast identification and retrieval of data. Indexing is done by row key, column key and timestamp.

I. Rows:

The row key can be defined as an atomic integer in any case of the number of column being written or read in that same row. This gives efficiency to the identification of data in spite of presence of concurrent updates to the same row.

Row keys are essentially the “Asset ID” of the exploration or production unit. It is given to be able to identify uniquely each source of generated data. In exploration and/or production weather it is a production well or exploratory well it is called an asset and given a unique ID which is called
Asset ID. This enables reduced communication with time and interaction with a limited number of systems. The search algorithm first refers to the row key as per the request made by the client, the master indexer contains the location of the required row key mapping to get the data at a higher rate of access.

II. Column:

Column keys are assembled into sets identified as “column genus”, this is the basis of access control. In a column genus same type of data is stored (e.g. electric log, gamma log, etc). The column genus has to be separately generated before data can be stored in it. The number of genus is kept low and less susceptible to any change during the different operations.

A column key consists of “Genus + ID”, the ID is an arbitrary integer. The genus is a name of the database type in which it is saved (e.g. Oracle, MS-SQL, MySQL, etc) and the ID is given to any particular database. The ID also enables an anchor to figure out the location of the genus.
Figure 3.5. Structure Of Data Model-1

Figure 3.6. Structure Of Data Model-2
III. Timestamps:

The indexing is done by timestamps to enable multiple variants of the same data. In an electric log sample in a particular location varies with time and that is required to understand the current scenario of the reservoir and compare it with the former one so that a prediction could be made accordingly. The timestamp is generated by the synchronized system through the database. The indexing by the timestamp is done by descending order so that the most recent data is kept at the top which is read first.

3.4. DATA IDENTIFICATION AND ASSOCIATION ALGORITHM.

Algorithm 1:

// create table in HANA

1. Create Table *T = OpenTable ("Virtual table Name in HANA");
2. Row Mutation r1(T, "asset id"); // Write a new anchor and delete an old anchor
3. r1.Set ("anchor:Asset ID", "376594");
4. r1.Delete("anchor:12345");
5. Operation op;
6. Apply (&op, &r1);
In this algorithm the basic table is created in the HANA database and the table is populated from the source remote database.

Algorithm 2:

//Row mutation $r_1$ algorithm

//Merging of two partial tree for virtual organization in HANA

**Input:** $T_1$ and $T_r$ \{two input partial trees\}

**Output:** $T_m$ \{merged partial tree\}

1. $k \leftarrow$ the least key in $T_r$
2. $h_l \leftarrow$ GetHeight($T_l$)
3. $h_r \leftarrow$ GetHeight($T_r$)
4. $h_{\text{min}} \leftarrow$ GetMin($h_l, h_r$)
5. if $h_l \leq h_r$ then
6. $l_m \leftarrow$ the left most node in $T_r$ at $h_{\text{min}}$
7. add $k$ to $l_m$
8. $\text{merged} \leftarrow$ merge the root of $T_1$ and $l_m$
9. return $T_m \leftarrow T_r$
10. else
11. $r_m \leftarrow$ the right most node in $T_1$ at $h_{\text{min}}$
12. add $k$ to $r_m$
13. $\text{merged} \leftarrow$ merge the root of $T_r$ and $r_m$
14. return $T_m \leftarrow T_1$

15. endif

This algorithm performs row mutations in HANA database to create the required primary table on which further operations are performed.

Algorithm 3:

// User input required

1. Anchor Asset ID
2. D data type required
3. R Range of Time Stamp
4. C Association condition for data

These are the user inputs which are received from the user interface.

// Read Data from Table in HANA

1. Scanner scanner(T);
2. ScanStream *stream;
3. stream = scanner.FetchColumnFamily("Anchor");
4. stream SetReturnAllVersions("R");
5. scanner.Lookup("D");
6. for (; !stream Done(); stream Next()) {
7. printf("%s %s %lld %s\n", scanner.Anchor(),
9. Stream ColumnName(),
10. Stream Timestamp (),
11. Stream Value ();

}

**Algorithm 4:**

// Searching for null value in the streamed data

1. For all streamed data
2. If D(value) =
3. Delete from stream r_l
4. Else
5. Send D to Procedure (C).
6. End If.

// procedure association

1. If C ≠ 0
2. Apply C on Stream r_1
3. print Output
4. else
5. Print r_1
6. End If
3.5. WORKING OF THE ALGORITHM.

The above given algorithm basically works in two stages to form a table, in the first stage it accepts the asset-id from the user and creates a table in the HANA database by pulling in data from all the remote databases where the asset-id is common, and then the different rows are joined together to form the base table which contains all the different tables of an asset together. Next part creates a snow flake schema in the database for handling of three dimensional data as shown in figure 3.8. then further processing of data is performed according to the user input described in the algorithm.

3.6. IMPLEMENTATION OF THE PROPOSED ARCHITECTURE.

In the formal circumstances the architecture of the whole architecture is out of question for any E&P company, so the way out is to be able to modify the existing architecture in a way that neither it causes an abrupt shutdown of the day to day business nor require a huge investment for a parallel infrastructure. Thus the architecture has to be such which needs to be able to incorporate the existing hardware in such a way that it does not create any disturbance during the procedure of implementation. This could be easily possible for this architecture because of two reasons, primarily of the cloud storage which could incorporate any type of software as well as hardware which are already in action and secondly a light and agile algorithm which is able to communicate to any type of system at a very high speed with minimum failure rate.
3.6.1. THE DATA IDENTIFICATION SCENARIO.

In the industry today the different ways to associate data is by asset-id, field-id, etc which is described in figure 3.7 as given below. This leads to ambiguity in the data identification process and loss of time. The standard process could feature any one of them which could relate to any other of the components. This not only reduces ambiguity but loss of time. Under certain condition, if it is required to find the associated data with something else other than Asset-ID, it could also be achieved by the relationship with the Asset-ID. So ultimately it does not create any inadequacy in the system.
Figure 3.7. Data Relationship
3.6.2. SOFTWARE WORKING DESCRIPTION.

The developed software is a prototype to act as a technology demonstrator; it is not a fully fledged deployment package. This software demonstrates the successful working of the algorithm designed for the solution of the problem at hand. It is not in any way made to handle each and every aspects of E&P data manipulation in its current condition. Due to the constrained time and funds, developing such a package is beyond the scope of this work. But the basic structure of the software is well grounded and tested to work efficiently as expected. So any further work could easily be based on this work without any or with a minimum changes.

3.7. Testing.

In this chapter the testing of the above algorithm is described. It discusses about the setup of the testing platform, the tables, the databases as well as the forms created.

3.7.1. TESTING ARCHITECTURE.

A scaled down architecture is designed for the testing purpose. The design of which is pictographically represented in figure 3.7. In this architecture for the convenience of understanding and monitoring of data integration three separate databases are created in MS-SQL Server 2010. These three data bases contains multiple table of the same table consisting of
1. Well Table
2. Pressure & Temperature Table
3. Production Table
4. Steam Injection Table
5. Assay Table
6. Structure Table

The components of the tables are same as would have happened if the databases would have been in three different locations of drilling. The concept here is to create a scaled down version of the real scenario.
Figure 3.8. Architecture of the Setup.
3.7.2. THE SETUP.

The environment consists of two major components the databases are created in MS-SQL Server 2010 and the Human Machine Interface (HMI) is developed in Visual C++. The operating system used is Windows 7.

The fields and the data types are of the three tables are described in the following three tables. The data entered in these tables are random numbers confirming to the required data types. This enables the system to be tested correctly as far as possible without infringement of any type of national and international law. The tables are as designed below.
Figure 3.9. Primary Table Structure.
These are the six tables which to be created for testing purpose. The structure of these tables is described below.

### 3.7.3. TABLE STRUCTURE.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>Varchar2</td>
</tr>
<tr>
<td>Co-ordinate -X</td>
<td>Float</td>
</tr>
<tr>
<td>Co-ordinate -Y</td>
<td>Float</td>
</tr>
<tr>
<td>Co-ordinate -Z</td>
<td>Float</td>
</tr>
<tr>
<td>Condition</td>
<td>Varchar2</td>
</tr>
<tr>
<td>Type</td>
<td>Varchar2</td>
</tr>
</tbody>
</table>

Table 3.3. Asset
<table>
<thead>
<tr>
<th>Fields</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Drill Head ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>X</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Y</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Z</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>MD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>INCL</td>
<td>FLOAT</td>
</tr>
<tr>
<td>AZIM</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DX</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DY</td>
<td>FLOAT</td>
</tr>
<tr>
<td>TVD</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Calc DLS</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>DATE-TIME</td>
</tr>
</tbody>
</table>

Table 3.4. Well
<table>
<thead>
<tr>
<th>Fields</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Cu (ppm)</td>
<td>INTEGER</td>
</tr>
<tr>
<td>AU</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Interval (m)</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Sample ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>DATE-TIME</td>
</tr>
</tbody>
</table>

Table 3.5. Assay

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Type</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>α</td>
<td>INTEGER</td>
</tr>
<tr>
<td>β</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Calc Dip</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Fields</td>
<td>Data Type</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Calc Dim</td>
<td>FLOAT</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>DATE-TIME</td>
</tr>
<tr>
<td>Null value accepted</td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Fields</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>BWPD</td>
<td>Integer</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>DATE-TIME</td>
</tr>
</tbody>
</table>

Table 3.7. Steam Injection

<table>
<thead>
<tr>
<th>Fields</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset ID</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>well</td>
<td>Integer</td>
</tr>
<tr>
<td>Oil Production(BBL)</td>
<td>Integer</td>
</tr>
<tr>
<td>Oil Production cumulative (BBL)</td>
<td>Integer</td>
</tr>
<tr>
<td>Fields</td>
<td>Data types</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Asset ID</td>
<td>Varchar2</td>
</tr>
<tr>
<td>Record time</td>
<td>Date Time</td>
</tr>
<tr>
<td>Time elapsed</td>
<td>Time</td>
</tr>
<tr>
<td>Pressure (Psia)</td>
<td>Float</td>
</tr>
<tr>
<td>temperature</td>
<td>Float</td>
</tr>
<tr>
<td>comments</td>
<td>Varchar2</td>
</tr>
</tbody>
</table>

Table 3.9. Temperature & Pressure
The cube structure is as follows.

![Data Cube Structure](image)

**Figure 3.10. Data Cube Structure.**

![Star Schema](image)

**Figure 3.9. The star schema**
For processing purpose this star schema is created to handle the three queries by the intended software, the queries are as follows.

The three questions to be answered by the system are

1. Maximum producing oil well.
2. Average water injection in a year.
3. Minimum fluid pressure in the last quarter.
3.7.4. THE FRONT END.

Figure 3.12. User Interface 1.

1. Query for generating list of asset_id

```
SELECT ASEST_ID FROM ASSET_MASTER
```

For the button “Generate Table”.
Figure 3.13. User Interface 2.

1. Query for “select data family”

------ Code Start --------

declare @val_to_search varchar(50), @column_name varchar(50)
Select @val_to_search = 'DDH245', @column_name='Asset_ID'

declare tbl cursor for

select table_name from information_schema.columns where column_name=@column_name

declare @tablename varchar(200),@qstr varchar(max)
declare @datapen table(table_name varchar(200))

open tbl
fetch tbl into @tablename
while @@fetch_status=0

begin
select @qstr='select top 1 '''+@tablename+''' from '''+@tablename+' where '+ @column_name + ' ='''+@val_to_search + ''''
insert into @datapen
exec(@qstr)
fetch tbl into @tablename
end
close tbl
deallocate tbl
select * from @datapen pen

------ Code End ------

2. Query for Select primary & secondary component. (List of column name in a table)

SELECT COLUMN_NAME,* FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'selected table' ORDER BY ORDINAL_POSITION

3.7.4.1. ASSOCIATION RULES.

1. Minimum
2. Maximum
3. Sum
4. Average
5. Difference
3.7.4.2. QUERIES FOR ASSOCIATION

1. Query for Maximum production

SELECT Asset_ID, MAX(Oil_Production_Cumulative_BBL) FROM Production_2 group by Asset_ID

2. Average steam injection.

SELECT AVG (DISTINCT BPWD) FROM SteamInj_3

3. Maximum pressure

SELECT MAX(Pressure_PSIA) FROM Pressure_Temp_2 WHERE Real_time >='2013-07-05 11:15:00.000' AND Real_time <= '2013-07-05 12:10:00.000'.

4. Minimum pressure

SELECT MIN (Pressure_PSIA) FROM Pressure_Temp_2 WHERE Real_time >='2013-07-05 11:15:00.000' AND Real_time <= '2013-07-05 12:10:00.000'.

Questions to be answered

The three questions to be answered by the system are:

1. Maximum producing oil well.
2. Average water injection in a year.
3. Fall of fluid pressure in the last quarter.
3.7.5. WORKING OF THE SOFTWARE.

To begin with data is entered as per the constraints given in the databases into all the tables in the three different databases. Then the update button is executed for inclusion of all those data into the central data warehouse to the predefined specific data bases created as per the requirement.

Once the backup button is clicked it initiates another form consisting of start and stop button. This commences the Identification & Integration Algorithm; the algorithm looks into the different databases and equates the time stamp of the central data warehouse tables with the remote databases time stamp, if the time stamp of the remote database is greater than that of the central data warehouse. The algorithm generates an updated query automatically thus updating the data into the central data warehouse.

3.7.6. THE BENEFITS OF THE SYSTEM.

Here the salient features and the benefits of the designed system is enumerated below for clear understanding of the properties.

1. Streamlined architecture.

2. Utilization of existing hardware infrastructure.

3. Easy integration of existing tools.

4. Highly reduced cost of technology migration.

5. Full control of the data flow.

6. Implementation of cloud makes it highly platform independent.
7. Very high throughput by utilizing in-memory computing and optimized algorithm.

3.8. Results

Within the constrained limits of the testing the algorithm shows expected output on all the aspects of data identification and integration. The concept behind the algorithm being simple makes it highly efficient and fast. The time of query generation was set from every 4 seconds to 20 seconds and each time ran flawlessly. The targeted destinations of the records in the central data warehouse were accomplished accurately with all entries. Even when the amount of data was increased the results were absolute.