# ENHANCING DATA STAGING AS A MECHANISM FOR FAST DATA ACCESS

REAGAN MURIITHI GATIMU

**MASTER OF SCIENCE**

**(Computer Systems)**

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE TECHNOLOGY**

**2016**

# Enhancing data staging as a mechanism for fast data access

**Reagan Muriithi Gatimu**

**A thesis submitted in partial fulfillment for degree of Master of Science in Computer Systems in Jomo Kenyatta University of Agriculture Technology**

**2016**

## DECLARATION

This thesis is my original work and has not been presented for award of a degree in any other University.

Signature…………………….................Date………………………………..

**Gatimu Reagan Muriithi**

This thesis has been submitted for examination with our approval as University supervisors.

Signature…………………….................Date………………………………..

**Dr. Michael Kimwele, PhD**

**JKUAT, Kenya**

Signature…………………….................Date………………………………..

**Dr. Wilson Cheruiyot, PhD**

**JKUAT, Kenya**

## DEDICATION

Firstly, I take this opportunity to thank the Almighty God, for giving me the strength and good health to carry out this research study to completion.

Secondly, I offer my regards and blessings to my lovely wife Sophia and son Raul for being patient at all times when I was very busy with research. Raul, your cries at night were timely because they woke me up to continue with data analysis.

Finally, my caring parents and siblings, thank you for supporting me in all aspects towards the completion of this course. Your encouragement and financial support benefited me a lot and may God bless you abundantly.

## ACKNOWLEDGEMENT

I am very grateful to my supervisors, Dr. Michael Kimwele, PhD and Dr. Wilson Cheruiyot, PhD for their constant guidance and support from the beginning to the end. Thank you a lot for accepting to be my supervisors and you mentored me well throughout the study. You were there to respond to all my requests in a timely fashion and I admired the way you handled my queries.

To Mombasa CBD campus, special thanks for allowing me to pursue the course there by providing support from the teaching and administrative staff.

To my employer, CP CIM-POOL AG, thank you for granting me the study leaves that enabled me to give the research full attention. You also allowed me to use the company's resources for research. This was a great motivation and it helped me to tackle the core problem with utmost professionalism.

## TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| **DSA** | Data Staging Area |
| **DTR** | Data Transfer Request |
| **DW** | Data Warehouse |
| **ETL** | Extract Transform Load |
| **ERP** | Enterprise Resource Planning |
| **FIFO** | First In First Out |
| **ISO** | International Organization for Standardization |
| **HANA** | High-Performance Analytic Appliance |
| **OLAP** | Online Analytical Processing |
| **OLTP** | Online Transaction Processing |
| **SAP** | Systems Applications Products |
| **SQL** | Structured Query Language |
| **UDF** | User Defined Functions |
| **EDI** | Electronic Data Interchange |

## DEFINITION OF TERMS

| | |
|---|---|
| **Data Staging** | Special feature of data warehouse for handling data from ETL |
| **Data mart** | A subset of the data warehouse that is usually oriented to a specific business line |
| **Extraction** | Retrieving of data in whichever format from sources |
| **Source system** | Support system where data is retrieved from |
| **Target system** | Destination system for the ready data |
| **Transformation** | Change of data state by subjecting to functions, procedures or algorithms |
| **Loading** | Transfer of the ready data to the data warehouse or mart |

## ABSTRACT

Most organizations rely on data that is generated after performing their daily transactions and operations. This data is retrieved from different source systems in a distributed network hence it comes in varying data types and formats. The source data is prepared and cleaned by subjecting it to algorithms and functions before transferring it to the target systems which takes more time. Moreover, there is pressure from data users within the data warehouse for data to be availed quickly for them to make appropriate decisions and forecasts. There has been a lot of delay in data delivery to the business users due to immense data explosion emanating from millions of transactions running concurrently. The current legacy systems cannot handle large data levels due to processing capabilities and customizations. The performance degradation has raised concerns since organizations invest a lot of resources to establish functioning data warehouses. Data staging, a technological innovation within data warehouses is targeted since most data manipulations are carried out here. It determines which data is to be integrated, harmonized by the staging functions, cleansed, verified, and archived for future use. The population selected to carry out the study was chosen amongst large organizational databases available online for research purposes. The stratified random sampling method was used to determine the sample frame for study. Several tools including Ms Excel, SQL Server Analysis and Integration Services were vital during data analysis and experimentation. The deterministic prioritization algorithm was developed and tested with a focus on data staging performance and efficiency. The proposed solution highlights the necessities of pre-determining the expected data loads and ways of prioritizing them and optimizing the execution plans. The experiment test runs for the different scenarios demonstrated in the study shows that data staging processing time improved by 2.66% and consequently the loading process time improved by 93.44%. Therefore, a recommendation to data warehouse practitioners and business intelligence designers was put forward to implement the Deterministic Prioritization algorithm providing enhancement for future design of Extraction, Transformation and Loading processes in data warehouse development.

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background

The growing number of business transactions in any enterprise is directly proportional to growth of data size. This data comes from variant source systems and applications and needs to be organized in a workable state so that it remains relevant and meaningful to the users. Technological development has led to the rise of Data Warehouse (DW). Inmon (2002) defined a data warehouse as “collection of integrated, subject-oriented databases designated to support the decision making process”. Both Kimball and Inmon (2002) agreed that a DW had to be integrated, subject-oriented, nonvolatile and time variant. This concept of time-variant was so crucial and ultimate concern and set the basis for this research. The foundations of a DW as explained by Zineb, Esteban, Jose-Norberto, and Juan (2011) encompassed integration of multiple different data sources. This allowed the provision of complete and correct view of the enterprise operational data which was synthesized into a set of strategic indicators and measures that the users of the data could associate with.

DW has business intelligence implemented in three major processes used to prepare data to match user’s needs. They are commonly referred to as ETL processes which are Extraction, Transformation and Loading. Extraction process retrieves data as is from source systems before subjecting it to any manipulations. Transformation process also referred to as transportation phase is the operational base and the most intriguing of all. Business rules and functions are some of the operations applied to the extracted data. Loading process involves moving the desired data as determined by the users to the DW. It’s important to note that, the flow of data from the sources to the destination is not as simple and smooth as it sounds. There results in impeding system performance observed across all the ETL processes raising more bottlenecks to data movement in data warehousing environment.

El-Wessimy, Mokhtar, and Hegazy (2013) showed the relevance of DW in decision making in today's environment. "The best decisions are made when all the relevant data is taken into consideration. Today, the biggest challenge in any organization is to achieve better performance with least cost, and to make better decisions than competitors. That is why data warehouses are widely used within the largest and most complex businesses in the world."

## Problems in ETL that call for use of Data staging

According to Javlin (2015), Data Staging means that the data is simply dumped to the location (called the Staging Area) so that it can then be read by the next processing phase. This is important if the transformation step fails, it should not be necessary to restart the Extract step. We can ensure this by implementing proper staging. IBM (2016) explains that data stage is keeping data before the job begins execution and the data is moved back to archives when a job has finished execution. The Data Staging Area is temporary location where data from source systems is copied. Rogers (2010) discussed that a staging area is mainly required in a Data warehousing architecture for timing reasons.  All required data must be available before data can be integrated into the Data Warehouse. According to Russom (2012) explained that data staging areas evolved from temporary storage platforms to pre- and post-processing platforms, they typically moved out of the data warehouse proper (where they were simply a few tables where data landed) and onto standalone database instances. Data staging emerged as a new technological development with an attempt to handle issues regarding low performance on data loads to the warehouses. Kimball and Ross (2002) stated that data staging was available in the extraction and transformation phases of ETL framework. In some legacy systems, data stage existed as a location that interconnected Online Transaction Processing Systems (OLTPs) to the Online Analytical Processing Systems (OLAPs). Although data staging was not a completely new technology since it had been researched before, the focus had been shifted to designs and development of data staging frameworks. Little attention had been given to its operability and its significant role in speeding the ETL process. It was a vital location on the basis that it linked the source systems (environment) to the target systems. The concern that arose was whether to deal with data staging as a process in ETL or the developed frameworks which were customarily available at the moment. Due to the vast amount of time and energy used in development of a framework the research did not dwell much on this but focused on enhancing the available frameworks with the realization of performance improvement.

In most of the developed systems for example Ascential DataStage and InfoSource from SAP AG (2002), usability was key and ideal for its simplicity of use. The data staging area was a technical layer where only the experts had capability to understand the processes. With this regard, the novice users at the presentation layer were abstracted from the many operations taking place inside. Eventually, the readily transformed data was moved to the DW for use and further categorized in the data marts repositories. This was referred to as the loading process. It depended on the laid down network infrastructure according to the designed distributed database system. The presentation systems and applications that linked to the business users could readily and easily access the data from DW and data marts for report generation, analysis and audit confirmations.

Aksoy, Franklin, and Zdonik (2001) explained that in large and busy organizations deployed in the production environments, the number of transactions was quite high as a result of many applications generating different kinds of data. There arose a performance problem where data flow speed reduced immensely. Another hurdle came in since there were no proper selection algorithms to pre-determine what category of data was clean for transfer to the data warehouse. El-Wessimy, Mokhtar, and Hegazy (2013) noted that all the data from the source systems was forwarded to the data warehouses without checking the relevance to the user decisions. To overcome the cited problems, a solution that would prioritize cleaned and verified data ready for loading would be a great move towards enriching data warehousing technologies.

Data staging remains without doubt a debated topic because of the conventional approach to data integration where extraction of all data from the source systems is done and later on the entire set is integrated again. This demonstrates a waste of CPU time since it involves handling all the data every time transformation is carried out. This approach is inefficient because the data delivered is not processed in real time due to the delay in the integration process. Kimball, Reeves, Ross, and Thornthwaite. (2008) found out that on the other hand, existing solutions put across are based on filtering records according to a timestamp column or flag that shows the changed data. These solutions are not productive and they require modifications in the applications from which they have been deployed.

## 1.2 Statement of the problem

Manipulating and loading large volumes of data concurrently is a time and resource consuming task. Data being manipulated by a certain process might be needed by another process to continue its task forming a cycle referred to as resource deadlock. The implication would be intense and could bring the system to a halt. Data growth in a company is a progressive trend which requires advance mechanisms to handle the vast amounts of data either on First In First Out strategy or concurrent processing. Lack of well designed process interrupts, scheduling plans and algorithms causes the systems to fall into unnecessary deadlocks. This makes data integration from the various source systems difficult thus increasing overhead that may result from the large number of individual queries being executed in data retrieval. These individual query procedures cannot predict the business users' dynamic needs, to match their demands then a lot of comparisons have to be done for the data that is already retrieved. This normally takes a lot of CPU running time hence impacting negatively on the performance of ETL processes. The core systems should adapt to change and restoring the operation space without its dependants noticing these changes. Fixing the noted problems in current systems requires dwelling deeper into the ETL framework development because they exist due to poor designs at the initial stages. This has to be applied at the application level where the ERL processes are carry out their different operations.

## 1.3 Objectives

The study was guided by the following objectives that were categorized as broad and specific objectives. Although quality had been had been given precedence in previous research works, the study aimed to balance between performance and quality in the resulting data presented to the data warehouses.

### 1.3.1 Broad Objective

To improve the speed of accessing, retrieving and processing data integrated from various source systems, transferred and synchronized with specific target systems. The improvement was to bring change on the performance of ETL processes in data warehouses when handling voluminous and bulky data.

### 1.3.2 Specific Objectives

1. To generate and implement prioritization algorithms for selecting appropriate data for collection from source systems.
2. To create scheduling plans for operations within the Transformation phase and optimize data load instead of using batch transfers or increasing processing space.
3. To streamline and integrate data processing strategies to achieve concurrent operation through parallelism of jobs with an aim of providing real time data processing.
4. To provide a scalable solution that would be easily integrated in new and existing data staging frameworks without the need to change development design and architecture.
5. To find out the performance change by testing the developed algorithm that proposes a solution to the performance issues within data warehousing environment.

## 1.4 Research Questions

1. How would prioritization algorithms change the data selection and collection from the sources?
2. What impact would scheduling plans have on the order of executing jobs in a staging area?
3. What is the business value after ensuring parallelism in concurrent data processing?
4. How would the solution integrate with existing and new systems highlighting incompatibility issues that may arise?
5. What support does the proposed solution provide in the growth of data staging technologies in future?

## 1.5 Justification

One of the greatest requirements for most any new systems is the ability to be very responsive, accurate and up to date for the business users to enjoy high performance. This should be maintained despite the significant delays introduced due to the nature of operations at the backend. The growth of information technology field saw new technologies such as cloud computing, virtual networks and new-age distributed systems emerge.

There was great push from the data warehousing community to implicate these speed improvements so that the real-timeliness of a system could be replicated across the board. In the previous systems, data was copied directly from the source transactional systems to the target systems to reduce the workload. This resulted in a lot of ambiguous and repetitive data which was rendered unusable. Data staging could leverage benefits from meta-data, additional data about data which was carried with the moved data.

## 1.6 Scope of study

The study revolved around the DW environment targeting the business processes involved in the flow of organizational data from source to destination. The complexity of the procedures depended on the phase under investigation in the ETL process. The research majorly targeted data from large business enterprises whose database organization supported real time data processing with constant data changes and modifications. Inclusively, data from companies that had already established DW environments were considered. The skill base in the research work targeted users with technical knowledge on flow business processes and logic.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1 Introduction

In this chapter the focus is on familiarizing with the field of study by investigating on research work that had been carried around this area by different researchers. It is vital to give credit to the authors for their work and their contribution to efficient and effective systems. Kimball and Ross (2002) said that data warehouse managers should ensure the right data is published depending on the type of business involved. A view of data warehouse environment in terms of location yielded four major regions namely; operational source systems, data staging area, data presentation area, and data access tools. There was an assumption that data fetching from the source systems was carried differently from the way data warehouses were queried for results. The source systems maintained little historical data to the greater advantage that relieved it the responsibility for keeping track of old data. The user queries fired against the source systems were minimal and procedural i.e. one query at a time hence the data flow was severely restricted to the demands of user requests for data.

## 2.2 Related Work

Kimball and Ross (2002) suggested that the key architectural requirement for the data staging area was that, it was off-limits to business users and did not provide query and presentation services. Data staging was an essential part of ETL phases and was considered to be the most crucial stage of data warehousing where maximum responsibility of data quality efforts resided. Ralph and Margy (2002) demonstrated the data staging area within a DW using an illustration in figure 2.1.

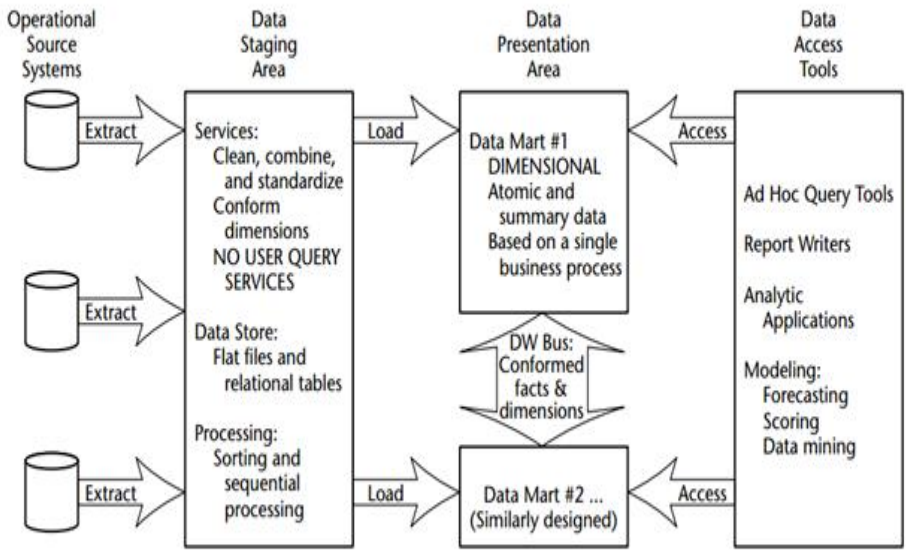


**Figure 2.1 Basic elements of the data warehouse Ralph and Margy (2002).**

Kimball and Margy (2002) noted that when staging dimension tables, they often handed a complete copy of the latest, greatest source data. “It would be wonderful if only the changes since the last extract, or deltas, were delivered to the staging area, but more typically, the staging application has to find the changed dimensions”.

Inmon (2005) defined “a data warehouse is a collection of Integrated, Subject-Oriented, Non Volatile and Time Variant databases where each unit of data is specific to some period of time”. Muller, Studer, Fondement, and Bézivin (2005) tackled the issue of ETL development and said that ETL process development constituted the most costly part of a data warehouse project, in both time and resources. The complexity of integration solutions continued to grow, with higher-quality data demands more-robust metadata and audit ability requirements.

There had been substantial amount of effort by other researchers to discuss deeper about ETL process modeling approaches, ETL software architectures and ETL frameworks as demonstrated by Thomsen and Pedersen (2009).

Bézivin (2005) stated that the data staging area of the data warehouse was both a storage area and a set of processes commonly referred to as extract-transformation-load (ETL). The data staging area was everything between the operational source systems and the data presentation area. The proposition here was to have prioritization mechanisms used to filter or sort data reducing the amount of flow to the data staging area. For this requirement, an algorithm was to be created to determine and decide beforehand what data was vital and must be moved to the DW therefore not all data was copied. There were limitations to full copying which looked like performing full backups e.g. bandwidth, transfer rates e.t.c. It had been investigated and the authors came up with either Opportunistic scheduling, pre-fetching or data caching approaches.

Vassiliadis (2009) further explained the Extraction, Transformation, and Loading Processes and their key role in data warehousing architecture. He stated that data was extracted from the source data stores, which could be in a relational and/or a semi-structured format. In typical cases, the source of data stores could be On-Line Transactional Processing (OLTP) or legacy systems, files under any format, web pages, and various kinds of documents or even data coming in a streaming fashion. After this phase, the extracted data was propagated to a special-purpose area of the warehouse, called Data Staging Area (DSA), where their transformation, homogenization, and cleansing took place.
Ranjit and Kawaljeet (2010) mentioned that "as data warehousing is gaining eminence in many organizations, problems arise in populating a warehouse with quality data". These organizations had become aware of the benefits of decision oriented and business intelligence oriented data bases. They further informed, "…data warehouses are one of the foundations of the Decision Support Systems of many IS operations.

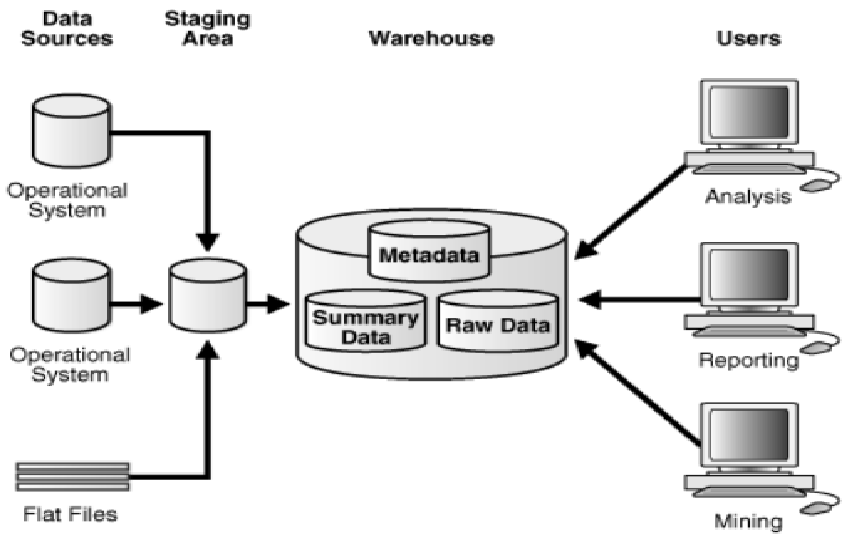


**Figure 2.2 Data Warehousing Structure Ranjit and Kawaljeet (2010).**

Akkaoui, Munoz, and Trujillo (2011) categorically stated that "there are many commercial Extract-Transform-Load (ETL) tools, of which most of them do not offer an integrated platform for modeling processes and extending functionality". This drawback complicated the customization and integration with other applications, and consequently, many companies adopted internal development of their ETL systems. A possible solution to this drawback was creating a software framework for ETL.

Zineb, Esteban, Jose-Norberto, and Juan (2011) enlightened us that ETL processes were the core component of a data warehouse, since they supplied the data warehouse with the necessary integrated and reconciled data from heterogeneous and distributed data sources. Corlatan, Mariaus, Valentina, Octavian (2014) noted that other than improving the system hardware, operating system and SQL server settings, the main factors that affected speed of query execution were as listed below.

i. Missing indexes
ii. Inexact statistics
iii. Badly written queries
iv. Deadlocks

v. T-SQL operations which do not rely on a single set of results
vi. Excessive fragmentation of indexes
vii. Frequent recompilation of queries.

Moreover, they put more stress on the subject of Missing indexes This particular factor has been found to affect the most SQL Server's performance. When missing indexing of a table, the system goes step by step through the entire table in order to find the searched value and return the results to the users. This leads to overloading RAM memory and CPU, thus considerably increasing the time execution of a query. More than that, deadlocks can be created. The SQL Server query optimizer is based on cost, meaning that it decides the best data access mechanism, by type of query, while applying a selectivity identification strategy.

## 2.3 ETL phases of a data warehouse

Data warehousing is made up of the following phases which are discussed separately to indicate the significance and relation to one another. This was the contribution and discussions as reviewed from different authors within DW environment.

### 2.3.1 Extraction Phase

Kimball, Reeves, Ross, and Thornthwaite (2008) informed that the extraction process consisted of two phases, initial extraction, and changed data extraction. In the initial extraction, data from the different operational sources to be loaded into the DW was captured for the first time. This process was done only one time after building the DW to populate it with a huge amount of data already available in the source systems. The next phase involved incremental extraction also referred to as changed data capture (CDC). In this process, it was necessary to extract only the newly added or modified data to the source systems since the last extraction process. This process was periodic according to the recursive cycle of collecting and submitting the data as well as business needs of the organization. There were some methods of capturing the changed or modified data since the last extraction such as use of audit columns, database log, system date, and delta technique, although they were not efficient.

Russom (2012) discussed the fact that most data staging areas were optimized for detailed source data. The advanced forms of analytics was growing aggressively and it required detailed data source because of the discovery nature of data mining, statistical analysis, extreme SQL, and natural language processing. Much of the content of big data by most definitions involved giant volumes of detailed data source.

SAP AG (2002) invested in ETL as part of DW enhancement. They noted that "Extractors are part of the data retrieval mechanisms in the SAP source system…the Persistent Staging Area (PSA), define the load process with an Info Package in the scheduler." To enable data and metadata extraction from non-SAP sources on the application level, SAP NetWeaver BI provided open interfaces known as staging Business Application Programming Interfaces (BAPIs). BAPIs were standardized programming interfaces that offered external access to the business processes and data in a SAP system.  Staging BAPIs allowed users to use SAP Business Objects Data services and certified third-party tools (like Extraction, Transformation, Loading) to integrate data from non-SAP sources. BAPIs were provided for scheduling the data transfer job. These could be used to define application-specific parameters, to use parameters and values for input help options in the InfoPackage and to send the data request to the extraction tool. The data transfer can be triggered from BI or using the extraction tool. During transfer, the data was transformed into the relevant BI format. SAP AG (2002) warned that care was needed to make sure that the transfer structure in BI and the data structure for the extraction tool matched. If possible, transformations runs for technical clean-ups (data conversions for example) at the extraction tool level were carried out. BAPIs also helped the user to monitor the data transfer, allowing them to find the requested status and the log transfer from the extraction tool. The illustration is as follows.

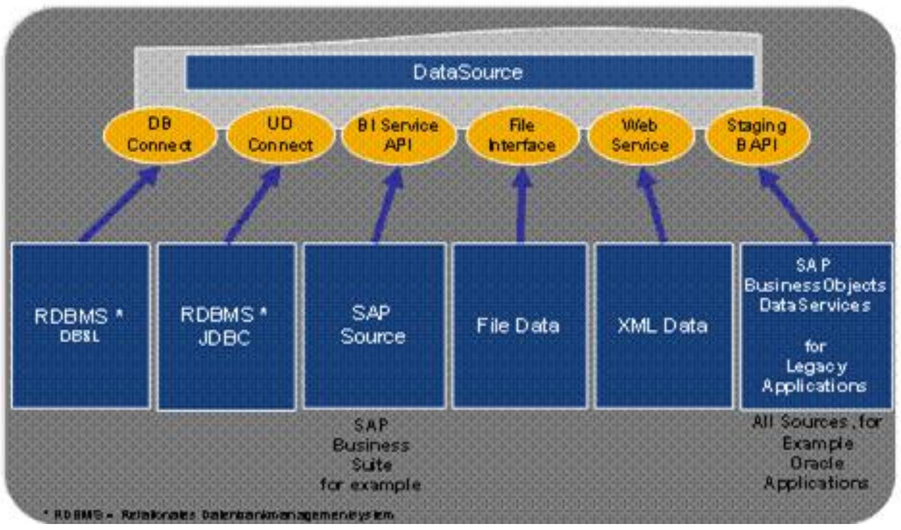

DataSource
DB Connect
UD Connect
BI Service API
File Interface
Web Service
Staging BAPI
RDBMS * DBSL
RDBMS * JDBC
SAP Source
File Data
XML Data
SAP BusinessObjects DataServices for Legacy Applications
SAP Business Suite for example
All Sources, for Example Oracle Applications
* RDBMS = Relationales Datenbankmanagementsystem

**Figure 2.3 SAP NetWeaver server SAP SE (2012)**

Stephen (2013) informed that "The staging tables usually get populated by some outside source, by either pulling or pushing the data from the source systems. This process is usually an insert only process and therefore does not rely on statistics for its successful execution. (pp.1)"

### 2.3.2 Transformation Phase

Once the data was extracted to the staging area, there were numerous potential transformations, such as cleansing the data (correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats), combining data from multiple sources, reduplicating data, and assigning warehouse keys. These transformations were all precursors to loading the data into the data warehouse presentation area. Unfortunately, there was still considerable industry consternation about whether the data that supports or results from this process should be instantiated in physical normalized structures prior to loading into the presentation area for querying and reporting.

Erhard and Hong (2000) elaborated on activities within transformation phase towards clean data. These included data analysis that focused on meta-data and due to fewer integrity rules it couldn't guarantee sufficient data quality of a source. Two approaches had been put across to assist in data analyses which were data profiling and data mining. Data profiling focused on the instance analysis of individual attributes. It derived information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, typical string pattern providing an exact view of various quality aspects of the attribute. Data mining helped discover specific data patterns in large data sets, e.g., relationships holding between several attributes.

Erhard and Hong (2000) explained that putting concern on the performance, then it was important to dwell in defining data transformations which consisted of multiple steps where each step performed schema and instance-related transformations i.e. introducing parallelism. It allowed data transformation and cleaning system to generate transformation code and thus reduce the amount of self-programming. A more general and flexible approach was the use of the standard query language (SQL) to perform the data transformations and utilize the possibility of application-specific language extensions, in particular, user defined functions (UDFs) supported in SQL:99. UDFs could be implemented in SQL or a general purpose programming language with embedded SQL statements. They allowed implementing a wide range of data transformations and supported easy reuse for different transformation and query processing tasks. Furthermore, their execution by the DBMS could reduce data access cost and thus improve performance. The normal cleaning steps and procedures in the order of execution are listed below using UDF implementation with cleaning logic

- Remove misspellings in data within fields
- Conflict resolutions
    - Extract data from free-form attributes(repeated data in different fields)
    - Instance matching and duplicate elimination
    - Sort and reorder values
- Validation and correction for each instance (cyclic process) to remove errors.
- Spell checking – dictionary lookup

- Standardization
    - Help in instance matching and integration
    - Convert to consistent format(uniformity)
- Deal with multi-source issues
    - Restructure schemas
    - Split, merge, fold and unfold of attributes and tables
    - Remove data overlaps and duplicate representation

The hindrance to this approach was that determining matching instances with such an approach was typically a very expensive operation for large data sets. Calculating the similarity value for any two records implied evaluation of the matching rule on the Cartesian product of the inputs.

### 2.3.3 Data staging area

The data staging area was dominated by the simple activities of sorting and sequential processing. In many cases, the data staging area was not based on relational technology but instead it consisted of a system of flat files. After the users' finished data validation on conformance with the defined one-to-one and many-to one business rules, it might be pointless to take the final step of building a full blown third-normal-form physical database. This had disadvantage in that the creation of both normalized structures for staging and dimensional structures for presentation meant that the data was extracted, transformed, and loaded twice i.e. once into the normalized database and then again when the dimensional model was loaded.

Obviously, this two-step process required more time and resources for the development effort, more time for the periodic loading or updating of data, and more capacity to store the multiple copies of the data. At the end, this typically translated into the need for larger development, ongoing support, and hardware platform budgets. With regard to Firestone (1998) the nature of the file determined how it was manipulated and "almost all processing in the data staging process is sorting, followed by a single sequential pass through one or two tables." He also concluded that “the data staging area will archive and store data for a number of purposes. Conformed dimensions are created in the data staging area and replicated out to all the requesting data marts. They must be permanently housed in the data staging areas as flat files ready for export. The data staging area may be the best place to hold data for emergency recovery operations….”

Kimball, Reeves, Ross, and Thornthwaite (2008) provided scenarios in data stage area that depicted the processes carried out. The data flow was set up so that it came out of the source system, moved through the transformation engine, and into a staging database as shown in the figure 2.4.

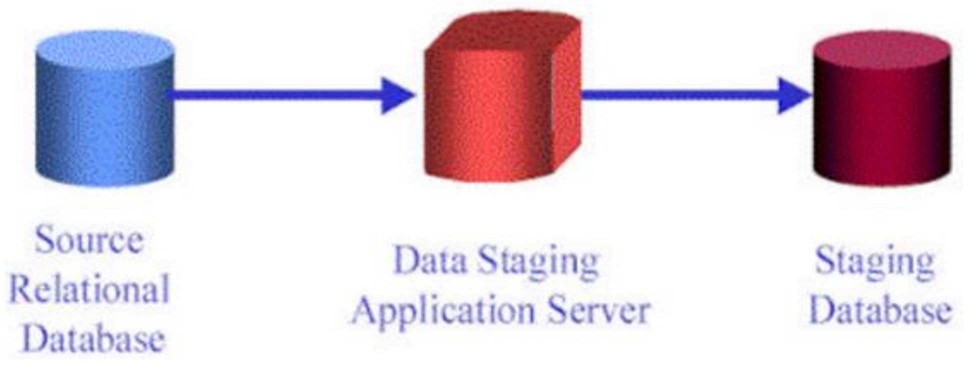


**Figure 2.4 First Data staging Scenario Kimball, Reeves, Ross, and Thornthwaite. (2008).**

In the second scenario, Extraction of the sought after data from mainframe legacy system into a flat file system followed. Then moving the file to a staging server, transformed its contents, and loaded transformed data into the staging database as seen in figure 2.5.

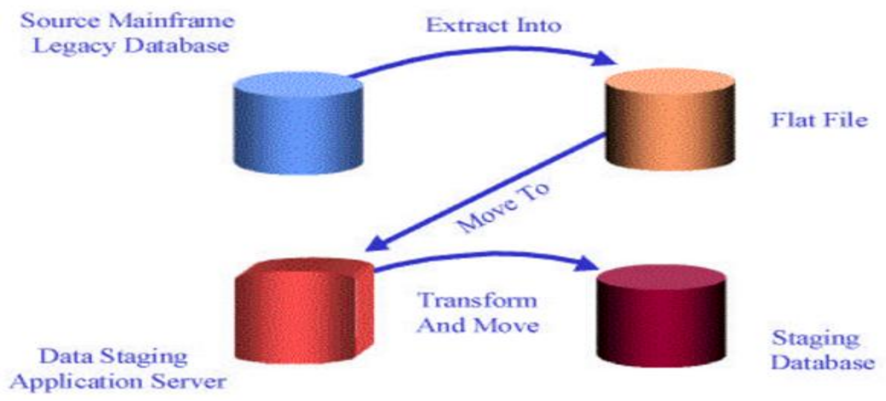

Source Mainframe Legacy Database
Extract Into
Flat File
Move To
Transform And Move
Data Staging Application Server
Staging Database

**Figure 2.5 Second Data Staging Scenario Kimball, Reeves, Ross, and Thornthwaite (2008).**

Abbasi et al. (2010) noted that data staging services moved output data from computer nodes to staging or I/O nodes prior to storage with an aim to reduce I/O overheads on applications' total processing times., "Explicit management of data staging offers reduced perturbation when extracting output data from a petascale machine's compute partition. Experimental evaluations of Data stager on the Cray XT machine at Oak Ridge National Laboratory establish both the necessity of intelligent data staging and the high performance of our approach, using the GTC fusion modeling code and benchmarks running on 1000 plus processors".

### 2.3.4 Loading Phase

Loading in the data warehouse environment usually took the form of presenting the quality-assured dimensional tables to the bulk loading facilities of each data mart. The target data marts must then index the newly arrived data for query performance. When each data mart had been freshly loaded, indexed, supplied with appropriate aggregates, and further quality assured, the user community was notified that the new data had been published.

Aksoy, Franklin, and Zdonik (2001) introduced a workable approach to data staging concerns raised at his time. They based their work on broadcast scheduling and data staging area. According to them the key design considered for development of large scale on-demand broadcast server was the scheduling algorithm selection useful for selecting of items to be broadcast. Their algorithm aimed to choose the beneficial data items only based on unfulfilled previous requests as outlined below

1. Recognize that work had been done in development of online scheduling algorithms for data broadcast.
2. Key concern to get high bandwidth downloads by the introduction of a low overhead algorithm which aimed at fully utilizing the available broadcast bandwidth.
3. Assumption that data items would be available beforehand was not true because of
   a. The sheer size of source data to load and highly-skewed nature to access the data.
   b. The cost to keep all the data in cache was prohibitive even though it was possible to cache the entire set of data in memory.
   c. The high latency led to system performance degradation therefore scheduling efficiency was not able to guarantee the full utilization of the download channel.

Firestone (1998) analyzed file manipulation modes and found out the different natures and ways of handling files. Firestone (1998) thought that the issue was broader and encompassed both the database format and process logic characteristic of: the data staging application, the archival repository, metadata and associated meta-model driving the data staging process.

Kimball and Ross (2002) asked whether the data staging area was relational or sequential for processing flat files. He concluded that most data staging activities used sequential processing. If the incoming data was in flat-file format then it should be handled by data staging processes as flat files before loading it into a relational database. Kimball and Caserta (2004) did not advocate using structured relational data staging area. Kimball and Ross (2002) further warned that “It is acceptable to create a normalized database to support the staging processes; however, this is not the end goal. The normalized structures must be off-limits to user queries because they defeat understandability and performance".

Eckerson, and White (2003) report showed the impact of ETL tools in controlling the source systems and the business intelligence applications. ETL tools needed to be expanded to match the growth in complexity of the BI environments. They stated that focus should be placed on real-time and parallel processing instead of batch-oriented processing that load data in bulk.

Aksoy, Franklin, and Zdonik (2001) came up with three complementary approaches to overcome the above mentioned issues and discussed them as follows.

1. Increasing bandwidth utilization/opportunistic scheduling
    a. Avoid server stalling between broadcast not to miss data items.
    b. Prioritizing technique/algorithm to help decide which the most beneficial data item to load. If most beneficial data item is not available at that time, then the available one should be loaded and broadcasted instead to avoid stalling the server to user responses. This was referred to as opportunistic scheduling.
2. Decreasing the need to fetch an item
    a. Making best use of available server memory space.
    b. Key success in data caching was keeping only those data items that were most likely to be scheduled. Used scheduling algorithm to scale them as either hot (popular) and cold (not so popular) items. This was referred as Hint based cache management.
3. Decreasing the fetch latency
    a. Reduce access latency from obtaining data from slow or remote sources before they were needed. This brought data items were likely to be scheduled in the near future in the cache. This was referred to as pre-fetching.

According to Inmon (2002), “Most of the transformed data are often loaded into DW which are dimensional databases usually structured according to star schemas and accessed by decision support systems...” About 70% of resources to implement a DW were consumed during the ETL project Li (2010). There were many commercial open-source ETL suites available, namely, Pentaho Data Integration, CloverETL Software and Data Integration Talend Open Studio, amongst others. DaSilva, Times, and Kwakye (2002) reiterated that, there was lack of standardization of the set of ETL mechanisms and of their graphical representations, as well as there was no consensus on the scripting languages available on the interfaces of these tools.

Flinn, Sinnamohideen, Tolia, and Satyanarayanan. (2003) stressed on a more architectural and closer view of ETL process. They stated that data staging improved the performance of distributed file systems running on small, storage-limited pervasive computing devices by opportunistically caching data on nearby surrogate machines. Flinn, Sinnamohideen, Tolia, and Satyanarayanan. (2003) statistically proved the performance increase of interactive applications running on the Compaq iP AQ handheld by up to 64%. The performance could be increased further, by redirecting cache misses to staged data on a nearby surrogate while still maintaining the consistency guarantees of the underlying file system. In their paper they reported on the feasibility of data staging on untrusted surrogates with the prototype implementation based on the Coda file system. They also outlined that for bursty, short-term workloads, data staging improved file operations by up to 77% by removing the cumulative delay. They ran tests and confirmed these results by replaying long-term traces of file-system activity and the experiments showed average reductions in file operation latency of 59%. Their focus in their paper was on the file system aspects of data staging and this raised questions on whether the file formats impacted on performance.

Some developed frameworks existed such as PygramETL Thomsen and Pedersen (2009) which was a programmable framework for developing ETL applications. PygramETL was aimed at optimizing the processing of large volumes of data, and was used in the implementation of physical parallelism to take advantage of the current multi-core processors. Thomsen and Pedersen (2011) continued to inform that the demerits of PygramETL lay in the flexibility and generality properties of PygramETL. They had not been evaluated so far, and only the performance had been evaluated. PygramETL was readily available thus a good testing framework that could readily integrate with other object oriented application since it was available as Python code instead of a graphical user interface.

Zheng (2009) recommended filtering relevant and useful data and ensuring the staging resources were available always. They should be dynamic and used whenever available and desired. The aim of filtering data was to reduce output volumes and restricting it in query session. Such filtering leveraged the increasing gap between the ability to speed up computations via faster and more numerous CPUs verses having to store or move data across increasingly deep memory hierarchies. With staging the application performance was decoupled from the run time performance of the back end storage system.

Zheng (2009) said "the massive amounts of data generated by petascale applications can cause performance bottlenecks when running these applications and in addition, they can put extreme pressure on the highly parallel cluster storage systems attached to petascale machines…". They decided to use the staging approach to address the problem by moving output data in a small storage staging area and operate on it for performance optimization. They called it data pre-Analytics-Pre-DatA. They also sort data using rapid data extraction with an aim of achieving real-time operations. Unfortunately, this approach resulted in reduced performance due to reorganization of data at the output.

Another framework based on PygramETL, called ETLMR by Liu et al. (2011), was based on MapReduce and enabled the building of DW stored in cloud computing environments. While PygramETL and ETLMR were directed to the optimization of ETL processes for building DW, FramETL was a more general and flexible framework for enabling the creation of data repositories and customized ETL transformations. The operation of an ETL System as described by FramETL had two phases; Metadata Definition and Operation. DaSilva, Times, and Kwakye (2012) informed that in the metadata definition phase, the specification of the system inventory and modeling of ETL processes occurred, while in the operation phase, the loading of metadata, the access to data sources, and the execution of ETL processes were performed.

DaSilva, Times, and Kwakye (2012) discussed about some approaches and tools that had been introduced to handle the challenges that data staging experienced on efficiency. These were Data Transfer Request, which was a tool that monitored request and response of data from sources as per changes made. Another approach was investing in complex and powerful hardware and software which was very costly. Hardware would include the machines with high processing power, high RAM memory ranging to several GBs, and many processors scheduled to work concurrently under configured settings. The software would be management systems as well as specialized databases such as in-memory HANA and oracle that relied on high memory caching. This might not be viable for SMEs due to affordability and risk ratio. We checked on the demerits of these approaches before coming up with a better solution.

El-Wessimy, Mokhtar, and Hegazy (2013) decided to try out three techniques to help in speed improvement. “In this context, they explored three scheduling techniques (First-In-First-Out (FIFO), Minimum Cost, and Round Robin (RR) based on time and records) for scheduling the ETL process. They experimentally showed the behavior of these techniques in terms of execution time with respect to the sales data and discussed the impact of their implementation.

When it came to moving data to DW Stephen (2013) informed that “The biggest question for the staging area is – how do we keep the statistics up-to-date such that the statistics for a particular daily load are always available and reasonably accurate. This is actually more difficult than it sounds. If the partitions would only be analyzed in the first quarter of the month each night, going to every other night and eventually each week because of the 10% stale setting. This obviously leaves us with a problem…. In order to have the statistics available for the latest day which is loaded, the statistics would have to be gathered after the staging tables have been loaded but before the ETL process starts".

#### 2.3.4.1 Metadata

Firestone (1998) recognized that the data staging process was driven in an essential way by metadata, including business rules. Metadata was used along with administrative tools to guide data extractions, transformations, archiving, and loading to target data mart and data warehouse schemas. SAP AG (2002) mentioned that Metadata from non-SAP systems could be defined or updated either manually or using BAPI functionality in BI. If you accessed the BAPI interface, the non-SAP software's extraction tool could automatically read the BI metadata from the source system or define the metadata in the extraction tool. The tool could then transfer the metadata to BI using the BAPI interface. To change metadata in BI manually, it was required to enter the needed data in the transfer structure maintenance transaction. In addition to transferring and updating metadata from the extraction tool to BI, the BAPIs could also be used to transfer BI metadata to the extraction tool.

During transformation, there were several problems Vassiliadis (2009) which included: schema-level issues that affected naming and structural conflicts; Record-level issues such as duplication and contradicting records; Value level issues that affected representation and interpretation of values formats. He also noticed the dilemma in the choice between bulk loading data through a DBMS-specific utility or inserting data as a sequence of rows. Clear performance reasons strongly suggested the former solution, due to the overheads of the parsing of the insert statements, and the maintenance of logs. Simple SQL commands were not sufficient since the 'open-loop-fetch' technique, where records were inserted one by one, was extremely slow for the vast volume of data to be loaded in the warehouse.

### 2.3.5 Query Processing and optimizations

The speed of query execution determines the overall performance of the ETL phases from the beginning to the end. Connolly and Begg (2005) noted that one of the major criticisms of current database management systems (DBMS) often cited was inadequate performance of queries. Since then, a significant amount of research has been devoted to developing highly efficient algorithms for processing queries. There are many ways in which a complex query can be performed, and one of the aims of query processing is to determine which one is the most cost effective.

Query processing involves four phases namely;

i. Query decomposition: - High-level language (SQL) is converted to low-level declarative language (relational algebra) to parse, validate optimize and execute the query.
ii. Query Optimization: - Is either based on heuristic rules/approach or those that follow cost estimation approach in determining the most optimal strategy for minimal resource usage.
iii. Query code generation: - the optimized relational algebraic expression is converted to a language the computer understands.
iv. Query Execution: - Using the optimal strategy that is chosen, the query is fired to yield the results in the minimal time and cost.

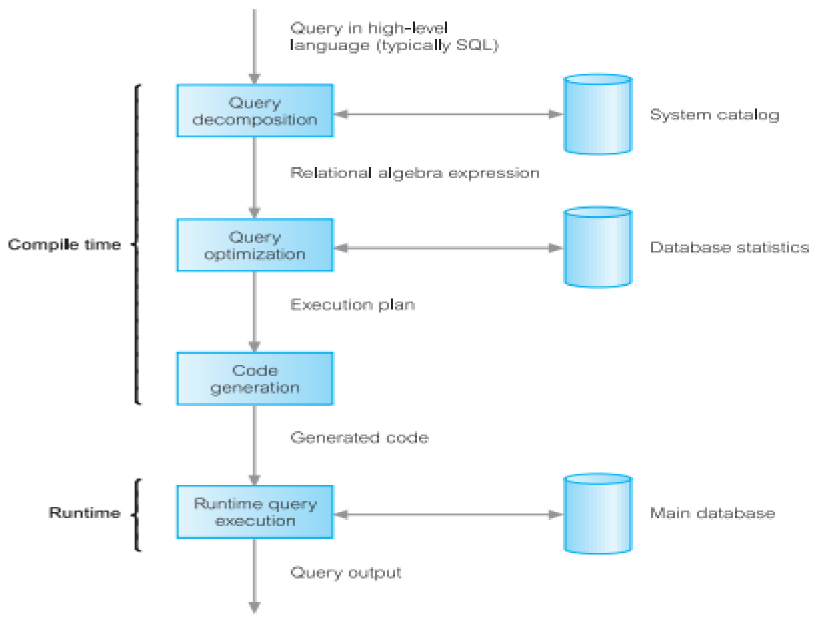


**Figure 2.6 Phases of Query Processing, Connolly and Begg (2005)**

The efficiency of repetitive cycle of query processing depends on whether the optimal strategy is saved in memory or disk for future accesses instead of compiling the same strategy over and over again.

Bamnote and Agrawal (2013) informed that the cost of query evaluation could be measured in terms of a number of different resources, including disk accesses, CPU time to execute a query, and, in a distributed or parallel database system, the cost of communication. The response time for a query-evaluation plan would account for all these costs, and could be used as a good measure of the cost of the plan. In relation to large database systems, they observed that, disk accesses were usually the most important cost, since disk accesses are slow compared to in-memory operations. The query optimization engine generated a set of candidate evaluation plans where in heuristic theory, some produced a faster, more efficient execution.

Nanda (2015) noted that the objective of query performance enhancement was to minimize the response time for each query and to maximize the throughput of the database server. He also observed that Query Processing and Optimization techniques as one of the factors that hinder performance of database management systems. He also observed that in a distributed environment like cloud, data is distributed to a number of sites, stored in its entirety on all sites or spilt on many sites. Here the query is processed and optimized in a different way.

#### 2.3.5.1 Query Hints

Connolly and Begg (2005) mentioned that the cost-based optimizer also takes into consideration hints that the user may provide. A hint specifies a comment specially formatted within an SQL statement to change the behavior of execution of that query. There are a number of hints that can be used to force the optimizer to make different decisions, such as forcing the use of the rule-based optimizer; n a particular access path; and Indexes Information about indexes.

#### 2.3.5.2 Stored execution plans

Connolly and Begg (2005) further informed that there may be times when an optimal plan has been found and it may be unnecessary or unwanted for the optimizer to generate a new execution plan whenever the SQL statement is submitted again. In this case, it is possible to create a stored outline using the CREATE OUTLINE statement, which will store the attributes used by the optimizer to create the execution plan. Thereafter, the optimizer uses the stored attributes to create the execution plan rather than generate a new plan.

## 2.4 Summary of Literature

Other than appreciating the current developments in this field of research, it was important to be aware of the future developments in technology with regard to DW. This was to avoid major changes to the improved frameworks being scraped off due to existence of a new product. Consistency was to be complete and accurate maintaining scalability. Most of the authors had based their research on quality of data rather than performance enhancement. The desire to make strong decisions led to loading almost all the data from the sources. The organizations had been able to cope with the current performance in tools because they were custom made to fit to their operational environment. The problem of handling data with speed as the unit of measure escalated at high rate with the growth of organizational data. Investing in the study was a great move to advance technology further as well as prepare for future compatibility with newer systems that met the users' needs in time.

## 2.5 Gaps in knowledge

Considering the suggested improvements as discussed from great works of other authors, it was important to critique the literature by finding gaps in existing systems. The following were identified as the missing links on the body of knowledge as regards the investigation on the works of previous authors in the field of data warehousing.

- Handling manipulation and transfer of bulk data was time consuming activity requiring a lot of resources. The dependency level for batch processing had diminished literally due to growth of technology.
- There lacked a stable and standard staging framework to use for any warehouse setting.
- The performance measure of the ETL process had been ignored because focus is invested on quality.
- There was no proper definition of scheduling plans for jobs in ETL processes.

## 2.6 Conceptual Framework

The research gaps identified and discussed in the previous section, helped to draw probable solutions which would be implemented to tackle the noted issues and finally improve the general performance in data warehouse environment. This study proposed to create and implement deterministic prioritization algorithm within the data staging area which helped to show the relationship between the ETL processes and improved performance of data access and retrievals. This approach was active immediately upon deployment working on the data collected from the source system.

Less activity was experienced in the extraction phase but the actual data work area was within the staging area. All the data would be subjected to the new deterministic prioritization algorithm that planned to prioritize and selectively coordinated the appropriate data that needed to be moved to data marts. The algorithm implementation was targeting use in two areas. The first would be an implementation in an open source environment where the algorithm would be built to target all general purpose staging frameworks available in the market. This would mean that the deterministic prioritization algorithm would be freely available for use by these organizations without subscriptions. The second area of implementation would be within specific organizations that have customized staging frameworks. The idea to test in both areas is a measure on the flexibility of the solution from different scenarios of the market since not all users of data staging frameworks would be willing to introduce the change directly into their systems. It was observed that previous activities performed within staging area were very important and none would be eliminated hence the approach aimed to improve on the order of execution to avoid redundancy and repetition of tasks. This concept is illustrated in figure 2.6 that shows the relationship among the ETL processes of a data warehouse.

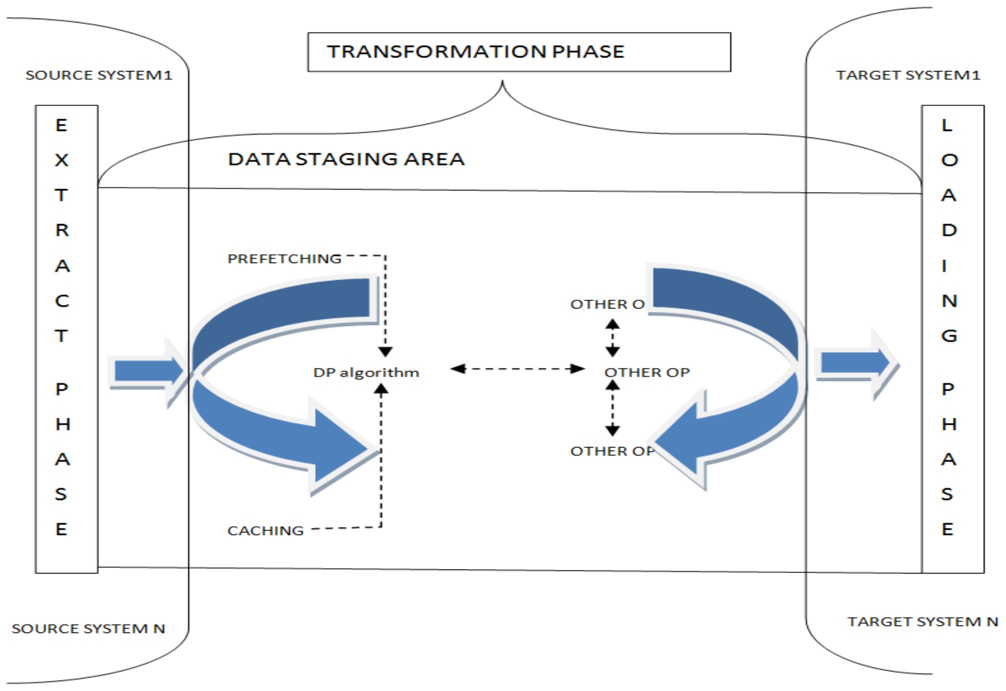


**Figure 2.7 Proposed Conceptual frameworks**

KEY:

DP algorithm – Deterministic prioritization algorithm

OTHER OP – Other transformation operations such as cleaning and verification

## 2.7 Broad to specific literature

Normally tables in any relational database were relational based on the logical bindings that resulted from querying data from them. Tables had relationship with at least one or more tables which was random in nature. The data access and retrieval from non-linked entities was batch-oriented where successive processing of requests in the queues was sequential. They were staged in the hope that in the next phase of loading there would be a process that would identify the relationship with other tables and during such load a relationship would be established. It depended on the operating system to provide free resources to handle the activities from source to destination. Data staging area greatly differed in this aspect because it encapsulated the three ETL processes as a functional area of a data warehouse and maintained the relationship interfaces that were held prior to the movement of data from the sources to destinations.

## 2.8 Included Improvements

The included changes would provide a stable and scalable algorithm that would be easily deployed and utilized in existing and newer data staging to support concurrent data processing. This would magnify the core benefits of owning and maintaining intelligent data warehouses that would be supportive to the top-level of decision making process.

According to Cecilia and Mihai (2011), the use of indexes on database queries improved the performance of the whole system. Clustered indexes performed better than nonclustered indexes when the expected returned records were many and were to be set for the most unique column of a table. This proposition supported the use of indexes in the staging area as illustrated in the positive test results shown above.

Grant (2012).explained about execution plan management which was a task done by query optimizer. The database relational engine performed logical reads within the cache memory while the storage engine performed physical reads directly from disk. Improvements were highly realized mostly for data manipulation language statements since the engine needed to parse the query for correctness. The SQL server generated statistics against the indexes and sent them to the optimizer to determine the execution plan.

Per-Åke L. et al. (2013) highlighted the improvements in the SQL Server 2012 released. It had enhancement in batch processing through introduction of column store indexes. The main idea was to have the table’s primary key being treated as the clustered index for any storage structure (heap or B-Tree). They categorically informed us that some customers had reported major performance improvements (10X to 200X) as a result of this change.

El-Wessimy, Mokhtar, and Hegazy (2013) similarly did an enhancement in the data warehouse staging area by using different techniques (FIFO, MC,RR time and record rotation) targeting the loading phase. The tests ran captured the time taken to transfer data in each stage of the ETL process and suggested the most suitable technique. They did a comparison amongst all techniques and noted that FIFO performed better for less data set while Record Limit Based Round Robin was best for large data sets. Their research related to this new enhancement in data staging on the basis of reducing the overall time taken to deliver data from source to destination.

Stephen (2013) elaborated an approach in Oracle environment that “Most ETL applications used a staging area to stage source system data before loading it into the warehouse or marts. When implemented within an oracle environment a partitioning strategy was usually employed such that data that was not required any longer could be removed from the tables with minimum amount of effort.”

Costel, Marius, Valentina, and Octavian (2014) did a research on query execution and optimization in the MSSQL Server and put across the missing of indexes as a contributor to low performance of query execution. They informed that when a table missed indexes, the search engine had to parse through the entire table step by step to find the searched value. The resources spent on this process ended to be enormous and considerably increased time to execute the queries.

The proposed algorithm would provide cost effective mechanisms to decide which data need to be collected beforehand and also how to change order of transferring the data to the destination systems.

# CHAPTER THREE

# METHODOLOGY

## 3.1 Introduction

There was great need to measure the impact of changes introduced in order to demonstrate and justify the purpose of this study. Moreover, the nature of the problem that the research explored called for a convenient approach and in particular Experimental methodology was selected. The comparison and contrasts expected from the study would make use of available staging tools and alternatives e.g. Pygram ETL and Microsoft data profiler tool. The resultant summary of findings, observations, and recommendation would be adequately documented in the research report.

The problem domain fell in the production/operational domain where vast transactions are normally carried out generating the workable data. Each enterprise has its own business processes but the concern of the study focused on the ETL phase that involved collection of data for the DW and how to make improvements. For quality decisions, all the relevant and related data had to be merged and eventually availed to decision makers. The activity posed to be difficult given the magnitude of data to be searched and the lack of algorithms to prioritize the collected data. To maintain scalability with previous ETL phases, the operational space was not expanded and this resulted to better, quality and smart ideas for generating queries of handling DW data.

## 3.2 Dependent and independent variables

The study investigated the different variables that associated to the research topic and how they impacted on system performance within any data staging area. The essence of discussing these independent and dependent variables was to assist in the development of a research design that would appropriately solve the problems reported by data warehouse users. The different variables that were considered for this study are illustrated in figure 3.1 and their description follows suit.

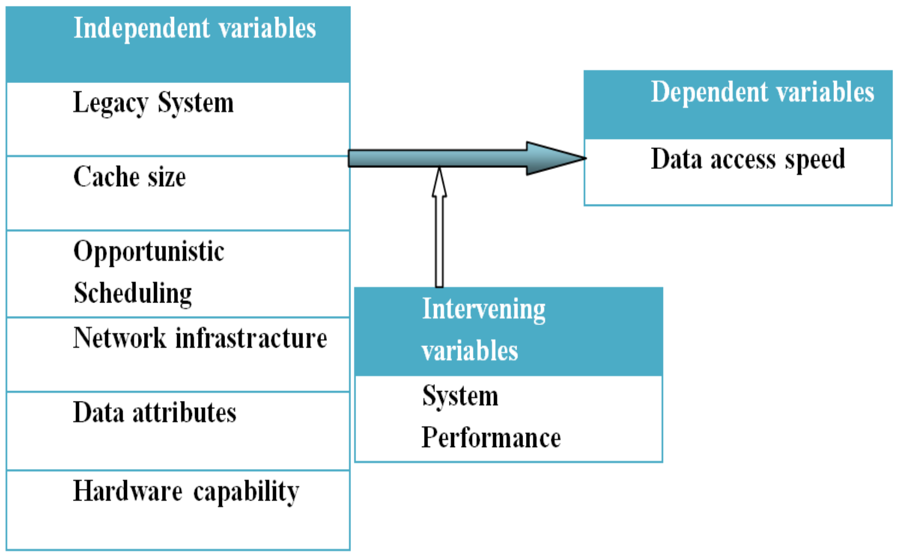


**Figure 3.1 Relationship between dependent and independent variables**

**a. Legacy systems and System Speed Performance**

Legacy systems used stacking where a job was pushed into the pool of tasks and popped out of the stack when its time of execution reached according to the queue. The order of execution was not highly dependent on pre-arranged structure. This procedural mode of execution degraded performance. The speed of job execution fully depended on the freeness of the particular system processor. The legacy systems were also limited by the number of these processors and job sharing resulted in overworked system hence affecting the overall system performance.

**b. Cache and System Speed Performance**

The cache offered pre-fetching capability where data storage was done temporarily for the most used data. The scheduled procedure looked firstly in the cache memory before checking secondary storage locations hence minimizing on the search time. The limitation was tied on the cache size. The determinant of the number of jobs to keep within the cache was the priority of the job scheduler and number of hits or requests. Therefore, a job with fewer requests was dropped from the cache because its importance in the delivery of particular data (priority) was not determinable beforehand.

### c. Opportunistic Scheduling and System Speed Performance

To achieve concurrent operation there needed to be prioritization algorithms to decide the execution order of queries fetching data from the source systems to the data staging area. With Opportunistic scheduling there was high probability of improving speed of data retrieval and access. Similarly to the storage of jobs in cache, the probability of selecting the important data was not clearly defined. The overall system performance was affected where the resultant deliverable couldn't meet the expectations of top management.

### d. Network infrastructure and System Speed Performance

Remote connection to source systems affected the speed of retrieval and query execution was delayed by the time-lapse for distributed systems. This impact on the nature of ordering results from query execution and thus optimization should be introduced to work with stored procedures and cache facility. Even after the integration of new technologies the underlying factor is the speed of connection. Considerable system performance in delivering adequate and real time decisions was observed for dedicated networked systems.

### e. Data Attributes and System Speed Performance

Data characteristics were defined by type and formats since it came from disparate systems. Data was not moved to the target systems before some modifications to match destination requirements. Poor data manipulation functions resulted in longer time processing the data slowing down the systems. The functions for manipulating flat files were different from the ones for relational tables and databases. The greatest barrier noted was the existence of different system formats as per the language of localization. When the data analyzer converted from one format using EDI tools, this was not 100 % complete and manual corrections were needed impacting on overall performance.

**f. Hardware capability and System Speed Performance**

Newer systems such as SAP HANA databases had high processing capability which was meaningless if there was no proper scheduling of resources. This could result to lots of losses of resources not being manned properly. Hardware being static and data being dynamic at some point provided discrepancies when scheduling job execution since the available hardware might not be sufficient to handle the data affecting performance. Although, scheduling played a vital role in the performance implications of systems, the balance between the operating software affected performance. It represented the effect being sought and it was measurable to make comparison.

## 3.3 Rationale of the research design

The research design was selected based on the need to demonstrate improvement and the availability of measurement tools to many users of a data warehouse. It was also desirable because of its simplicity in use and richness of analysis to the problem. The changes to be made in data staging were very crucial and their impact was checked to see the implication to be either positive or negative. The research methodology fit well to counter the problems outlined by previous authors as noted previously; because it conformed to any new user requirements introduced within the ongoing project. This flexibility was advantageous in observing any interruptions to the normal operation of a DW. Corrections that needed to be effected when the model was in development stage were tailored to match the specific problem defined with ease due to its less strict nature. The success of meeting the user requirements was to ensure functional requirements were delivered while abstracting the users from the complexity of algorithms used to achieve this.

This resulted in creation of a new algorithm was to be introduced from data extraction to help determine what data and how much to capture initially from the heterogeneous sources. The criterion was never based on the type of source system but on formulating the source to a format where the data was readable to get only the most vital data. Meta data was also collected but its use in the analysis of the problem was limited to a controlled level. This was so because the nature of study was not considering the deeper details of data and characteristics.

## 3.4 Feasibility study

It was important to accustom with the study area and understand better the problem domain from the actual site. Data staging is an adaptive topic meaning there is so much dynamism taking place depending on the source system or the configurations for the target systems. This hence dictated a static workflow that was recursive in every complete transaction and phase according to the setup of the scenarios and test system.

## 3.5 Research Site

The study targeted organizations and institutions that performed many transactions resulting in handling large data ranging from a few records to several million rows in their relations. This data would be available in mid-sized organizations and businesses. It was to the researcher's convenience and constrained resources to collect the research data through online means. This was highly contributed by the lack of local organizations implementing data warehouse technologies due to lack of resources. Some organizations had packaged downloadable databases deployed on their websites for researchers from online community. The available databases were availed in different data formats resulting from the variant source systems and also the need to match the user's specific database server type. The databases were varied in sizes depending on the number of relational tables and records they contained. It would be envisage running active tests in those manufacturing companies that had already implemented SAP business solution products and any other appropriate ERP systems like Oracle because their data is already organized and managed by a management system. It is the flow of data obtained from these large enterprises, that the explanation for data staging enhancement would be depicted clearly.

## 3.6 Population Sampling Procedure

The sample size was selected from those organizations and enterprises identified to provide online materials for research. The aim was to capture databases with large volumes of data ranging from several hundreds of rows to thousands of records. This was vital for this study because it was a requirement to experimentally show the change introduced and technical features improved by the new algorithm implementation with regards to data access speeds. There were multiple organizations that offered research data for download as availed from their websites. The stratified random probability sampling method was used on the identified research sites to aid in this selection. The sample was a representation of the whole population which couldn’t be investigated due to the time constraints and authorization access. The sampling method had been chosen purposely because of its high degree of representing the whole population and its abilities to overcome biasness in the study. The random selection was performed on several organizations whose databases were freely downloadable from their websites. Among the organizations investigated were;

- PPSOne demo databases in relational format from CP CIM-POOL AG
- SAP SBO-Common databases
- AdventureWorksDW2008R2 from Microsoft
- ACM SIGMOD Records in XML format
- TPC-H Relational Database Benchmark in XML format
- Annual Capital Expenditures Survey from Data.Gov in csv format

The following factors were considered while weighing on which database was most useful for this research study to base on.

- Accessibility to the database – requirements of pass codes and subscriptions.
- The data types for the databases – flat files, csv, xml, relational tables
- Database sizes in Giga-bytes
- Number of records within the tables and their normal form
- Real data to match production environment
- Currency of data
- Availability of tools to analyze and work on the data

Based on the above characteristics on the data smaller groups were identified that formed the strata. Each stratum came from individual company names from which the databases would be sought. All these company names were copied in a single column of Excel in any order. In a second column, the "Randomize" Excel function was to be applied and generate values after the "rand ()" function was fired. Additional details about the companies were optionally included although they would not be used anywhere in this sampling process but provided more details on the data sets. After executing the randomization function and sorting by this second column, the values were ranked in an orderly manner either in ascending or descending order. The last row number of the selected strata provided the sample size for the study to represent the whole population. In this case the stratum selected was from Microsoft's AdventureWorksDW2008R2. The database conformed to match with the available SQL server version that would enable easy analysis and data manipulations.

## 3.7 Sampling

### 3.7.1 Sample size calculation

To calculate the sample size for the study, the Slovin's formula invented in 1960's was used as shown below.

$$n = \frac{N}{1 + Ne^2}$$

$Equation\ 3.1$

Where **n** = Number of samples;

**N** = Total population;

**e** = Error tolerance; and (**0.01 <= e >= 0.05)**, the lower the error tolerance value, the higher the accuracy of result.

**For Example Using N = 5;**

e = 0.02;

Then

$$n = \frac{5}{1 + 5 * 0.02^2}$$

$$n \cong 4.99$$

## 3.8 Rationale for sample and size selected

Slovin's formula was used to figure out what the sample size to use since the whole population could not be investigated based on time constraints. Moreover, it was not it was not possible to know every characteristic of the population early enough. It was also a simple and direct formula with minimal variables hence less complicated. The methodology called for recursive tests and measurements on the new algorithm and hence required several test runs of the experiment so as to compare the values from each test scenario. The average of the results would bring clarity to the actual performance of the algorithm based on the test environment and available resources. This improved on effectiveness, quality and also was within time boundaries of the project deliverables given the vast amount of data dealt with.

### Sampling techniques and procedures

The stratified random sampling method fit well for this study analysis due to the nature of data distribution on the customer systems. This procedure managed to subdivide the customers' systems selected from the population into workable units or single test scenarios. The customer's databases that had large data were opted for since they could easily show by measurements the performance change. The sampling method overcame a lot of biasness because the customer selections were not based on the type of business but on data size or number of transactions. All the data attributes and properties were considered when comparing amongst customer data. When several customers matched the filtering criteria, a tie was easily separated by randomly selecting the final subject's proportionally from the different strata. The merit for working in groups was that it was quite easy and fast to note the change when comparison was carried out between systems vis-à-vis the data size. The study highlighted a specific subgroup within the population performing better than the others where the changes had not been implemented. This method had higher statistical precision and needed a small sample size so as to save on time, effort and money.

## 3.9 Data collection

### 3.9.1 Data collection tools and instruments

Research data was collected from the available data availed online for research. Several databases were downloaded from the group that matched the stratum specification after random sampling procedure. Microsoft SQL website provided data sets for relational databases for example AdventureWorksDW2008R2, matched the SQL server version running in the researcher's notebook. The availability of analysis tools in close proximity was a core deciding factor to attach the database for this research. The databases were extracted and attached to the SQL server. To actually match the exact version of customer database, some upgrades were carried out so that the data could be readily retrieved from the tables without constraints. The SQL server had tools for analysis and configuration which was important in the experimentation stage where simulation of data warehousing was depicted clearly. The SQL Server Integration Services software tools and package were added to the installed version of SQL Server and really assisted in the analysis of the problem and data collected.

### 3.9.2 Pretesting of the study instruments

The pretesting was done in a different site rather than the randomly selected one for the study. The databases were also attached in a remote system and controlled locally in a networked environment. This was to establish its validity and reliability before the actual study took place. The system setup for both the old and new were pre-tested in one machine then deployed to the network. This will cover the majority part of interviewees as per the sample inclusion criteria. Pre-analysis was done to check on the appropriateness of the data collection tools, and the identified gaps and overlaps were rectified before the actual study took place.

### 3.9.3 Data collection procedures

The study was conducted after successful approval by the Board of Postgraduate Studies and the appropriate administrative officers from the organizations. Since the study was neither inventing the wheel nor creating a pioneer project, secondary data comprised of research materials in both soft copy (electronic documents published online) and hardcopy (books, journals, and articles). Primary data was captured by running tests to determine the mode of operation using the actual systems in selected companies. This data was collected by downloading sample databases of the organizations for further analysis. More data was also collected through online materials and documentation, where experts in ETL tools in an organization guided the researchers vigorously through videos. These systems were used widely to manage business processes and customer relations while streamlining decision making by management in an organization.

The captured workflows differed among organizations because of the business logic outlined in the ETL process of the specific organization hence a clear distinction was noted. The research process was a continuous activity that was projected to take eight months from the time of research proposal acceptance to delivery date. Data collection and gathering was done from the third to the sixth month while analysis followed suit.

### 3.9.4 Data Analysis

The data needed to be analyzed to bring out the problem clearly as was the case at the site. The researcher had to capture all affected variables since they would play an important role in looking for a valuable solution. It was a mind opening venture because the different aspects of business processes were studied as per the set logic.

### 3.9.5 Statistical tools used

There existed tools to analyze the collected data to make work easier for the study. The most applicable tools used were Microsoft Excel, Visio 2007, Gantt chart creator, SQL Server, SSIS, and linear scheduling method tool. They fit to the study because of the need to compare the results from different sources as well as among companies. They were also readily available to many users due to their simplicity in use even though some were costly to acquire.

## 3.10 Data presentation techniques

After completion of data analysis, the resultant data were presented to the project funders through a committee to show and convince them that the change in performance was realizable. The summarized presentation through power point slides explained how the research design fit to the problem domain. Upon completion of the analysis of study, the deliverable product was subjected to full testing for the proposed changes or enhancements and finally deploying it for use by the organizations.

# CHAPTER FOUR

# RESEARCH FINDINGS AND DISCUSSION

## 4.1 Introduction

The study proposed an algorithm that would work well on limited or more cache memory since freeness of the memory was operating system dependent. The running space was not an issue since it only acted as holding place for the recursive processes that were being controlled by the operating system. The creation of test simulations and test scenarios was designed to match the actual production environment where the decision support of a data warehouse provided quality answers to requests coming from the business users who were abstracted from the general operations of data warehouse environment. The users only requested for information in form of SQL queries and expected timely and current results from the intelligent data warehouses.

## 4.2 Development of Deterministic Prioritization Algorithm

The new algorithm that was created during this study involved the following major steps that were performed in a recursive manner.

### 4.2.1 Deterministic Prioritization Algorithm Definition

Problem: How to determine which data to collect and improve the speed of loading the data to the intended destination location?

Step 1: Extraction of data from source tables through table and columns scan.

Step2: Building the SQL query clauses with focus on SELECT type.

Step 3: Optimization of the queries selection through clustered indexing.

Step 4: Execution of the queries noting the change in execution time or cost.

### 4.2.2 Deterministic Prioritization Algorithm Design

According to the algorithm procedure, the kind of operations performed at schema level was similar for all of the relations such as connection management to the SQL server. The SQL queries were generated with a purpose of fetching content from the source relational table and then availing the collected data to transformation operations before loading to the data warehouse. The query analysis focused on ways of reducing the cost of query execution in-terms of actual operation time and minimal resource usage. With this in mind a sample query was considered for one of the users' request to load customer data to the data warehouse. This was important in analyzing the estimated I/O cost when a query was executed.

```
SELECT STCustomerID, CustomerKey, GeographyKey,CustomerAlternateKey,
    FirstName, LastName, NameStyle, BirthDate, MaritalStatus,Gender,
    EmailAddress,YearlyIncome, TotalChildren, NumberChildrenAtHome,
    EnglishEducation, SpanishEducation, FrenchEducation, EnglishOccupation,
    SpanishOccupation,FrenchOccupation, HouseOwnerFlag,NumberCarsOwned,
    AddressLine1, Phone, DateFirstPurchase, CommuteDistance, MiddleName,
    STStartDate

FROM  newStaging_Customer sc

WHERE sc.STStartDate > '2014-01-01 00:00:00.000' AND sc.  STCustomerID IS
    NOT NULL

ORDER BY sc.STCustomerID
```

The relevance of the kind of cost selection approach depended on the database statistics which were saved in the system catalog at the creation of the data staging tables. During query decomposition the query was converted to relational algebraic expression as described in the following discussion. In this scenario a single relation, **R** defined by attributes or columns $\mathbf{A = \{A_1, A_2 \ldots\ldots\ldots, A_n\}}$, and which were defined over **p** and **q** to denote Predicates. The above query provides a conjunctive selection with operations for users' requested data that formed an individual query selection applicable to every source relation used in the study experiment.

$$\boldsymbol{\sigma}_{p\wedge q}(R) = \boldsymbol{\sigma}_{p}\ (\boldsymbol{\sigma}_{p}(R))$$

Expounded in the example as

$$\boldsymbol{\sigma}_{(\text{STStartDate} > \text{'2014-01-01 00:00:00.000'})\ \wedge\ (\text{STCustomerID IS NOT NULL})}\ (\text{newStaging\_Customer})$$

$$= \boldsymbol{\sigma}_{(\text{STStartDate} > \text{'2014-01-01 00:00:00.000'})}\ (\boldsymbol{\sigma}_{(\text{STCustomerID IS NOT NULL})}\ (\text{newStaging\_Customer}))$$

Where **σ** stands for **SELECT** and ^ stands for **AND**

Once the query was expressed in relational algebraic expression, the processor needed to find the most optimal strategy to execute the SQL query. The deterministic prioritization algorithm introduced the use of clustered indexes to manipulate the default the procedure taken by query processor to process query selections. The selection queries utilized in the algorithm followed the first rule of heuristic processing strategies. The rule states that any selection operation when done early, it results in reduced relation cardinality and also reduces the subsequent processing of that relation. Since the selection was performed on a single relation, the predicate **p** involved only the attributes in the selection clause.

$$\boldsymbol{\sigma p(R)} \text{ where } p \in \{A_{1,} A_{2\ldots\ldots\ldots,} A_n\}$$

Expounded in the example as

$$\boldsymbol{\sigma}_{(\text{STStartDate} > \text{'2014-01-01 00:00:00.000'})}\ (\text{newStaging\_Customer}) \text{ where } _{\text{STStartDate}} \in \{$$
$$\text{STCustomerID, CustomerKey}_{,\ldots\ldots\ldots,}\ \text{STStartDate}\ \}$$

The sequence for adding the clustered index to each table based on the newly created (derived and distinct) column was discussed to bring out the cost estimation for the converted relational algebraic expressions of the queries used. This aimed to depict how this procedure ended up to be the optimal strategy chosen for all proceeding data staging queries. The SQL server uses formulas to estimate the cost of a number of strategies and chooses the most efficient of them all based on reduced cost. In query processing the query cost is highly concerned with disc block accesses or hits rather than memory accesses which are managed by cache memory.

The success of estimating the size and cost of resulting relational algebra operations depended on the amount and currency of the statistical information that the SQL Server had generated at the time of experiment. The clustered index was based on and not limited to a single external column, $\mathbf{A_n}$ that was added in every table scan during the building of the selection query. This had two advantages where the index needed not to be recompiled every time it was run and any change in the source relation's attributes at runtime would not interfere with the clustered index generated or cardinality of the query.

**For each base relation R | R ∈ Schema**

- nTuples (**R**) represent number of rows in relation **R** forming its cardinality
- bFactor (**R**) represents the block factor of relation **R** meaning the number of rows that form one block
- nBlocks (**R**) represents the total number of blocks generated on relation **R**

This results to an equation that calculates the number of disk block accesses, nBlocks in relation **R** which in essence represents the division of the select operation on small operation chunks.

$$\textbf{nBlocks (R) = nTuples (R) / bFactor (R)}$$

The greatest advantage of query formation in terms of these blocks comes out clearly when a scheduler plan is employed mainly during runtime to determine the sequence of processing. For instance if the opportunistic scheduling algorithm is implemented, the number of blocks, nBlocks (**R**) represent quantum for the round robin algorithm. This scheduling algorithm is best suited for large data sets and it is very efficient in queue and parallel transaction processing. In this case each block is given equal share of time or slices as per the number of blocks until the entire query is completely executed.

The procedure to build the select clause of the Selection query continues deeper into the attributes of the relation R. The cycle is nested such that for every relation R, the all the attributes are collected. This is represented as follows.

**For each attribute A of base relation R| A ∈ R and A = {$A_1, A_2$.........., $A_n$}**

- $nDistinct_A$ (R) represents number of distinct values for column **A** in relation R
- $nNull_A$ (R) represents number of null values for column **A** in relation R
- $SC_A$(R) represents the cardinality of selection of column **A** in relation R for those rows that match the selection criterion.

This analysis on relation attributes helped to determine the confidence levels of the columns and their capability of constituting a candidate key. Their strengths would guarantee prioritization of these attributes in the creation of the clustered indexes.

**$SC_A$(R) = nTuples(R) / ($nDistinct_A$(R)) * nBlocks** for A in {A1, A2………., An} negating obsolete attributes nTuples(R)/ ($nNull_A$ (R))*nBlocks for A in {A1, A2………., An}

This results with only the required and determined attributes that satisfy the requests of the user to form a selection clause and hence the process ends up being deterministic. The prioritization of individual columns that were of high confidence levels was done by addition of clustered indexes based on those columns. This meant that in every relation an index was generated as described below.

**For each index I of base attribute A | I ∈ A and A = {$A_1, A_2$.........., $A_n$}**

- $nLevels_A$(I) represents the number of levels in **I** meaning number of attributes that satisfy to be added as clustered index. The assumption within the study was using n=1, where the clustered index was on a single attribute (the derived distinct column).

This resulted in the following SQL query for the addition of the named clustered index to a relation **R**.

```
CREATE CLUSTERED INDEX I ON R (A)
```

Expounded in the example as

```
CREATE CLUSTERED INDEX IX_ newStaging_Customer
ON newStaging_Customer (STStartDate)
```

In conclusion, the selection operation in the relational algebra demonstrated in the algorithm worked on a single relation **R**. The result set after selection formed a new a relation, **S** containing only those records of **R** that satisfy the specified predicate as per the user request. Hence this narrows down to $\mathbf{S} = \boldsymbol{\sigma}_{\mathbf{p}}(\mathbf{R})$. The optimal strategy considered as per the clustered index added above is equality condition on clustering (secondary) index. The estimated I/O cost of this strategy for the entire Selection operation is summarized as follows.

$$\textbf{Estimated I/O cost} = \mathbf{nLevels_A (I) + SC_A(R)}$$

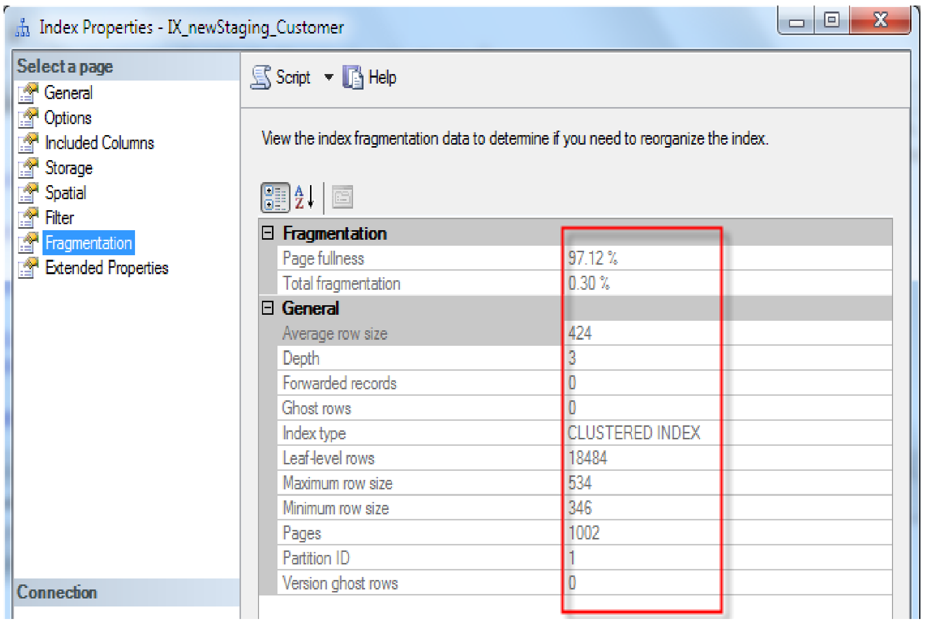
If the predicate condition of the query involves an equality condition on column A, that is not the primary key , then the index is used to retrieve the required records.

## 4.2.3 Deterministic Prioritization Algorithm validation

The following Calculations are used to validate the algorithm. The values are based on database statistics provided by the system catalog for the relation **newStaging_Customer** which was investigated after clustered index was employed. The database statistics are illustrated in figure 4.1 after the clustered index was created in that relation.

Index size = 4, Index depth = 3,

nBlocks = 424, nTuples = 18484



View the index fragmentation data to determine if you need to reorganize the index.

| | |
|---|---|
| **Fragmentation** | |
| Page fullness | 97.12 % |
| Total fragmentation | 0.30 % |
| **General** | |
| Average row size | 424 |
| Depth | 3 |
| Forwarded records | 0 |
| Ghost rows | 0 |
| Index type | CLUSTERED INDEX |
| Leaf-level rows | 18484 |
| Maximum row size | 534 |
| Minimum row size | 346 |
| Pages | 1002 |
| Partition ID | 1 |
| Version ghost rows | 0 |

**Figure 4.1 Database statistics for the Index on newStaging_Customer**

**bFactor (R)** = **nTuples (R) / nBlocks (R)**

bFactor (R) = 18484 / 424 = 43.594

**nLevels$_A$ (I) = (Index Depth/ Index size)**

nLevels$_A$ (I) = (3/ 4) = 0.75000

**SC$_A$(R) = nTuples(R) / (nDistinct$_A$(R)) * nBlocks**

SC$_A$(R) =18484/(18484)*424 = 0.00236

**Estimated I/O cost = nLevels$_A$ (I) + SC$_A$(R)**

Estimated I/O cost = (0.75 + 0.00236) = 0.75236

The estimated operation cost as per the SQL server on the same query was 0.74461

Error Margin = (0.75236 - 0.74461) = 0.00775

This indicated an error margin of **0.00775** that was almost negligible when compared to the actual operation time of the SQL server. The strategy was thus selected as the optimal query optimization by the query processor.

### 4.2.4 Priori Analysis of the algorithm

Upon creation of the SQL query commands generated in a cyclic manner depending on the number of tables that were extracted from the source systems, the queries were passed to the control flows of the SSIS tool. Each control flow represented data flow from one source system hence the number of control structures varied with number of relations available. The start and end task of each control flow representation had some scripts added that measured the performance of the algorithm by recording execution time and stored it on specific variables. The performance of the algorithm assumed that all other factors remained constant such as disk space, number of processors and processor speed. More tests were carried comparing on performance of the new algorithm over existing solutions such as:-

a. Checking different file formats and data types from the source systems e.g. flat files, relational tables in files. The relational OLE-DB databases were highly used for movement of data from source systems to target systems. Extraction in relational databases was quite fast and easy to measure the time factor of the algorithm complexity.
b. Horizontal or Column and Vertical or row based data store to exploit which method used less disk space. The space factor of the algorithm complexity was not affected since the data processing space was not increased or decreased.
c. Compared existing scheduling algorithms strategies and their reported performances:-
    i. Scheduling data transfers and batch processing – job queues existed but no predefined order of execution was noted in existing frameworks.
    ii. Pre-fetching data/Cache storage – The tests depended on system operating system to manage the cache history

iii. FIFO – Mode of handling fast growing data with minimal delays. Limitations observed were due to the latency introduced by incapable legacy systems.

iv. Minimal cost – The suggested solution for handling small data levels.

v. Round robin – The strategy suggested for handling large amounts of data with processes depending on a quantum or time slice.

#### 4.2.5 Posterior Analysis of the algorithm

The deterministic prioritization algorithm was implemented using C# Programming language. This is an object oriented programming language that builds applications that run on systems that have windows operating systems. There were several scripts that were attached to the major phases of the ETL as represented by the SSIS tools that contained the experiment. The scripts captured the Start and End time after execution of the Extraction, Transformation, Staging, and Loading processes. The execution time recorded was recorded in a log file that was accessed after the experiment run was complete depending on the number of times the tests were performed.

The difference between the End and Start times provided a delta change that was compared to the average performance of the algorithm. This was graphically represented to provide a more vivid look at the performance after the new improvement introduced by deterministic prioritization algorithm.

### 4.3 Statistical Analysis of Raw Data used in the study

#### 4.3.1 Source of Data used in the study

The research data was obtained from available online samples provided for research. The focus was to obtain large data from heterogeneous data sources. The procedures undertaken were to demonstrate the binding of data obtained from different sources, performing the ETL operations and fed data to the data warehouses. Projecting towards the research problem, the transfer issue needed to be highlighted in the created scenario. Firstly, there was need to clearly and correctly show from the set scenario that a problem existed before any improvements were carried out. This would show the relevance of putting much effort in generating the new algorithm that would solve the problem demonstrated. The data flow process from the sources was depicted clearly where data staging came in handy at the central processing region. To further understand how the actual data moved from the sources to the data warehouses, there was need to investigate the structure of these sources. The source data form source applications was directly linked to the data extraction process and thus data was availed in varied formats and types. The deliverable for this activity was having a simulation model of the existing situation from one or more data sources. The selected data source was entity related and the relationships between the tables were shown by link joins on key or unique identifiers. Furthermore, to understand data organization on these tables, data modelling was found suiting and most fitting to explain the underlying table designs. The data dimensional modelling procedure was helpful in deriving data models and relational attachments tied to the entities dealt with.

### 4.3.2 First Scenario

Since the research methodology stated in research design was by Experimentation, there was need to demonstrate the current situation within ETL processes. The tests performed depicted the flow of data from source to destination while the data stage was located within the data transformation. To illustrate the tests, the SQL Server Integration Services (SSIS) tool was used since it had the business intelligence management studio component. This tool was readily available as an upgrade package for the Standard and Enterprise edition of the Microsoft SQL server.

The aim of the test was to view and report on Products sold to customers from different geographical regions over a period of time. The data source used was AdventureWorksDW2008R2 database available for download from Microsoft Website. The figure 4.1 shows the data modelling diagrams generated after dimensional modelling in the analysis phase.

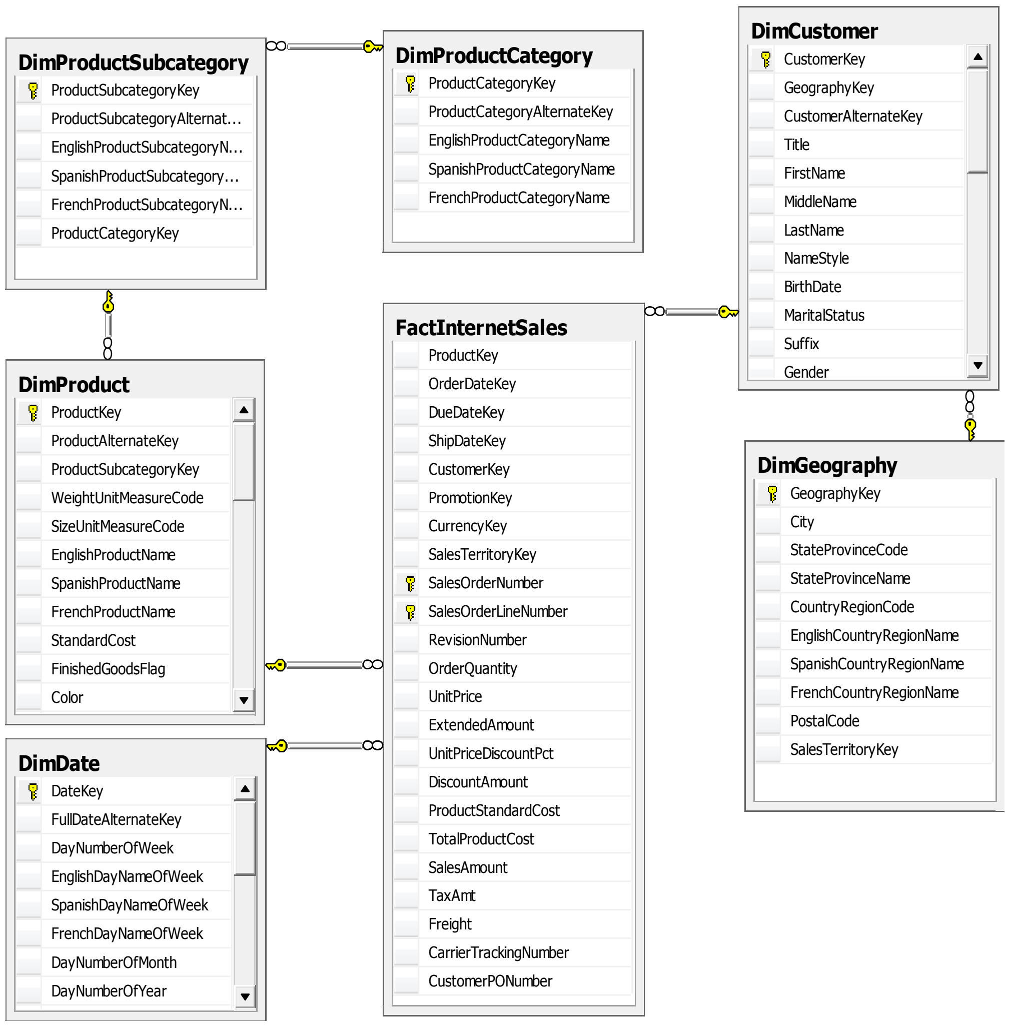


**Figure 4.2 Dimension data model**

The dimension data model shown above is a star-schema. It has a fact table named "Fact Internet Sales" connected to the related six dimensional tables .The fact table contained all the dimension tables' surrogate keys which formed its composite key. Each dimension had a distinct primary key that uniquely identified the records per entity while others were snow-flaked to deeper levels generating new dimensions but maintained the foreign relationships.

#### 4.3.2.1 Issues found from the dimensions

The following issues were observed and formed the variables to be measured in data modelling and analysis.

i. Redundant columns due to regional difference i.e. Education and Occupation were given for different regions i.e. English, Spanish and French. This reduced confidence levels of the column to be a candidate key.

ii. Nullity Measure of columns – Some columns did not have data at all yet they occupied resources.

iii. Column Confidence Levels – Measured columns for their capability to be Candidate Keys. This was shown by the distinctiveness of the columns.

#### 4.3.2.2 Individual Table examination in the developed scenario

The tables from the data model were further examined to understand the indexing of columns in the table structure using the identified variables obtained above. Using SSIS each table data was extracted and attached to the SQL Server data source and then migrated to an Excel workbook target. The figure 4.2 was a snapshot of migrating Customer table using the tool.

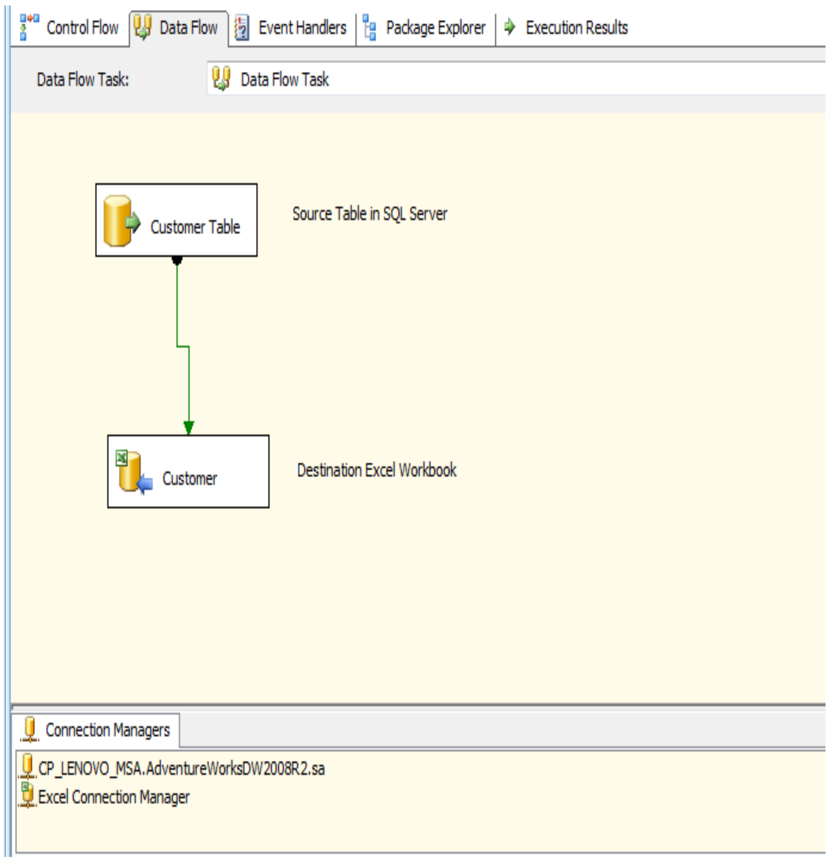


**Figure 4.3 Export of individual tables to the Excel Workbook**

### 4.3.2.3 Table structures Examination results

The following were the findings of measuring the variables against each table to determine the priority of the columns used as main reference points of a full data load to the warehouse. Filtering of table columns in data staging process required the determination of their levels of nullity. The nullity factor was combined with removal of obsolete columns that were never used anymore at the target systems. The following query was used to collect null levels of columns for each table examined.

```
SELECT COUNT(*) - COUNT(EndDate) Nulls, COUNT(EndDate) NonNulls

FROM DimProduct
```

**Table 4.1 Customer Dimension**

| CUSTOMER TABLE SUMMARY | | | | | | |
|---|---|---|---|---|---|---|
| Column Name | Title | Suffix | AddressLine2 | CustomerKey | CustomerAlte | AddressLine1 |
| NULL Columns | 18383 | 18481 | 18172 | 0 | 0 | 5686 |
| Distinct columns | 101 | 3 | 312 | 18483 | 18483 | 7111 |
| Total Rows | 18484 | 18484 | 18484 | 18483 | 18483 | 12797 |
| Nullity Percentage % | 99.4536 | 99.98377 | 98.31205367 | 0 | 0 | 44.43228882 |
| Distinct Percentage % | 0.54642 | 0.01623 | 1.687946332 | 100 | 100 | 55.56771118 |

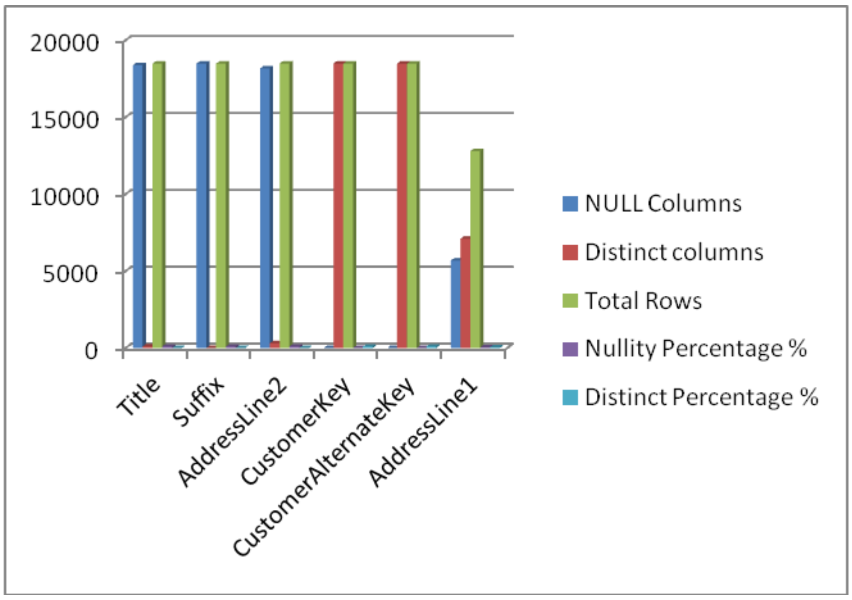


**Figure 4.4 Customer Dimension Columns Analysis**

CustomerKey and CustomerAlternateKey qualified to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness as shown in figure 4.3. The table 4.1 had two unique keys in order to separate between the children relations associated with Customer's dimension.

**Table 4.2 Geography Dimension**

| GEOGRAPHY TABLE SUMMARY | | |
|---|---|---|
| **Column Name** | **GeographyKey** | **City** |
| **NULL Columns** | 0 | 0 |
| **Distinct columns** | 655 | 562 |
| **Total Rows** | 655 | 655 |
| **Nullity Percentage %** | 0 | 0 |
| **Distinct Percentage %** | 100 | 85.80153 |

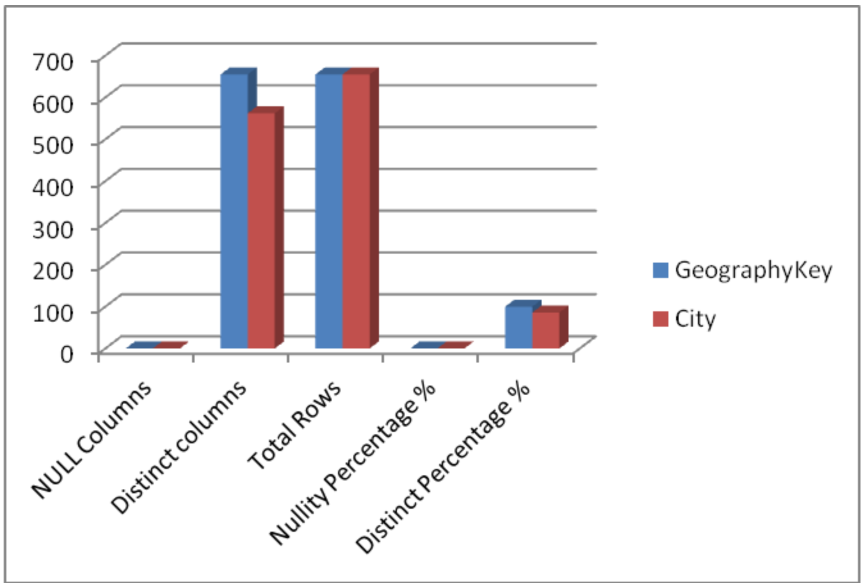


**Figure 4.5 Geography Dimension Columns Analysis**

GeographyKey and City columns were determined to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness as seen in figure 4.4.

**Table 4.3 Date Dimension**

| **DATE TABLE SUMMARY** | | |
|---|---|---|
| **Column Name** | **DateKey** | **FullDateAlternate Key** |
| **NULL Columns** | 0 | 0 |
| **Distinct columns** | 1188 | 1188 |
| **Total Rows** | 1188 | 1188 |
| **Nullity Percentage %** | 0 | 0 |
| **Distinct Percentage %** | 100 | 100 |

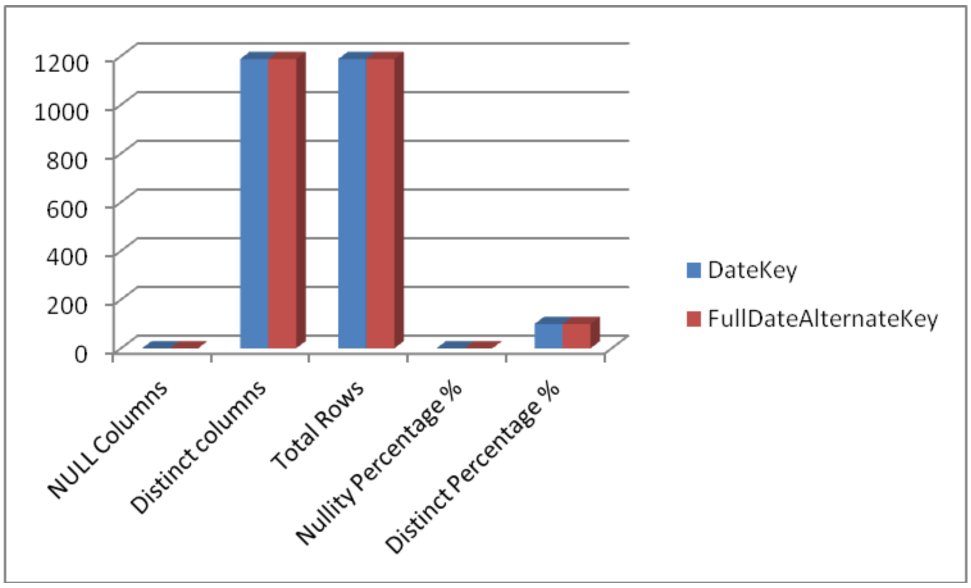


**Figure 4.6 Date Dimension Columns Analysis**

DateKey and FullDateAlternateKey qualified to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness in comparison to the other columns as observed in figure 4.5.

**Table 4.4 Product Dimension**

| PRODUCT TABLE SUMMARY | | | | |
|---|---|---|---|---|
| Column Name | ProductKey | ProductAlternateKey | ProductL | Class |
| NULL Columns | 0 | 0 | 226 | 276 |
| Distinct columns | 606 | 606 | 4 | 4 |
| Total Rows | 606 | 606 | 606 | 606 |
| Nullity Percentage % | 0 | 0 | 37.29373 | 45.54455446 |
| Distinct Percentage % | 100 | 100 | 0.660066 | 0.660066007 |

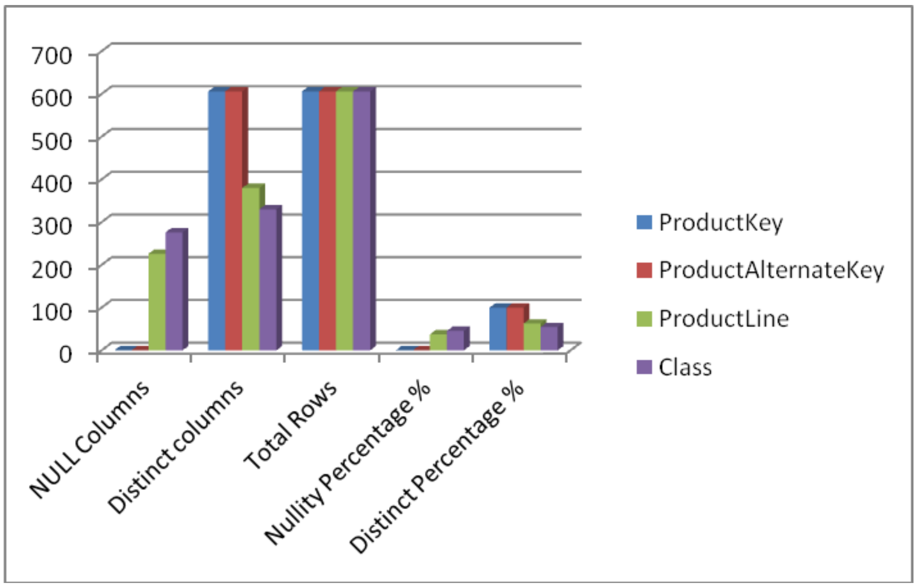


**Figure 4.7 Product Dimension Columns Analysis**

ProductKey and ProductAlternateKey qualified to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness. For instance the degree of distinct column ProductKey was 100% as noted in figure 4.6.

**Table 4.5 ProductCategory Dimension**

| PRODUCTCATEGORY TABLE SUMMARY | | |
|---|---|---|
| Column Name | ProductCategoryKey | ProductCategoryAlternateKey |
| NULL Columns | 0 | 0 |
| Distinct columns | 4 | 4 |
| Total Rows | 4 | 4 |
| Nullity Percentage % | 0 | 0 |
| Distinct Percentage % | 100 | 100 |

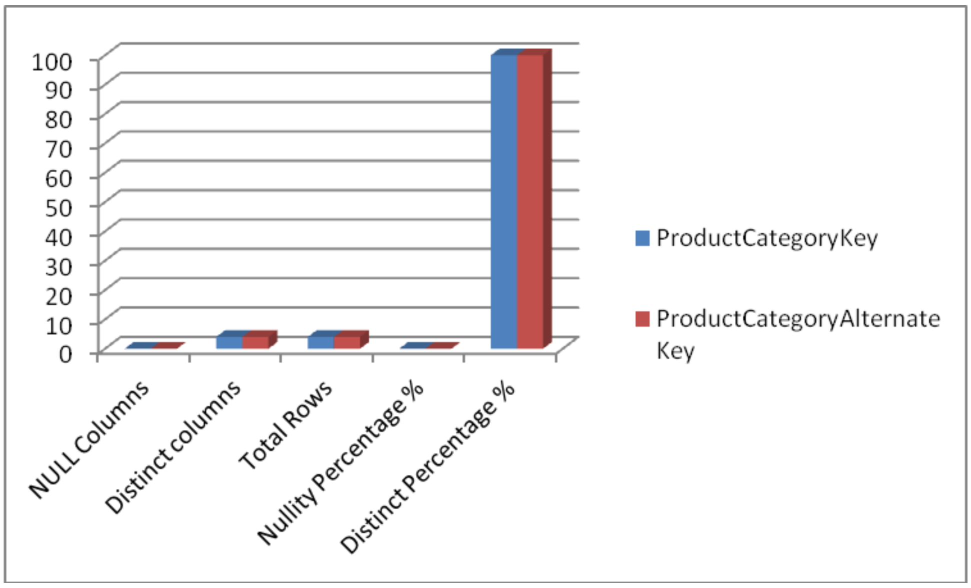


**Figure 4.8 Product Category Dimension Columns Analysis**

ProductCategoryKey and ProductCategoryAlternateKey qualified to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness as seen in figure 4.7.

**Table 4.6 ProductSubcategory Dimension**

| PRODUCTSUBCATEGORY TABLE SUMMARY | | |
|---|---|---|
| Column Name | ProductSubcategoryKey | ProductSubcategoryAlternateKey |
| NULL Columns | 0 | 0 |
| Distinct columns | 37 | 37 |
| Total Rows | 37 | 37 |
| Nullity Percentage % | 0 | 0 |
| Distinct Percentage % | 100 | 100 |

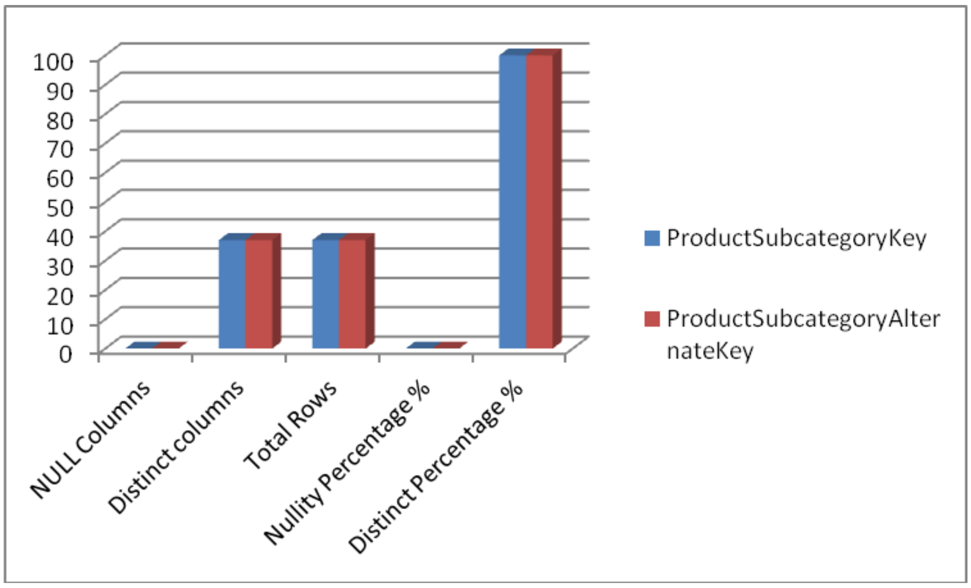


**Figure 4.9 Product Subcategory Dimension Columns Analysis**

ProductSubcategoryKey and ProductSubcategoryAlternateKey qualified to be the candidate Keys for this dimension due to their high confidence levels and their distinctiveness as seen in figure 4.8.

**Table 4.7 Internet Sales Fact**

| INTERNET SALES TABLE SUMMARY | | | | |
|---|---|---|---|---|
| **Column Name** | **ProductKey** | **OrderDateKey** | **DueDateKey** | **SalesOrderNo** |
| **NULL Colum** | 0 | 0 | 0 | 0 |
| **Distinct colum** | 158 | 1124 | 1124 | 60398 |
| **Total Rows** | 60398 | 60398 | 60398 | 60398 |
| **Nullity in %** | 0 | 0 | 0 | |
| **Distinct in %** | 0.261598066 | 1.860988774 | 1.860988774 | 100 |

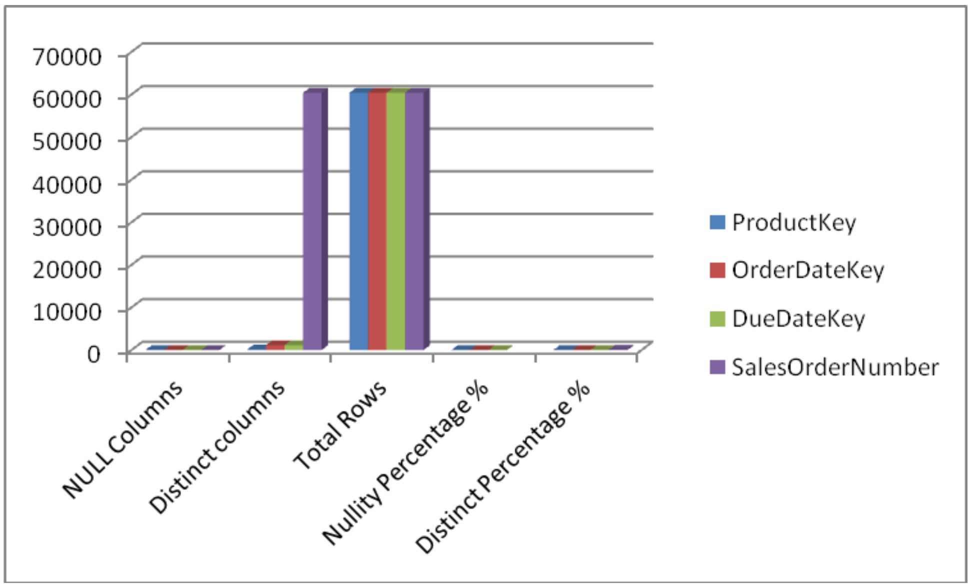


**Figure 4.10 Internet Sales Dimension Columns Analysis**

The composite key for the above fact table was generated as a combination of all the foreign keys (Surrogate keys from related tables) together with the SalesOrderNumber column. Each of the dimensions was represented in the fact based on the associative key as demonstrated in figure 4.9.

#### 4.3.2.4 Significance of table structure examination

The main aim of examining the table structures of data sources was to understand the table designs with regard to uniqueness of data stored within the tables. The data selection during search within the tables was based on the primary keys set on each table. Since the user queries were dynamic, there was need for creation of procedures to collect the most vital data. This would be based on several factors that needed optimization namely;

- Relevance of the data to transfer
- Previous common selections procedures
- Actual data demands by presentation users
- Maintaining integrity of the data association after selection and transfer.

The goal was to figure out if there was a predetermined sequence of data selections from the sources that could be noted after deeper analysis. It wanted to establish if there existed filtering mechanisms for certain data as inclined to the business processes of a sales environment and whether this sequence would build some intelligence to determine future selections.

Based on the data content and selection queries obtained from the data warehouse users, the table columns confidence levels were noted to be deterministic in collection of data i.e. selecting only data that is needed and minimizing duplication of data at the data warehouses.

There was need to maintain the entity relationships at the destination side to match those set in the data sources. The integrity of loaded data would be termed correct when their original relationship is also depicted and maintained at the target systems accordingly. This would help to mirror single systems available at the sources and the business users would not identify the difference at operational level.

## 4.4 Demonstrate Data Staging within ETL Process

The experiment was set-up and prepared using the SSIS tools. The graphical design involved adding data control flow diagrams that simulated the actual data flow from the sources to the destination. Further configurations were done in the backend where the new algorithm was implemented within the data control structures on every ETL stage with addition of execution scripts. The experiment diagrams are shown in figure 4.10, figure 4.11, figure 4.12 and figure 4.13.

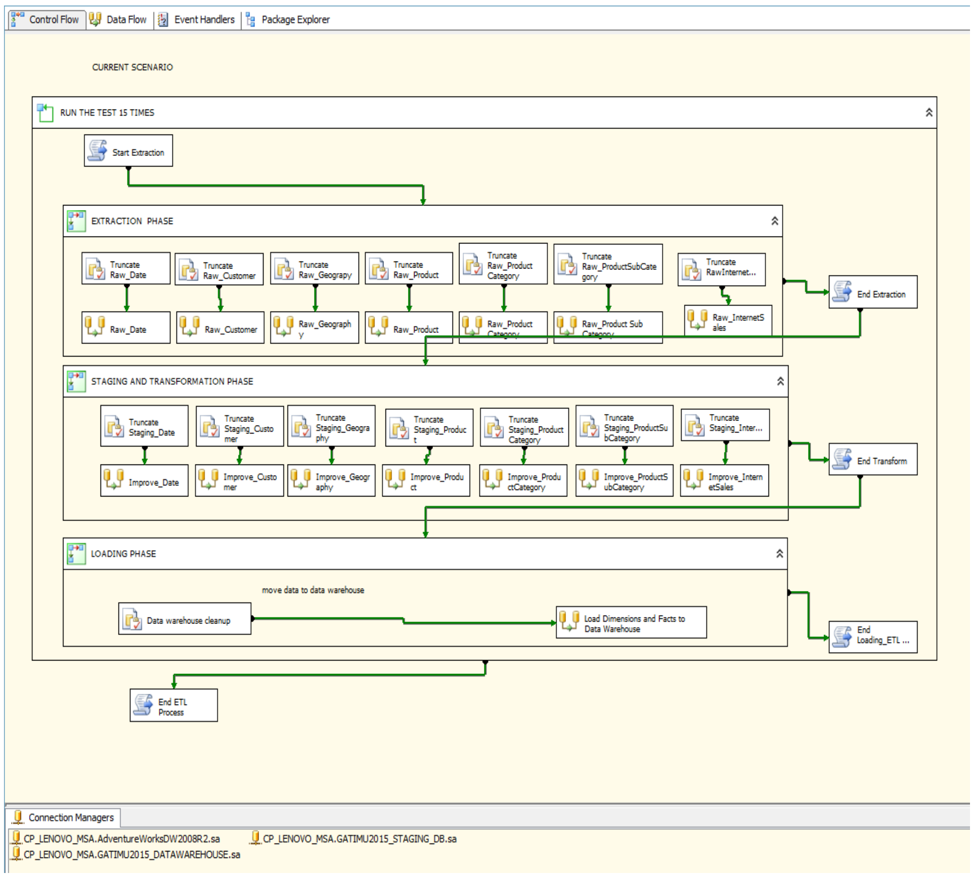


**Figure 4.11 Demonstration of all ETL phases in experiment simulation**

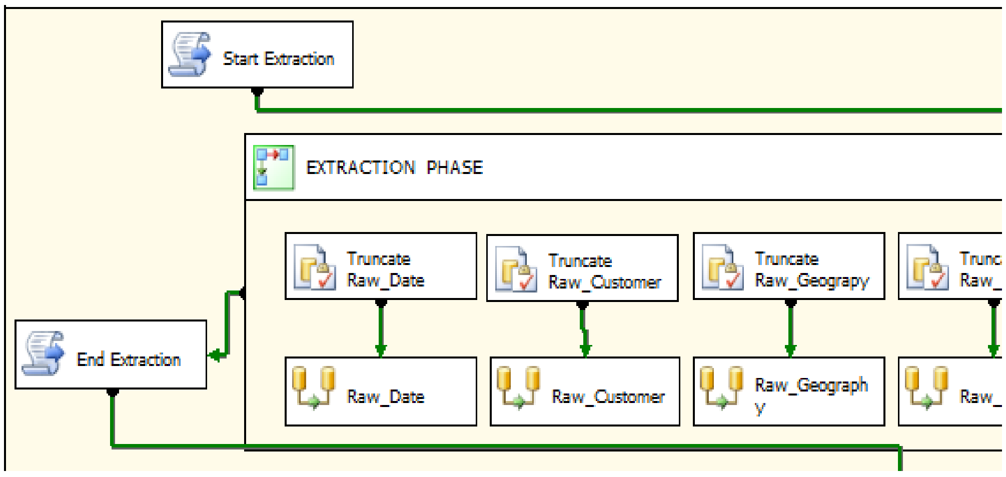


**Figure 4.12 Demonstration of part of extraction process from experiment**

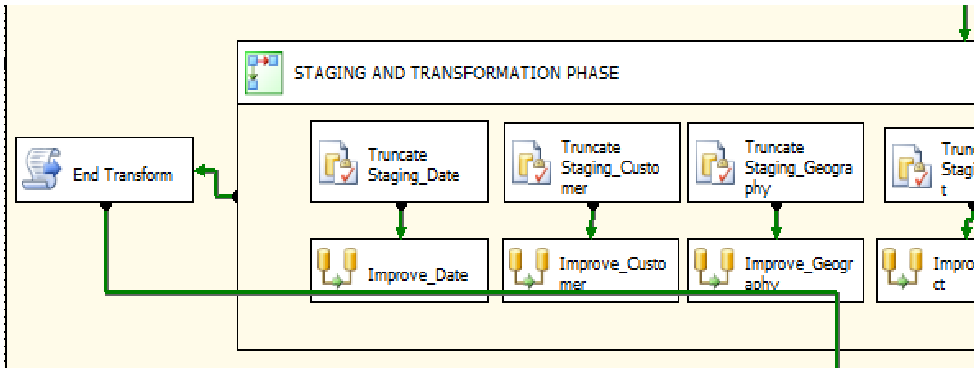


**Figure 4.13 Demonstration of part of staging process from experiment**

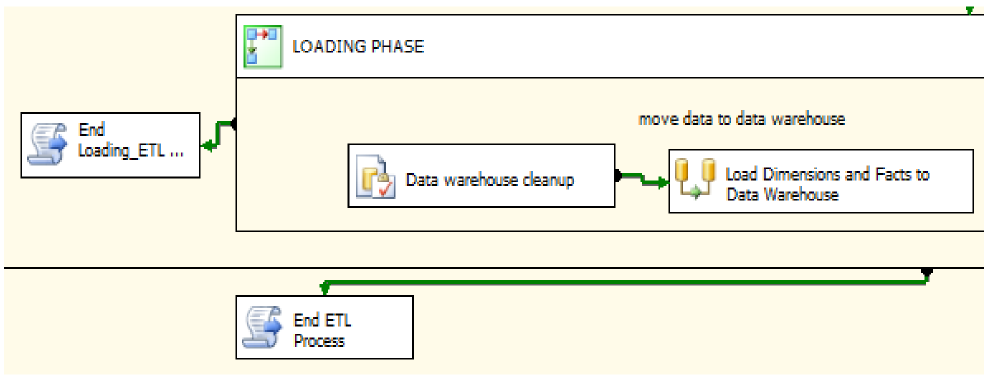

LOADING PHASE
End Loading_ETL ...
move data to data warehouse
Data warehouse cleanup
Load Dimensions and Facts to Data Warehouse
End ETL Process

**Figure 4.14 Demonstration of part of loading process from experiment**

### 4.4.1 Scenario setup

The data sources used in the project were added and connection was created using the various installed connection managers such as SQL Server Native Client 10.0. The connection manager depended on the type of the data source e.g. OLE_DB Source used the Native OLE_DB SQL Provider.

The data was cleaned to remove any “dirty data” such as repetition, duplicates and obsolete fields. More transformation tasks could be added in the scenario later such as sorting, merging, conversions, deriving new columns, e.t.c. In the simple setup the study added one transformation task for deriving new columns as illustrated in figure 4.14.

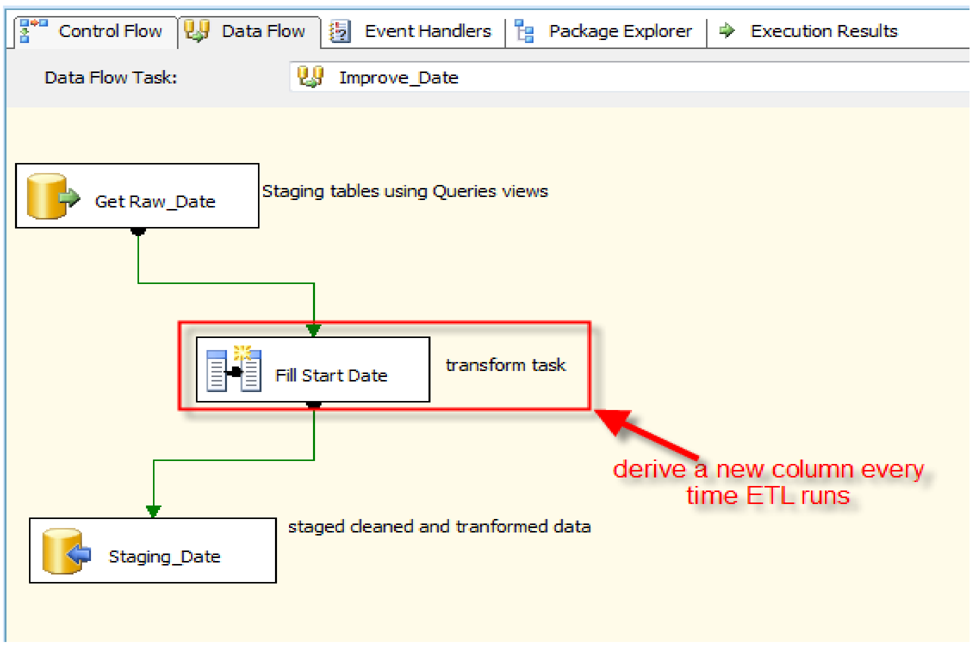


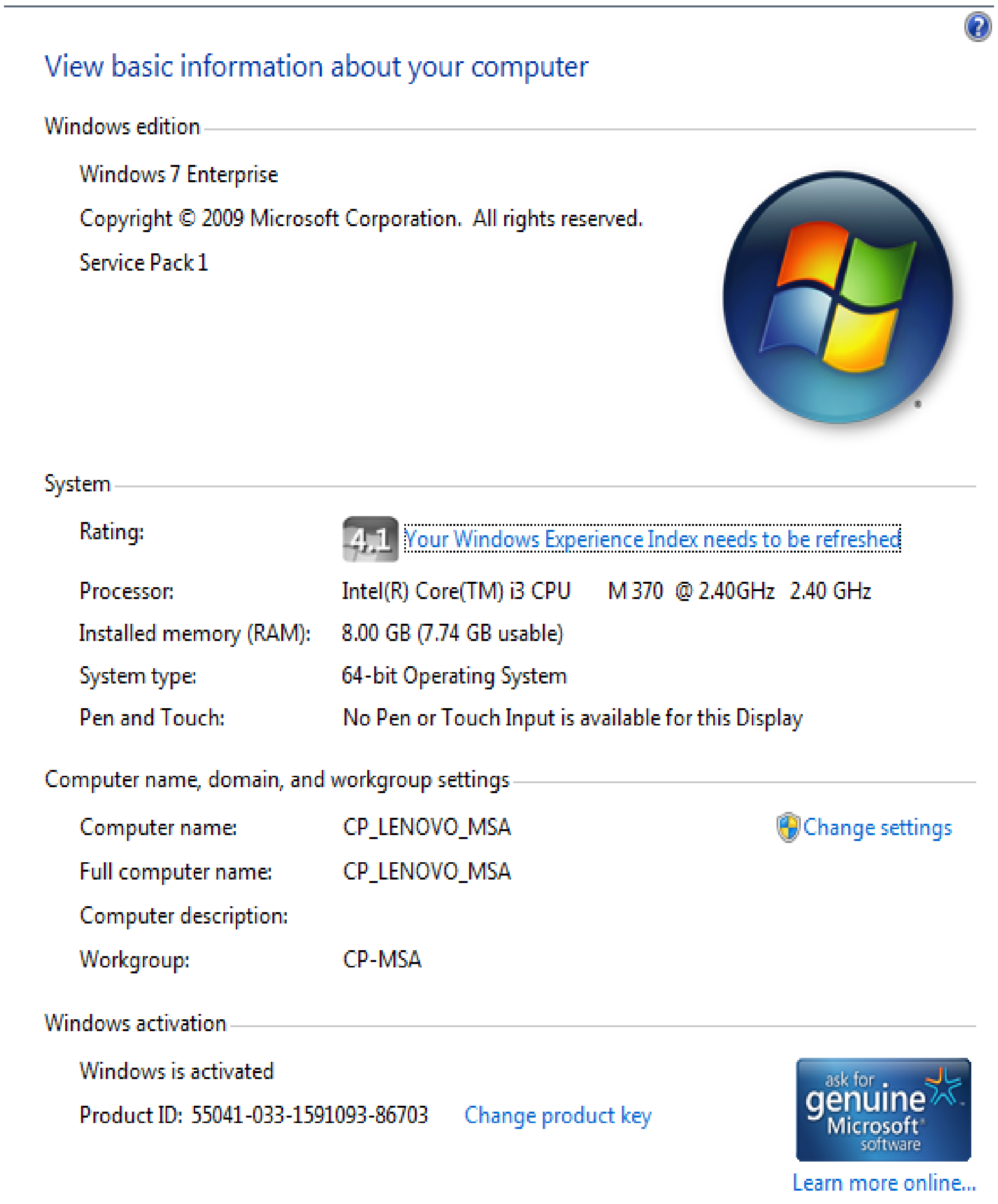
**Figure 4.15 Transform Task**

More data conversions could be done here to prune the output as per organization's customization before it could to be moved to the data warehouse. The project setup was a Microsoft Visual Studio package that was executed to measure performance and efficiency variables. Several scripts were included in the project to collect the variable changes over the number of times the tests were executed. The tests carried out were based on the data staging tables' approach where the extracted data from the sources was stored in staging tables. The study showed the change introduced when the Deterministic Prioritization approach was used in the setup.

**System properties**

The tests were carried based on AdventureWorksDW2008R2 database which was freely available for download from Microsoft website link http://msftdbprodsamples.codeplex.com/releases/view/59211

The Computer system used had the following specifications shown in the figure 4.15.



View basic information about your computer

Windows edition

Windows 7 Enterprise

Copyright © 2009 Microsoft Corporation. All rights reserved.

Service Pack 1

System

| | |
|---|---|
| Rating: | 4.1 Your Windows Experience Index needs to be refreshed |
| Processor: | Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz 2.40 GHz |
| Installed memory (RAM): | 8.00 GB (7.74 GB usable) |
| System type: | 64-bit Operating System |
| Pen and Touch: | No Pen or Touch Input is available for this Display |

Computer name, domain, and workgroup settings

| | |
|---|---|
| Computer name: | CP_LENOVO_MSA |
| Full computer name: | CP_LENOVO_MSA |
| Computer description: | |
| Workgroup: | CP-MSA |

Windows activation

Windows is activated

Product ID: 55041-033-1591093-86703

**Figure 4.16 System specifications**

## 4.4.2 Situation before enhancement

The test scenario environment utilized three databases to simulate the production environment to separate operations and tasks. The reason for this was that the data had dynamic state and to note its change at different times, one needed to store it in separate locations. This was keenly implemented in the ETL process.

i. Source databases – From the AdventureWorksDW2008R2 database
ii. Staging database - Had Raw and staged tables
iii. Data warehousing database - For final loading and presentation purposes.

### Running the scenario

When the setup project was run, the process execution order started with the Extraction stage and the components found within each stage ran concurrently before moving to the next stage as shown in figure 4.16 and figure 4.17.

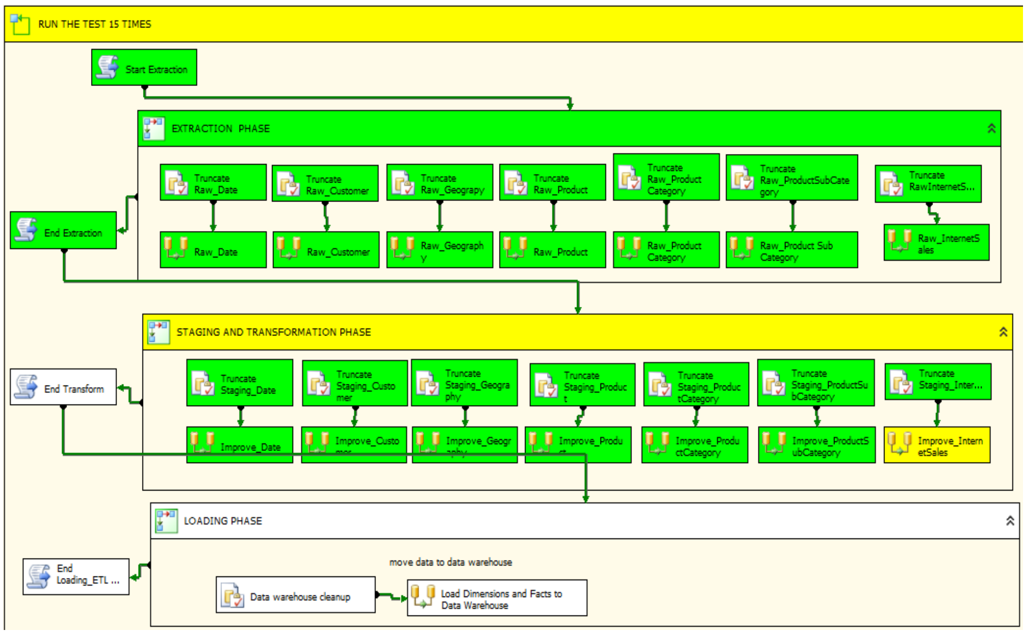


**Figure 4.17 Scenario execution**

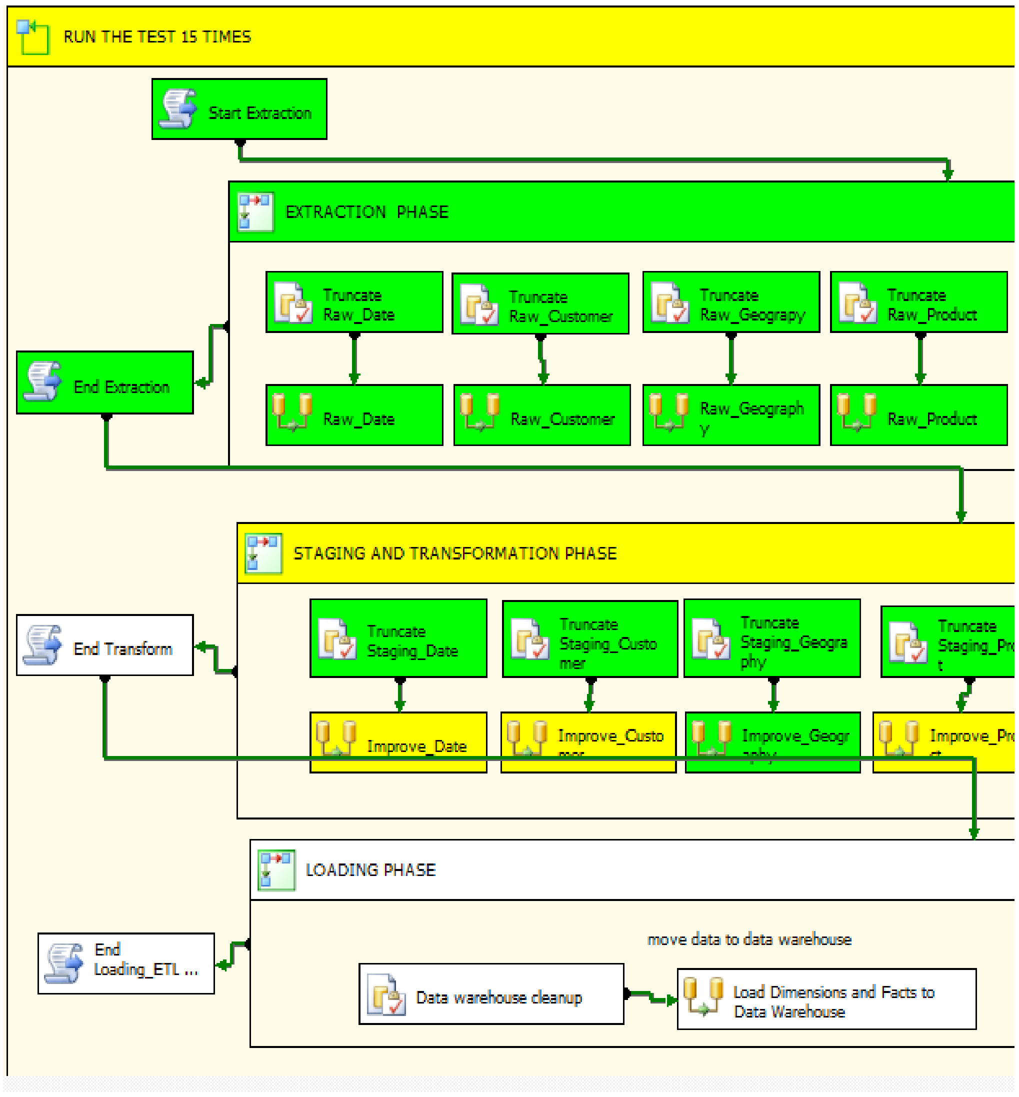


**Figure 4.18 Expanded view of Scenario execution**

A log file was created after complete ETL iteration when running the project since it had information on the variables to measure.

### 4.4.3 Situation After enhancement

Deterministic and Prioritization approach (DP) had been utilized throughout the ETL process as discussed below. The same setup used in the demonstration of before enhancement had been used to illustrate the improvements made when the new proposition was in place.

#### 4.4.3.1 Deterministic and Prioritization Implementation

The new algorithm that was described in chapter three was based on two concepts namely; Determinism and Prioritization. These key concepts were covered exclusively especially on their role to provide an amicable solution to the underlying problems noted within data staging area.

Firstly, the term deterministic means the nature of retaining of state by a procedure when given the same inputs, it yields the same output with the exception of running cost. With regard to the analysed data, the key to data to be loaded to the data warehouses relied on data selections as per user needs. Occasionally, not all data from the sources was loaded to the warehouses depending on the nature of operations within the ETL processes that filtered out some data. All these activities either were controlled by the needs of business users at the destination systems or the customizations of available data staging frameworks under which filtering was done with a purpose. These limitations could only be visible after analysing the data sources structures as well as destinations systems structures.

However, since the data flow was unidirectional from the data sources to target systems, the nature of transfer considered which data was transferred from one phase of ETL to the next. Reducing on the amount of data involved in these transactions would mean applying several functions to filter out data exclusively from the load capacity. Through data modelling the scanning of the table structures illustrated the data attributes with regards to the columns arrangement, importance (confidence levels), and their relation to within and without other relations.

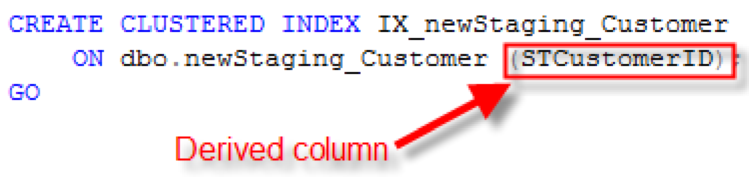
The external binding would later form the relational model of a database system. Although the physical aspects of the data as regards the diagrammatic representation shown from the experiment setups and configurations remained constant, the approach was deterministic in nature; same input gave same output results. The idea was to have a way of determining and deciding beforehand which columns of tables were to be used in creating the staging tables after extraction process. The knowledge on table columns' confidence levels and access levels was necessary to build some intelligence prior enough to the loading process hence determining what to migrate to the data warehouse. Since the filtering of data by columns was done specifically by organizations that had customized frameworks, the study extracted all the data from the sources as is and transferred it to the data staging area for further manipulations as per the new algorithm.

Secondly, the term prioritization means evaluation of a group of items and ranking them in the order in urgency or importance. The most ranked data is said to be of high priority and the urge to transfer it to the data warehouse faster than the low priority data. To achieve this there needed to have mechanisms of informing the systems on how to handle the different levels during transfers. The prioritization algorithm affected the logical aspects enforced with respect to the business rules of an organization thereby maintaining the dimensional model. The data staging area was to be optimized to run its operations in a timely fashion and with ability to run multiple tasks at the same time, concurrency could be achieved. Upon knowing what kind of data needed to be loaded to the data warehouses through the staging area, priority was to be given to such data to speed-up its availability to the requesters on the target systems, while the low priority data followed suit.

The general problem was how to separate the data based on priority at loading time considering the high number of query data requests and the growing number of rows per table. The algorithm established a way of changing the speed to query executions which was implemented by adding indices to specific columns of high priority. The query execution would therefore speedup based on the clustered indices added to the table under scrutiny. The power behind indexing as a solution to priority issue was the ability to manipulate the order in which SQL queries were handled during execution.

**Prioritization by indexing specific columns**

Most often, the staging tables were generated through execution of SQL query commands, and this strategy implemented prioritization of the selection columns by using named clustered indexes. New distinct derived columns were added to each staging table and they were used to create indexes. The advantage of using these external columns generated only at runtime was that they did not affect the data from the source in any way ensure consistency and also the relationship of data from the sources was maintained.

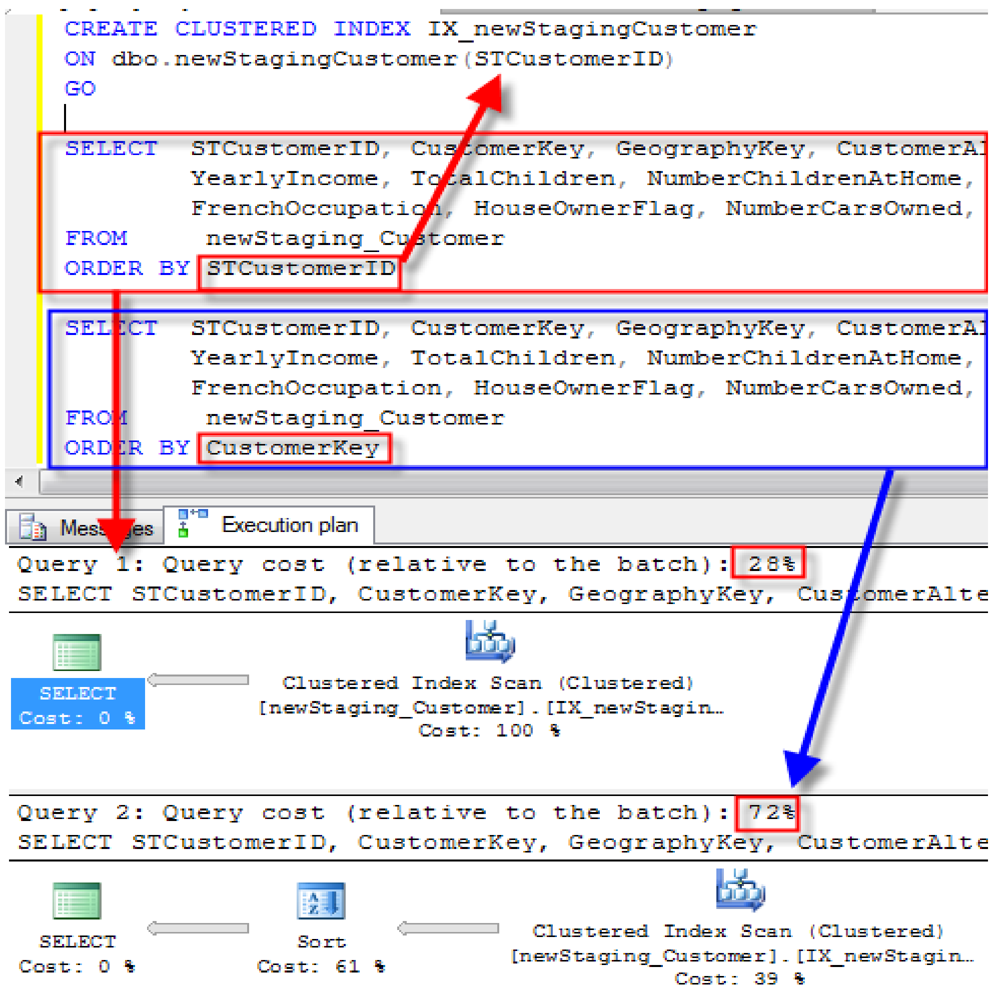


```
CREATE CLUSTERED INDEX IX_newStaging_Customer
    ON dbo.newStaging_Customer (STCustomerID);
GO
```

**Figure 4.19 Clustered index**

The figure 4.18 demonstrates how prioritization was achieved for one of the tables named newStaging_Customer”. A clustered index was created and given a name “IX_newStaging_Customer”. The priority was set on the derived unique column named “STCustomerID” on that staging table named “newStaging_Customer”. Moreover, the created index is not limited to only one column as shown in figure 4.18. It could include either new columns or existing columns. The newly defined index hinted the change in order of execution of the Data Definition Language (DDL) queries submitted to the query server. The resulting impact was to override the server’s query execution plan.

A query execution plan has an ordered set of steps used to access data in a SQL relational database management system. The plan represents how data flows from child operators to parent operators in a form of tree structure. The query execution plan depends on the query optimizer. The query optimizer is a feature of database server that attempts to determine the most efficient way of executing a given query. The query optimizer checks for the best plan before passing the query to the query executor that gives the results after firing. Before adding the clustered index to the table, the order of query execution is controlled by the Primary key which is a unique identifier added upon table creation. After adding the clustered index to the table, the query optimizer suggests a better and efficient plan for query execution over the original plan of using the primary key. Prioritization by distinct columns enhanced the efficiency and performance of the query execution plan on the server during query search by adding hints. This resulted in optimized selection costs yet maintaining the quality of data to the data warehouse. The optimal measure of change by use of hints to prioritize data columns in the selection query was demonstrated as in figure 4.19.

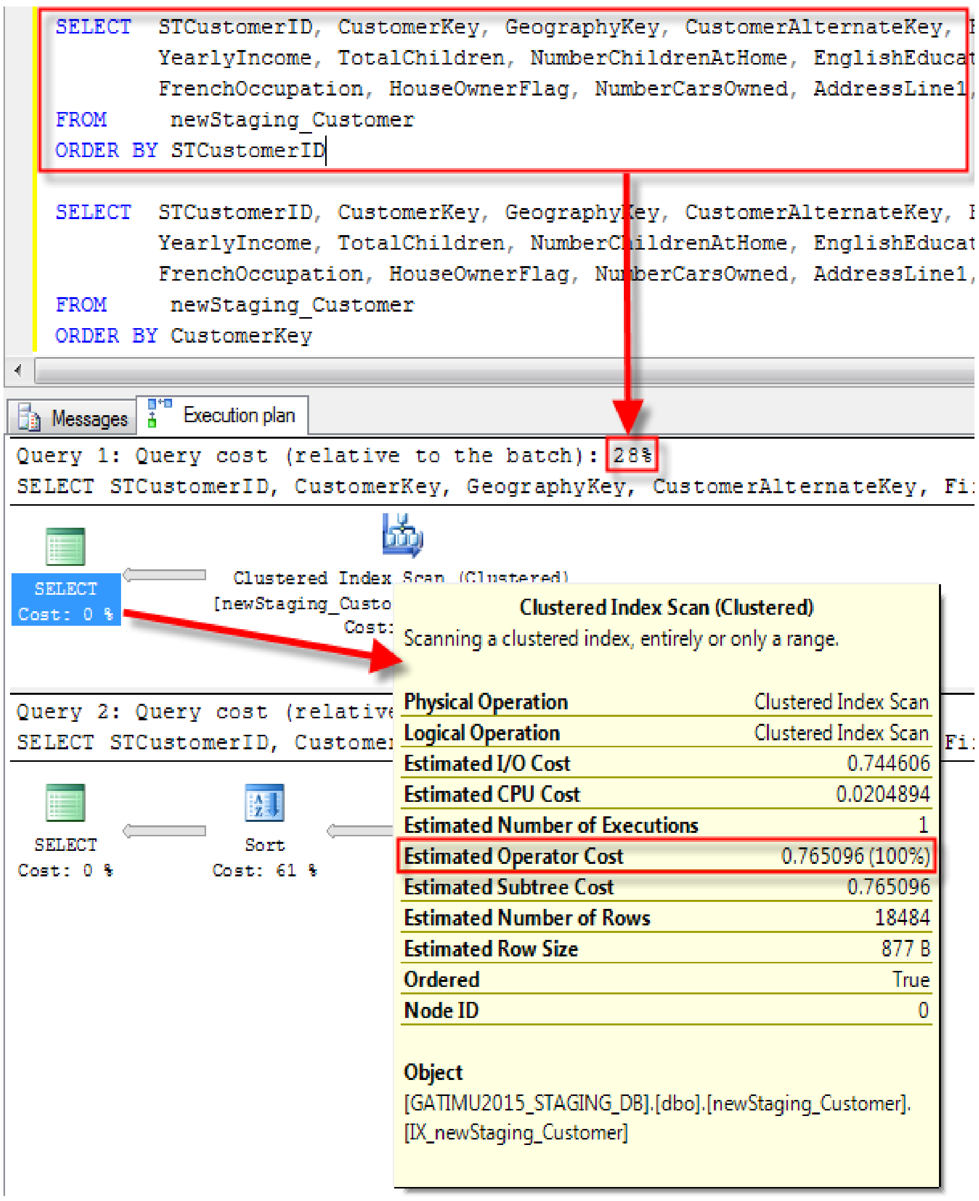


```
CREATE CLUSTERED INDEX IX_newStagingCustomer
ON dbo.newStagingCustomer(STCustomerID)
GO

SELECT   STCustomerID, CustomerKey, GeographyKey, CustomerAl
         YearlyIncome, TotalChildren, NumberChildrenAtHome,
         FrenchOccupation, HouseOwnerFlag, NumberCarsOwned,
FROM      newStaging_Customer
ORDER BY STCustomerID

SELECT   STCustomerID, CustomerKey, GeographyKey, CustomerAl
         YearlyIncome, TotalChildren, NumberChildrenAtHome,
         FrenchOccupation, HouseOwnerFlag, NumberCarsOwned,
FROM      newStaging_Customer
ORDER BY CustomerKey
```

```
Query 1: Query cost (relative to the batch): 28%
SELECT STCustomerID, CustomerKey, GeographyKey, CustomerAlte

SELECT        Clustered Index Scan (Clustered)
Cost: 0 %     [newStaging_Customer].[IX_newStagin...
              Cost: 100 %

Query 2: Query cost (relative to the batch): 72%
SELECT STCustomerID, CustomerKey, GeographyKey, CustomerAlte

SELECT        Sort          Clustered Index Scan (Clustered)
Cost: 0 %     Cost: 61 %    [newStaging_Customer].[IX_newStagin...
                            Cost: 39 %
```

**Figure 4.20 Optimized Query Execution plan**

STCustomerID was the newly derived column while CustomerKey was the Primary Key for the table. The index was created on the derived column to show the difference with the primary key column.



```
SELECT  STCustomerID, CustomerKey, GeographyKey, CustomerAlternateKey, 
        YearlyIncome, TotalChildren, NumberChildrenAtHome, EnglishEducat
        FrenchOccupation, HouseOwnerFlag, NumberCarsOwned, AddressLine1,
FROM     newStaging_Customer
ORDER BY STCustomerID

SELECT  STCustomerID, CustomerKey, GeographyKey, CustomerAlternateKey, 
        YearlyIncome, TotalChildren, NumberChildrenAtHome, EnglishEducat
        FrenchOccupation, HouseOwnerFlag, NumberCarsOwned, AddressLine1,
FROM     newStaging_Customer
ORDER BY CustomerKey
```

```
Query 1: Query cost (relative to the batch): 28%
SELECT STCustomerID, CustomerKey, GeographyKey, CustomerAlternateKey, Fi
```

```
Query 2: Query cost (relative
SELECT STCustomerID, Custome
```

**Clustered Index Scan (Clustered)**

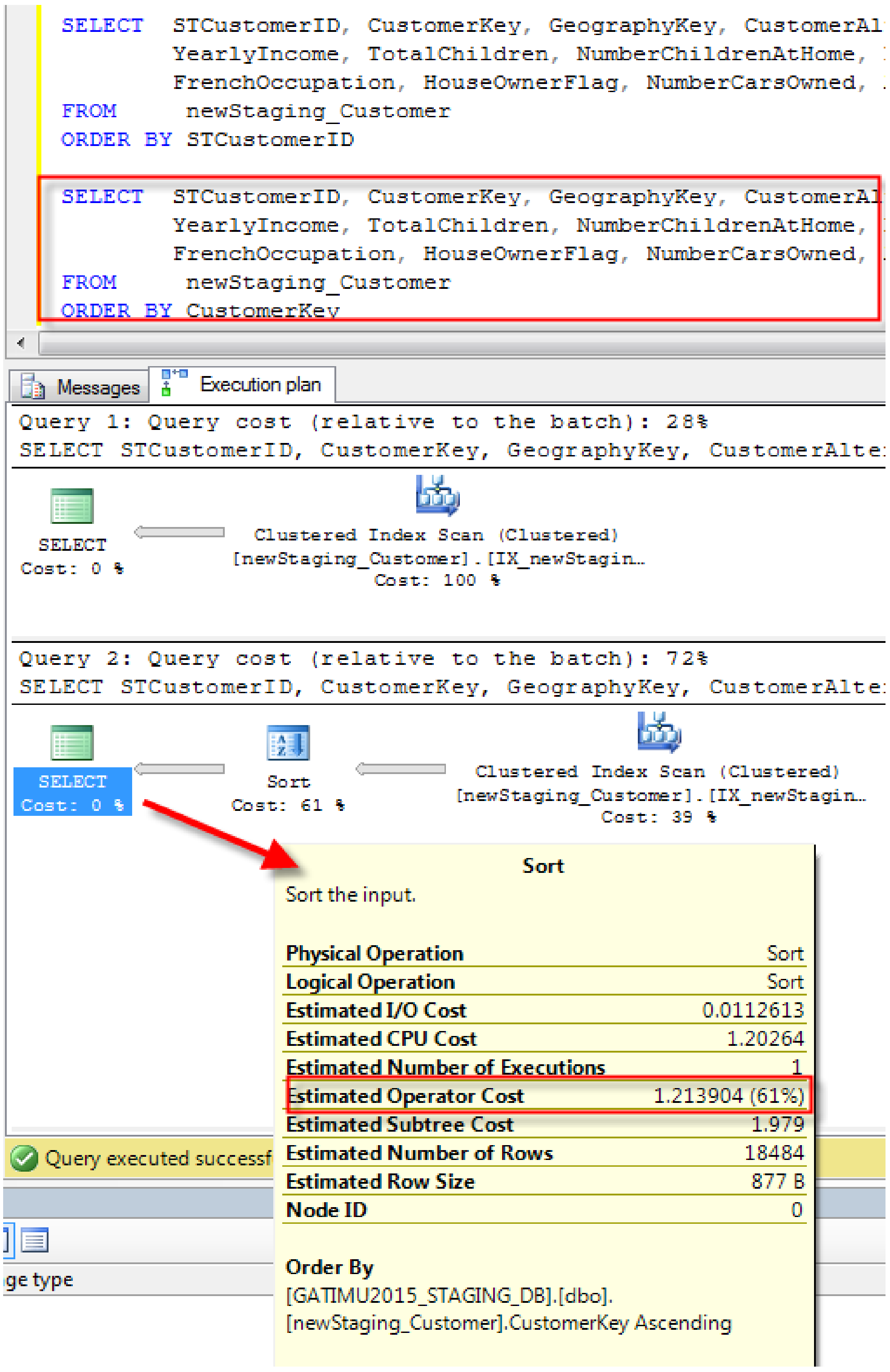
Scanning a clustered index, entirely or only a range.

| | |
|---|---|
| **Physical Operation** | Clustered Index Scan |
| **Logical Operation** | Clustered Index Scan |
| **Estimated I/O Cost** | 0.744606 |
| **Estimated CPU Cost** | 0.0204894 |
| **Estimated Number of Executions** | 1 |
| **Estimated Operator Cost** | 0.765096 (100%) |
| **Estimated Subtree Cost** | 0.765096 |
| **Estimated Number of Rows** | 18484 |
| **Estimated Row Size** | 877 B |
| **Ordered** | True |
| **Node ID** | 0 |

**Object**

[GATIMU2015_STAGING_DB].[dbo].[newStaging_Customer].[IX_newStaging_Customer]

**Figure 4.21 Demonstration with new clustered index**



```
SELECT   STCustomerID, CustomerKey, GeographyKey, CustomerAl
         YearlyIncome, TotalChildren, NumberChildrenAtHome,
         FrenchOccupation, HouseOwnerFlag, NumberCarsOwned,
FROM      newStaging_Customer
ORDER BY STCustomerID

SELECT   STCustomerID, CustomerKey, GeographyKey, CustomerAl
         YearlyIncome, TotalChildren, NumberChildrenAtHome,
         FrenchOccupation, HouseOwnerFlag, NumberCarsOwned,
FROM      newStaging_Customer
ORDER BY CustomerKey
```

```
Query 1: Query cost (relative to the batch): 28%
SELECT STCustomerID, CustomerKey, GeographyKey, CustomerAlte

SELECT        Clustered Index Scan (Clustered)
Cost: 0 %     [newStaging_Customer].[IX_newStagin…
              Cost: 100 %

Query 2: Query cost (relative to the batch): 72%
SELECT STCustomerID, CustomerKey, GeographyKey, CustomerAlte

SELECT        Sort          Clustered Index Scan (Clustered)
Cost: 0 %     Cost: 61 %    [newStaging_Customer].[IX_newStagin…
                            Cost: 39 %
```

**Sort**

Sort the input.

| | |
|---|---|
| **Physical Operation** | Sort |
| **Logical Operation** | Sort |
| **Estimated I/O Cost** | 0.0112613 |
| **Estimated CPU Cost** | 1.20264 |
| **Estimated Number of Executions** | 1 |
| **Estimated Operator Cost** | 1.213904 (61%) |
| **Estimated Subtree Cost** | 1.979 |
| **Estimated Number of Rows** | 18484 |
| **Estimated Row Size** | 877 B |
| **Node ID** | 0 |

**Order By**
[GATIMU2015_STAGING_DB].[dbo].
[newStaging_Customer].CustomerKey Ascending

**Figure 4.22 Demonstration with previous primary key**

The figures 4.19 through figure 4.21 illustrate the contrast on selection queries on tables with clustered indexes and without. The focus is on the Estimated Operation Cost. This shows the percentage cost taken by the operator in this case it is the SELECT operation. This is also associated with the resource I/O and CPU availability. It helps to determine if the query is I/O or CPU-intensive.

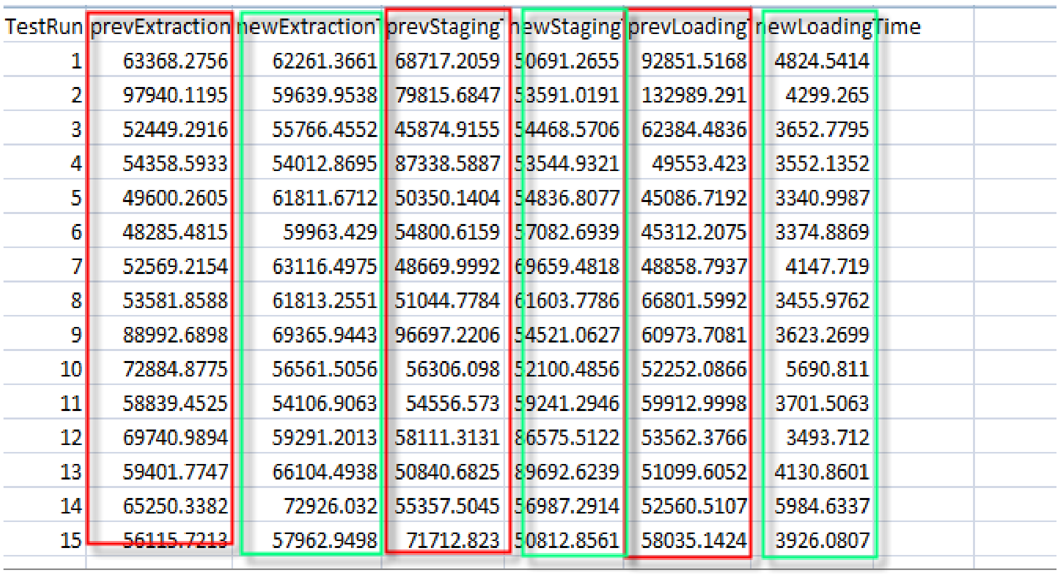
The graphical demonstration of the Actual execution plan indicates that the Operation cost when the tables make use of clustered indexes to prioritize data selections is lower than without hence the use of this concept in prioritization algorithm.

**Test results using the scenario in SSIS tool**

The situation before enhancing was simulated and yielded the following results that were recorded when the scenario was run for fifteen times recursively. The reason for running the tests for multiple times was to see whether the results of each test were consistent in order to draw a conclusion on the performance of the new algorithm. Another reason was that the experiment depended on measured variables to draw conclusion. The change in values to these variables depended on the resources available for the system on which the experiment was done. I real-life situations some resources would be free at different times and hence the need to observe the change in performance at different times after runs. The comparison of the results was separated into two scenarios i.e. the situation before enhancement and the new situation after enhancement.

**Comparing all stages of ETL processes before and after enhancement**

The figure 4.22 is a line graph illustrating the performance levels for all ETL processes combined showing situation before and after enhancement.



| TestRun | prevExtraction | newExtraction | prevStaging | newStaging | prevLoading | newLoadingTime |
|---|---|---|---|---|---|---|
| 1 | 63368.2756 | 62261.3661 | 68717.2059 | 50691.2655 | 92851.5168 | 4824.5414 |
| 2 | 97940.1195 | 59639.9538 | 79815.6847 | 53591.0191 | 132989.291 | 4299.265 |
| 3 | 52449.2916 | 55766.4552 | 45874.9155 | 54468.5706 | 62384.4836 | 3652.7795 |
| 4 | 54358.5933 | 54012.8695 | 87338.5887 | 53544.9321 | 49553.423 | 3552.1352 |
| 5 | 49600.2605 | 61811.6712 | 50350.1404 | 54836.8077 | 45086.7192 | 3340.9987 |
| 6 | 48285.4815 | 59963.429 | 54800.6159 | 57082.6939 | 45312.2075 | 3374.8869 |
| 7 | 52569.2154 | 63116.4975 | 48669.9992 | 69659.4818 | 48858.7937 | 4147.719 |
| 8 | 53581.8588 | 61813.2551 | 51044.7784 | 61603.7786 | 66801.5992 | 3455.9762 |
| 9 | 88992.6898 | 69365.9443 | 96697.2206 | 54521.0627 | 60973.7081 | 3623.2699 |
| 10 | 72884.8775 | 56561.5056 | 56306.098 | 52100.4856 | 52252.0866 | 5690.811 |
| 11 | 58839.4525 | 54106.9063 | 54556.573 | 59241.2946 | 59912.9998 | 3701.5063 |
| 12 | 69740.9894 | 59291.2013 | 58111.3131 | 86575.5122 | 53562.3766 | 3493.712 |
| 13 | 59401.7747 | 66104.4938 | 50840.6825 | 89692.6239 | 51099.6052 | 4130.8601 |
| 14 | 65250.3382 | 72926.032 | 55357.5045 | 56987.2914 | 52560.5107 | 5984.6337 |
| 15 | 56115.7213 | 57962.9498 | 71712.823 | 50812.8561 | 58035.1424 | 3926.0807 |

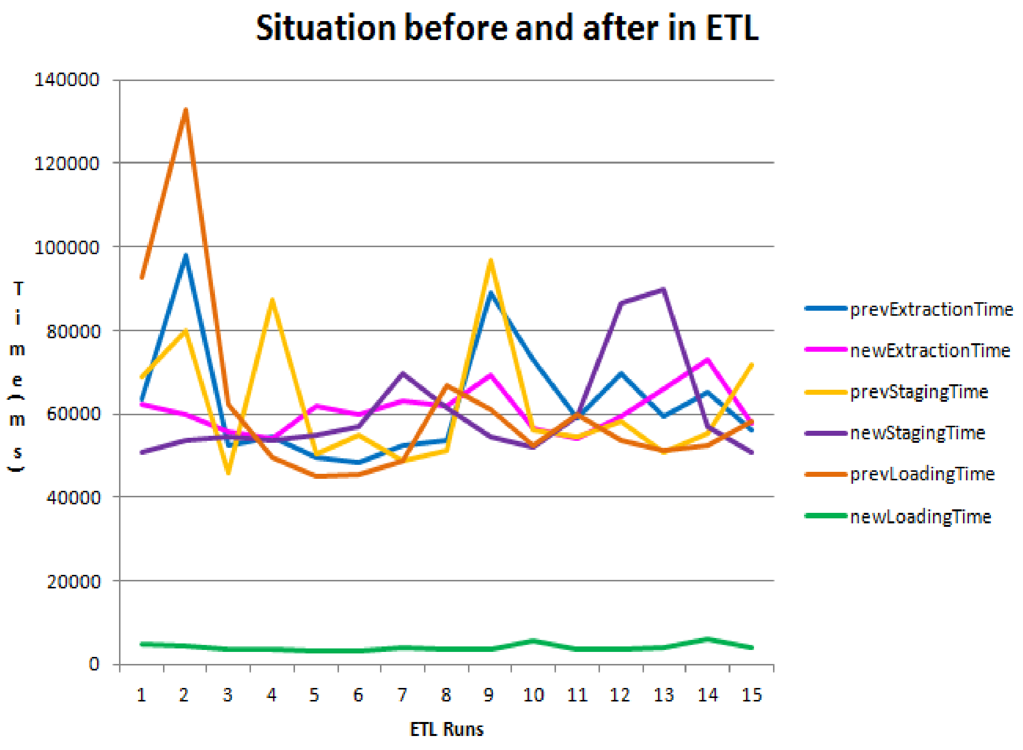
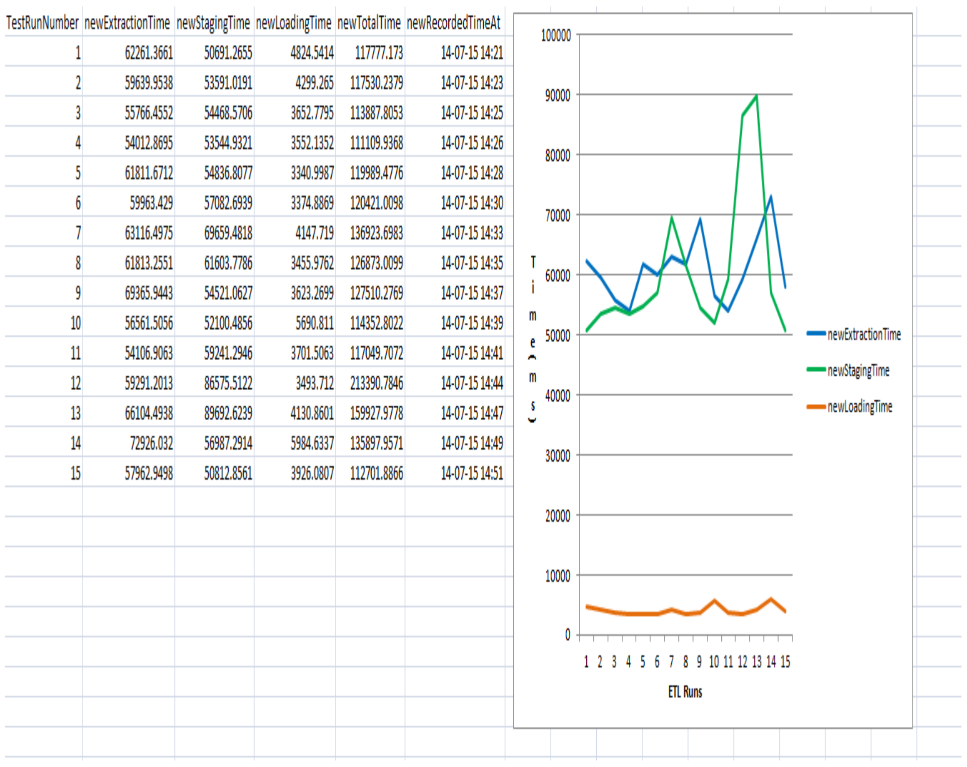


**Figure 4.23 Compare all ETL processes**

The figure 4.23 was an illustration of the current situation after enhancement for all stages of the ETL process.

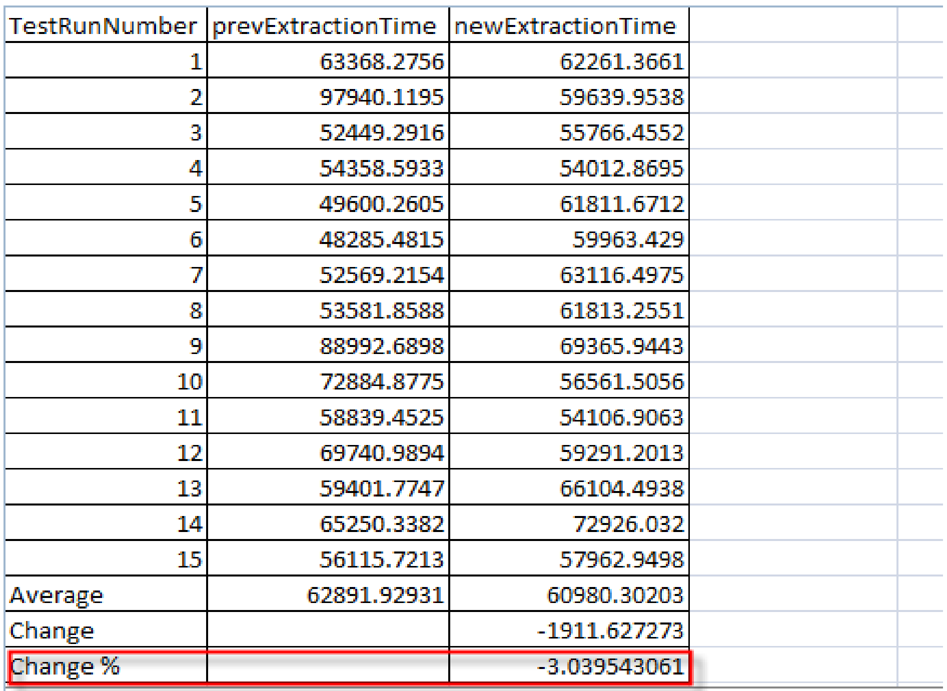


| TestRunNumber | newExtractionTime | newStagingTime | newLoadingTime | newTotalTime | newRecordedTimeAt |
|---|---|---|---|---|---|
| 1 | 62261.3661 | 50691.2655 | 4824.5414 | 117777.173 | 14-07-15 14:21 |
| 2 | 59639.9538 | 53591.0191 | 4299.265 | 117530.2379 | 14-07-15 14:23 |
| 3 | 55766.4552 | 54468.5706 | 3652.7795 | 113887.8053 | 14-07-15 14:25 |
| 4 | 54012.8695 | 53544.9321 | 3552.1352 | 111109.9368 | 14-07-15 14:26 |
| 5 | 61811.6712 | 54836.8077 | 3340.9987 | 119989.4776 | 14-07-15 14:28 |
| 6 | 59963.429 | 57082.6939 | 3374.8869 | 120421.0098 | 14-07-15 14:30 |
| 7 | 63116.4975 | 69659.4818 | 4147.719 | 136923.6983 | 14-07-15 14:33 |
| 8 | 61813.2551 | 61603.7786 | 3455.9762 | 126873.0099 | 14-07-15 14:35 |
| 9 | 69365.9443 | 54521.0627 | 3623.2699 | 127510.2769 | 14-07-15 14:37 |
| 10 | 56561.5056 | 52100.4856 | 5690.811 | 114352.8022 | 14-07-15 14:39 |
| 11 | 54106.9063 | 59241.2946 | 3701.5063 | 117049.7072 | 14-07-15 14:41 |
| 12 | 59291.2013 | 86575.5122 | 3493.712 | 213390.7846 | 14-07-15 14:44 |
| 13 | 66104.4938 | 89692.6239 | 4130.8601 | 159927.9778 | 14-07-15 14:47 |
| 14 | 72926.032 | 56987.2914 | 5984.6337 | 135897.9571 | 14-07-15 14:49 |
| 15 | 57962.9498 | 50812.8561 | 3926.0807 | 112701.8866 | 14-07-15 14:51 |

**Figure 4.24 After enhancement ETL processes**

**Comparing Individual stages of previous and current situations in ETL processes**

The following was an illustration of the individual comparison per stage of the ETL process to have a clear view of the improvement performed. Explanation for each comparison followed for every illustration. The negative sign indicated that it was in the reducing direction thus showing the enhancement had taken place by reducing time taken in execution of a task.



| TestRunNumber | prevExtractionTime | newExtractionTime |
|---|---|---|
| 1 | 63368.2756 | 62261.3661 |
| 2 | 97940.1195 | 59639.9538 |
| 3 | 52449.2916 | 55766.4552 |
| 4 | 54358.5933 | 54012.8695 |
| 5 | 49600.2605 | 61811.6712 |
| 6 | 48285.4815 | 59963.429 |
| 7 | 52569.2154 | 63116.4975 |
| 8 | 53581.8588 | 61813.2551 |
| 9 | 88992.6898 | 69365.9443 |
| 10 | 72884.8775 | 56561.5056 |
| 11 | 58839.4525 | 54106.9063 |
| 12 | 69740.9894 | 59291.2013 |
| 13 | 59401.7747 | 66104.4938 |
| 14 | 65250.3382 | 72926.032 |
| 15 | 56115.7213 | 57962.9498 |
| Average | 62891.92931 | 60980.30203 |
| Change | | -1911.627273 |
| Change % | | -3.039543061 |

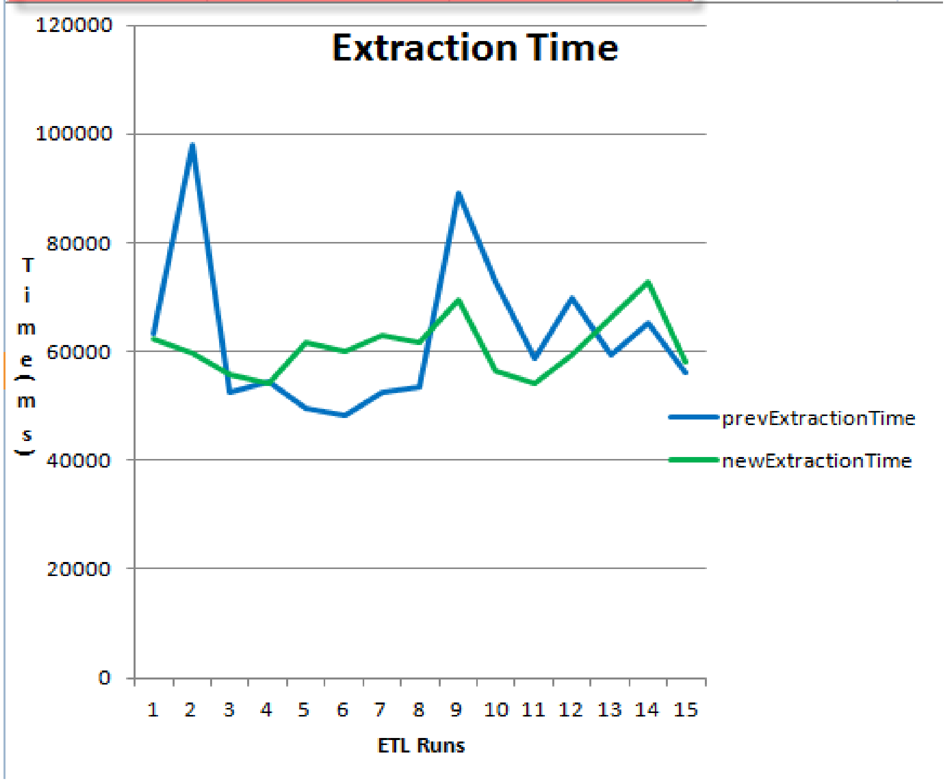
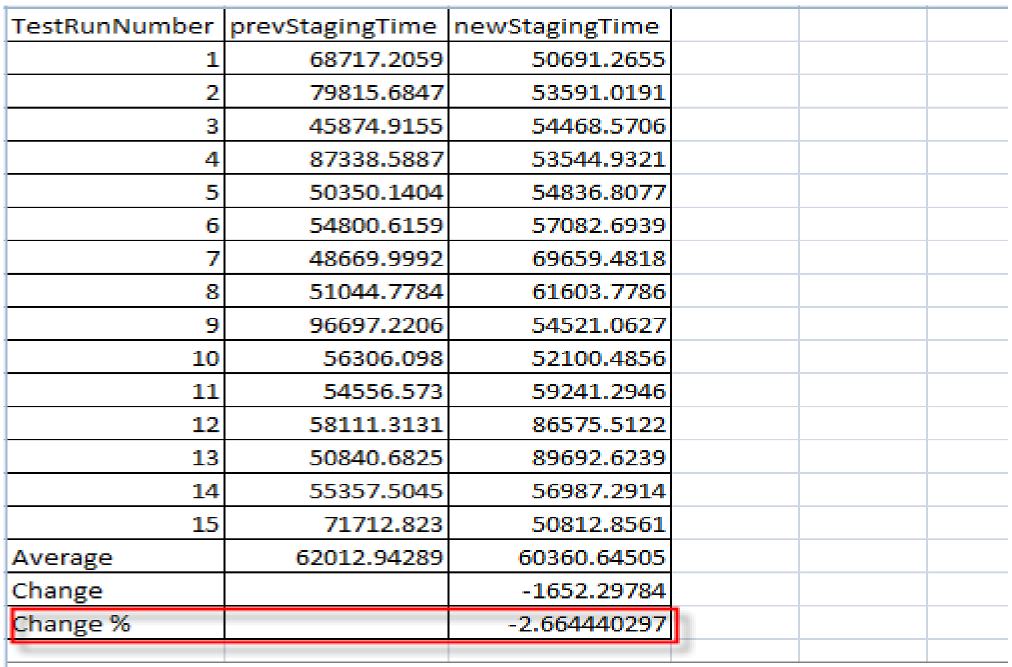


**Figure 4.25 Extraction phase**

The time taken for extraction using deterministic prioritization algorithm had reduced by 3.04% compared to the previous extraction time as shown in figure 4.24.



| TestRunNumber | prevStagingTime | newStagingTime |
|---|---|---|
| 1 | 68717.2059 | 50691.2655 |
| 2 | 79815.6847 | 53591.0191 |
| 3 | 45874.9155 | 54468.5706 |
| 4 | 87338.5887 | 53544.9321 |
| 5 | 50350.1404 | 54836.8077 |
| 6 | 54800.6159 | 57082.6939 |
| 7 | 48669.9992 | 69659.4818 |
| 8 | 51044.7784 | 61603.7786 |
| 9 | 96697.2206 | 54521.0627 |
| 10 | 56306.098 | 52100.4856 |
| 11 | 54556.573 | 59241.2946 |
| 12 | 58111.3131 | 86575.5122 |
| 13 | 50840.6825 | 89692.6239 |
| 14 | 55357.5045 | 56987.2914 |
| 15 | 71712.823 | 50812.8561 |
| Average | 62012.94289 | 60360.64505 |
| Change | | -1652.29784 |
| Change % | | -2.664440297 |

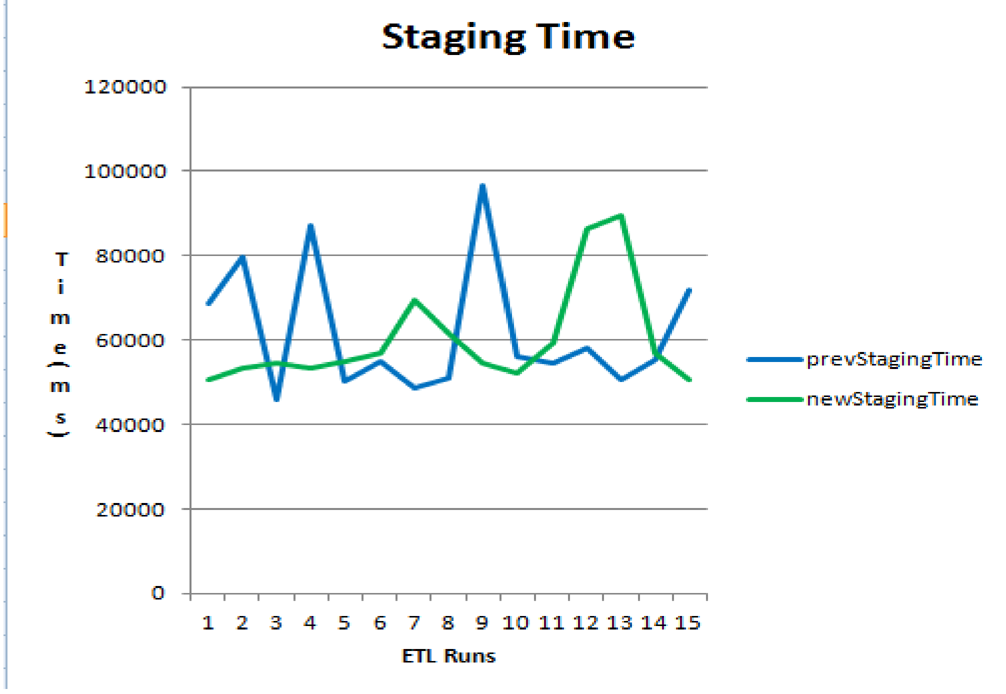
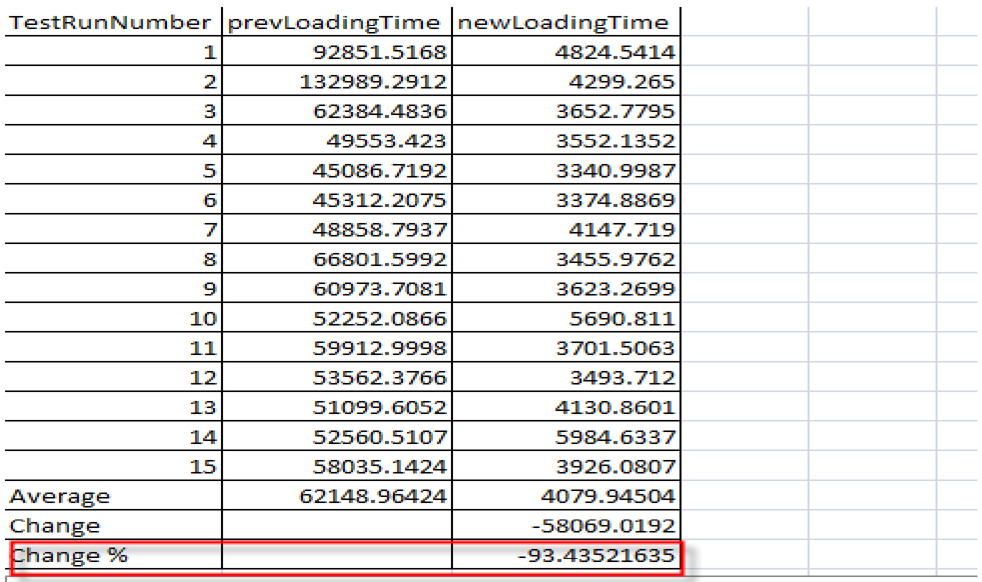


**Figure 4.26 Staging phase**

The time taken in staging area using deterministic prioritization algorithm had reduced by 2.66% compared to the previous extraction time as shown in figure 4.25.



| TestRunNumber | prevLoadingTime | newLoadingTime |
|---|---|---|
| 1 | 92851.5168 | 4824.5414 |
| 2 | 132989.2912 | 4299.265 |
| 3 | 62384.4836 | 3652.7795 |
| 4 | 49553.423 | 3552.1352 |
| 5 | 45086.7192 | 3340.9987 |
| 6 | 45312.2075 | 3374.8869 |
| 7 | 48858.7937 | 4147.719 |
| 8 | 66801.5992 | 3455.9762 |
| 9 | 60973.7081 | 3623.2699 |
| 10 | 52252.0866 | 5690.811 |
| 11 | 59912.9998 | 3701.5063 |
| 12 | 53562.3766 | 3493.712 |
| 13 | 51099.6052 | 4130.8601 |
| 14 | 52560.5107 | 5984.6337 |
| 15 | 58035.1424 | 3926.0807 |
| Average | 62148.96424 | 4079.94504 |
| Change | | -58069.0192 |
| Change % | | -93.43521635 |

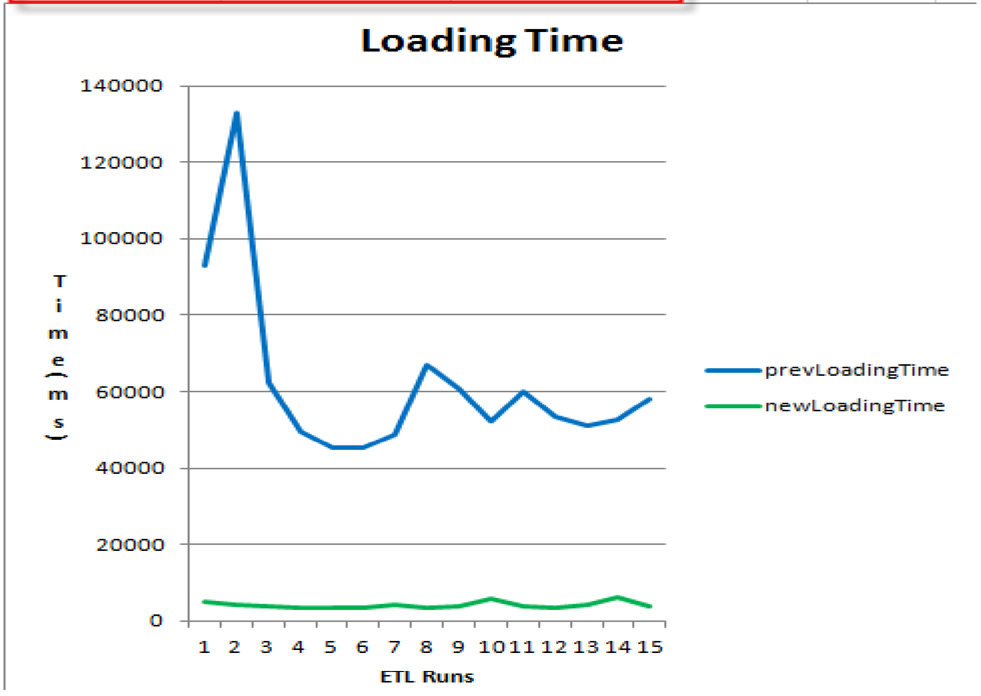


**Figure 4.27 Loading phase**

The time taken in loading using deterministic prioritization algorithm had immensely reduced by 93.44% compared to the previous loading time as shown in figure 4.26. This was attributed to the fact that the number of operations that took place at this stage being minimal and less traffic to the destination targets for the data which were the data warehouses. However, with increased traffic of data especially over the network, the loading time percentage might reduce further.

## 4.5 DISCUSSION

The implementation of the new algorithm of deterministic prioritization showed massive improvement in terms of performance and data quality selections. According to the conceptual framework, the set of procedures outlined the operation of the proposed solution which indicated the different roles carried out within ETL processes. This was advantageous because the complexity of designing new staging frameworks in future was abstracted from the general users of data warehouses. The solution called upon the optimization of the order of query execution by changing the default order of execution that was based on unique identifiers generated at table creation. With massive data requests coming from the business users through query commands executions, the query optimizer became inefficient. The requirement of having data warehouses that were business intelligent, created the roadmap of generating an algorithm that would be cost efficient and add to the existing knowledgebase. The aim was to find the best procedures of collecting the data from the source systems, and performing operations that would optimally load the selected data to the data warehouses. The proposed solution compared relatively to the research by Costel, Marius, Valentina, and Octavian (2014) on the query execution and optimizations in the MSSQL server. They found out that missing indexes in a table resulted in low performing query executions.

With the introduction of clustered indexes, the study was changing the default order of SQL query server's execution plan. This manipulated on the execution plan resulting in better and guaranteed performance as long as data was always readily available for loading in good time. Cecilia and Mihai (2011) agreed to this proposed solution because they indicated that the use of indexes on database queries improved the performance of the whole system. Most of the highlighted processes were sought to be repetitive and were being activated when new data was realized and was ready to be loaded to the data warehouses. The run of the algorithm impacted positively the results obtained after running the tests. It was possible to measure the performance of the algorithm from different stages of ETL processes and compare the performance with previous existing systems. In particular, there was research that had been done by El-Wessimy, Mokhtar, and Hegazy (2013) with tests on scheduling algorithms. They tested several techniques to measure time taken throughout the ETL processes. The observation noted from their research work was that there was reduction in time taken to transfer data in each stage. This compared greatly with the study results on enhancing data handling yielding fast accesses.

The results were demonstrated per every stage and they indicated reduction in the execution times. Within the extraction phase, the results showed an impressive gain of 3.04%. There was more room for improvement on the extraction time if the collection of data was pre-determined and repetitive in every cycle. This would mean reduced procedures of determining which data to collect every time the cycle was performed. The limitation of the test carried in this section was collecting all the data from the source tables so as to maintain integrity of data. The time taken for data manipulation in the staging area reduced by a rate of 2.66%. Considering that the staging area being the working space during data transformation, many operations were involved. The scheduling plans allowed the delivery of data immediately it was staged in the staging tables unlike before where it was put in a queue awaiting batch processing. The declining implementation of batch processing within job execution made the study to dwell in parallelism mode of execution which was supported by an object oriented programming language and ended to be fast.

Finally, the availing of data to the data warehouses and data marts depended on the sequence at which staging was performed. The faster the staging the faster was loading process. Moreover, the loading phase had minimal tasks to perform and this was clear in the results of the study that showed execution time reduction of up to 93.44%. Depending on the architecture and setup of the data warehousing environment, the values of the results performed would change. With proper configurations the impact of implementing the algorithm would match the suggested results or exceed further. With tests on higher performing hardware, the results would be even more impressive.

CHAPTER FIVE

# SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Introduction

Data staging area is part of the data warehouse techniques of improving data handling when migrating it to the data warehouses. The previous researches as reviewed in existing literature had partly covered on the enhancing data staging to realize fast working data warehouses. Most authors focussed on the general data flow targeting quality assurance with few discussing the speed and performance of data staging area. Many advances in hardware have created a pool of resources available to meet current data needs but with the rate at which the data sizes continue to grow, these resources may not be enough to guarantee performance stability in future. In most cases due to incompatible legacy systems, it definitely ends up being compromised. The proposition to enhance data staging was to provide a lee way for future developments in this area that could scale well to new technologies without a complete overhaul of the improvements suggested in this study. The aim was to have the deterministic prioritization algorithm deployed in a production environment to improve on the data handling performance especially on access and retrieval. The benefits were observed mostly in large data sets where retrieval had for long been a bottleneck characterized by network jamming and slow data flow resulting to almost deadlocked systems.

The problem emanated in the existing systems at the data selection point with regard to incremental loading situations. After the initial full data load, the consequent ETL processes were done as increments of the full load. To distinguish the changed data in time, several methods that were used relied on checking the timestamp at which the change occurred and storing this data in the database for consideration in future loads. The new systems, took the advantages of the deterministic prioritization algorithm to provide advancements by prioritizing only the changed data. It became more specialized in operation since the recursive clustered index ran comparison on the previous loads before determining the next data loads. This became a merit in the study by saving computation time and resources especially in fast growing data.

Other than the enhancement that was introduced, it was suggested that continuous monitoring of the database servers be performed regularly so as to identify the trend of data growth. The technological improvements proved within the ETL phases suggested an overall change in the field of data warehouses. The reason for this was that it would be important to compare the impact on performance for the proposed and tested solution over older and newer distributed database systems. One of the roles of this study was to provide a guideline to the data warehouse and business intelligence experts during the design of a staging location within the ETL processes. This purpose was achieved based on positive experimental results analysed in the previous chapter. It highlighted the necessities of pre-determining the expected data loads, ways of prioritizing them and optimizing the query execution plans. The key skills needed were on creation of the clustered indexes and ability to interpret the output generated from the query executions plans.

El-Wessimy, Mokhtar, and Hegazy (2013) indicated that one of the old algorithms being in use in data staging was First In First Out strategy (FIFO). It collected data from the sources as soon it was ready and transferred it to the destination. There was no business intelligence added thus the initial copy was similar to the one at the destination systems. For data to be of great use in the decision making at the data marts, some processing was required where filtering unwanted data helped reduce bandwidth use in transfers. The enhancement came in handy in implementing the business intelligence required for a simple test setup that was ready for the production environment.

This was supported by the fact that a small change in data staging had massive change in the loading phase and the ETL process as a whole. The stages correlated well in the configurations; therefore, there was no biasness since the different parts were not examined separately.

## 5.2 Summary

This research work was started to face a real-life problem that existed in most of organizations whose daily transactions were enormous resulting in large volumes of data. Based on the available resources at these organizations, it became a bottleneck to handle the fast growing data hence the need to invest more time in this research with an aim of solving the mentioned problem. The solution provided was based on a completely new idea that led to the development of the deterministic prioritization algorithm, which was an addition to the body of knowledge (BOK). The main challenge encountered in the research was selecting the best performing algorithms that would solve the problems identified at the research site. The study was to viewed in two perspectives where it considered what data to collect and how to speed-up the process of loading the chosen data. Coming up with a direct solution was difficult and could result in a complex algorithm whose performance could not be measured or tested easily. This resulted in having a solution in two sections; where one section covered on determining the data to collect while the other was concerned on ways of achieving higher priority to the chosen data.

Once the newly developed algorithm was put to test, the previous hurdles were reduced immensely because clarity in the order of operation within the ETL phases was brought out openly through the algorithm procedures. The simplicity of the solution indicated that the solution could be utilized with ease by experts of data warehouse environment to aid in improving the performance of data access.

The ETL implementation procedures when performing data loads varied a lot depending on the instance of load. For the very first time this was ran, a full load was expected because it involved setting up the systems from scratch and also the target system had no structures to hold new data loaded. The tables were created for the first time and this took more time to implement when the data records spanned several thousands of rows. This was not an exception for any new system since it had to undergo through this activity.

In the subsequent runs also referred to as incremental loading, only the modified or changed data in the sources were transferred to the data warehouse. With reference to the performance of the new deterministic prioritization algorithm, the search criteria were highly improved such that changed data was easily fetched and loaded to data warehouses improving the overall system performance.

## 5.3 Conclusions

The data access and retrieval to the data warehouse was immensely improved as shown from the statistics of the tests carried out in this study. Every stage of the ETL process showed some considerable change in particular the data retrieval, processing and loading stages. The deterministic prioritization algorithm selected the appropriate data as per the business logic of organizations which valued data to be of high value and priority in their daily transactions and existence. The new data was synchronized with the already existing data depending on what was modified since the last ETL process run. It was noted that this was a deterministic approach where once the configurations were done on the various stages, the execution of extraction, transformation and loading ended up to be cyclic in nature. This mode of operation continued depending on two factors namely; the operation time of the data warehouse without shutdown and the availability of data to load to the data warehouses.

In cases where there was vast data to process and transfer to the data warehouses, deterministic prioritization algorithm prioritized on the loading process. This was achieved by manipulating on the order of query executions, thereby reordering the flow of data to the data warehouses. The created priorities formed change of schedule for data load. The implementation of the new solution did not add more processing space or utilize batch job processing but rather optimized the SQL query executions.

Since the operations within the extraction and transformation phases of ETL were very explicit depending on the logic behind data selections, more than one operation were intertwined together to support multitasking. The data was subjected to various functions at the same time resulting in concurrent operations during the test of the algorithm. The performance of the algorithm were depicted based n the statistics collected for the affected variables of execution time. The data delivery from one stage to another ended up to be in real time.

All in all the results obtained from the experimental test runs were positive and clearly showed that with implementation of the deterministic prioritization algorithm, there was guaranteed enhancement in the data staging area and ETL processes in general. Improving the staging of data in the data warehouse was a progressive process that couldn’t be concluded in one day. Several considerations had to be made with respect to the other bounding processes in the ETL anatomy so as to maintain data consistency and high quality data availed to the users.

## 5.4 Recommendations

With regard to the findings of this research it is highly recommended to have the implementation of deterministic prioritization algorithm in existing and new data staging frameworks at architectural level. The impact has been seen to enhance the data retrieval and access from the sources all the way to the destinations. It will assist in building business intelligent warehouses where it is possible to pre-determine which data need to be loaded to the data warehouse or data marts based on the available columns and allocating priority indexes to those columns.

The new deterministic prioritization algorithm is flexible to implement in the current ETL and data staging frameworks for those organizations that have customized data warehouse configurations. This has been contributed to the fact that the solution only touches on the backend where the data fetching processes exist. The abstraction of the strategy has reduced complexity of the interlocking relationships among tables and this has enabled compatibility to any data formats. All the data from organizations can now be matched after conversion using the Electronic Data Interchange (EDI) tools. The conversion is determined by the source and target systems. They can easily be incorporated in the data warehouse system. The knowledge base required for ETL processes is limited by the availability of resources and the users' training.

The common behavior of data warehouse practitioners is to periodically extract data from the transaction applications e.g. OLTP and storing it in dedicated servers that have batch programs to transfer to destination servers then to data warehouse. This operation is overkill since the extraction of data from the dedicated servers could have been the base of selecting the important data to load. This could reduce redundancies with a big margin. The enhancement algorithm was implemented using the syntax of visual studio C# programming to complement the semantics of data staging in transformation level. The experiment setup was on a localhost workstation that played as a client and server. Implementing the deterministic prioritization algorithms in an expansive networking environment e.g. cloud computing and in relation to SAP HANA databases is a possible investigation for future work.

# REFERENCES

Abbasi, H., Wolf, M., Eisenhauer, G., Klasky, S., Schwan, K., and Zheng, F. (2010). *Datastager: scalable data staging services for petascale applications. Cluster Computing*, *13*(3), 277-290.

Akkaoui, Z. E., Munoz, E. Z., and Trujillo .J. A. (2011). *Model-Driven Framework for ETL Process Development*. *In Proceedings of the international workshop on Data Warehousing and OLAP*. pp. 45–52 Glasgow, Scotland, UK.

Aksoy, D., Franklin, M. J., and Zdonik, S. (2001). *Data staging for on-demand broadcast.* In VLDB (Vol. 1, pp. 571-580).

Bamnote, G.R., and Agrawal, S.S. (2013).Introduction to Query Processing and Optimization. *International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 7. ISSN: 2277 128X*

Bézivin, J. (2005). *Survey report on the unification power of models. Software and System Modeling*, 4(2):171–188

Cecilia, C., and Mihai, G. (2011). Increasing Database Performance using Indexes. *Database Systems Journal vol. II, no. 2/2011*. Economic Informatics Department, Academy of Economic Studies Bucharest, Romania.

Connolly, T and Begg, C. (2005). *Database Systems: A practical Approach to Design, Implementation, and Management* (2nd ed). Harlow, England, Addison Wesley: Pearson Education Limited.

Corlatan, Mariaus, L.,Valentina,L.,Octavian,T.(2014). Query Optimization Techniques in Microsoft SQL Server. *Database Systems Journal vol. V, no. 2/2014*. PETRICICĂ University of Economic Studies, Bucharest, Romania

Costel, G.C., Marius, M. L., Valentina, L., and Octavian, T. P. (2014). Query Optimization Techniques in Microsoft SQL Server. *Database Systems Journal vol. V, no. 2/2014*. University of Economic Studies, Bucharest, Romania.

Da Silva, M.S., Times, V.C., and Kwakye, M.M. (2012). *Journal of Information and Data Management.*3 (3).

Eckerson, W., and White, C. (2003). *Evaluating ETL and data integration platforms*. *Seattle: The DW Institute*.

El-Wessimy, M., Mokhtar, H. M., and Hegazy, O. (2013). Enhancement techniques for data warehouse staging area. *International Journal of Data Mining & Knowledge Management Process*, *3*(6).

Erhard, R., and Hong, H.D. (2000). *Data Cleaning*: Problems and Current Approaches. *Journal IEEE Data Eng. Bull.23 (4), 3-13.*

Grant, F. (2012). SQL Server Execution Plans. 2nd Ed. Simple Talk Publishing. ISBN: 978-1-906434-92-2.

Firestone, J. M. (1998). Dimensional modeling and ER modeling in the data warehouse. *White Paper No, Eight June, 22*.

Flinn, J., Sinnamohideen, S., Tolia, N., and Satyanarayanan, M. (2003*). Data Staging on Untrusted Surrogates. In FAST (Vol. 3, pp. 15-28)*.

IBM. (2016). Data staging. Retrieved from http://www.ibm.com/support/knowledgecenter/SSFJTW_5.1.0/com.ibm.cluster.loadl.v5r1.load100.doc/am2ug_datastaging.htm

Inmon, W. H. (2002). *Building the Data Warehouse*. New York: Wiley.

Inmon, W. H. (2005). *Building the data warehouse*. New York: John Wiley and Sons.

Kimball, R., and Caserta, J. (2004). *The data warehouse ETL toolkit*. New York: John Wiley and Sons.

Kimball, R. and Margy, R. (2002). The Data WarehouseToolkit: The Complete Guide to Dimensional Modeling. Second Edition. New York: John Wiley and Sons,

Kimball, R., Reeves, L., Ross, M., and Thornthwaite, W. (2008). *The Data Warehouse Lifecycle Toolkit*, (2nd ed). *Practical Techniques for Building Data Warehouse and Business Intelligence Systems*.

Muller, P. A., Studer, P., Fondement, F., and Bézivin, J. (2005). *Platform independent Web application modeling and development with Netsilon. Software & Systems Modeling,* 4(4), 424-442.

Nanda, R. (2015). Review of Query Processing Techniques of Cloud Databases. *Suresh Gyan Vihar University Journal of Engineering & Technology*. Vol . 1, Issue 2, pp.12-16

Per-Åke, L., Cipri, C., Campbell, F., Eric, N. H., Mostafa, M., Michal, N., Vassilis, P., Susan, L. P., Srikumar, R., Remus, R., and Mayukh, S. (2013). *Enhancements to SQL Server Column Stores.* ACM 978-1 -4503-2037-5/13/06. New York, USA.

Rogers, D. (2010). Data Warehousing Architecture - *Designing the Data Staging Area*. Retrieved from http://www.databasejournal.com/sqletc/article.php/3888696/Data-Warehousing-Architecture-Designing-the-Data-Staging-Area.htm

Russom, P. (2012). BI Experts: Big Data and Your Data Warehouse's Data Staging Area. *TDWI Best Practices Report, Fourth Quarter*. Retrieved from http://tdwi.org/articles/2012/07/10/big-data-staging-area.aspx

SAP AG. (2002). Business Information Warehouse – *Data Staging* Retrieved from http://scn.sap.com/docs/DOC-8100.

Stephen, B. (2013). Staging, Statistics and Common Sense: *Oracle Statistics Maintenance Strategy in an ETL environment* Retrieved from http://www.seethehippo.com/

Vassiliadis, P. (2009). A survey of Extract–transform–*Load technology*. *International Journal of Data Warehousing and Mining (IJDWM)*, *5*(3), 1-27.

Zineb, A., Esteban, Z., Jose-Norberto.M., and Juan, T. (2011). *A Model-Driven Framework for ETL Process Development*. In DOLAP 11 Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP Pages 45-52. ACM New York,, USA. ISBN: 978-1-4503-0963-9