Abstract – Many organizations rely heavily on their data warehouse for enterprise level decision making. Since a data warehouse pulls data from various heterogeneous sources, the extract and load processes play very important roles in building an optimized data warehouse solution. In this paper we compare Extract, Transform and Load (ETL) approach and Extract, Load and Transform (E-LT) approach for loading data into a data warehouse. We will gauge the performance difference for both ETL and E-LT approaches.

KEYWORDS: ETL, E-LT, DATA WAREHOUSE, PERFORMANCE MEASUREMENTS

INTRODUCTION

In today’s world everyone including commercial companies, non profit organizations,
universities and government agencies are spending millions of dollars on building data warehouse solutions for various reasons like forecasting, profitability, process improvement, reporting etc. Since a data warehouse is a central repository for reporting and analytical purposes, the extraction of data from various heterogeneous sources and loading it into a target data warehouse plays a very important role in building an optimized data warehouse. There are different approaches that can be used for extracting data from various heterogeneous source systems and loading this data into an Enterprise Data Warehouse. The aim of this paper is to compare traditional Extract, Transform and Load (ETL) approach vs. Extract, Load and Transform (E-LT) approach for loading data into data warehouse.

In this paper, we have also performed and compared performance benchmark tests between ETL and E-LT approaches.

**RELATED WORK**

The research presented in [1] covered the Extract, Transform and Load (ETL) approach for building a data warehouse solution. The old materialized view approach is discussed and compared to show how the ETL approach is better. It further focused on how to build a data warehouse solution by optimizing the ETL approach.

In contrast, [2] demonstrated how to build a successful data warehouse solution using the Extract, Load and Transform (E-LT) approach. It presented the advantages in terms of cost and time in building the E-LT solution over the traditional ETL approach.

The next generation of building data warehouse solutions based on Corporate Information
Factory is discussed in [3]. It also presents the importance of the ETL approach in building the proposed solution. The ETL approach and its key features like metadata, audit trails and data quality are well presented in [3].

Both the ETL and the E-LT approaches are thoroughly discussed in [4], where the pros and the cons of both approaches are also presented.

TECHNICAL BACKGROUND

Extract Transform and Load (ETL) Approach:

The traditional Extract, Transform and Load (ETL) approach operates by first extracting data from various heterogeneous sources like databases, flat files, ERP systems, CRM systems and main frame systems [2]. Different business rules are applied on the data extracted from various sources by the proprietary, middle-tier ETL engine [2]. This massaged and transformed data is finally loaded into the target data warehouse system or integration system. The process is often designed from the end backwards, in that the required output is designed first. In so doing, this informs exactly what data is required from the source. The routines designed and developed to implement the process are written specifically for the purpose of achieving the desired output, and only the data required for the output is included in the extraction process [4].

Business rules that define how aggregations are achieved and the relationships between the various entities in both the source and target, are designed and therefore coded into the routines that implement the ETL process. This approach leads to tight dependencies in the routines at each stage of the process [4].
**Strengths:**

- Designing from the output backwards ensures that only data relevant to the solution is extracted and processed, potentially reducing development, extract, and processing overhead, thus reducing the time to build the solution. [4]
- Due to the targeted nature of the load process, the data warehouse contains only data relevant to the presentation. [4]
- ETL can perform more complex operations in single data flow diagrams (data maps).

**Weaknesses:**

- The data transformation step of the ETL approach is the most compute-intensive and is performed entirely by the proprietary ETL engine on a dedicated server. This increases the job’s runtime as well as more hardware costs.
- The ETL engine performs data transformations and sometimes data quality checks on a row-by-row basis. This can easily become the bottleneck in the overall process [4].
• The data is moved over the network twice – once between sources and the ETL server and again between the ETL server and the target data warehouse [2].

• Since only the relevant data in captured in the data warehouse, data needed for any future requirements might not exists in the data warehouse and will need to be added to the ETL routines. Due to nature of tight dependency between the routines developed, this often leads to a need for fundamental re-design and development. As a result this increases the time and costs involved.

**Extract, Load and Transform (E-LT) Approach:**

The Extract, Load and Transform (E-LT) approach incorporates both the manual coding as well as leveraging ETL approach in the same solution [2]. The data is extracted in the same way as in the ETL approach. This data extracted from different sources is now loaded into the target data warehouse system. Once loaded, the transformations and business logics are applied using native SQL drivers. This helps in saving cost and extra processing needed by ETL middle-tier. Thus the ELT approach leverages the power of the Relational Database Management System (RDBMS) engine.
Figure 2 (from [4])

Strengths:

- In general, in an E-LT implementation all data from the sources are loaded into the warehouse as part of the extract and load process. This, combined with the isolation of the transformation process, means that future requirements can easily be incorporated into the warehouse structure [4].
- Once the data is loaded on the target platform, all transformations/business rules are placed on the RDBMS engine. This reduces network congestion.
- Since no extra server, technology, or skill requirement comes into play, the E-LT architecture provides optimal performance and scalability and easing the management of the integration infrastructure [2].

Weaknesses:

- There are fewer E-LT tools available in the market [4].
- This approach works better for the larger volume of data set. As the volume of data increases, so does the performance.
- Since this approach is relatively newer, there are very fewer developers who have a good understanding of the underlying principles.

EXPERIMENTS

Experimental Setup:

Our experimental system consists of Informatica Power Center 8.6.1 Hot Fix 9 server running
on a Sun SPARC M9000 series machine with 8 GB RAM. Both the source and the target database are running on Teradata V2R6.2.2.

**Data Set:**

We have used the sample data set provided by T-Mobile’s EDW team ([www.t-mobile.com](http://www.t-mobile.com)). We performed three different sets of experiments to compare between the ETL and the E-LT approach.

In the first set of experiments which performed comparison between the ETL and the Full Pushdown E-LT, 32044 records were read from the write_off_stg_delta table and loaded into the write_off table. This data set was only used for experimental purposes.

In the second set of experiments which performed comparison between the ETL and the Target Push down E-LT, 154271 records were read from the tax_extract_stg_delta table and 462813 were loaded into the tax_extract_tst table.

In the third set of experiments which performed comparison between the ETL and the Target Push down E-LT, 154271 records were read from the tax_extract_stg_delta table and 154271 records were loaded into the tax_extract table.

**Procedures:**

We performed our experiments on the development server used in the T-Mobile EDW department. The earlier versions of Informatica PowerCenter tool could only perform ETL tasks. With the greater increase in use of E-LT approach, Informatica Corp also introduced E-LT processing (called Push down Optimization) within the PowerCenter tool. Pushdown optimization
allows you to “push” PowerCenter transformation logic to the Teradata source or target database. The PowerCenter Integration Service translates the transformation logic into SQL queries and sends the SQL queries to the database. The Teradata database executes the SQL queries to process the mapping logic. The Integration Service processes any mapping logic it cannot push to the database. Therefore all the experiments were conducted using Informatica PowerCenter 8.6.1 HF9 advanced version which enabled us to run E-LT processing.

There are different types of pushdown techniques to perform the ETL and the E-LT processing:

- None: If the Pushdown option isn’t selected, the Informatica PowerCenter will work purely as an ETL tool. All the business logic processing is handled by the ETL engine.
- Source-side: The PowerCenter analyzes the mapping from the source to the target or until it reaches a downstream transformation it cannot push to the database. It pushes as much transformation logic as possible to the source database.
- Target-side: The Informatica PowerCenter analyzes the mapping from the target back to the source or until it reaches an upstream transformation it cannot push to the database. It pushes as much transformation logic as possible to the target database.
- Full: The Informatica PowerCenter attempts to push all transformation logic to the target database. If all transformation logic cannot be pushed to the database, it performs both source-side and target-side pushdown optimization. Both the source and the target need to be on the same database for full pushdown.

The ETL code and E-LT code was run against the same test data and same code base to study and compare the following performance tests:

- **Data Load Throughput**: Data throughput is very important for optimized data
warehouse applications since it usually involves large volumes of data (millions to billions of rows). Data load throughput is calculated as number of rows loaded per second in the target database. Experiments were run against the ETL code as well as E-LT code to compare the data load throughputs against the same dataset.

- **Memory Usage:** Memory consumption is really important in a process which involves applying transformations and loading of large volumes of data (gigabytes in size). The experiments involved tests to monitor the memory consumption during the run time for both the ETL and the E-LT workflows.

- **CPU Utilization:** Both the ETL and the E-LT code was executed against the same data set in the same environment to compare the CPU utilization.

**Experiment 1: ETL vs. E-LT with Full Pushdown**

The first set of experiments were run to compare and calculate performance differences between the ETL code using Informatica PC 8.6.1 HF9 and the E-LT code using Full Pushdown Optimization option of Informatica PC 8.6.1 HF9. For the first set of experiments we loaded 32044 rows from the source view to the target table in Teradata test database. For the first set of experiments the source data was read from a teradata table (tax_write_off_stg_delta) and the data was loaded into the target table (write_off). This data set was only used for experimental purposes. Since the target table was empty, therefore this test was performed as an initial load test.

**ETL job:** Informatica Server handled all the code processing and generated its own internal SQL.
**E-LT Job (Full Pushdown):** In the full pushdown, Informatica Server pushed all the code to RDBMS engine and Informatica Server worked purely as an E-LT tool. Teradata database engine generates its own native Sql.
Results: Approximately 18 Times more performance gain using E-LT Full Pushdown.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Data Read (rows)</th>
<th>Data Load (rows)</th>
<th>Runtime (second)</th>
<th>Throughput (rows/sec)</th>
<th>Memory</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETL</td>
<td>32044</td>
<td>32044</td>
<td>110</td>
<td>291</td>
<td>DB/ETL Server</td>
<td>Database/ETL Engine</td>
</tr>
<tr>
<td>E-LT Full</td>
<td>32044</td>
<td>32044</td>
<td>6</td>
<td>5340</td>
<td>Only Database Memory</td>
<td>Only Database CPU</td>
</tr>
</tbody>
</table>

Table 1

Experiment 2: ETL vs. E-LT with Target-side Pushdown:
The second set of experiments were run to compare and calculate performance differences between the ETL code using Informatica PC 8.6.1 HF9 and the E-LT code using Target Pushdown Optimization option of Informatica PC 8.6.1 HF9. In the second set of experiments, 154271 records were read from the tax_extract_stg_delta table and 462813 were loaded into the tax_extract_tst table. This data set was only used for experimental purposes. Since the target table was empty, therefore this test was performed as an initial load test.

**ETL Job:** Informatica Server handled all the code processing and generated its own internal SQL.
**E-LT Job (Target Pushdown):** Informatica Server pushed code processing on target database to the Teradata database engine.
Results: No performance gains as both the ETL and the E-LT jobs used ETL Engine as well as Database Engine resources (CPU and Memory).

<table>
<thead>
<tr>
<th>Approach</th>
<th>Data Read (rows)</th>
<th>Data Load (rows)</th>
<th>Runtime (second)</th>
<th>Throughput (rows/sec)</th>
<th>Memory</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETL</td>
<td>154271</td>
<td>462813</td>
<td>1879</td>
<td>246</td>
<td>ETL/ Partial DB</td>
<td>ETL/ Partial DB</td>
</tr>
<tr>
<td>E-LT Target</td>
<td>154271</td>
<td>462813</td>
<td>1887</td>
<td>245.26</td>
<td>DB/ E-LT Server</td>
<td>DB/ E-LT Server</td>
</tr>
</tbody>
</table>

Table 2

Experiment 3: ETL vs. E-LT with Source-side Pushdown:
The third set of experiments were run to compare and calculate performance differences between the ETL code using Informatica PC 8.6.1 HF9 and the E-LT code using Target Pushdown Optimization option of Informatica PC 8.6.1 HF9. In the third set of experiments which performed comparison between the ETL and the Source Push down E-LT, 154271 records were read from the tax_extract_stg_delta table and 154271 records were loaded into the tax_extract table. This data set was only used for experimental purposes. Since the target table was empty, therefore this test was performed as an initial load test.

**ETL job:** Informatica Server handled all the code processing and generated its own internal SQL.

![Image of ETL Code](image)

**Figure 7 ETL Code (from [6])**

**E-LT Job (Source Pushdown):** Informatica Server pushed down code processing on source side to the Teradata database engine. While performing the experiments, we found out that we
couldn’t run job if we used Informatica sequence generator transformation in the pushdown option. Therefore we had to drop sequence generator transformation from the mapping and created teradata level sequence generator to create unique rows. The sequence generator work only for Oracle and DB2 databases.

Results: No performance gains as both the ETL and the E-LT jobs used ETL Engine as well as Database Engine resources (CPU and Memory).

<table>
<thead>
<tr>
<th>Approach</th>
<th>Data Read (rows)</th>
<th>Data Load (rows)</th>
<th>Runtime (second)</th>
<th>Throughput (rows/sec)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RESULTS

Three different set of experiments were performed to compare the performance differences between the ETL and the three E-LT approaches (Full Pushdown, Target Pushdown and Source Pushdown) with different data sets, in terms of throughput, memory and CPU utilization. We found out that significant performance gains were obtained when full E-LT pushdown was used to run a data warehouse job where both the source and the target tables resided on the same database platform. We also observed that there was no performance difference in terms of running a job to load data into data warehouse tables if complete pushdown powers of E-LT jobs were not used. In those jobs as presented in experiments 2 and 3, both the ETL approach and E-LT approach used concurrent utilization of both the ETL Server and the Database Server CPU and memory. We also discovered out that although full E-LT approach performed better in our experiments, there were some cases where E-LT did not work at all. In building a Real Time Data warehouse, where data is fetched continuously from the source systems, only ETL approach works.

CONCLUSION AND FUTURE WORK

In this paper, we discussed both the ETL and the E-LT approach for loading data into a data
warehouse. The strengths and the weaknesses of both the approaches are presented in this paper. Various experiments were performed on the same data set in the same environment to show performance differences between both the approaches. The paper also demonstrated which approach is more favorable than other and under what scenarios. The future work which proposes use of hybrid approach (which is combination of ETL and E-LT approach) also known as ETL-T (ETL and ELT hybrid) is also presented in this paper.

ACKNOWLEDGEMENT

The experiments performed in this research paper were performed against the sample data set provided by T-Mobile’s EDW team. All the experiments were performed on the test server of T-Mobile’s EDW laboratory. Therefore I am thankful to the manager of T-Mobile’s ICC team. I am also thankful to my wife, Sapna Wason for her constant support and keeping me motivated throughout my research work.

REFERENCES


[7] Teradata, Available at: [www.teradata.com](http://www.teradata.com)
