Optimization of Data Warehouse Design and Architecture

Wahab Munir

Master of Science Thesis
Stockholm, Sweden 2011

TRITA-ICT-EX-2011:126
Master Thesis

Optimization of Data Warehouse Design and Architecture

Wahab Munir
Master’s in Software Engineering of Distributed Systems
Kungliga Tekniska Hogsköla
Stockholm
Sweden
Abstract

A huge number of SCANIA trucks and busses are running on the roads. Unlike the trucks and buses of the past, they are hi-tech vehicles, carrying a lot of technical and operational information that pertains to different aspects like load statistics, driving time, engine-speed over time and much more.

This information is fed into an analysis system where it is organized to support analytical questions. Over the period of time this system has become overloaded and needs to be optimized. There are a number of areas identified that can be considered for improvement. However, it is not possible to analyze the whole system within the given constraints. A subset is picked which has been thought of to be sufficient for the purpose of the thesis.

The system takes a lot of time to load new data. Data loading is not incremental. There is a lot of redundancy in the storage structure. Query execution takes a lot of time in some parts of the database.

The methods chosen for this thesis includes data warehouse design and architecture analysis, end user queries review, and code analysis.

A potential solution is presented to reduce the storage space requirements and maintenance time taken by the databases. This is achieved by presenting a solution to reduce the number of databases maintained in parallel and contains duplicated data.

Some optimizations have been made in the storage structure and design to improve the query processing time for the end users.

An example incremental loading strategy is also implemented to demonstrate the working and idea. This helps in the reduction of loading time.

Moreover, An investigation has been made into a commercially available Data warehouse management System. The investigation is
mostly based on hardware architecture and how it can contributes to better performance. This portion is only theoretical.

Based on the analysis recommendations are made regarding the architecture and design of the data warehouse.
Acknowledgements

This master thesis was carried out at Scania Technical Center in Sodertalje, Sweden. This is a part of my master’s degree in software engineering of distributed system at The Royal institute of technology (KTH) in Stockholm, Sweden.

I would like to thank my supervisor Ann Lindqvist, who has shown complete trust in my abilities. She has always helped me with the problems I faced from time to time related to the system and specially domain understanding. Her kind and helping words and full of life attitude has always been an inspiration. Moreover I would like to thank Ms. Alvarado Nadesska. She has been so kind with her responses for my unplanned and ad hoc questions that I have been bombarding her with for the last 4 months.

Mr. Johan Montelius is my supervisor from the university. I would like to pay my gratitude for the grace he has shown in accepting me as his student for the master thesis.

I would further like to thank Mr. Joakim Drott who was going to be my supervisor but he could not continue due to some unavoidable circumstances. His technical expertise and strong grasp of the system has been of great help until he left.

In the end I would also like to thank Mr. Magnus Eriksson and everyone else at RESD for supporting and helping me out during this time.

Wahab
Table of contents

1 Introduction ........................................................................................................... - 1 -
  1.1 SCANIA ............................................................................................................ - 1 -
  1.2 Data Usage ....................................................................................................... - 1 -
  1.3 Problem Statement .............................................................................................. - 2 -
  1.4 Methodology ...................................................................................................... - 4 -
2 Data warehouse Architecture and Designing ...................................................... - 5 -
  2.1 Parallel Processing Hardware ........................................................................... - 5 -
  2.2 Data Warehouse Management Software .......................................................... - 8 -
3 Teradata .................................................................................................................. - 16 -
  3.1 Hardware Architecture ....................................................................................... - 16 -
  3.2 Query Execution ................................................................................................ - 18 -
  3.3 Rows Partitioning ............................................................................................... - 19 -
  3.4 Linear Scaling .................................................................................................... - 19 -
  3.5 Use of BYNET ................................................................................................... - 20 -
4 System Analysis .................................................................................................... - 22 -
  4.1 Data Flow .......................................................................................................... - 22 -
  4.2 Hardware Architecture ....................................................................................... - 23 -
  4.3 Clean and Unclean Databases ......................................................................... - 23 -
  4.4 Data warehouse Design ..................................................................................... - 24 -
  4.5 Loading Strategy ............................................................................................... - 24 -
5 Proposed and Tested Optimizations ................................................................. - 26 -
  5.1 Database Space Reduction .............................................................................. - 26 -
  5.2 Incremental Loading .......................................................................................... - 29 -
  5.3 Partitioning ....................................................................................................... - 30 -
6 Future Work ......................................................................................................... - 32 -
7 Discussion & Conclusion .................................................................................... - 33 -
9 Bibliography ........................................................................................................ - 34 -
11 Terminology ...................................................................................................... - 35 -
Appendix A ............................................................................................................ - 36 -
What is SSIS? ......................................................................................................... - 36 -
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current System Architecture</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Compressed and uncompressed data transfer</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Time vs. Disk Space</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Star Schema Example</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>BTree Index Structure (Green highlights one of several search paths)</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Teradata High level Architecture</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>Data Access Request flow path.</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Data movement to new node</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Shifted Time Space Graph</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>Dataflow Architecture</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>Current System Star Schema Structure</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>Data duplication in different databases</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>Data warehouse cleaning process</td>
<td>27</td>
</tr>
<tr>
<td>14</td>
<td>Accessing clean and unclean data.</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>SSIS Package for incremental loading of data.</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>Partitioning: Row counts for Vector names</td>
<td>31</td>
</tr>
</tbody>
</table>
The document contains the following sections:

Introduction section includes some information about Scania and the importance of the data it uses. It presents the current dataflow architecture and physical design of the data warehouse. It mentions the methods used in the thesis study including literature study and others.

Data warehouse architecture and designing section include the theoretical information about good architecture and design practices.

Teradata section discusses the technical architecture and working of Teradata as a data warehouse management system. How the data is kept in a distributed fashion and how the query processing works.

System Analysis section includes the information about current system and its working strategy.

Proposed and tested optimizations section includes the information about the possible improvements that can be made in the system.

Discussion & Conclusion section presents the summary of the results achieved. It includes some recommendations that can improve Database usage and productivity.

Future work includes the areas that can be further investigated.
1 Introduction

1.1 SCANIA

Scania is one of the world’s leading brands of heavy transport vehicles, industrial and marine Engines. It has been successfully operating for over 100 years now. Its success over the years is based on the fact that it has been constantly investing on the improvement of its products through Research & Development.

Scania has presence in more than 100 countries. It has more than 34,000 employees, out of those around 2400 are involved in R & D operations. Production units are mainly located in Europe and Latin America.

Scania has a modular product system which allows high customization at low cost. It also makes it possible to develop vehicles tailored for specific transport needs.

This leadership is maintained with the help of the constant usage of cutting edge technology and R & D. Efficient data accumulation and analysis is a key to timely and correct decision making.

1.2 Data Usage

Unlike the trucks that were manufactured 100 years ago, today’s trucks carry a lot of information. This information ranges from engine revolutions per second to drivers drive timings, number of gear changes, fuel consumption and the list goes on. The data collected is a valuable input for the improvement and optimization of future generation of products. It can be used in analyzing the usage of different components.

Data is also generated for the Diagnostic Trouble Codes (DTCs) produced in the trucks. This data is useful in analyzing faulty components.

Now, if we are talking about the data flowing in from more than a thousand of trucks or buses at irregular intervals, It becomes quite a task to structure and organize that data in order to make it readily accessible.

The data can be used for different purposes ranging from operational analysis to strategic planning, data mining applications and statistical analysis. This is only possible if structure and design of data storage allows it.
1.3 Problem Statement

The Business intelligence platform or the data warehouse being used with operational data for statistical and operational analysis was developed a few years ago to cater for the growing need of analysis and increased amount of data relayed from the vehicles to the central system.

1.3.1 Current System

![Current System Architecture](image)

As shown in the Figure 1 the current system consists of multi stage databases and data warehouses.

- Initially the data is accessed from multiple transactional systems to a staging area through a SQL Server Integration Services (SSIS) package. *(See Appendix A)*
- From the staging area, data is routed to two different databases one of them is clean and the other one is not. Cleaning is performed in the staging area when data has already been transferred to unclean database. After completing the cleaning process the data is transferred to clean database.
- From each of the databases mentioned in the above step, data is transferred to data warehouses designed for analytic query processing.
- Once the data is in the data warehouse it can be accessed by the end users, who run ad hoc queries by using different presentation and data processing client applications.
The reason behind keeping both clean and unclean databases is due to the fact that the end users often require to query one or the other.

Microsoft SQL Server is mainly used as a database management system. For transformation and cleaning SQL Server procedures are mainly used.

1.3.2 Potential Improvement Areas

1. The SSIS package mentioned in the first step (current system) always performs the full load of the system. Full load means that everything is deleted from the staging database which is the destination for the SSIS package. Consequently the data which came into the system a year ago and was never updated again will be deleted and reinserted, apparently for no reason. This should be addressed somehow.

2. The processes that moves data to clean and unclean databases from staging area involves heavy cleaning and transformation operations. All the business logic is implemented in the procedures that involve cross referencing databases. This makes it hard to maintain. Any change in the source system requires corresponding changes in the transformation logic.

3. There are 5 databases maintained in all for the complete flow to work as a coherent system. There is a lot of duplicated data stored at different points. An obvious example is the case of clean and un clean databases. Unclean is a super set of cleaned with much redundancy.

4. Index usage in the analysis databases can be revisited and redesigned. The large tables can be partitioned with well chosen partition functions.

5. Investigate architectural differences and performance characteristics of different Database Management System (DBMS) tools other than those currently used.

6. Revisit column level duplication in the tables inside the data warehouse.

7. Investigate the possibility and impact of partitioning on the basis of the columns frequently appearing in the filtering criteria of queries.

In addition to the above mentioned areas, a commercially available data warehouse management system, Teradata, will be investigated for its architectural and design attributes.
1.4 Methodology

The methodology followed in this thesis work is described below:

1.4.1.1 Literature Study:

A review of the literature in the area of high performance decision support systems has been performed in accordance with the given time constraints. This includes books, research papers and technical magazines. The purpose is to consider recent advances in the area of data warehousing technology and shortlist any potential concepts that can lead to a better solution for this problem.

1.4.1.2 End User Queries Review:

An analysis of column usage was performed on a random sample of queries executed by end users for statistical analysis. Checking row counts of all the columns for appropriate index creation.

1.4.1.3 Code Analysis:

The Extract Transform Load (ETL) Code responsible for moving the data from source to staging database and to the data warehouse has been analyzed. The purpose of analysis is to check the maintainability and resilience to change.

1.4.1.4 Design Analysis:

An investigation of the physical design of the data warehouse has been conducted to find out any possible areas of improvement. It also includes table partitioning, indexing strategies analysis. A random set of queries have been studied to find out ways to reduce data access time.

1.4.1.5 Architectural Analysis:

This analysis is performed to find out the possible optimizations on the architectural level, including ETL strategy and storage strategy. In the ETL strategy the main task is to find out how it can be modified into an incremental extraction and loading. The storage strategy review includes the data redundancy analysis at different points in the databases.
2 Data warehouse Architecture and Designing

A review has been conducted into the available literature and research going on in the field of high performance data warehouse systems.

When designing a data warehouse or a database system there are a number of desirable characteristics that one is keen to optimize. It should be scalable, accessible, response time should always be in acceptable limits. It should be easy to maintain and so on. There are a number of design and architectural level methodologies proposed for achieving optimal results from the final system. The factors that affect the final outcome includes the choice of hardware architecture, software product and the design of the data warehouse. If everything has been chosen wisely and according to the requirements and priorities, the end product will also be well behaving. In this chapter we will analyze the different aspects that contribute to the success of the hardware, software and design.

We will study the hardware approaches that are being used by different products. In terms of hardware we should give high priority to the scalability of the system as data warehouse is always growing and more data is always pouring in. If we have a scalable system it will help us in accommodating more data without changing the entire platform underneath. If we do not opt for a scalable system in the beginning we may end up in replacing the entire hardware because it simply would not be able to handle the increased workload.

By software we mean the data warehouse management system software. It also plays a major role in making a successful data warehouse. There are some fundamental differences in the method of data storage that are being used by different products. We will investigate some characteristics that contribute to the high performance on the software level.

On the design level, we will investigate different data warehouse and database design methodologies that plays an important role in optimal performance and maintenance of the system.

2.1 Parallel Processing Hardware

The trend of the last few years has shown that the cost of hardware has gone down and processing efficiency has gone up continuously. It has become easier to afford computers with more and more processing power at reduced prices. To achieve high performance we can use bigger and faster machines.
addition to this brute force method of pouring in more resources to increase performance, there are some more creative approaches as well:

We can introduce parallelism into picture and see if this is practical and how can it help us in achieving satisfactory results. There are at least three possible approaches that can exploit the parallelism capabilities for performance improvement (DeWitt, Madden, & Stonebraker, 2011)

Shared Memory
Shared Disk
Shared Nothing

2.1.1 Shared Memory

It means using multiple processors all of them sharing single memory and disk. All the data access requests made by multiple processors are terminating at a single memory source. If we examine the shared memory system, in terms of data warehouse load requirements, it is evident that it is not scalable as the memory requests are carried over the same data bus that eventually becomes a bottle neck.

Another issue with shared memory systems is to keep the cache consistent. As the number of processors keep on increasing, it becomes more and more difficult to keep their cache memory consistent. In some cases it may require specialized hardware as well. This makes it much more expensive to scale up.

Another term used for this kind of architecture is Symmetric Multiprocessing (Inmon, Rudin, Buss, & Sousa, 1999).

2.1.2 Shared Disk

Shared disk architecture involves a number of processors each with its own memory. A disk or a set of disks are shared between all processor-memory sets. The disks can be connected to the processors using a network. This system also suffers from the scalability limitations as the network bandwidth can become a bottleneck in the face of increasing loads. In order to lock tables when concurrent access is performed we need to have some sort of data structure that should be accessible to every node. As every node has its own memory and no memory is shared between nodes, it becomes cumbersome to store the lock table. Either it should be dedicated to one node, or we can have a distributed lock management system. Both of these methods limit the scalability of the system. If we have a single dedicated node for lock
management we would have a single point of failure and single node would not be able to handle increasing loads.

Another way to work around this problem is by using a shared cache. In this mechanism, when a node have to access any data, it first checks the local cache memory, then it checks if the resource is in the cache of some other node, if it is not than it can access it from the disk. In terms of data warehousing workload this approach also cannot work as the cache misses will be too frequent, reason being that data warehouse queries normally access much more data than OLTP queries as it has to summarize large fact tables. This results in minimal benefit of cache as most of the queries will be requiring hard disk access.

2.1.3 Shared Nothing

In this approach every processor has its own memory and disk(s). One typical approach in this kind of architecture is to divide the data rows amongst all the available nodes. Usually each table is horizontally divided amongst the nodes. This helps in balancing the load on the overall system as each node is only responsible for the subset of the whole data. Nodes are responsible locking and access management to the data as each has its own memory.

Table 1 below highlights the different parallelism approaches taken by different Data warehouse and database management systems.

<table>
<thead>
<tr>
<th>Shared Memory</th>
<th>Shared Disk</th>
<th>Shared Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server</td>
<td>Oracle RAC</td>
<td>Teradata</td>
</tr>
<tr>
<td>Postgre SQL</td>
<td>Sybase IQ</td>
<td>Netezza</td>
</tr>
<tr>
<td>MySQL</td>
<td></td>
<td>IBM DB2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnterpriseDB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GreenPlum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vertica</td>
</tr>
</tbody>
</table>

**Table 1:** Different parallelism approaches taken by different data warehouse DBMS vendors. Source: (DeWitt, Madden, & Stonebraker, 2011)

2.1.4 Comparison of Parallel Approaches

As we have discussed three different parallel approaches that can be used. Scalability is a useful measure that can gauge the advantages and
disadvantages of one over the other. Scalability means that a system can be adapted or scaled up to accommodate increasing requirements without undergoing major changes. More specifically, as the data is seldom or never deleted in a data warehouse so it requires the hardware to be scaled up with time. Let us analyze each of the above three approaches in the light of this requirement. Shared memory architecture is tightly coupled and it is usually designed to be as it is. We may add more memory but that is not an unlimited possibility. In the case of shared disk system the above mentioned problem can be resolved as we can add more CPU-memory to access the disk but the data transfer bus cannot be upgraded to increase data transfer rate from disk to physical memory. This problem limits the scalability in the data warehouse. Lastly, in the shared nothing architecture there is a possibility to add new nodes (CPU-memory-disk) to redistribute the increased load from the existing nodes. Theoretically, this process can be repeated many times. So it makes this type of architecture more scalable than the other two. Next chapter presents a detailed description of a shared nothing commercial system.

2.2 Data Warehouse Management Software

The software which is being used to manage the data warehouse data also plays a crucial role in achieving desired performance and design goals. There are tens of popular vendors out there which are selling database and data warehouse management software. While most of the products provide same standard features there are some notable variations as well. In addition to the traditional method of storing data by adjacently placing row data on the disk. There is a different approach called column based storage.

2.2.1 Row based storage

In the traditional database systems which are row based systems the data is physically stored on the disk in the form of rows. In other words the data of all the columns related to one row is contiguously placed on the disk. This storage mechanism results in achieving much more efficient write operations as all the data belonging to a record can be written to the disk with a minimum amount of disk write operations.

This storage strategy is not very optimal for data warehouse data access. The reason is that a data warehouse is not very frequently updated. The data is loaded after a certain period of time depending on the circumstances. This loaded data is then used for the queries and reports generation over a period of time until it needs to be updated again. When DBMS is running a data read
query requiring only a few columns out of many in a table. The system must bring to memory all the column values related to each row and then filter out those which are not required by the query. This additional I/O cost can be avoided if we find a way to avoid bringing non-required columns into memory. (Stonebraker, o.a., 2005)

2.2.2 Column based storage

Recently the focus of research has been to partition the data vertically based on columns and store contiguously on the disk. All the data in one column of a table would be stored alongside each other. This would be a read optimized solution because it would avoid the problems mentioned in the above paragraph. The idea behind this approach is that a query normally accesses only a few columns, from often de-normalized tables, in a data warehouse. If we store data in the form of columns rather than rows, it would result in a much better performance. If we need to access only C - D columns out of C number of columns (where C and D are positive integers greater than zero) in a table. We first have to access all the columns of the qualifying rows from the storage and then filter them out. The additional I/O involved in accessing non-required columns is saved by column based storage. (Stonebraker, o.a., 2005)

2.2.3 Compression in data storage

As mentioned in the hardware section above, it is becoming cheaper and cheaper to afford more powerful machines. Cost per byte to store and process data has reduced, however, the cost to transfer a byte from storage to processing can be translated into a bottleneck in performance. As the data bus size has not increased much relative to advances in processing capabilities and storage capacity, It is wise to compress the data when stored to considerably reduce the I/O cost.

As shown in the Figure 2 below if we compare the data stored in compressed form with the one normally stored, we can easily see that compressed database will take less time to get transferred to the main memory. There will be an additional cost of uncompressing the data but that would be much faster as that would be an in-memory operation.
2.2.4 Data Warehouse Designing

There are number of possible design approaches which can be used for optimal storage and retrieval of a data warehouse data. The design tradeoff involves choosing a proper balance between available space and time constraints. The design techniques described below are those that can help in making appropriate choice between space and time constraints. The Figure 3 shows a graph which explains the relationship between disk usage and time to access data. If we suppose that we have a design that attempts to reduce the Disk space requirement than the time to access will increase. On the other hand if we have no limitation on space usage we can compute and store virtually every information that is possible to be extracted out of the system. This will minimize the time to access information. The design process revolves around finding a proper balance between memory usage and time constraints to access the information. The more we remain close to the origin of the below graph while accessing information the better our design is.
2.2.4.1 Normalization

Normalization is a process to reduce the redundancy from a database. When normalizing a database we are actually optimizing the space requirements to store data. It usually involves tables containing multiple columns that are broken down into smaller tables having less number of columns. They are then related by primary key-foreign key relationships.

2.2.4.2 Denormalization

De-normalization is the reverse process to the normalization process. It is the most common method when designing for performance. There are at least three main denormalization techniques: pre-aggregation, column replication and pre-join. Pre-aggregation means that the columns that are most often aggregated in user queries should be aggregated and stored preemptively. In this way when the users access the aggregation there is no on-the-fly calculation involved, resulting in faster queries execution. As described above that pre aggregation is for the aggregation operations that are repeated by the users. Similarly pre-join denormalization means that we should preemptively join the tables that are frequently joined by the user queries. In this way the join cost at the time of access can be avoided. Column replication is almost similar to the pre-join with the only difference that in this technique we only replicate one column in the joined table instead of fully joining the tables together. (Inmon, Rudin, Buss, & Sousa, 1999)
2.2.4.3 Star Schema

A star schema is a database design method. In this method the focus is to structure the data in such a way that we can reduce join operations in the queries. As JOINING cost is one of the most expensive operations in any query.

We store the data in two different types of tables called fact tables and dimension tables. Fact tables usually contain foreign keys from dimension tables and it also contains measurable columns such as price, quantity etc. Dimension tables normally contain descriptive columns which can be arranged in hierarchies.

Figure 4 below shows a sample star schema diagram with a fact table in the center and 4 dimension tables.

![Star Schema Example](image)

2.2.4.4 Snowflake Schema.

A snowflake schema is similar to the star schema except the following difference, the dimension tables are normalized to generate more tables. The resulting structure is called the snowflake schema. This allows us to save some space in comparison to having large denormalized dimensions.
A more detailed treatment of snowflake schema and star schema can be found in (Ponniah, 2001) or any other book covering dimensional data warehouse modeling.

2.2.4.6 Database Objects

There are a number of database constructs commonly available in many DBMS platforms in different flavors. Optimal use of these constructs can also result in optimized performance of data warehouse. They are briefly described below.

2.2.4.6.1 Indexes

An Index is a data structure that can be used to reduce the time to find a row in a table. It usually works by minimizing the requirement of full table scan row by row. An index can be based on binary tree structure, hash map or a bitmap as well. Index is usually built on a single column or a group of columns in the same table. This is called the index key. These keys map to rows in a table. When DBMS engine has to find the location of the row, it uses the index to locate the key from the index structure which normally takes much less than O(n) (See Appendix B) time to find the row.

The Figure 5 below represents the concept of binary tree index. The green arrows show a search path that can spare the requirement of visiting all the rows. It is made possible by sorting the keys in a certain order.

![Figure 5 BTree Index Structure](image)

Figure 5 BTree Index Structure (Green highlights one of several search paths)
A bitmap index works best on a column where the number of unique values is very small. The idea is to create a bitmap for each distinct value and set the bits to ‘1’ where the value occurs. For example the figure below shows the usage of bitmap index on a column with only three possible distinct values.

A hash map index works by organizing data into key-value pairs. A direct access is possible by using key for a particular value. It also reduces the requirement of full table scan.

2.2.4.6.2 Materialized Views
Materialized views are views with indexes. Just as we can build an index on a physical table(s) column(s), we can also do so for a view. The DBMS takes care of the changes in the index corresponding to the changes in the underlying table data itself. This allows us to have different visualizations of data without physically storing the data itself. The only additional cost is for storage and maintenance of index.

2.2.4.6.3 Partitioning
In the data warehouses normally the tables gets millions of rows deep and sometimes thousands of columns wide. This increases maintenance cost (e.g. index maintenance) and access time. Partitioning is a way forward for such situations. We can partition a table both horizontally and vertically. In vertical partitioning we create groups of columns from a table and divide them into two or more tables. Each of these groups can then be stored as a separate unit. By doing this partition we separate the columns that are more frequently accessed from those that are not. This improves the data access time. In horizontal partitioning we distribute rows in a table into groups. The advantage we gain is the decrease in maintenance cost as the number of index levels decrease due to decrease in number of rows per partition. The access cost also decreases as index traversing is faster due to the same reason.

2.2.4.6.4 Period-End Batch Reporting
Another technique that can be used for performance improvement on design level is the usage of period end batch reporting. In this method one can decide on the reports that are required often. Write program which can be run in the off-peak hours to generate the data the tables. Data population frequency can be decided on need basis. It may be daily, weekly or monthly as per requirement. The advantage of this method is that we can quickly access the data in the format it is required as it is not calculated on the fly. The disadvantage on the other hand is that the user does not have the flexibility
while doing analysis as the format is pre decided. If some variation in the format is required it has to be programmed again.
3 Teradata

As described in the previous chapter that shared nothing architecture is the most scalable architecture. Numerous factors contribute to the choice of Teradata as a Data warehouse management system. It is a pioneer in data warehouse technology. It is solely focused on data warehouse management systems. It is an instance of a shared nothing architecture. It is not just a software but a complete solution with specialized hardware. We will have a detailed discussion about the internal architecture and working of Teradata in order to find out how does it attempts to solve the problems with other systems.

Teradata architecture and design has been studied as an example of a parallel processing database management system (DBMS). This software is primarily marketed as a data warehouse platform not as a transaction processing system. This DBMS is designed considering the requirements of a data warehouse processing, unlike most of the commercial systems that have evolved from being a transaction processing system into an OLAP system.

As different hardware architectures were introduced in the beginning of the document; a detailed study has been performed on the Teradata hardware architecture and design.

3.1 Hardware Architecture

The hardware running below the Teradata software is based on Symmetric Multi Processing (SMP) technology. The hardware of SMP systems can be connected by using a network to form a massively parallel processing (MPP) system. SMP systems have already been described in the Literature study section above. Finally, the MPP systems are the result of connecting two or more SMP nodes.

BYPNET is the network that connects the SMP nodes. Its purpose is to provide services to processors that include broadcasts, multicasts and point to point communication support. It can be considered as a highly sophisticated system bus with advanced functions. The environment can usually have at least two BYNETS. This provides additional fault tolerance, thus If one bus fails the other can take over its job.

For storage, Teradata uses redundant Array of Independent Disks (RAID) technology. Data availability is ensured using clique feature. A group of nodes
called clique have access to common disk array units and connections are made using fiber channels.

Teradata also uses virtual processors to eliminate the dependency on specialized physical processors. There are two types of virtual processors:

- PE (Parsing Engine)
- AMP (Access Module Processor)

![Teradata High level Architecture](image)

The PE is used for session controlling and dispatching tasks. It communicates with clients and AMPs. It is used for session management, decomposition of SQL statements and for sending the query results back to the clients. A PE has the following components: parser, optimizer, generator, dispatcher and session control. The Parser decomposes SQL statements, the optimizer finds the most efficient data access path, it also passes the optimized parse tree to generator. The generator generates steps by transforming the optimized parse tree. The dispatcher receives and posts parsing steps towards the AMP that are most suitable. The session control manages the users sessions and takes care of activities like logon, password validation and failure recovery.

The AMP is a virtual process that performs main database tasks such as retrieving data rows, filtering, sorting, joining etc. It is also responsible for
disk space management. As each data table is distributed across the AMPs, each AMP manages the portion of the table it is responsible for.

The architecture is also given in Figure 6 Teradata High level Architecture.

3.2 Query Execution

The data access request is initiated by the client and it ends up in the parser, which is one of the components of the Parsing Engine. A cache of previous requests is also maintained by the parsing engine. The parser will check the cache for the same requests. Let us consider the case when a request is not in the cache. After syntax checking, the objects referred in the query are resolved into internal identifiers. Access rights are checked by the security module and it forwards the request to optimizer if the access rights are valid or stops it if they are not valid. The optimizer finds out the best method of executing the SQL commands and it creates an optimized parsed tree. The generator transforms the optimized parse tree into non-concrete steps. The gncApply, which is the next component handling the query, takes the non-concrete steps produced by the generator and makes them concrete. Later, it passes the concrete steps to the dispatcher.

The Dispatcher is responsible for the order in which the steps are executed. It transfers the query execution steps to the BYNET one by one and after completion response of one it passes on the next step. For every step it also informs the BYNET if it has to be broadcasted to all the AMPs or only one of them.

The AMP is responsible for accessing the disk and obtaining requests which are required to be processed. The AMP can also use BYNET to transfer message to another AMP or set of AMPs (Teradata Corporation, 2006). The data access request flow path is also shown in the Figure 7 Data Access Request flow path.
3.3 Rows Partitioning

Teradata distributes the rows evenly among the AMPs. This is done in order to distribute the work load evenly and fully exploit the capabilities of the parallel processing architecture. There is a hash function in place which acts as a load balancer and distributes the rows evenly across the AMPs. When a row needs to be inserted, it is passed to the hash algorithm which returns the AMP that should store this row. Similarly, when we have to access the same row again, the hashing algorithm is used to figure out the AMP that has stored this row. (TeraData Corporation)

3.4 Linear Scaling

There is another important feature that is interesting to mention, it is called linear scaling. Linear scaling means that the system can be upgraded to handle increased work load. If the performance of the system falls below acceptable limits, Teradata will allow us to add additional resources. These additional resources will linearly improve the performance of the overall system.
As shown in Figure 8, when a new node(s) is(are) added to the existing set of nodes only some of the existing data is transferred to the new node(s). The hash maps are updated according to changes made. The AMPs read the data that they own and transfer that portion which they should not contain according to the updated hash maps.

3.5 Use of BYNET

It is a fully scalable banyan switching network. Teradata platform makes an intelligent use of the connection network by minimizing the traffic across the nodes. It is used in message passing, data movement, results collection and also coordinates inter AMP activities. When any aggregation is required in a query it is done at the AMP level first. The sub-results are moved between dynamically formed AMP groups to generate final results. This movement is also kept at a minimum level by keeping the number of AMPs that exchange data to low. Ordinary parallel database management systems use standard communication protocols like TCP/IP for internodes communication. These protocols use headers in addition to the payload and the actual bandwidth attained never reaches close to the theoretical limit. BYNET on the other hand is specially designed for a parallel database operation so it is generally more efficient.

As we have discussed some important features of Teradata in the above paragraphs. It is important to link these to the potential bottlenecks we discussed in the Literature study section.
In the hardware section we pointed out that shared memory systems are not scalable because we have a single data bus that connects all the processors to the memory. This becomes a bottleneck during increase workloads. In the shared disk systems the bottleneck is the network which is connecting the disk(s) to the processing units. These problems are avoided in the Teradata system by the help of shared nothing architecture. Whenever the system becomes overloaded we can add new nodes and redistribute the load (Teradata Corporation, 2006).

It is worth mentioning what effect a shared nothing multi processor system like Teradata could have on the graph of Figure 3. As shown in the Figure 9 below that it should, in principal, make a shift towards the origin as it reduces the overall access time constraints and storage space requirements.

![Figure 9 Shifted Time Space Graph](image-url)
4 System Analysis

In this section we will describe the current architecture, design and working of the data warehouse system in detail. As described in the introduction section that it consists of a multi stage databases and data warehouses. Each database has a significance of its own.

4.1 Data Flow

The operational data from the vehicles is transferred from the truck to the workshop and then from the workshops to the database which later becomes a source for the data warehouse. All the manufactured products (vehicles, engines etc) data is populated in the manufactured vehicles database as the vehicles are being produced. This data is then transferred to the data warehouse environment. Another source of data that is used to load data warehouse is the product’s configuration related data source that is also loaded into the data warehouse.

Figure 10 Dataflow Architecture

From all of these sources data is transferred to a staging database. This staging database is a partially normalized structure. The data loading frequency is not well defined and data is usually loaded at irregular intervals ranging from as lows as two weeks interval to may be a month long interval. In the staging database the data is copied into two independent databases. One of them is called clean database and the other is called unclean. There are a number of
database procedures written that reformats and transfers the data to the unclean database. Once the transfer to the unclean database is finished the dirty data is deleted inside the staging database. the remaining clean data is then transferred to the clean database. This is also done through Sql server procedures.

There are almost no structural differences between staging and clean / unclean databases. Staging database is only used as a holding area to separate clean and unclean data.

After population of clean and unclean databases from staging database. The data is further transferred to data warehouse. There exists two separate data warehouses. One is used to hold clean data and the other is used for unclean data. End users can access either of the data warehouses as per their requirements.

4.2 Hardware Architecture

In terms of hardware only one physical system is used to hold the all the staging databases, clean and unclean databases and the data warehouses. But the source data is situated on separate physical machines. It is transferred over the network.

4.3 Clean and Unclean Databases

These databases contain duplicated or redundant data. Most often the purpose of introducing redundancy into the data warehouse is to gain some performance benefit. Thus, one should be careful that while attempting to reduce redundancy, performance requirements should not be compromised.

The Cleaning process starts when the data from the staging database has been transferred to the unclean database and data warehouse. The cleaning process sanitize the data by removing rows in the staging database. The removal logic depends on many business rules. The removed rows, only the ones with removal date and identification column, are put into new tables to be used later on for lookup. The related data from connected tables is deleted to keep the database in a consistent state. After removal, the data is copied to the clean database for any changes in the existing data with the same primary keys. Any new additions are also copied to the clean database.

At this point it is important to note that the data that does not fall under cleaning logic is present in the unclean database and the same is also copied to
the clean one. In other words all the data in the clean database is also present in the unclean database. If we suppose that 30% of data that fell under cleaning logic was deleted. The other 70% would be in both of the databases. this duplication is later transferred into the star schema data warehouse as well (see Figure 12 for data redundancy overview in different databases).

4.4 Data warehouse Design

As data warehouse is tuned for performance so obviously there must be a difference in the storage structure. Similar is the case in this environment and the data is stored in a star schema structure. It is a denormalized structure.

Figure 11 below shows the star schema structure. It can be noted that it looks fairly simple with very few number of tables. The reason of this simplicity is that many participating tables from the partially normalized databases before has been denormalized to one as a result we have only five tables left. The central table is called the fact table and the tables exporting their primary keys to the fact table are the dimensions.

![Figure 11 Current System Star Schema Structure](image)

4.5 Loading Strategy

Current loading strategy in the data warehouse is to remove everything from the staging database. After removal populate everything from the source
databases again that includes new data as well as that was loaded in previous iteration. For clean / unclean databases and data warehouses the loading strategy is to insert data in the destination tables with new primary keys. For the data against existing primary keys every column is checked for any changes and updated if required. This approach is time consuming as every data item is read and compared.
5 Proposed and Tested Optimizations

In this section we will discuss the proposed and tested optimization methods in terms of performance factors discussed in the above literature study section. The sample implementations made to demonstrate the approaches discussed in the literature study section will also be covered in this section.

5.1 Database Space Reduction

As mentioned before, in the System Analysis section, the data warehouse comprises five databases in all. One of them is used as a staging database. Then, we have a clean and unclean set, each consisting of a fairly normalized database and a denormalized star schema data warehouse. Figure 12 below shows the duplication in different databases of the current system.

One of the possible solution to decrease data redundancy is to keep the clean and unclean data in the same database as separate tables. This will reduce the overall space consumption and It will also reduce the number of databases to three.

To achieve this, we first have to analyze what business rules are used for the cleaning of data. They are as follows:

- Remove snapshots with special value of mileage.
- Remove snapshots with no chassis number.
- Remove trucks with no chassis number.
• Remove snapshots for Scania trucks.
• Remove snapshot with chassis number less than a specific value.
• Remove truck and chassis data for Scania trucks.
• Remove snapshot if any related sample value is negative.
• Remove ECU (Electronic Control Units) Charts if breakpointidx is a special value.
• Remove ECUs related to removed snapshots.
• Remove ECUs if there is no related ECU chart
• Remove ECU Charts if they are related to removed ECUs.
• Remove duplicates in snapshots in terms of mileage (same mileage).
• Remove duplicates in snapshots if the difference between reading dates or mileage is not more than a decided limit.
• Remove samples if there are corresponding removed ECUs and ECU charts.

Figure 13 below shows the cleaning process as a block diagram.
In order to optimize the space usage in the databases / data warehouses, the following modifications can be made in the current environment. In the staging area, where the data is cleaned and pushed to later stages, for all tables from where the data is deleted, we can make shadow tables containing exactly the same columns as the original tables and two additional columns named removal date and removal reason code. Removal reason code is a foreign key from the ReasonCode table. This is a new table created to hold descriptions of all the data removal reasons.

Now, when any row of data is deleted as part of the cleaning process, it is copied to the shadow table with the reason code and removal date as additional columns. The data is not distributed to clean and unclean databases / data warehouses parts with high redundancy. Thus, the number of databases / data warehouses are reduced from five to three. Moreover, overall space consumption is also reduced.

To provide the same interface of data access to the user as before, we have created views on top of clean and unclean tables. When the user wants to query unclean / clean data she accesses the physical tables, and when the user requires data from both clean and unclean tables she can access it by using a view. The view is simply a union of clean and unclean tables (Figure 14).

The same approach is also applied in the clean and unclean data warehouses.

Figure 14 Accessing clean and unclean data.
5.2 Incremental Loading

As mentioned before in the Introduction chapter that the current loading strategy of the data warehouse is to fully load the system whenever an update is required. In other words, it means that the system will also load the data which may already be present in the data warehouse and was never updated in the source.

A simple way to work around this is to change the current loading process to incremental loading. Only load the data which was either new or updated. The updated data is the data which has changed against the same primary key in a table. One or more columns except the primary key columns has changed in the source.

An SSIS package has been implemented to change the loading of staging database. The loading strategy has been changed from full to incremental load. A last modified date column is available in the tables with most amount of data. This column is updated with the current time stamp when ever any data changes against a row.

A new table is added in the staging database which contains the last load start timestamp and last load end time stamp. The load start time stamp is updated with current date and time whenever data reading starts from the source for update. The last load end time stamp is updated with current date and time when the SSIS package execution finishes. For the next load only those records are picked that have last modified timestamp greater than the last load time stamp.

This approach considerably reduces the time required to refresh the data warehouse data. The exact loading time is a relative figure and depends on the number of days we chose to load in one go.

Figure 15 below shows the figure of the implemented SSIS package.
5.3 Partitioning

If there are columns that frequently appear in the WHERE clause of the queries, we can partition the data based on those columns, e.g.,

SELECT *

FROM TABLE

WHERE id = ‘xx’

In the above case, if the table was partitioned on id, then the queries using id on the WHERE clause would only require to access a specific partition of the table, and no partition would have to interact with the other partition for joining purposes (if there is any in the query). This would result in improved performance. (Chaziantoniou & Ross, 2007).

An implementation was made in SQL Server 2005, by partitioning a table with approximately four hundred million rows. The table was divided into 50+ partitions as per load constraints. Each partition contains between 3.5 and 16 million rows.


**Partitioning scheme:** The partition scheme was decided after analyzing a random sample of queries and by getting input from end users. The column `vectorName` has been selected as partition basis on the table, it has approximately 350 distinct values. The partition was made in alphabetical order and by considering the count of rows against each `vectorName`. In the Figure 16 below the color denotes the partition boundaries, the columns are showing vector names and row counts against each `vectorName`.

![Partitioning scheme](image)

The table below shows the time differences in the queries tested for comparison of partitioned and non partitioned implementation.

**Performance improvements**

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Non Partitioned Implementation</th>
<th>Partitioned Implementation (W/O Index)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Time 339M rows</td>
<td>1 hour 10 min</td>
<td>1 hour 10 min</td>
</tr>
<tr>
<td>Summary data for a specific vector.</td>
<td>3:33 min</td>
<td>2:19</td>
</tr>
<tr>
<td>Detailed data for a specific Vector</td>
<td>3:50 min</td>
<td>2:43 Min</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Summarized data - access to 4 different partitions.</td>
<td>3:23 min</td>
<td>2:13 Min</td>
</tr>
<tr>
<td>Summary for a full table</td>
<td>3:34 min</td>
<td>2:18 min</td>
</tr>
</tbody>
</table>

An index was also created on the `vectorName` column. This is a non clustered and non unique index. It further improves the performance of the queries mentioned in the table.

### 6 Future Work

This work can be extended to compare the performance of different data warehouse management systems available in the market. The testing can be performed under different load conditions, specially t

The parallel processing solutions can be tested to find out how they compare to other conventional solutions. Additionally, there are many types of indexes available in some products but not every type is available in every solution, one can compare these and see the results with the aim of selecting the best possible technology.

The cleaning process is currently based on SQL server procedures. These procedures make it hard to maintain the data warehouse as the changes occur in the source system. This process can be investigated and reprogrammed in any graphical ETL tool like Microsoft SSIS. The graphical tools makes it easier to visualize the data reformulation and transformation steps, and subsequently to maintain them in case of any changes.
7 Discussion & Conclusion

In this thesis work the primary task was to investigate the data warehouse architecture, designing and development practices for the betterment of the data warehouse. A literature study was performed in this area to find out the research and development going on in this area. In addition to the current research and development the established engineering principles towards making high performances data warehouses were also studied.

The current implementation can be improved in a lot of ways as demonstrated in the proposed optimizations section. The space usage can be reduced by keeping the clean and unclean data in the same database. the query performance can be improved by implementing partitioning on the deep tables. The loading strategy can be changed from full load to incremental loading by using the approach recommended above. The choice of data warehouse management system is a difficult question but it is recommended to use a parallel processing system with shared nothing architecture for best scalability and performance. A study is performed towards the architecture of Teradata as a sample parallel processing system. This is only a theoretical part and is subject to more work in order to make the conclusion that the system would really be high performing and scalable if parallel processing hardware would be used.

Moreover, we can conclude that shared nothing architecture is the most scalable. If we use a software that uses compression adds to the performance of the data warehouse. A software system that stores data column wise is much more efficient for data warehouse queries.
9 Bibliography


http://db.lcs.mit.edu/madden/high_perf.pdf


11 Terminology

**DBMS** Database Management System

**SSIS** SQL Server integration services

**DW** Data warehouse

**SSAS** SQL Server Analysis Services

**ETL** Extract Transform Load

**MPP** Massively Parallel Processing

**SMP** Symmetric Multi Processing

**RAID** Redundant Array of Independent Disks

**ECU** Electronic Control Units

**RESD** System Architecture and Product Data
Appendix A

What is SSIS?

“Microsoft Integration Services is a platform for building enterprise-level data integration and data transformations solutions. You use Integration Services to solve complex business problems by copying or downloading files, sending e-mail messages in response to events, updating data warehouses, cleaning and mining data, and managing SQL Server objects and data. The packages can work alone or in concert with other packages to address complex business needs. Integration Services can extract and transform data from a wide variety of sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations.

Integration Services includes a rich set of built-in tasks and transformations; tools for constructing packages; and the Integration Services service for running and managing packages. You can use the graphical Integration Services tools to create solutions without writing a single line of code; or you can program the extensive Integration Services object model to create packages programatically and code custom tasks and other package objects” (Microsoft Corporation, 2011)