Using Dimensional Hierarchy Analysis in Data Warehouse Fragmentation Process

Yusor Rafid Bahar
College of Education for Pure Sciences / Ibn Al-Hatham
University of Baghdad

ABSTRACT

The data warehouse becomes extremely large in size and its implementation normally uses very expensive platforms, typically based on high-end servers or high-performance cluster. Therefore, emerged the need to develop a distributed data warehouse to meet requirement specific to a departmental or restricted community of users.

The transition from the using of a central data warehouse to the distributed data warehouse needs to fragmenting the central data warehouse into several smaller warehouses (or data marts) physically distributed across the nodes in computer network. There are a lot of algorithms used in the process of fragmentation such as genetic algorithm, affinity-driven algorithm, etc. In this research we will use the principle of hierarchy located in the data warehouse as the basis of the fragmentation process, we proposed framework includes the representation of hierarchical property in the form of inverted tree with several levels depending on the existing data, so we can conduct searches on this tree using searching algorithms to get to the best level that could be adopted in the fragmentation process. In this framework a greedy search algorithm will be used because of its advantages to support the process of selecting the appropriate level that will use in fragmentation process.

The use of distributed data warehouse as small data warehouses (data marts) after the use of fragmentation process was the first attempt to solve the problems of space, performance, and cost of special hardware.

1- INTRODUCTION

A data warehouse is a repository of data from legacy and transaction database systems. Unlike a typical transactional database used for day-to-day operations, a data warehouse is structured to maintain large amounts of related historical data that transform it into a dimensional or normalized data store [1].

The basic idea behind the data warehousing approach is to extract, filter, and integrate relevant information in advance of queries [2,3].

The data warehouse would be seen as a centralized repository, whereby data from all sources would be imported into that large centralized repository for analysis. Nowadays the speed and bandwidth of wide area computer networks enable a distributed approach. There are two major forces that have contributed to the importance of parallel and distributed data warehousing. First, the fact that data warehouses can be extremely large and highly resource demanding, so it requires a special type of hardware with highly efficient. The second is that, queries and analyses must be answered within acceptable time limits has led to a series of specialized techniques that were developed specifically for them, including view and cube materialization, indexing structures and implementations of distributed data warehouse [2,4].

The distributed environment satisfactorily the need to handle huge data sets efficiently, in both query processing and other concerns such as loading or creation of auxiliary structures [3,6].

2- Related Work

A number of researchers have recognized new concepts in the traditional data warehouse model and suggested a series of views at the conceptual and practices level. The Shuigeng Zhou and other proposed a design of new type of data warehouse, called hierarchically distributed data warehouse (HDDW), which can be built quickly by a bottom-up, level by level method. HDDW is constructed in as follows: starting from low level, building a data mart for each town separately then moving up to county level, then building a data mart for each county, which contains the economic data of all town located within that county. On the basis of county-level data marts, the provincial level data marts are built so that each provincial data mart holds data generalized from the county level data marts located within the corresponding province. Finally, a nation-wide data warehouse is created over the provincial level data marts, and there is only one DW at the nation level. This leads to a kind of distributed DW, which is also called federated DW [7]. This method is useless in case of having a large central data warehouse needed to be a distributed data warehouse (top-down approach) because no fragmentation schema has been used. Other approach "Handling Big Dimensions in Distributed Data Warehouses using the DWS Technique" by Marco Costa and Henrique Madeir in which a data warehouse striping (DWS) this technique allow the distribution of large data warehouses through a cluster of computers. The round-robin partitioning approach partitions the fact table strict row-by-row through all nodes and replicates the dimension tables, where the replication of the dimension tables creates a limitation to the applicability of the DWS technique to data warehouses

with big dimensions. Therefore, the researchers proposes a new approach called selective loading to deal with data warehouses with big dimensions in DWS systems. The selective load technique proposed explores the fact that the subset of the fact table rows stored in each node are only related to a small part of the rows of the big dimension and not related to all of them. Thus, the idea is to store in each node only the dimension rows that are related to fact rows stored in that node, and not to replicate the entire dimension [6]. This method faces a problem because the data size will be bigger over time and the round-robin partitioning approach must be applied again to partitioning the fact and dimension table row-by-row through all nodes.

3- Distributed Data Warehouse Technology

A distributed data warehouse (DDW) can be defined as a logically integrated collection of shared data that is physically distributed across the nodes of a computer network. The distributed data warehouse architecture involves the merging of two diverse concepts namely, integration through the data warehouse element and distribution through the networking element as shown in the following figure (1) [1].

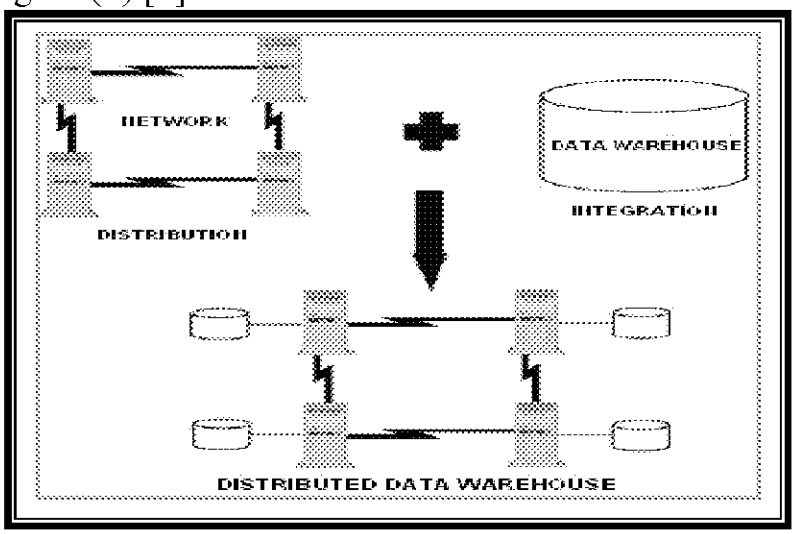


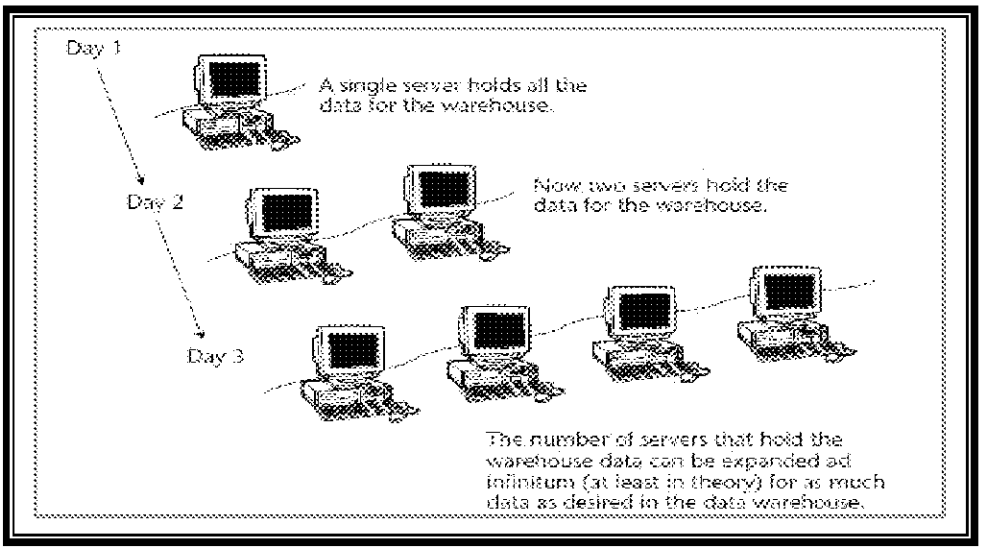
Figure 1: Data Integration and Distribution

The infrastructure required to implement a distributed data warehouse is just a set of inexpensive computers (called system nodes, typically PCs or server blades) connected by fast Ethernet (share nothing configuration) and having a same DBMS installed in each node [6].

Physically storing and administrating the data warehouse as a distributed database helps in the storage and access of large and small pieces of datasets. The first advantage of a technologically distributed data warehouse is that the entry cost is cheap. In other words, the cost of hardware and software for a data warehouse when initially loaded onto distributed technology is much less than if

the data warehouse were initially loaded onto classical, large, centralized hardware. This could reduce the need for massive central computing resources, network delays, OLAP query can be executed in parallel fashion, and provides the transparent local datasets access to the clients [4,8]. The second advantage is that there is no theoretical limit to how much data can be placed in the data warehouse. If the volume of data inside the warehouse begins to exceed the limit of a single distributed processor, then another processor can be added to the network, and the progression of adding data continues in an unimpeded fashion. Whenever the data grows too large, another processor is added. Figure (2) depicts a world in which there may be an infinite amount of data in the data warehouse. This is appealing because a data warehouse will contain much data (but n

ot an infinite



amount) [3,4].

Figure 2: The Progression of Adding More Servers to Hold Data Warehouse

4- Fragmentation

The data fragmentation technique is the result of the data partitioning process and its aim is to reducing the number of disk accesses for query execution as a result of minimizing accesses to irrelevant data. The data warehouse modeled using a star schema have an important property in terms of join operations between dimensions tables and the fact table (i.e., the fact table contains foreign keys for each dimension) [9]. The joins in data warehouses (called star join queries) are particularly expensive because the fact table (the largest table in the warehouse by far) participates in every join and multiple dimensions are likely to participate in each join. To speed up star join queries, many optimization structures were proposed: redundant structures (materialized views and advanced index schemes) and non redundant structures (data fragmenting and parallel processing). Recently, data fragmenting is known as an important aspect of physical database design (Sanjay, Narasayya & Yang, 2004; Papadomanolakis & Ailamaki, 2004). Two types of data fragmentation available which is vertical and horizontal fragmentation [1]. The vertical fragmentation (VF) allows tables to be decomposed into disjoint sets of columns, where Horizontal fragmentation (HF) allows tables, materialized views and indexes to be partitioned into disjoint sets of rows that are physically stored and usually accessed separately.

HF may affect positively (1) query performance, by performing partition elimination: if a query includes a partition key as a predicate in the WHERE clause, the query optimizer will automatically route the query to only relevant partitions and (2) database manageability: for instance, by allocating partitions in different machines or by splitting any access paths: tables, materialized views, indexes, etc [10]. Logically, two types of HF are distinguished by Ceri, Negri & Pelagatti, 1982 and Özsu & Valduriez, 1999:

- **Primary HF:** The primary HF consists in fragmenting a table T based only on its attribute(s).
- **Derived HF:** The derived HF consists in splitting a table S (e.g., the fact table of a given star schema) based on fragmentation schemas of other tables (e.g., dimension tables). This type has been used in (Stöhr, Märtens & Rahm, 2000). The primary HP may be used in optimizing selection operations, while the second in optimizing join and selection operations since it pre-computes them [1].

In this research, we will concentrate on fragmenting the warehouse data horizontally. We consider that the warehouse data is modeled using a star schema. This schema has two kinds of tables: dimension tables D = {D1, D2, ..., Dd}, where each table Di has a primary key KD, and a fact table F where its primary key is composed by the concatenation of the keys of dimension tables. Fragmenting of warehouse data is more complex and challenging compared to that in relational and object databases due to the several choice of fragmenting

of a star schema. In warehouse, either the dimension tables or the fact table or both can be fragmented [9].

1- Fragmenting only the dimension tables using simple predicates defined on these tables, a simple predicate p is defined by a P: Ai θ Value; where Ai is an attribute, θ € {=, <, >, ≥, ≤}, and Value € Domain (Ai). This situation is not suitable for OLAP queries because the sizes of dimension tables are generally small compared to the fact table. Most of OLAP queries access the fact table, which is huge. Therefore, any partitioning that does not take into account the fact table is discarded [10].

2- Fragmenting only the fact table using simple predicates defined on this table because it normally contains millions of rows and is highly normalized. The fact table is composed of foreign keys and raw data. Each foreign key references a primary key on one of the dimension relations. These dimension relations could be time, product, customer, etc. The raw data represent the numerical measurement of the organization such as sales amount, number of units sold, prices and so forth. In a data warehouse modeled by a star schema, most of OLAP queries access dimension tables first and after that to the fact table, it is necessary to fragment both of them. This choice is also discarded [1,10].

3- Fragmenting some/all dimension tables (using their predicates) by applying the primary HF firstly, and use them to fragmenting the fact table using the derived HF. This approach is suitable because it takes into consideration the star join queries requirements and the relationship between the fact table and dimension tables [9].

5- The Greedy Algorithm

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum, it can be used as a selection algorithm to prioritize options within the search. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time [11].

The greedy algorithm is a stepwise, iterative procedure. At each step, it scans each planning unit that is not yet in the portfolio and selects the one that satisfies the determined condition until all element goals have been satisfied to the extent possible. It produces reasonable efficient solutions. The greedy algorithm always takes the best immediate, or local, solution while finding an answer. Imagine the following decision tree scenario in following figure (3). Greedy walks through the following tree, beginning from the root node, and at each node looks around and picks the local optimum. As described, there are always

two routes to take; it picks the one that currently looks best (in this case: lowest) heuristic value [12,13].

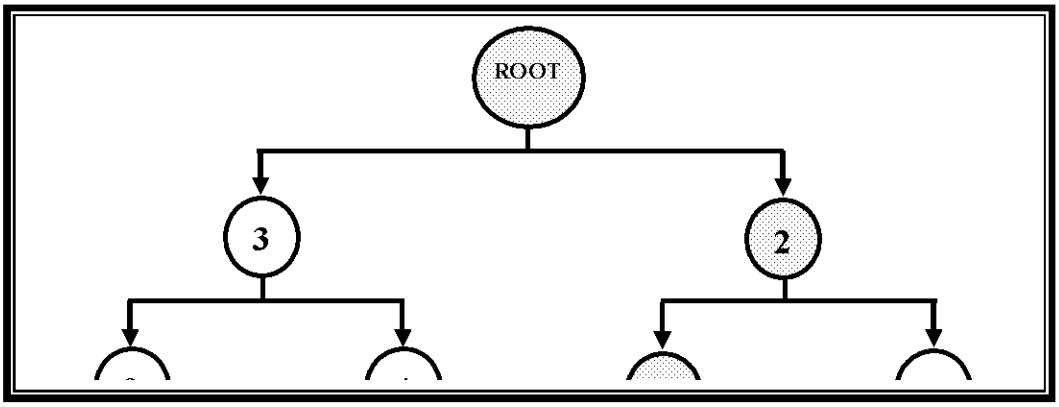


Figure 3: Greedy Decision Based on Local Optimum

The greedy algorithm will consider a hierarchy of dimension tables, which is in the form of a tree and we get the minimum and maximum number of fragments that data warehouse administer will desire. The greedy algorithm selects one dimension table hierarchy randomly, and then selects a level of the selected hierarchy. Once the selection is done, the fact table will be fragmented with derived horizontal fragmentation based on the primary horizontal fragmentation schema of the selected hierarchy of dimension table. Then compute the number of fragments (N) of the fact table, where (N) must be smaller or equal to max and bigger or equal to min. The algorithm selects another level of hierarchy and repeats the same process until the conditions are satisfied. Otherwise, the selected level of hierarchy will be removed and select another level and will keep repeating the same process until the conditions are satisfied or get to the end of the hierarchal level of this dimension table. Then the algorithm moves to select another hierarchy of another dimension table and repeat the same procedure until obtain a partitioned data warehouse.

6- Proposed Framework

This section is dedicated to present the phases of the proposed framework that being from the dimensional hierarchy analysis as an introduce to represent the hierarchy as a inverted tree structure and exploit it to searching for the best candidate level to fragmentation. A description of the overall framework architecture is explained in details. Finally, all steps were translated as an algorithm.

6.1 Dimensional Hierarchy Analysis

A hierarchy is a set of binary relationships between dimension levels. A dimension level participating in a hierarchy is called hierarchical level. Hierarchies can express different structures according to the criteria used for analysis and it is important part of any business where it is used to identify the

chain of administrative, regulating levels of customers, products, and to perform planning and analysis. When used to analyze, the user can view summaries first and then move to different levels of detail. For example, the user can see the total sales for the whole year and then can move down to the level of half of the year to look at the sales by individual half of the year. After that, the user moves down to the level of quarters or moves down further to the level of individual months to look at monthly sales number. Where note that, the hierarchy of the time dimension consists of a year, half, quarter, and month level. So the dimension hierarchy can be defined as a path for drilling down or rolling up in the analysis process. Hierarchies resulting naturally from the business process, so it must be arranged and sub divided in order to provide efficient control of the business activity. The nature of the hierarchy in the data warehouse is often represented by the (parent-child) relationship. In this type of relationship, one parent can have multiple child, and the child can not be belonging to more than one parent only. For example, the employer may be assigned to one department of the company, but the department may contain more than one employers. It is possible to represent this type of hierarchy as a tree structure and will call it hierarchical tree, as will be explained in the following figure (4).

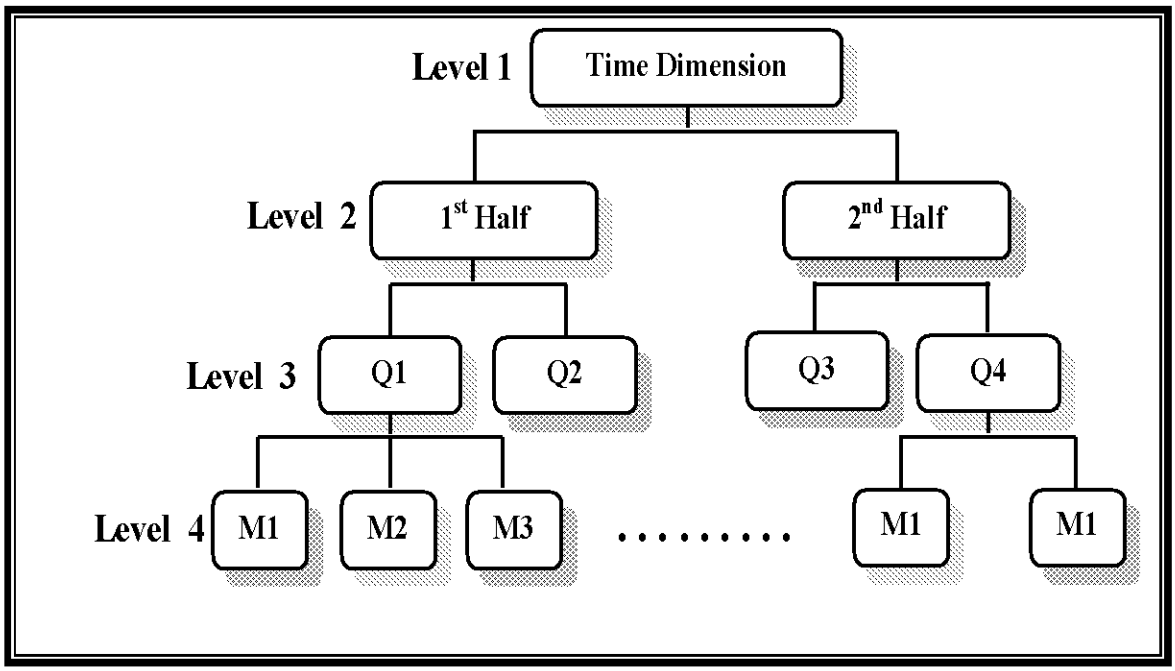


Figure 4: Simple Hierarchy Tree for Time Dimension

Each member in the hierarchy tree is called a (node), the topmost node in the hierarchy tree is called a (root node) which represents summaries of the data, the bottommost node is called (leaf node) and represents details of data. It is possible to use the hierarchical tree in the process of fragmentation. Therefore it possible to make the search process to the tree structure using the greedy search

algorithm in order to get the best level of hierarchy, which will be used in the fragmentation process to provide the efficient performance in the OLAP query and flexibility in the management and maintenance.

6.2 Overall Framework Design of Distributed Data Warehouse

This work presents a framework to handle the class fragmentation problem during the design of distributed data warehouse. The idea is using a hierarchy property available within the data warehouse properties and harnessed for the purpose of fragmentation of the warehouse data. The use of the greedy algorithm in this work provides the highest flexibility in the selection process to the best candidate dimension table and specific hierarchy and hierarchical level within this dimension table to be used in the fragmentation process phase. It provides a high percentage of transparency for the data warehouse administer to select or identify a minimum and maximum number of fragments that will result from the fragmentation process. The proposed framework is shown in the following figure (5) and all steps illustrated in figure (6).

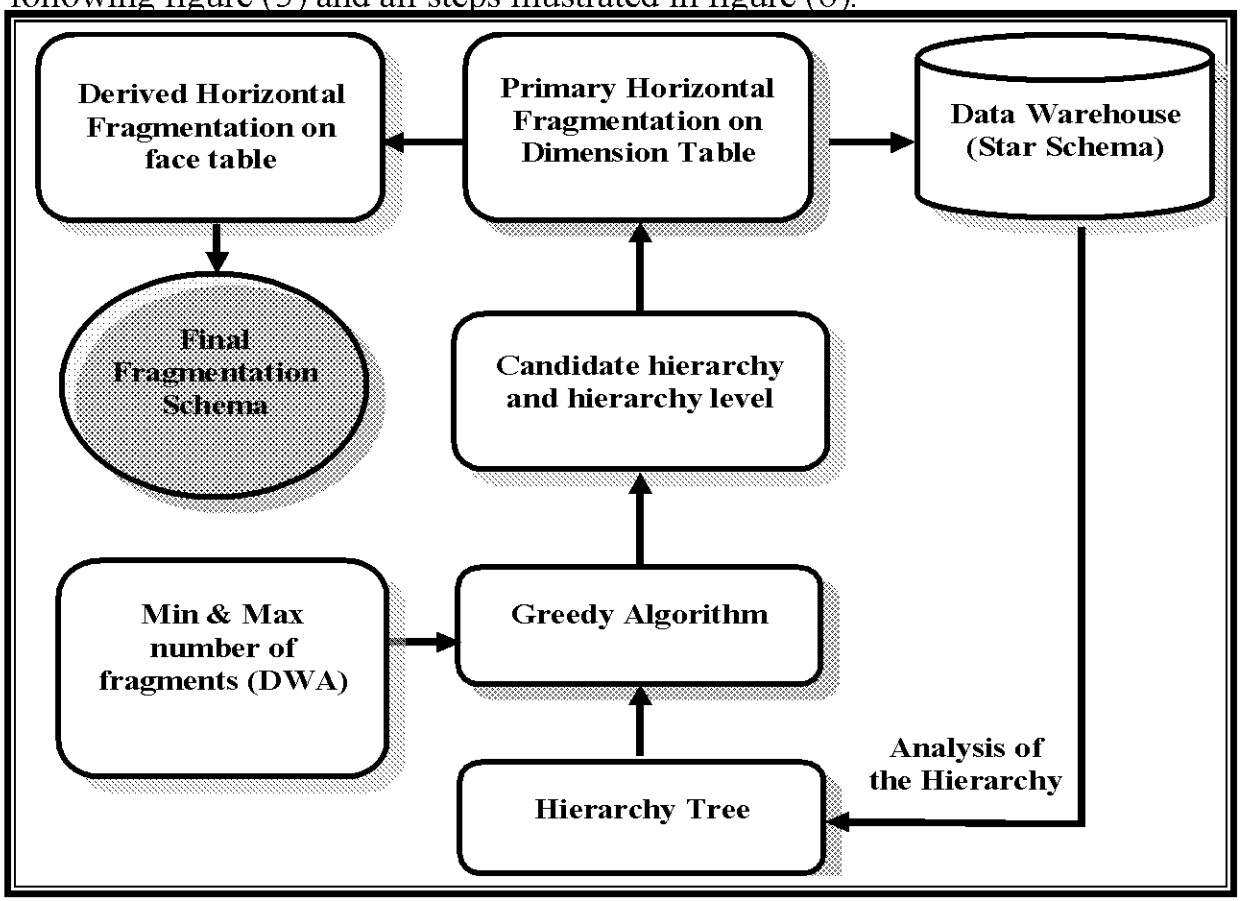


Figure 5: Overall Framework for Design of Distributed Data Warehouse

Initially the distributed designer must provide input information about the data warehouse semantics such as the operation that will be executed over stored data, quantitative and qualitative information, and additionally the number of

fragments that the data warehouse administers desires, this information then is passed to the greedy algorithm. A greedy algorithm then searches in the hierarchy tree (which is reflected in the dimensional hierarchy analysis) to find the best level of hierarchy that will be depended in the fragmentation. The output of the greedy search algorithm is a candidate dimension table and specific level of hierarchies in this table. As a result, it can apply primary horizontal fragmentation on this table depending on this nomination by greedy algorithm, and then apply derived horizontal fragmentation of fact table based on the fragmentation of dimension table which obtained by applying the primary horizontal fragmentation. Finally we obtain a set of fragment as a result of applying fragmentation process of the data warehouse based on hierarchy and greedy algorithm.

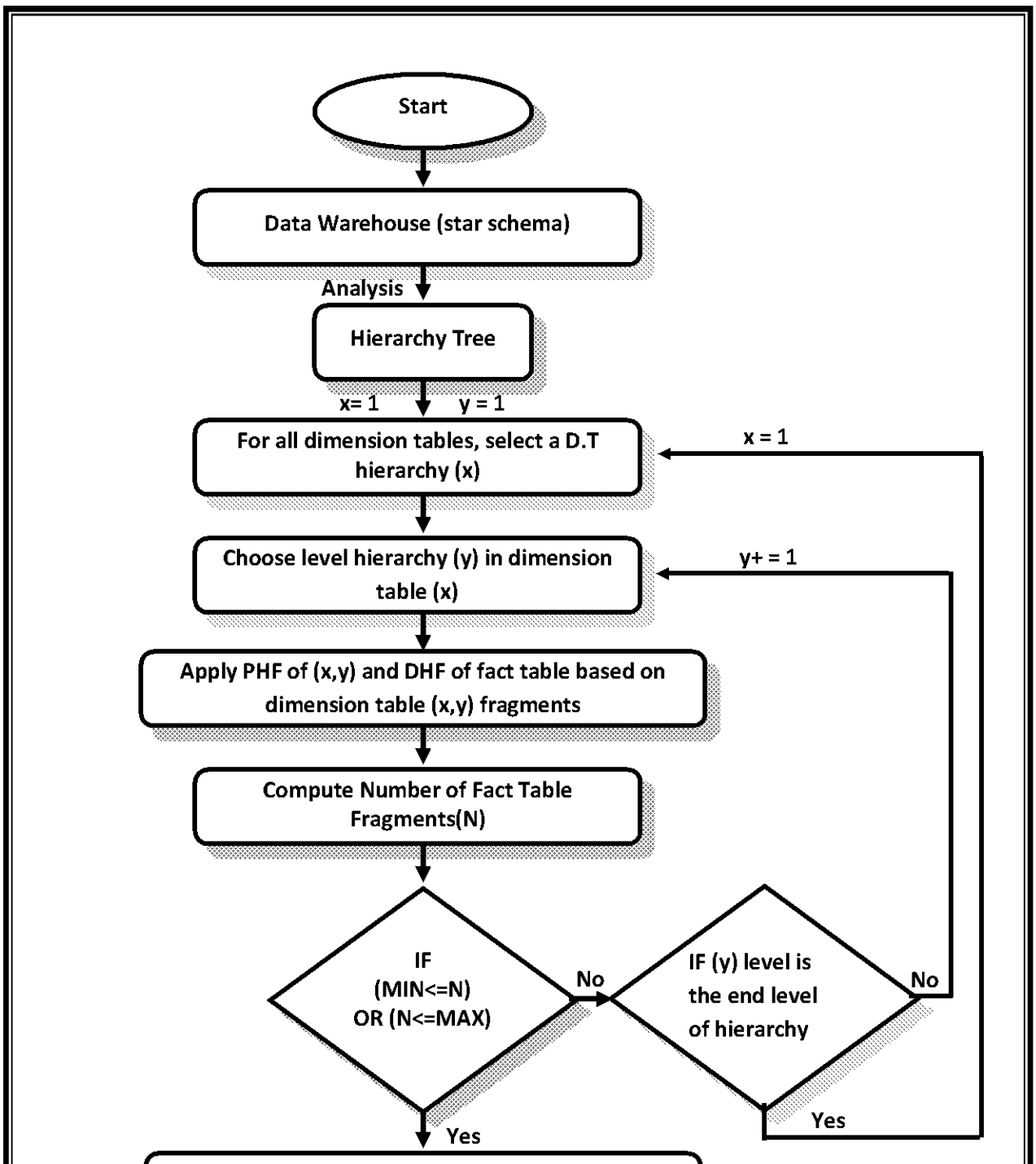


Figure 6: Framework Steps in Details

6.3 Fragmentation Algorithm for Proposed Framework

Input:

Tree HT (x, y) // Hierarchy tree for centralized data warehouse.
Integer MIN, MAX. .
Integer E // End level of hierarchy.

Output:

Integer N // Number of Fact Table Fragments.
Boolean Find solution.

Start

1- x = 1 ; y = 1 ;
2- S = Select (HT (x , y));
3- A = Apply PHF (S (x , y));
4- Apply DHF (fact table) based on (A);
5- Compute (N)

If (Min<= N) OR (N=>Max) then // Where min & max is the number of fragments specific by Data Warehouse Analysis.

{Find solution = True; Print ("Candidate Level is", x, y)Goto End; }

Else

If (y = E) then { x+ = 1; Find solution = False ; Goto (2); }

Else

{ y+ = 1; Find solution = False ; Goto (2); }

End.

7- Query Execution Time

The query response time in the online analytical processing and decision support systems is crucial and very important. Therefore, the implementation of

the query on the fragmented data provides fast response and thus speeds up decision-making. For example, ten complex queries were carried out to the centralized data warehouse and then calculate the rate of query response time and compare it with the response time of the same queries performed on fragmented data (data mart) to find that the rate of implementation of the query on the central data warehouse takes six times more than the time spent of the query implementation on the fragmented data (data mart) as shown in following table (1) and figure (7).

Table 1: Query Response Time

Query Number	Characteristics	Response time in Data Warehouse in second	Response time in Fragmented Data DM in second
1	3 join, 3selection	0.0094	0.0010
2	4 join, 4selection	0.0095	0.0011
3	5 join, 5selection	0.0096	0.0012
4	6 join, 6selection	0.0096	0.0013
5	7 join, 7selection	0.0097	0.0014
6	8 join, 8selection	0.0098	0.0015
7	9 join, 9selection	0.0099	0.0015
8	10 join, 10selection	0.0100	0.0016
9	11 join, 11selection	0.0102	0.0018
10	12 join, 12selection	0.0105	0.0020
Total		0.0975	0.0144
Average		0.0972 / 0.0144 = 6.770	

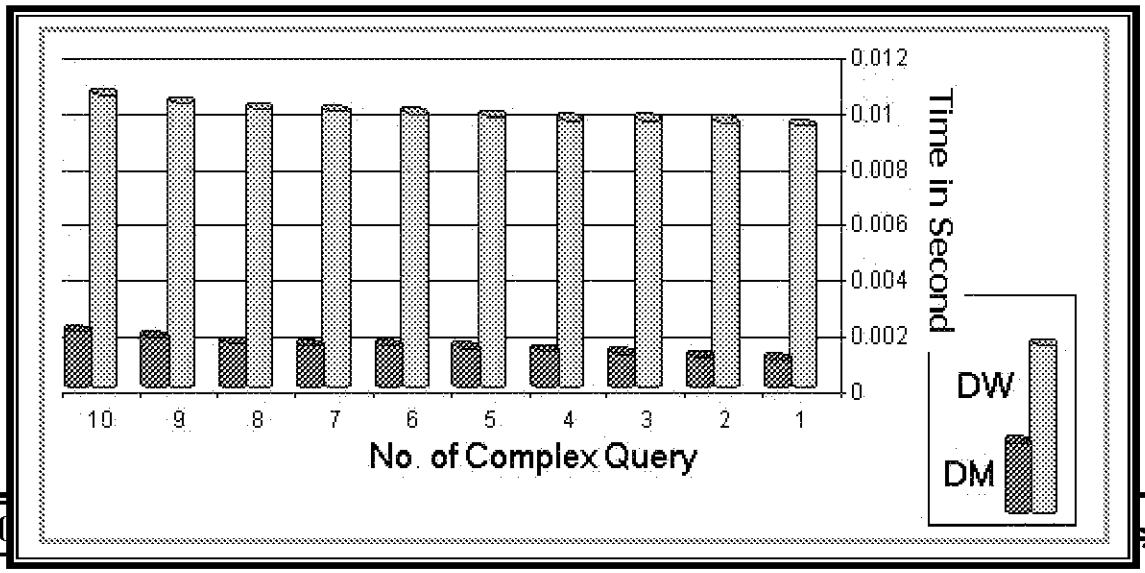


Figure 7: Query Execution Time Comparison

8- Conclusions

The fragmentation by a proposed framework ensures accepted number of resulted fragments (sub star schema) and thus facilitate the management, control, and maintenance of data warehouse as a small data mart distributed in a set of relatively cheap computer network (nodes) that be lower costs, higher reliability and reduce the need to use special central computer to build a large centralized data warehouse. On the other hand the OLAP query in distributed data warehouse will divided into a set of (sub query) sent to the corresponding nodes in network to be executed in parallel, so thus reduce the query execution time and make OLAP query more efficiently.

REFERENCES

[1] Moeller, R.A. Distributed Data Warehousing Using Web Technology: How to Build a More Cost-Effective and Flexible Warehouse. AMACOM American Management Association. New York. 2001.

[2] Bernardino, Jorge and Henrique Madeira. "Experimental Evaluation of a New Distributed Partitioning Technique for Data Warehouses ". IEEE. 2001.

[3] Rainardi, Vincent. Building a Data Warehouse: With Examples in SQL Server. Springer-Verlag. New York.2008.

[4] Inmon , William H. Building the Data Warehouse. Fourth Edition. Published by Wiley Publishing, Inc., Indianapolis, Indiana. 2005.

[5] Kimball, Ralph , and Joe Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley Publishing, Inc. Printed in the United States of America. 2004.

[6] Costa, Marco and Henrique Madeira. "Handling Big Dimensions in Data Warehouses Using the DWS Technique". ACM. 2004.

[7] Zhou, Shuigeng and other. "Hierarchically Distributed Data Warehouse". IEEE Computer Society Press. 2000.

[8] Wrembel, Robert, and Christian Koncilia. Data Warehouses and OLAP: Concepts, Architectures and Solution. IRM Press. Published in the United States of America. 2007.

[9] Bellatreche, Ladjel and Other. What Can Partitioning do for your Data Warehouse and Data Marts. IEEE Computer Society Press. 2000.

[10] Noaman, Amin Y. and Ken Barker. A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse. ACM. 1999.

[11] Cormen, Leiserson, and Rivest "Introduction to Algorithms" 1990, Chapter 17 "Greedy Algorithms" p. 329.

[12] M. G. C. Resende. "Greedy randomized adaptative search procedures". Technical Report 98.41.1, AT&T Labs. 1998.

[13] Slav`ık. "A tight analysis of the greedy algorithm for set cover". In Proceedings of the 28th ACM Symposium on Theory Of Computing (STOC), Philadelphia, PA, USA. 1996.

المستخلص

أصبحت أحجام مستودعات البيانات كبيرة جدا وتنفيذها عادةَ يتطلب استخدام خـــوادم مكلفة للغاية، لذلك برزت الحاجة الى اســـتخدام مـــستودعات البيانـــات الموزعـــة لتلبيـــة احتياجات المستخدمين من شركات ومؤسسات علمية وتجارية.

ان عملية الانتقال من استخدام مستودع بيانات مركزي الى استخدام مستودع بيانـــات موزع يتطلب عملية تجزئة المستودع المركزي الى مستودعات اصغر حجماً توزع ضمن شبكة من الحواسيب. هنالك العديد من الخوارزميات المستخدمة في عملية التجزئـــة، فـــي هذا البحث اقترح استخدام مبادئ الهرمية المتضمنة في مستودع البيانات كأســـاس لعمليـــة التجزئة ، حيث تم اقتراح اطار عمل يتضمن تمثيل خاصية الهرمية على شـــكل شـــجرة مقلوبة تتكون من عدة مستويات بالاعتماد على نوعية البيانات الموجودة ضـــمن مـــستودع البيانات عندها نتمكن من اجراء عمليـــات البحـــث ضـــمن الـــشجرة باســـتخدام إحـــدى خوارزميات البحث للوصول الى افضل مستوى يمكن اعتماده في عملية التجزئة .

في اطار العمل المقترح استخدمت خوارزمية البحث الجشعة (Greedy) بـــسبب مـــا تمتلكه من مزايا لدعم عملية اختيار المستوى المناسب الذي يستخدم في عملية التجزئة.

اخيرا ان استخدام مستودعات البيانات الموزعة كمستودعات صغيرة (Data Mart) وموزعة ضمن مجموعة من الحواسيب ضمن الشبكة تعد من اولى محاولات حل مـــشاكل الحجم والاداء والتكلفة الباهضة لأجهزة الخوادم المـــستخدمة فـــي مـــستودعات البيانـــات المركزية.