1. Introduction

Over the past few decades, each corporation depend massively on decision making system for business growth and stay competitive in business community. This type of decision making system massively depends on database and some kind of connected analysis tools for success. To approaching this objective, data of each corporation come from different independent sources are arranged and stored in special purpose database generally called DW. To enable OLAP data warehouse adopt Extract Transform and Load technique to combine heterogeneous info in corporation. DW are large database used by business analysts to extract useful information for make quick and more desirable decision. Old DB's like OLTP used to access limited rows but at the same time provide read and write access. On the contrary DW environment (OLAP oriented processed complex queries quickly and efficiently) contains large data sets that are read only and not frequently updated. DW OLAP process normally implement in many different ways e.g. filter, aggregation, count, min, max, sum, average and group by data also some customized aggregation operations. These OLAP queries are normally very complex and consume so much time to complete because data warehouse contains extremely large data often more than 2 billion rows for analysis and updated only periodically. In data warehouse environment, processing these queries faster is very critical issue. Most of time the result of these complex queries are falsy and unable to find relevant facts.

To speed up query response time and provide most beneficial visualization, OLAP application generally designed in such a way that provided multidimensionally modeled for viewing the data in many business aspects. For example, in pharmaceutical sales data warehouse, sales amount, sales volume and sales geographic area are some dimensions in which user show interest. Generally four types of analytical in OLAP include roll up or aggregation (decreasing detail or e level of aggregation like cities dimension rolled up country) and drill down (expending detail or level of aggregation like quarter dimension drill down in month) onward some other dimension dice and slice (projection and selection like filter cube by quarter one or include more dimension like filter cube by quarter and product) and pivot (rotate the dimensions give alternate view of data). In OLTP environment if we are trying to view data in multidimensional against the operational databases or traditional databases the result is unsatisfactory performance. Additionally, in operational databases data maybe missing that require for decision support; for example, if we want to make future prediction we need historical data, whereas operational databases deals only current data. Despite this fact, in data warehouse data come from different heterogeneous sources including several operational databases and external source like stock markets etc. Because data come from many different platforms that maybe inconstant, raw or badly formatted. That's why in multidimensional data cubes and operations generally require special data arrangement, different techniques for access and implementation. Old DBMSs that targeted for OLTP doesn’t need special methods because data is stored in it is highly normalized. But in data warehouse environment many research issues identified by [Wu97]. Data warehouse are needed new design approaches and tool for aligning RDB and multidimensional, aggregation support and

cost for new design models that need in design steps. Others issues like data warehouse architecture design in such a way that maintain efficient storage and easy way to retrieve large amount of data. Data warehouse maintenance like ETL process for data clean and cost maintenance. Operational and optimization issues like scanning large table and also space and time issues for multi dimension query. There are three types of storage modes in OLAP for store most used data in multi-dimensional cubes are MOLAP, ROLAP and HOLAP. The difference between MOLAP, ROLAP and HOLAP show in Table 1.

Table 1 Difference between MOLAP, ROLAP and HOLAP

MOLAP	ROLAP	HOLAP
MOLAP stand for Multidimensional online analytical Processing.	Stand for Relational Online Analytical Processing.	Stand for Hybrid Online Analytical Processing.
Data is stored in multidimensional cube.	Data is stored in relational database.	Most recently data used in multidimensional cube and detailed data stored in relational database.
Query response time in MOLAP is fast.	Query response time in ROLAP is slow.	Query response time in HOLAP is medium.
Processing time is fast.	Processing time is slow.	Processing time is slow.
Latency is high.	Latency is slow.	Latency is medium.
Storage space requirement is medium.	Storage space requirement is Large.	Storage space requirement is small.

Data warehouse focus on most recent data for analysis and mainly focused on multi view maintenance issues and query performance. MOLAP is suitable choice for recent analysis because query response time is and storage space requirement is less as compared to ROLAP.

During my search, we found many papers as well as several informal reviews within the area of OLAP query optimization in warehouse. In [1] purposed three techniques for OLAP query optimization. In which two of the algorithms focus mainly on how to produce a global plan from several connected local plans. The third algorithms focus mainly on creating a good overall plan without creating a local plan. The essential task in evaluating several related dimensional queries is to recognize and use common secondary tasks between queries. They provide several useful operators for this task as part of the relational implementation of dimensional data sets. Remember that the basic operation in this area is a star union. They consider two methods of star union: star union based on hashing and index. Each selective startup method, the index-based method is the best option. For non-selective queries, hash-based star union is a good solution. The local two-phase optimal algorithm (TPLO) is perhaps the most natural and straightforward way of tackling the problem of simultaneous optimization of dimensional queries. TPLO splits an MDX expression into multiple SQL queries. Because all database systems support OLAP queries per pre-calculation group, TPLO selects separately the best pre computed (materialized) group for each of these component sub queries. Once the "group by" goal has been determined for each component request, TPLO uses the SQL optimizer to generate the best plan for the queries. Finally, it generates a global plan by combining as much as possible the common tasks between the consultation plans, using the operators. The inspiration of the two-stage extended local greed (ETPLG) is to solve the disadvantages of the optimal two-phase algorithm (TPLO). TPLO always uses the best local plan for each multidimensional expression query and combines the plans at the query tree level to generate a global plan. The TPLO restriction is that you cannot create the base table interchange if the optimal plans for independent queries use different input tables. In some cases, it may be preferable for independent queries to use suboptimal base tables so that you can share tasks as they run. The two-phase Extended Local Encoding Algorithm (ETPLG) enables the heuristic to increase the exchange of base tables between multiple queries. But the construction of the overall plan by the ETPLG algorithm may not yet be a good overall plan. The reason is that ETPLG never "changes your mind" on which base table to use for a query. To find a better solution or a global plan, they extended the ETPLG to the Global Greedy (GG) algorithm. The GG algorithm increases the overall plan by attaching a new query to the plan, one at a time. The difference between the two lies in the fact that the GG algorithm allows a class to modify its shared base table to include the new group, which is not allowed in the ETPLG algorithm.

Ref. [10] proposed that the solution motivated by the architecture of the distributed data warehouse and traditional RR partitioning algorithm. Before presenting [10] approach, let's briefly explore the star schema and some key features of a data warehouse implemented in a relational repository. In OLTP environments, a highly normalized scheme provides excellent performance and efficient storage because each transaction is retrieved only a few records. The star schema offers same advantages for data warehouse where most requests add huge amounts of data. A star schema, as

illustrated in Figure 1. The fact table is the heart of the star schema and contains mainly fact data

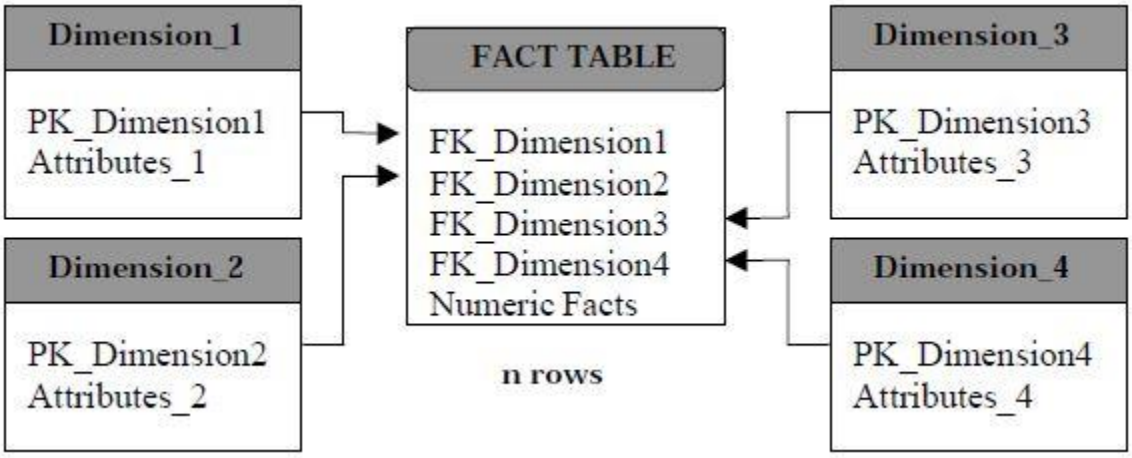

Figure 1 Example of star schema [10]

Such as counts and quantities. Typically, one or more star schemes invent a data warehouse and each star corresponds to a business process. The fact table represents most of the space occupied by all the star tables in most cases [5]. However, this fact table is fully normalized form, which means that it genuinely represents the most productive relative method (as far as storage space concerned) for storing facts. Measurement charts are very standardized, but generally represent a small portion of storage space in the star schema. The star schema additionally optimized for implementing complex queries that total a huge amount of data in fact table. To solve this issue related with distributed data warehouse, they propose a new approach by distributing the data warehouse. Their approach is motivated by the RAID-3 scheme [7] and so they call it DWS. This is because a small cube size (tapes) is used in RAID-3 that all disk access is distributed evenly across all disks in the array and increases disk access speed of N discs in the array. In this way, one of the key objectives of creating tape in the data warehouse is to improve the performance of query execution by adding N of computers in a DWS system (see figure 2).

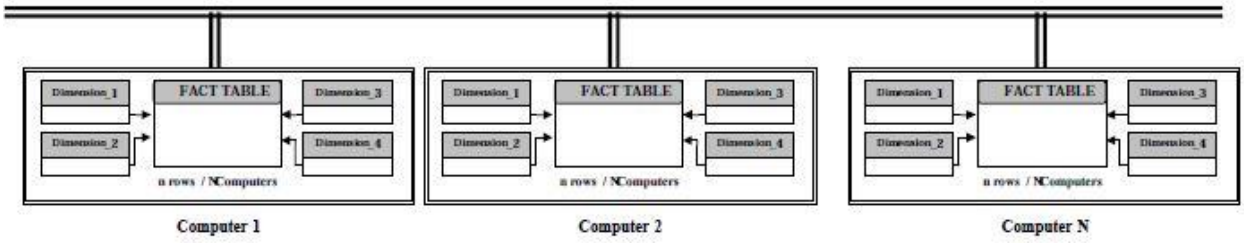

Figure 2 Data Warehouse Striping [10].

In DWS, data in fact table are spread over an arbitrary number of computers and queries are processed simultaneously on each computer, ensuring almost linear acceleration and significantly improving the response time to show result. Figure 3 show DWS cycle using 5 computers.
By spreading the often large fact table (this is usually more than 90% of the space occupied by all tables in the star schema) by many computers, might possible to grab exceptional gain in the speed of the query.

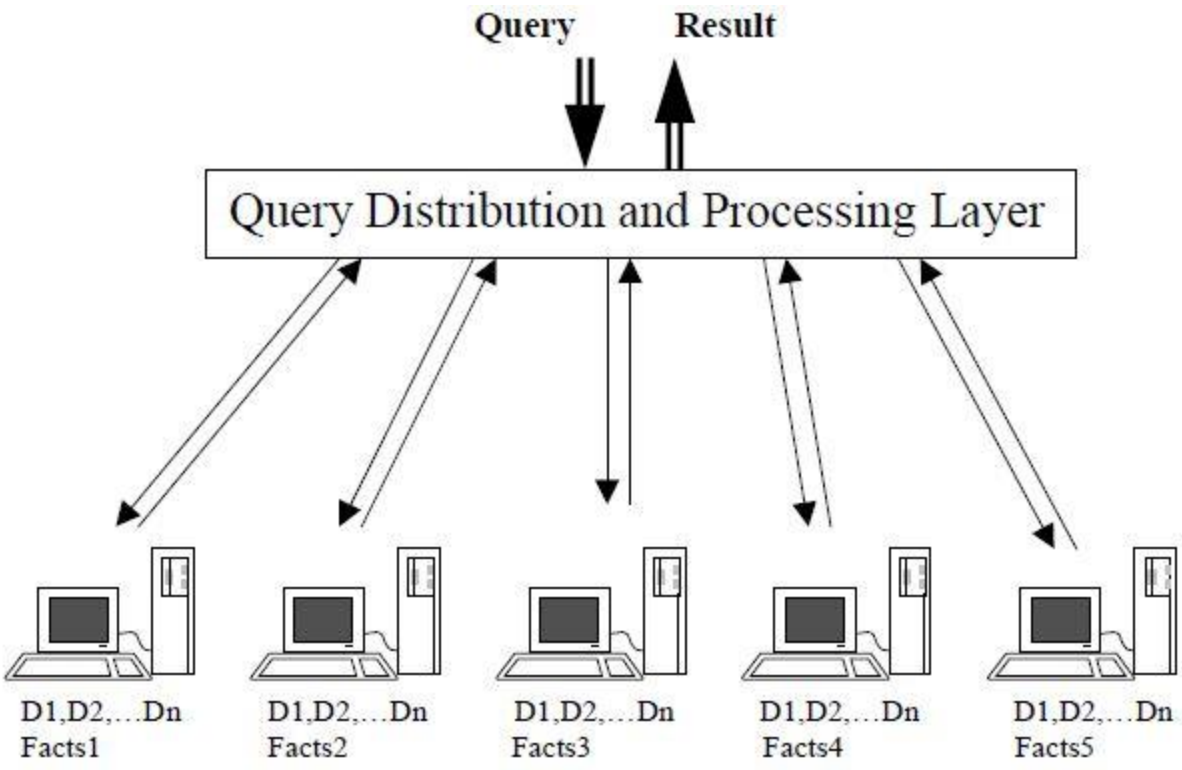

Figure 3 Distributed query execution using 5 computers [10].

DWS system has the following four phases:

1) **Distributed loading phase:** dimensions are distributed in all computer and fact table rows equally divided in RR fashion between all systems.
2) **Query Distribution phase:** query is divided into n partial queries and each computer execute independently.
3) **Query processing phase:** query must be execute in all computers at same time because distributed query is the key functionality of this approach.
4) **Result merging phase:** all computers return small partial result. In this phase merge all result in single set and return to end user.

Gupta and Mumick [6] have provided an A * heuristic algorithm which provides the selection problem in terms of maintenance cost perception and confirms that the heuristic A * can guarantee an optimal solution. There is a representation of * heuristic algorithm in the FIG. 3. The heuristic algorithm uses A * inverse order to obtain a series of relevant views. It explains an IT binary tree with the leaf wedges as the candidate solutions to this problem. At each stage of the search, the heuristic A * evaluates the advantages of the remaining branches downhill and selects the branch of the significant benefit to be reduced. Each tip of the binary search tree is labeled (Nx, Mx) (Mx Nx), where Mx is a set of opinions selected to highlight them and is considered to correspond to all Nx requests. While the A * heuristic algorithm can guarantee the

search for optimal resolution, it is an exponential algorithm in the worst case scenario and implementation of time can be too time-consuming.

The quality and operability of the evolutionary algorithm will be examined by [2] and will report on their results. Ref. [2] they suggest an evolving new algorithm to select the problem of selection of maintenance costs. The restrictions are placed in the algorithm through a stock-ranking procedure. In evolutionary algorithms, a string string genome group is generated randomly. It is the initial population. Each genome illustrates a possible solution to the problem to be solved. The length of this genome is the total number of additives in the cross section; 1 and 0 suggest that the vertices must be achieved or not alone. Genomics can be normalized as a genome = (x1, x2, x3 ..., xn), where the total total of the network peaks. At this point, Xi = 1 is selected if the scene is selected Vi and Xi = 0 if the view is not selected Vi. During a process of transition and modification, senior candidates will survive and the poor candidates will die. Following this, we will initiate stock-based sanctions and classification methods, and provide a new evolutionary algorithm.

[8] Propose different revision techniques to ensure high performance information during storage. DWs typically encounter SQL aggregation [8]. They tried to rewrite raw questions and run them on live databases and an optimized version of the previous ones. All the techniques used in this article explain in detail the use of statistics and questions on the use of assets.

1) **Deter FTS:** RTS retrieves each piece of information from the board, for example. it stimulates a huge amount of I / O on the disk. This type of access will always be a reference to the database as it relates to the recovery of billions of records. So avoid the fact that RTS in the database maximizes the issue strongly.
2) **Enforcing Multi-leveled Partition of Indexes:** This approach will bring the following benefits in terms of performance. (Loading data via a DML operation, loading data via a DDL operation, retrieving data via a SELECT statement). Maintenance also has advantages.
3) **Read-Only Tablespace(s):** In this approach, if board space is read only, the DBMS engine will avoid the read-through stability protocol, reduce overhead costs, and have a faster implicit throughput.
4) **Precocious Data Buffer Administration:** In this approach, by applying different sizes, we could designate objects / stores to separate data buffers and ensure that the data set always gets revalued data. We also redistributed the memory frames between db_cache_size (cache memory) and the pga_aggregate_target region (unique memory for the user) to DW promotion. Therefore, we must keep track of memory and memory specific to the user.
5) **Materialized View (MV):** Impose a copy of the relevant comments and we will allow ourselves to prepare pre-summarized tables in advance. Above all, this allows VMs to rewrite a question. Any question can then be asked on the preliminary summary and will be automatically examined to transmit the aggregated view. This prevents the RTS from being necessary and expensive (reducing I / O disk) instead of trying to optimize the rewrite issue. In fact, the database administrator must a) Report a time series to follow a survey path

through the ad workload, and then carefully build the best performing MVs. b) Examine the possibility of later creation of any VM, hence the reduction of disk I / O.

6) **Deem Star Query Optimization:** Star request tuning allows complex DDS queries to be processed more quickly. A bitmap index must be generated on each foreign key column of the fact table. We must also allow to start with the modified rewrite.
7) **Caching the Data:** Smaller and recurring dimension tables must be deposited to reduce I/O and delay.
8) **Multiple Block-Sizes:** In Mixed Block Size, each index of a database must be processed by means of a range analysis and the objects to be processed by means of RSI (s) or IFS (s) must be placed in a database block size of 32KB. Multiple block application can significantly improve I/O but requires knowledge of the I/O environment. Smaller block access requires random access, reduces block variance, and reduces the amount of overhead costs. RTS and sequential access to the size of the big block.

In [9] they develop 2 greedy algorithms. In these algorithms, take the top-down implementation plan by identifying the best ideas at each step, rather than the most promising question. The first algorithm, called Best View Firs (BVF), is simple: rather than building the overall implementation plan by asking the questions themselves (a bottom-up approach), they use a top-down approach. In the BVF at each cycle, the best plan is selected with the help, on the basis of a saving indicator, and all the questions covered by best view and not yet assigned to another opinion, in the overall plan. This process only ends when all inquiries are completed. The security metric is defined as follows: Let v ε MV and let VQ ⊆ Q be a series of queries that can be answered under v. MV and Cmin (q) = min (C (q.ui)) 1≤i≤ MV | [11] response using a relevant point of view. The complexity of the algorithm is polynomial.

The BVF will usually develop a small number of series, each series sharing the same star may have several questions. To solve this problem, they used other multi-algorithms. Version of the BVF, called MBVF (Multilevel Best View First). The idea of this new algorithm is that we can reconsider the arrangement presented by BVF by distributing some of the issues that appear to be lower in the network (ie more general comments). Lower the costs. The MBF First MBF First appeal asks BVF for a basic production arrangement, called the Lower Plan. At this point, he chooses in the lower plane the view v which is higher in the network (i.m., the widest view). It is not possible to respond in more detail to requests for information and to re-call the VFS for further comments and inquiries to build a new plan. V and its inquiries refer to the complete plan. If its cost is not expensive, the correct plan is the next new plan that stops the algorithm. In the strictest case, the algorithm was stopped after analyzing all the comments. Therefore, the complexity is O (| Q | 2. | MV | 2). In MBVF, it is possible to encrypt the plan more expensive when scenes are more accessible, but at the same time, the BVF setting cost will be lower.