**Chapter 1: Introduction**

Over the past few decades, each corporation massively depends on decision making system for business growth and stay competitive in the market. This type of decision making system massively depends on the 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 the corporation. DW very 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. A data warehouse 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 the data warehouse environment, processing these queries faster is a 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 shows 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 the month) onward some other dimension dice and slice (projection and selection like a filter cube by quarter one or include more dimension like a filter cube by quarter and product) and pivot (rotate the dimensions give alternate view of data). In OLTP environments if we are trying to view data in multidimensional against the operational databases or traditional databases the result is unsatisfactory performance. Additionally, in operational database data, maybe missing that require for decision support; for example, if we want to make a 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. A data warehouse are needed, new design approaches and tool for aligning RDB and multi-dimensional, 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. 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 Difference between Multidimensional, Relational and Hybrid OLAP

|  |  |  |
| --- | --- | --- |
| **MOLAP** | **ROLAP** | **HOLAP** |
| MOLAP stand for Multidimensional Online Analytical Processing. | Stand for Relational Online Analytical Processing. | Stand for Hybrid Online Analytical Processing. |
| The information is stored in a multi-dimensional cube. | Information is saved in a relational database. | Most recent data used in multidimensional cube and exhaustive data preserved in the relational database. |
| Query response time in MOLAP is fast. | Query response time in ROLAP is slow. | Query response time in HOLAP is medium. |
| Time to respond is fast. | Time to respond is slow. | Time to respond is slow. |
| Latency is high. | Latency is slow. | Latency is medium. |
| The space requirement is medium. | The space requirement is Large. | The 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 a fit choice for recent analysis because query response time is and storage space requirement is less as compared to ROLAP.

**Chapter 2: Critical Review of Related Literature**

During my search, we found many papers as well as several informal reviews within the area of OLAP query optimization in Data Warehouse. In (Yihong Zhao, 1998) purposed three techniques for OLAP query optimization. In which two of the proposed techniques focus mainly on how to produce a universal plan from individual connected narrow plans.

The third proposed technique focus mainly on creating a good universal plan beyond creating a narrow plan. The essential responsibility in appraising several relevant dimensional queries is to recognize and use frequent secondary tasks between each query. They provide several useful operators for this task as part of the relational implementation of dimensional data cubes. Remember that the essential process 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-choosy inquiry, the best solution is hash based star union. The local two Phase technique is maybe the nearly common and straightforward plan of tackling the issue of concurrent optimization of OLAP queries. TPLO splits an MDX query into many SQL queries. Because all DB systems support OLAP queries per pre-calculation group, TPLO selects separately the best pre computed (personalize) group for each of such sub request modules. Once the "group by" goal has been determined for each component request, to make the right request project, TPLO utilizes the SQL optimizing compiler. Finally, it accomplishes a universal 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 fix the disadvantages of the best two-stage technique (TPLO). TPLO utilizes the best local strategy for each multidimensional expression and combines the plans at the level of the query tree to create a universal project. The two-stage technique (TPLO) restriction that’s it you can’t build the core table interchange if the prefect ideas for independent queries use various input tables. In certain situations, it can be preferable to use sub optimal base tables in independent queries so that you may share your responsibilities while running. The two-phase Extended Local Encoding Algorithm (ETPLG) enables the heuristic to boost the exchange of core tables between various requests. But it may not yet be a good overall plan to build the overall plan by the ETPLG algorithm. The explanation is that on which base table to use for a query, ETPLG never "change your views". They extended the Extended Two Phase Local Greedy Algorithm to the Global Greedy technique to find a suitable approach or universal plan. The GG algorithm increases the overall schedule by attaching a new one query to the schedule one by one. The gap in the two lies in the fact that the Global Greedy algorithm grant a category to update its common core table to build the new panel, which is not allowed in the ETPLG algorithm.

The (Bernardino, 2000) proposed that the solution motivated by the architecture of the distributed data warehouse and traditional RR partitioning algorithm. Before presenting (Bernardino, 2000) technique, let's discuss the star framework and some key features of a data warehouse in a structured database. In OLTP domain, extremely normalized structured provides finest achievement and adequate cache because each transaction is retrieved only a few records. The star schema offers the same advantages for data warehouse where most requests, add huge amounts of data. The heart of the star schema is fact table and contains mainly fact data such as counts and quantities. Typically, more than one star schemes invent a DW and all-star corresponds to a trade progress. In most cases, the fact table occupies most of the area that all tables share in star schema (Kimball, 2011) .After all, this fact table is a fully structured form, which means that it genuinely shows the batter productive relative method (in term of storage space concerned) for storing facts. Measurement charts are very standardized, but a little portion of storage space is usually shown in the star schema. Similarly, the structure of star pattern optimized for implementing multiplex requests that total a huge data volume in a fact table. To solve this issue related to distributed data warehouse, they propose a new approach by distributing the data warehouse. Their perspective is motivated by the RAID-3 pattern (Patterson, 1988), so they call it DWS. This is because RAID-3 uses a small cube size (tapes) that all disk connection is uniformly distributed evenly across each disk in the collection and increases disk connection pace of M discs in the collection. One of the key objectives of creating tape in the data warehouse is thus to enhance the performance of request execution by adding M of computers in a Distributed Warehouse Striping network.

In DWS, data in the fact table are spread, the number of computers is arbitrary and queries are processed simultaneously on each computer, ensuring almost linear acceleration and significantly improving the response time to show results.

By spreading the often large fact table (this is normally more than 90% of the area that all-star tables occupy) many computers, might possible to grab exceptional gain in the speed of the query.

There are four phases of the DWS system:

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

Gupta and Mumick (Gupta, 1999) have provided an A \* heuristic algorithm which provides the selection problem in terms of maintenance cost perception and confirms that’s the heuristic A \* can promise 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 edges as the competitor solutions to solve this problem. The heuristic A \* assesses the benefits of the remaining branches downhill at each stage of the search and chooses the group of the major benefits to be reduced. Every point of the binary selection 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 (Roy, 2000), in the worst case scenario, it is an exponential algorithm and time-consuming implementation can be too far.

The quality and operability of the evolutionary algorithm will be examined by (Yu, 2003) and will report on their results. They (Yu, 2003) 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 genome group is produced randomly. It is the first community. Every genome illustrates a possible plan to solve the problem. This genome’s length is the total number of additives in the cross section; 1 and 0 suggest that the vertices must be achieved or not alone. Genomics could it be normalized as a genome = (x1, x2, x3 ..., xn), where the 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, good candidates are going to survive and poor candidates die. We would than launch stock-based penalties and classification methods and provide a new evolutionary algorithm.

(Myalapalli, 2015) Propose different revision techniques to ensure high performance information during storage. DWs typically encounter SQL aggregation (Myalapalli, 2015). 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 fetch each piece of information from in the board, for example. It stimulates a large volume of I / O on the disk. This kind of control will always going to be a reference to the database such 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 is going to bring the relevant advantages in terms of performance. (Data loading through a DML operation, data loading through a DDL operation, data recovery through a SELECT statement). Maintenance also has advantages.
3. **Read-Only Table space(s):** In this approach, if board space is just read it, the DBMS engine may avoid the read-through stability contract, reduce expensive costs, and have a quick 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 block of memory between db\_cache\_size (cache memory) and the pga\_aggregate\_target region (the user’s unique memory) to DW promotion. Therefore, we must keep record of specific memory 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 administrator of the database must a) Report a time sequence to follow a survey path create the best performing MVs carefully through the ad workload. b) Examine the possibility of later creation of any VM, hence the reduction of disk I / O.
6. **Deem Star Query Optimization:** Star request pitch allows complicated DDS queries to be processed more quickly. An index of bitmaps must be generated on each foreign key column of the fact table. We also need to allow the changed rewrite to begin.
7. **Caching the Data:** To reduce I / O and delay, smaller and recurring dimensional tables must be deposited.
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 the storage block size of IFS(s) must be 32 KB. Several block application could be significantly improve I/O but it does require research of the I/O environment. Lower level block access requires access randomly minimizes block variance, and reduces the amount of overhead costs. RTS and sequential access to the size of the big block.

In (Kalnis, 2003) 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 technique, called Best View Firs (BVF), is simple: they use a top-down approach instead of constructing the overall implementation idea by asking the questions themselves (a bottom-up 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 analysis are completed.

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. The 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 comprehensive vision. If the cost is not expensive, the correct vision is the next new plan that stops the technique. In the strictest case, the technique was stopped when all the comments were analyzing. 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.

**Chapter 3: Comparative Study of Related Literature**

Below mentioned table determines the comparison between various search papers that we have gone through:

Table 2. Comparison OLAP Optimization Techniques

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Research Topic | Authors | Proposed Solution | Strengths | Limitations | Operator Used | | Total Cities | Remarks |
| Simultaneous Optimization and Evaluation of Multiple Dimensional Queries | (Yihong Zhao, 1998) | TPLO, ETPLG, GG | TPLO splits into several SQL queries in multidimensional expression. TPLO pick the best pre-computed group for each of these sub queries components. To create the best strategy for each query, TPLO uses SQL Optimizer. Then combining the common tasks with operators to create a global strategy.  ETPLG is overcoming TPLO's problem. ETPLG gradually establishes a globally plan by greedily inserting queries. The algorithm determines the cost of running it individually to each query, and the cost of sharing a view with a query previously selected. The plan will be selected with the lowest cost.  Major difference between GG and ETPLG is that GG allows you to change the shared view of a group of queries to include a new query if this results in lower costs. | The TPLO algorithm combination of optimal local plans does not promise an optimized global action plan.  The ETPLG enables the heuristic to boost the exchange of core tables between various requests. But it may not yet be a good overall plan to build the overall plan by the ETPLG algorithm. | | Hash based, Index based and star based join. | 14 | When the quantity of materialized views expands, none of the above algorithms grow effectively |
| Multi-query optimization for on-line analytical processing | (Kalnis, 2003) | BVF, MBVF | The concept motivating Best View Firs algorithm, they use a top-down approach rather than constructing the global execution plan by inserting the queries one after another (bottom-up approach).  IN MBVF technique first of all called BVF to generate an initial plan then its check which view is larger in term of the lattice. If the cost is lower than the initial plan, the framework for new Plan will continue, otherwise the algorithm will end. | BVF starts to build a smaller amount of sets in which each set produces a lot of queries which share its same join star. This aggression can contribute to higher cost plans if there are many queries and a few materialized views.  The ETPLG enables the heuristic to boost the exchange of core tables between various requests. But it may not yet be a good overall plan to build the overall plan by the ETPLG algorithm. | | N/A | 25 | When the number of materialized views is low, BVF appears to produce bad plans. |
| Efficient OLAP query processing in distributed  data warehouses | (Akinde, 2003) | Skalla algorithm | A prominent feature of our method is that very few partial results are delivered,   never send of the detailed information. We are proposing a number of optimizations to reduce both in the traffic of synchronization and the local computation in each venue. | This approach works only in a distributed environment and study is based on TCP data. | | GMJD |  | They use GMJD operator for optimization to reduce computational costs. |
| Data Warehousing and OLAP: Improving Query  Performance Using Distributed Computing | (Bernardino, 2000) | DWS Algorithm | Equally distribute the rows that contains fact table in round robin fashion by the M number of PC’S and improve the response time of the query.  Provides load balancing for al computers. | A factor of significance is the accessibility of computers that form a distributed data warehouse. We are testing strategies to make the DWS device stable enough to cope with one of the more computers currently unavailable. | | Star Schema | 24 | DWS system Provides excellent performance and flexibility even in large data warehouses. |
| Optimized Data Indexing Algorithms for OLAP Systems | (Bornaz, 2010) | N-Tree | N-tree index is a more generalized version of b-index tree. N-Tree index offer n dimensional data for optimization while B-Tree only offer uni-dimensional data.  The n-Tree index storage is much smaller than the B-Tree index space required. | The cost of the N-Tree index query is lower than the cost of the B-Tree index query, except when the number of cells is very small. | | Projection | 10 | The effectiveness of the n-Tree index becomes even more obvious when the number of cells in the cube increases. |
| Data Warehouse Performance Optimization  Implementing DHE Algorithm in Mortgage  Backed Security using Mondrian | (Momin, 2015) | DHA algorithm | Eliminates the original key from the fact tables and dimensions of that table, also reduces storage space, reducing the multi-table joins and element component is encrypted by mitigating the binary code through top to bottom for all dimensional structural properties. | Not applicable in distributed environment. | | Star schema | 10 | It reduces the disk I / OS considerably and improves the handling of OLAP queries. Because dimensions data is stored in the fact table in binary format. |
| The MD-join : An Operator for Complex OLAP | (Chatziantoniou, 2001) | MD-join | As conventional aggregation does, the MD-join doesn't really group rows for aggregation. Alternatively, a base value relation defines the collection of throughput rows, which also includes a selection of "group" keys. From now on, we will generally refer to base values rather than groups. | Algebraic expression transitions leading to a broad range optimization problem in md join. | | N/A | 21 | MDjoin technique offers immense experience when it comes to queries for decision aid. |
| Tree Based Indexes vs. Bitmap Indexes: A Performance Study | (JÜRGERNS, 2001) | Bitmap Index | Mainly focus on four points, optimal with respect to time, space, equality and range. | They can't be used for small tables.  The Cause deadlock  There is an overhead for maintaining this bitmap index if there is a huge number of records. | | selection | 15 | Provide low latency time to improve performance. |
| Optimized Data Indexing Algorithms for OLAP Systems | (Bornaz, 2010) | Tree Index | Access data randomly and sequentially.  Group by used for aggregation.  Search result is faster | They can't be used for large tables | | Selection | 17 | N/A |

**Chapter 4: Conclusion**

In this paper we have discussed different types of approaches that are: GG, TPLOG, ETPLG, A\*, B-Tree Index, Bitmap Index, DHA algorithm, DWS algorithm and Skalla algorithm. These tools are used in data warehouses to deal with data present in gigantic amount to make it useful for data analysts and many other individuals. When we talk about **ETPLG**, the cost of query ‘q’ is evaluated by executing it individually and by sharing the previously selected query with a view and then evaluating its cost. The plan occurring with the lowest cost is selected, though selecting the order of queries insertion affects the total cost, ETPLG uses ascending order and thus selecting the top most query in the lattice. Global Greedy (**GG**) works in the similar manner as ETPLG but it mainly focuses on sharing a view with group of queries to include a new query insertion only if it leads to lower cost. While **TPLOG**’s focal point is selecting materialized view ‘v’ for each query ‘q’ in a way that lowers its cost and uses SQL query optimizer for an optimal plan for query ‘q’ then the algorithm merges the subtasks at each query and merges them, but it doesn’t guarantee an optimal solution. The **A\*** heuristic that we have discussed in this paper describes how it searches to find the shortest path by ranking the alternatives with the pre informed cost. The **B-Tree Index** represents data in uni-dimensional form to index a spatial data but according to our finding the N-Tree indexing can represent data in n-dimensional form making it more suitable for spatial data. Coming towards the Dimension Hierarchal Encoding (**DHE**) algorithm, we gave determined so far that the binary code of all the dimensional attributes is computed for each and every dimensional member. Doing this reduces the multi table joins of fact and dimensions table and lessens the storage consumption by substituting the key which ultimately leads to fast computation. In **DWS** algorithm the star schema is sub-divided into ‘n’ number of computers having same star schema at each computer. This allows the input data to be processed parallel and generate results by merging results at the end, generated by the each star schema at each computer. This does makes the computing faster but also increases the cost of assets being used to make it work. The **Skalla** system consisted of coordinators and skalla sites. The skalla sites fetches the partial information and transmits them back to the coordinators which in turn synchronizes the local fetched data into global results and may return back to the skalla sites. Computation and query processing is done at the local sites. **Bitmap Index** Algorithm works in an adhoc fashion and the situations which can call for it includes: the column have low cardinality meaning less distinct values as compared to rows or say that the indexed table consists of read only restriction. Data is represented in the form of a column in binary where ‘1’ represents data availability and ‘0’ represents no record for the selected column. We can use bitmap indexing on two or more than two columns but it’s not considered to be the best practice, also it comes with restrictions like you can only use concatenated bitmap index when all the columns are filtered. Therefore, separate bitmap indexes are more flexible.

**References**

Zhao, Y., Deshpande, P. M., Naughton, J. F., & Shukla, A. (1998). Simultaneous optimization and evaluation of multiple dimensional queries. *ACM Sigmod Record*, *27*(2), 271-282.

Bernardino, J., & Madeira, H. (2000). Data warehousing and OLAP: improving query performance using distributed computing.

Kimball, R., & Ross, M. (2011). *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons.

Patterson, D. A., Gibson, G., & Katz, R. H. (1988). A case for redundant arrays of inexpensive disks (RAID) (Vol. 17, No. 3, pp. 109-116). ACM.

Gupta, H., & Mumick, I. S. (1999, January). Selection of views to materialize under a maintenance cost constraint. In International Conference on Database Theory (pp. 453-470). Springer, Berlin, Heidelberg.

Myalapalli, V. K., & Dussa, K. (2015, December). Optimizing SQL queries in OLAP database systems. In 2015 International Conference on Information Processing (ICIP) (pp. 833-838). IEEE.

Yu, J. X., Yao, X., Choi, C. H., & Gou, G. (2003). Materialized view selection as constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *33*(4), 458-467.

Kalnis, P., & Papadias, D. (2003). Multi-query optimization for on-line analytical processing. *Information Systems*, *28*(5), 457-473.

Akinde, M. O., Böhlen, M. H., Johnson, T., Lakshmanan, L. V., & Srivastava, D. (2003). Efficient OLAP query processing in distributed data warehouses. *Information Systems*, *28*(1-2), 111-135.

Bornaz, L. (2010). Optimized data indexing algorithms for OLAP systems. *Database Systems Journal*, *1*(2), 17-26.

Momin, I., & Amin, I (2015). Data Warehouse Performance Optimization Implementing DHE Algorithm in Mortgage Backed Security using Mondrian.

JÜRGERNS, M., & Lenz, H. J. (2001). Tree based indexes versus bitmap indexes: A performance study. *International Journal of Cooperative Information Systems*, *10*(03), 355-376.

Roy, P., Seshadri, S., Sudarshan, S., & Bhobe, S. (2000, May). Efficient and extensible algorithms for multi query optimization. In ACM SIGMOD Record (Vol. 29, No. 2, pp. 249-260). ACM.

Chatziantoniou, D., Johnson, T., Akinde, M., & Kim, S. (2001, April). The MD-join: An operator for complex OLAP. In Proceedings 17th International Conference on Data Engineering (pp. 524-533). IEEE.

|  |  |
| --- | --- |
|  |  |

Now a days if you want to deal with a large amount of a data you have to have an efficient data warehouses. When we talk about data warehouses the soul purposes of such tools is to make quick decision on specific set of data, meaning executing in effective manner which generates quick results. Most of the data is used in the (OLAP) Online Analytical processing which supports the online decision support with minimal response time.