International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

Query Optimization Using Hierarchical Approach to Data Extraction, Multidimensional Modeling and Aggregation

Neha Sharma
Assistant Professor
Northern India Engineering College
Delhi, India

Abstract

Databases are generally accessed by asking a set of queries which return some information according to set of retrieval criteria's. In this paper, some such methods/ steps are introduced by applying which query performance can be optimized. The process starts with first analyzing the database schema and then applying mapping rules to map it to make multi dimensional model (star schema) and while making multi dimensional model, one need to keep track of representing it in hierarchical format . Now, in the star schema so formed apply aggregation. Aggregate the fact tables. This approach has been analyzed using the case study of bank management system.

Keywords-Data-warehouse,multidimensional modelling, aggregation,Unified modelling language.

1. INTRODUCTION

A database is a data repository which includes a collection of entity sets in the form of rows and columns, and each of these entity sets contains any number of entities of the same type. A relational database is a shared repository of data [1]. These databases are not able to fulfill out-of –the-ordinary tasks of data analysis and therefore, a new concept named Data Warehouse came into the picture and the technique through which the data is extracted out from data warehouse is called Data Mining. The companies use this process to collect useful information from the raw data. Data mining efficiency depends up querying also. Without queries, there is no significance of data warehouses. Queries help us to collect useful data from the bulk of data present in different data sources. The information gathered after querying helps in making analytical decisions for business making processes. Different strategical decisions are made upon this information.

Data-warehouse has a wider range and is fully operational with the most up-to-date technologies at the bottom of the decision making process. So, it's very important to collect accurate information using query and for this effectiveness in query results is required. This paper focuses on some such techniques.

Firstly, data need to be presented in hierarchical format. A hierarchy defines the navigating path for drilling up and drilling down. All attributes in a hierarchy belong to the same dimension. Large dimensions usually posses multiple hierarchies [2]. Now, from this hierarchical database design Multi-dimensional model, star schema is designed. Since, star schema is one of the multi-dimensional models, which is being designed query centric. Using UML, database schema is being designed and presented in hierarchical format to show data at the lowest level of granularity.UML is used to extend the data extraction process and show the various levels of granularity from where the source data is being accessed. UML has been as modeling language owing to its wide acceptance and the possibility of using and extending various complementary diagrams for modeling various system aspects. Then after, depending upon the queries, frequently asked queries and the requirement of company's fact table aggregates need to be made. This will help in reducing the query run time.

2. LITERATURE REVIEW

Sukheja et al focused on query optimization technique which generates sequences of SQL statements in order to produce the requested information [3]. The analysis for this paper is exposed that many sequences of queries generated by commercial tools are not very efficient. Semantic query optimizer architecture is suggested for these applications. The main component is a SQO optimizer that accepts previously generated sequences of queries and rewrites them according to a set of optimization strategies, before they are executed by the underlying database system. Given a database and a query on it, several execution plans exist that can be employed to answer the query. In principle, all the alternatives need to be considered so that the one with the best estimated performance is chosen. Queries formulated using SQL query language provides little predictive information useful for estimating query performance. Internal knowledge of the database structure, data distribution, and semantic query optimizing strategy are necessary to develop effective query execution plan. This is possible if and only if when the indexing, referential integrity and naming conventions are properly defined at the database designing level so that the semantic query optimizer uses indexing, referential integrity to rewrite a new query.

Chaudhuri et al describes the System-R optimization framework since this was a remarkably elegant approach that helped fuel much of the subsequent work in optimization [4]. It focuses on the search space that is considered by optimizers and also provides the forum for presentation of important algebraic transformations that are incorporated in the search space. It addresses the problem of cost estimation and takes up the topic of enumerating the search space. It presents some of the recent developments in query optimization. Optimization is much more than transformations and query equivalence. The infrastructure for optimization is significant. Designing effective and correct SQL transformations is hard, developing a robust cost metric is elusive, and building extensible enumeration architecture is a significant undertaking. Despite many years of work, significant open problems remain. However, an understanding of the existing engineering framework is necessary for making effective contribution to the area of query optimization.

Balke et al addresses the problem of multi-objective retrieval in database query processing. Multi-objective retrieval is especially useful for personalization problems,

where multiple user preferences have to be taken into account, and one has to compromise between certain desired characteristics of database objects to deliver high quality results [5]. However, up to now only for two extreme cases of such retrieval, namely top k retrieval and skyline queries, efficient algorithms have been investigated. Handling cases involving several distinct objectives, still needs to access and compare all database objects. It presented a novel multi-objective retrieval algorithm and proved that it always retrieves a correct result set and uses only an instance-optimal number of object accesses. Moreover, it contains the respective optimal algorithms for top k retrieval and skylining as special cases. It subsequently enhanced it by allowing for a successive output of result objects at the earliest possible time while the algorithm is still running. Finally they have addressed preliminary practical experiences with applications of our algorithm. The algorithm can be easily integrated into practical personalization frameworks or relational query processing. Concerning the manageability of query results, they have also shown that the cardinality of the multi-objective result set is bounded by the size of Pareto-optimal sets over the minimum of the number of score lists and objective function limiting down the set's cardinality in most practical cases. Implementing an advanced control flow then addressed how to save additional object accesses in the case of skewed data distribution by focusing on the most prominent objects at an early time.

Pahwa et al had described an object oriented approach to model the process of data extraction as part of extraction, transformation and loading process [6]. The hierarchies of each data element have been explicitly defined, thus highlighting the data granularity and hence simplifying the data extraction process. The object oriented features of generalization, aggregation, composition and association have been incorporated. These features help in identifying and establishing the relations between various data sources, thereby making the process of data extraction more reliable. Solving queries has been made easier because data sources at every level of granularity can be identified and targeted directly.

Pahwa et al introduces a framework which proposes design methodologies to map the relational database into a multidimensional model. The process starts with first cleaning the relational database and then categorizing the attributes of this cleansed relational database into metrics and dimensional attributes by applying the proposed set of mapping rules. The design of a data warehouse from relational databases is made possible in a user-friendly and semi-automated manner. The mapping of relational database to multidimensional model has been done by practically implementable mapping rules. These mapping rules are successful at conceptual as well as practical level mapping and are easy to understand. The approach followed is generic and simplified in nature.

All the above mentioned approaches do not consider proper rules for query optimization and which rule to be considered in which case. This concept presents an approach to make query processing more efficient and easy. It also represents data Warehouse in more understandable form to developer as well as to end users. This framework also goes on to suggest a set of rules for performing the above mentioned rules to database schema.

3. THE PROPOSED METHODOLOGY

In this technique, different methodologies are presented to make query processing more efficient. The approach proposes three steps, by following them query processing will become comparatively more effective and efficient as compared to the present scenario:

- Hierarchical Representation of data.
- Use star schema, instead of snowflake.
- Aggregate Fact Tables.

Hierarchical Representation of data

Granularity is the extent to which a system is broken down into small parts, either the system itself or its description or observation. It is the "extent to which a larger entity is subdivided [8]. For example, a yard broken into inches has finer granularity than a yard broken into feet."

Data Granularity models the level of detail present in the data sources. The data sources has been categorized to exhibit the fine grained data source which helps in efficient extraction of relevant and detailed data from different databases. By using object oriented concepts we have defined relationships between various data entities which help us to understand the interdependence between these data entities on every level of hierarchy.

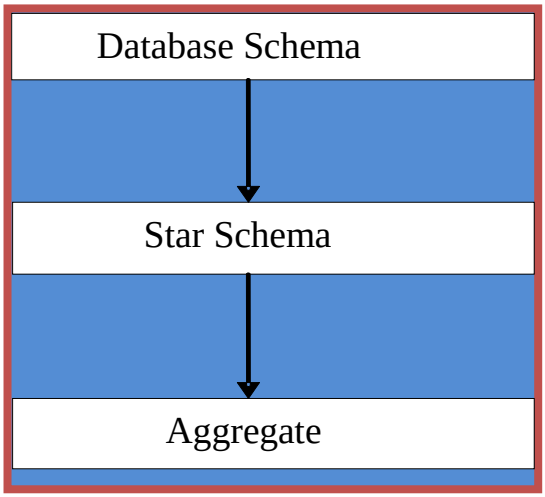

Figure 1: Framework

A hierarchy defines the navigating path for drilling up and drilling down. [9] All attributes in a hierarchy belong to the same dimension. Large dimensions usually posses multiple hierarchies. Rolling up and drilling down of data becomes more efficient by defining the data grain at each level of hierarchy. The fact table should to be at the lowest grain and there should to be multiple hierarchies present in the dimension tables.

So, hierarchical representation enhances the capabilities of query processing. With the help of this technique, number of joins required in query gets reduced, as a result query produce results faster and complexity also gets reduced.

A. Use star schema

Star Schema is a query centric structure; it means it is more suitable for query processing as compared to other multi–dimensional representations of the data warehouse [2]. In star schema, irrespective of the number of

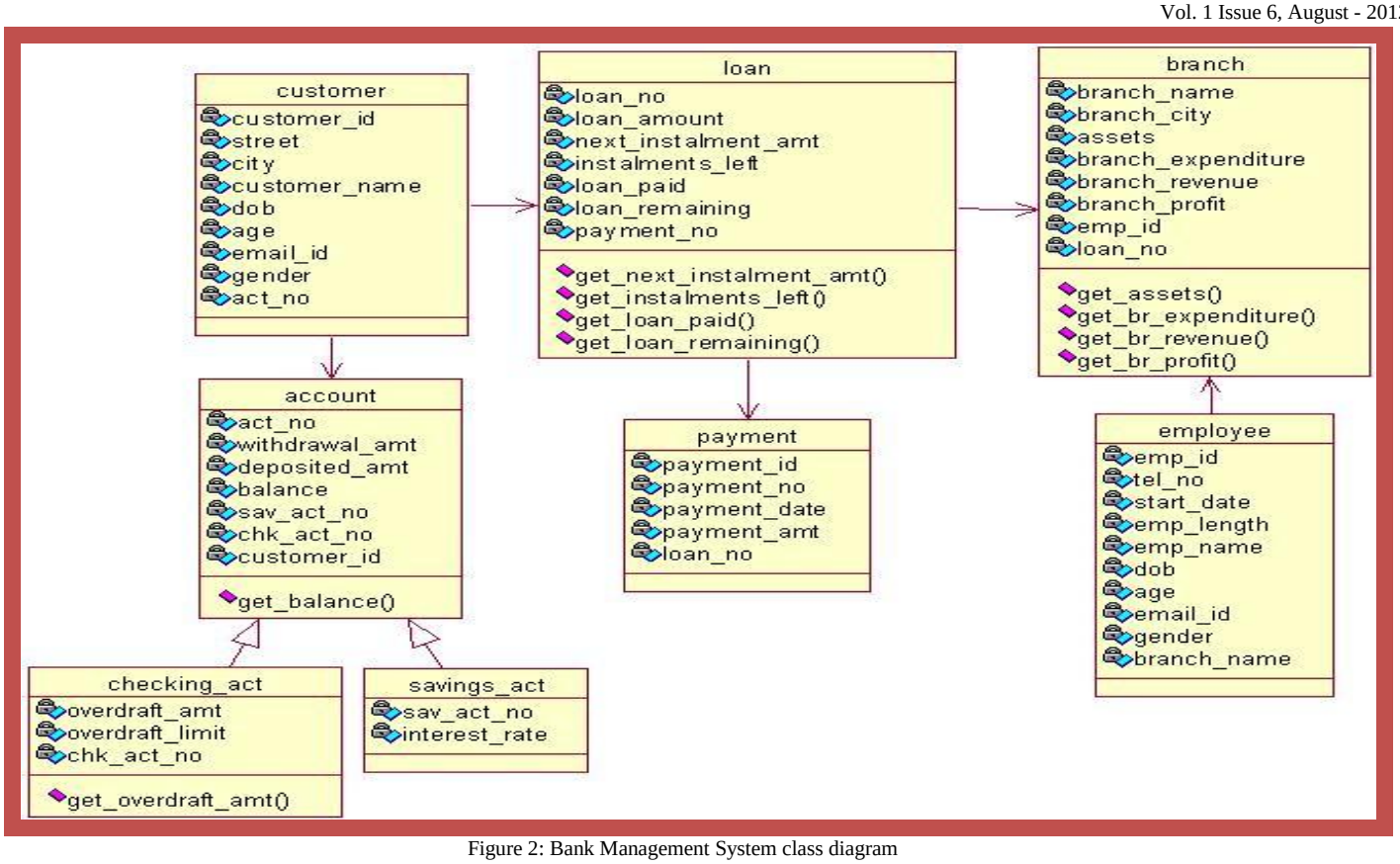

Figure 2: Bank Management System class diagram

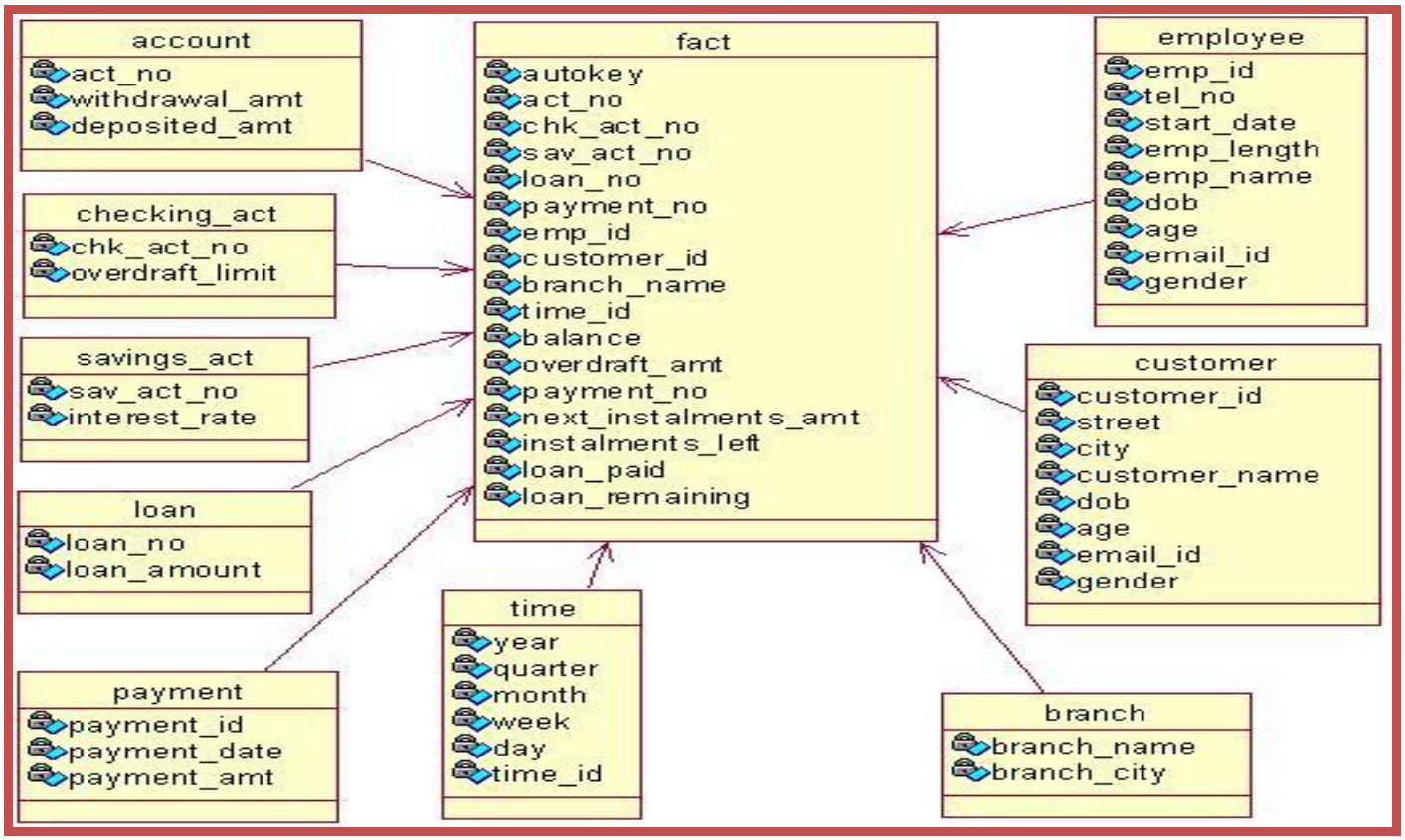

Figure 3: Star Schema

dimensions that participate in the query and irrespective of the complexity of the query, every query is simply executed first by selecting the rows from the dimension table using the filters based on the query parameters and then finding the corresponding fact table rows. Navigation is quiet simple and straightforward in star schema. So, Star Schema helps in query optimization by making visualization easier and decreasing query processing time.

B. Aggregate Fact Tables

Aggregates are the pre-calculated summaries derived from the most granular fact table. These summaries form a

set of separate aggregate fact tables [2]. Aggregate tables can be created across a number of dimensions and there can be any number of aggregates too, depending upon the requirement. When we run queries in data warehouse environment, it produces large result sets. These queries produces result sets after manipulating metrics of fact tables which are being got form thousands of table rows.
The handling of fact metrics and their calculation takes time and is very much complex. So, it will produce large set of data as a result and will take long time to process such queries.

Table 1: Dimension Table Attributes

TABLE_ NO.	TABLE_ NAME	ATTRIBUTES
1	account	act_no, withdrawal_amt, deposited_amt
2	checking_act	chk_act_no, overdraft_limit
3	savings_act	sav_act_no, interest_rate
4	loan	loan_no, loan_amt
5	payment	payment_id, payment_amt, payment_date
6	employee	emp_id, tel_no, start_date, emp_length,emp_name,dob,age,email,gender
7	customer	customer_id,street,city,dob,age,email,customer_name ,gender
8	branch	branch_name, branch_city
9	time	time_id, year, quarter, month, week, day

So, on the behalf of this information one can design number of aggregates, e.g. select all the customers, instalments_left from fact having loan_remaining >= 25,000 and branch_name = “delhi” and gender= “M”.
This data might be useful for bank and since there’s number of rows in fact table, so, always running query their will take long time (might be few minutes), whereas if will store aggregates initially, then query runtime will surely get reduced from minutes to few seconds.

Aggregate tables are used in the case when the data required to be analyzed requires number of dimensions and lots of rows of each dimension to be computed to calculate metrics of fact table.

So, formulation of aggregate fact tables is undoubtedly a very valuable method to improve query performance. Aggregate should to be hidden from the end users. Aggregates should to be chosen in the ways that do not increase overall storage space, i.e. aggregates with low sparsity.

1) Number of possible aggregates

There are thousands of possibilities for summarizing a dimension according to the level of summarization. The level of summarization is being decided upon the company’s requirement and often asked questions. The total number of aggregates possible can be easily determined by simply multiplying the number of levels in each dimension hierarchy.

2) How to store aggregates

There are mainly 2 possibilities to store aggregates:

- As new level field in already existing fact table.
- As new fact table

Out of the above two approaches, second approach (storing aggregates as new fact table) is a better method. Due to following reasons:

- If aggregates will be stored as a field in already existing fact table, then there are following problems with that:
  - There will be problem of double count.
  - Aggregates will be visible to the user.
- But if aggregates are stored as new fact table, then there are following benefits of this over previous method:
  - Double count problem conquered.
  - Aggregates will be invisible to the user.
  - Can be easily updated in future without any problem to tables.
  - Size of the field for aggregates does not affect the size of the field for the basic data.
  - Metadata will be simpler.

Aggregates are required when we have predictable queries with high load. In such a case, aggregates will help to get faster response, as we’ll be having results already stored in aggregates. Summary data must to be used only when critically needed.

3) When to Choose to Aggregate

There are two basic pieces of information which are required to select the appropriate aggregates:

- Usage and Analysis Pattern
  The most significant point is the expected usage patterns of the data. Depending upon this, one can easily choose best suited aggregate and this can be found after requirement phase.
- Base Table Row Reduction
  Secondly, one also needs to consider is number of rows in fact table and there distribution, wherever possible decrease the data load in fact table. One can check this, after loading the data in data warehouse and running some queries.

4) How much to Aggregate

There are 2 approaches to aggregation:

- No aggregation
  Sometimes, the size of data in the fact table is little that performance is adequate without aggregates. So, no need to create aggregates in such cases, it’ll increase the load. But in real time scenario, this is obviously not the case. It can be possible for new data warehouses or small scale companies.
- selective aggregation
  In this case, depending upon the requirement and the generally asked queries, required aggregates are created.

4. CONCLUSION

The proposed methodology for query optimization is a three step procedure and is a combination of various small techniques for query optimization. This technique is very unique in nature and will help in making query performance more effective and efficient. Query performance will become more effective in the way, that one need not to wait long for getting the results of the queries. This approach will help in decreasing the query run time and in the present scenario, time is very much important, even a microsecond matters during execution. The approach followed is generic and simplified in nature. This approach has been examined using the case study of bank management system. Further work going on in this field and will surly result in more efficiency and ease in query processing.

5. REFERENCES

[1] Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. Addison Weasely Pub Co. ISBN 0201542633 (2000).
[2] Paulraj Ponniah, Data Warehousing Fundamentals, Wiley India Pvt. Ltd., Reprint 2008.
[3] Sukheja D. & Singh K. U. , "Query Optimizer Model for Performance Enhancement of Data Mining Based Query" , International Journal of Computer Science & Communication (IJCSC) Vol. 1, No. 1, January-June 2010, pp. 235-237.
[4] Chaudhuri S. "An Overview of Query Optimization in Relational Systems", Microsoft Research, One Microsoft Way, Redmond, WA 98052.
[5] Balke W. , Güntzer U.," Multi-objective Query Processing for Database Systems" ,Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.
[6] Pahwa, P., Chaudhary, G., Jain, K., Sharma, N. and Gupta, R., 'Hierarchical Approach to Data Extraction using UML 2.0', Proc. of the International Conference on Advanced Computing and Communication Technologies (ACCT 2011), Copyright © 2011 RG Education Society, ISBN: 978-981-08-7932-7.
[7] Pahwa, P., Chaudhary, G., Jain, K., Sharma, N. and Gupta, R., "A User-Friendly Approach to Design a Data Warehouse from Operational Systems" International conference on Advanced Computing, Communication and Networks‟11.
[8] Wikipedia, http://en.wikipedia.org.
[9] Kimball, Ralph (2008). The Data Warehouse Lifecycle Toolkit, 2. edition. Wiley. ISBN 978-0-470-14977-5.