ISSN XXXX XXXX © 2019 IJESC

Research Article Volume 9 Issue No.1

Efficient Query Optimization in Data Warehouse using Indexing & Partitioning Techniques by Limit Clause

P.Arpitha[1], Dr.P.V.Kumar[2]
Research Scholar[1], Professor(Rtd)[2]
Department of Computer Science[1], Department of CSE[2]
RayalaSeema University, India[1], Osmania University, India[2]

Abstract:
Query optimization is of great importance for the performance of a relational database, especially for the execution of complex SQL statements. Query optimization is the bottleneck of database application performance especially those which store history i.e. data warehouse. Numerous researches has been introduced in the area of optimizing query performance, however a lot of research focused on online transaction processing (OLTP) database applications rather than data warehouse applications. SQL is used as query language because most data warehouses are based on relational or extended relational database system. As the information requests of the users are likely to be very complex. In order to reduce the complexity of the query generation process and in order to preserve portability to other database systems proposed semantic query optimization architecture is very useful. First, we offer some guiding principles for query optimization. A query optimizer determines the best strategy for performing each query. The query optimizer chooses, for example, whether or not to use indexes for a given query, and which join techniques to use when joining multiple tables. These decisions have a tremendous effect on SQL performance, and query optimization is a key technology for every application, from operational Systems to data warehouse and analytical systems to content-management systems. In present scenario data warehouses & mining turned out to be the common basis for the integration and analysis of data in modern enterprises. For retrieving faster execution and time performance we use efficient query optimization in data warehouse using different indexing and partitions techniques. In this thesis, we have also analyzed the factor that plays a vital role in the enhancement of query performance. The various factors that we have analyzed are optimizer, query equivalence rules, indexes, cost estimates and dependencies implied by SQL expression. The importance and their role in query optimization have also been discussed. The main aim of this thesis is to give guidance in constructing a query optimizer that is capable of optimizing large queries in a data warehousing setting and capable of producing evaluation plan with the shortest execution time or response time.

Keywords: Data Warehouse, OLAP, OLTP, Indexing, limit clause, query optimizer

I. INTRODUCTION

In today's era information and communication technology, every organization builds and maintains their vital data and information using powerful data base system that offers better performance and availability .apart from the typical transaction processing systems heavily for data warehousing and data mining to support different information systems of an organizations like decision support system ,executive support system, expert system and business intelligence systems that must run 24*7 around the globe.

II. DATA WAREHOUSE

Data warehousing is a collection of decision support technologies, aimed at enabling the knowledge worker (executive, manager, and analyst) to make better and faster decisions. A data warehouse is a "subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making.". Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports on-line analytical processing (OLAP), the functional and performance requirements of which are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported by the operational databases.

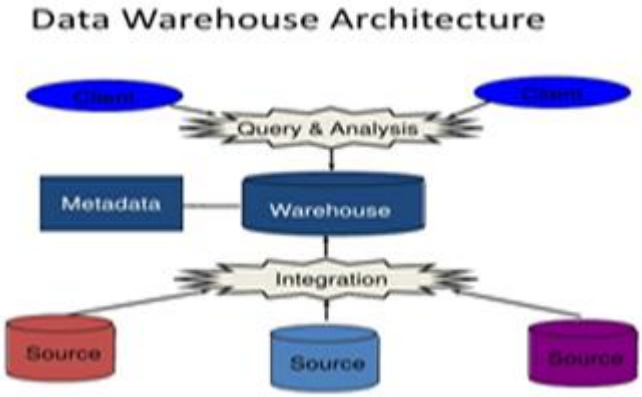


Figure.1. Data warehouse
Figure 1 shows a typical data warehousing architecture.

III. QUERY OPTIMIZATION:

Database performance depends on several factors at the database level, such as tables, queries, and configuration settings. This software constructs result in CPU and I/O operations at the hardware level, which you must minimize and make as efficient

as possible. As you work on database performance, you start by learning the high-level rules and guidelines for the software side, and measuring performance using wall-clock time. As you become an expert, you learn more about what happens internally, and start measuring things such as CPU cycles and I/O operations.

IV. INDEXING:

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table , the more this costs. If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially.

Most MySQL indexes (PRIMARY KEY, UNIQUE, INDEX, and FULLTEXT) are stored in B-trees.

V. PARTITIONING:

Partitioning (adatabase design technique) improves performance, manageability, simplifies maintenance and reduces the cost of storing large amounts of data. Partitioning can be achieved without splitting tables by physically putting tables on individual disk drives. Partitioning allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, therefore queries that access only a fraction of the data can run faster because there is fewer data to scan. There are two major forms of partitioning:

4)LIMIT Query Optimization

The LIMIT clause in a SELECT query sets a maximum number of rows for the result set. Pre-selecting the maximum size of the result set helps to optimize memory usage while processing a query. Originally, the value for the LIMIT clause had to be a numeric literal

This clause is useful in contexts such as:

- To return exactly N items from a top-N query, such as the 10 highest-rated items in a shopping category or the 50 hostnames that refer the most traffic to a web site.

- To demonstrate some sample values from a table or a particular query. (To display some arbitrary items, use a query with no ORDER BY clause. An ORDER BY clause causes additional memory and/or disk usage during the query.)

- To keep queries from returning huge result sets by accident if a table is larger than expected, or a WHERE clause matches more rows than expected.

MySQL sometimes optimizes a query that has a LIMIT ***row_count*** clause and no HAVING clause:

- If you select only a few rows with LIMIT, MySQL uses indexes in some cases when normally it would prefer to do a full table scan.

- If you combine LIMIT ***row_count*** with ORDER BY, MySQL stops sorting as soon as it has found the first ***row_count*** rows of the sorted result, rather than sorting the entire result. If ordering is done by using an index, this is very fast. If a filesort must be done, all rows that match the query without the LIMIT clause are selected, and most or all of them are sorted, before the first***row_count*** are found. After the initial rows have been found, MySQL does not sort any remainder of the result set.

- One manifestation of this behavior is that an ORDER BY query with and without LIMIT may return rows in different order, as described later in this section.

- If you combine LIMIT ***row_count*** with DISTINCT, MySQL stops as soon as it finds ***row_count*** unique rows.

- In some cases, a GROUP BY can be resolved by reading the index in order (or doing a sort on the index), then calculating summaries until the index value changes. In this case, LIMIT ***row_count*** does not calculate any unnecessary GROUP BY values.

- As soon as MySQL has sent the required number of rows to the client, it aborts the query unless you are usingSQL_CALC_FOUND_ROWS. In that case, the number of rows can be retrieved with SELECT FOUND_ROWS().

- LIMIT 0 quickly returns an empty set. This can be useful for checking the validity of a query. It can also be employed to obtain the types of the result columns within applications that use a MySQL API that makes result set metadata available. With the**mysql** client program, you can use the --column-type-info option to display result column types.

- If the server uses temporary tables to resolve a query, it uses the LIMIT ***row_count*** clause to calculate how much space is required.

- If an index is not used for ORDER BY but a LIMIT clause is also present, the optimizer may be able to avoid using a merge file and sort the rows in memory using an in-memory filesort operation.

If multiple rows have identical values in the ORDER BY columns, the server is free to return those rows in any order, and may do so differently depending on the overall execution plan. In other words, the sort order of those rows is nondeterministic with respect to the nonordered columns.

One factor that affects the execution plan is LIMIT, so an ORDER BY query with and without LIMIT may return rows in different orders.

5)Using Limit in a query:

The LIMIT clause limits the number of rows in the result set. We will code the LIMIT clause with a single numeric argument. If you code a single argument, it specifies the maximum row count, beginning with the first row. Here we executed the query WIth and without LIMIT clause .Fetch and Execution time is less when we use LIMIT clause.

Simple select query:

Query 1:

A) QUERY WITHOUT LIMIT CLAUSE

MySQL> SELECT * FROM ORDERS_FACT;

0.046 sec.

B) QUERY WITHLIMIT CLAUSE

MySQL> SELECT * FROM ORDERS_FACT LIMIT 10;

0.00 sec.

Query 2:

A) QUERY WITHOUT LIMIT CLAUSE

MYSQL>SELECT * FROM CUSTOMER_DIM;

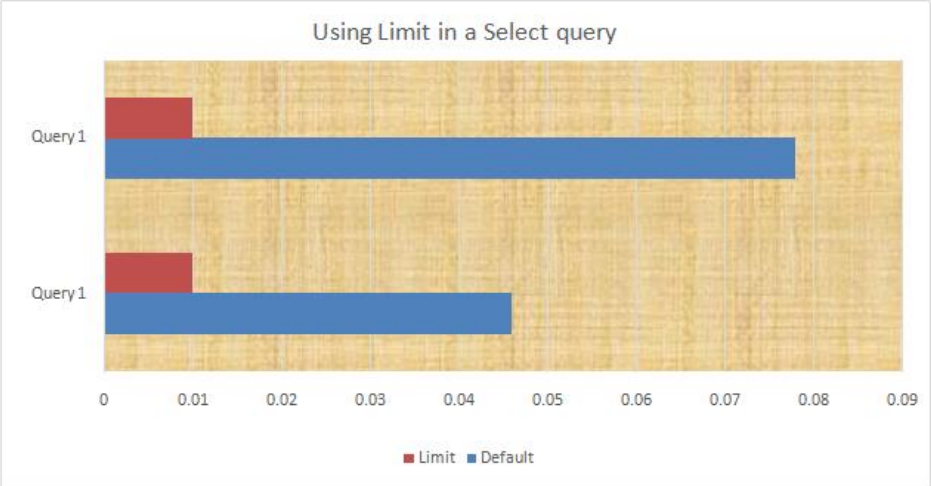
0.078 sec.

B) QUERY WITHLIMIT CLAUSE

MYSQL>SELECT * FROM CUSTOMER_DIM LIMIT 10;

0.00 sec

Result: we have fetched the records from the tables to know the result using LIMIT Clause. We applied above two conditions in a query as we found the following graph.



3)Inner Join:

An INNER JOIN Clause combine columns from two or more tables into a single result set. To join data from two tables, we code the name of the first table in the FROM clause and the name of the second table within the JOIN keyword. Here we executed the queries without LIMIT clause and with LIMIT clause.

A) INNER JOIN USING WITHOUT LIMIT CLAUSE

Mysql>select orders_fact.order_id

,orders_fact.customer_id,orders_fact.status,items_fact.order_id,items_fact.item_id,items_fact.product_id from

orders_fact inner join items_fact on orders_fact.order_id=items_fact.order_id;

Time 0.031sec

B) INNER JOIN USING WITH LIMIT CLAUSE

Mysql>select orders_fact.order_id

,orders_fact.customer_id,orders_fact.status,items_fact.order_id,items_fact.item_id,items_fact.product_id from

orders_fact inner join items_fact on orders_fact.order_id=items_fact.order_id **limit** 10;

Time 0.00sec.

Result: We applied the queries and observed that the time is optimized in query using limit clause.

4)Natural Join:

In MySQL, the NATURAL JOIN is such a join that performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.

A)NATURAL JOINS WITHOUT LIMIT CLAUSE

MYSQL> SELECT ORDERS_FACT.ORDER_ID ,ORDERS_FACT.CUSTOMER_ID,ORDERS_FACT.STATUS,ITEMS_FACT.ORDER_ID,ITEMS_FACT.ITEM_ID,ITEMS_FACT.PRODUCT_ID FROM ORDERS_FACT NATURAL JOIN ITEMS_FACT ;

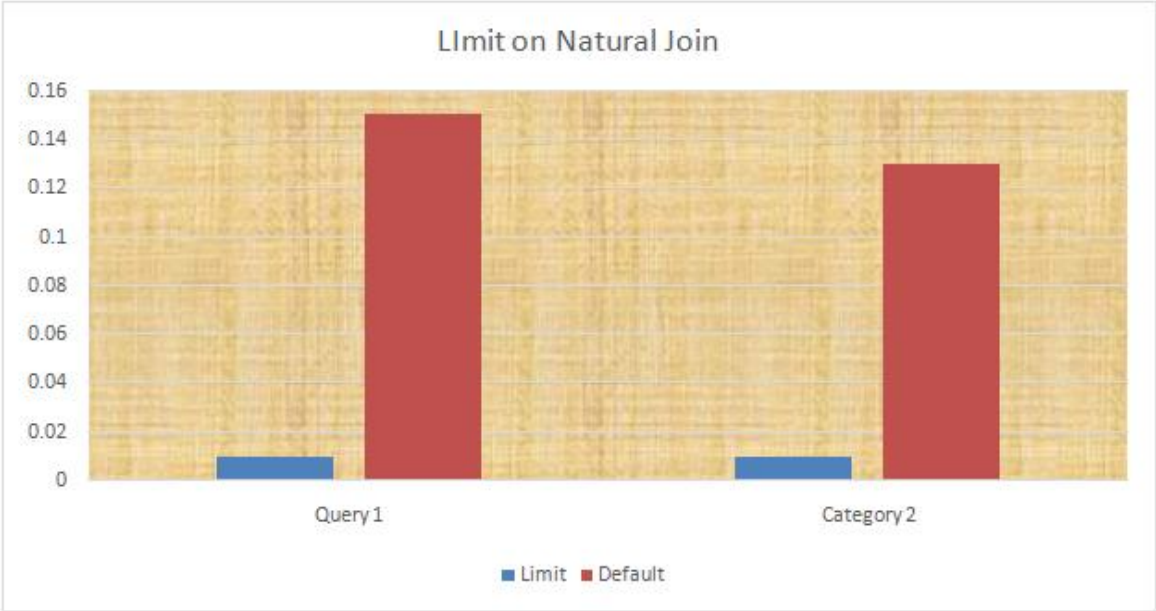
TIME 0.12 SEC.

B) NATURAL JOINS WITH LIMIT CLAUSE

MYSQL> SELECT ORDERS_FACT.ORDER_ID ,ORDERS_FACT.CUSTOMER_ID,ORDERS_FACT.STATUS,ITEMS_FACT.ORDER_ID,ITEMS_FACT.ITEM_ID,ITEMS_FACT.PRODUCT_ID FROM ORDERS_FACT NATURAL JOIN ITEMS_FACT LIMIT 10;

TIME 0.00 SEC.

Result: We applied the queries and we observed that the time is optimized in query using limit clause.



5)Count Function:

The COUNT(*) function returns the number of rows in a result set returned by a SELECT statement. The COUNT(*) function counts rows that contain no-NULL and NULL values.. This optimization is applied to MyISAM tables only because the number of rows of a MyISAM table is stored in the table_rows column in the tables table of the information_ schema database. therefore, MySQL can retrieve it very quickly.

A)COUNT FUNCTION WITHOUT LIMITCLAUSE

Mysql> select count (customer_id) from customer_dim;

Time 0.0.31 sec.

B)COUNT FUNCTION WITH LIMITCLAUSE

Mysql>select count (customer_id) from customer_dim limit 10;

Time 0.00 sec

Result: We applied this in the queries and observed that the time is optimized in query using limit clause.

6) Aggregate functions:

Aggregate Functions are all about performing calculations on multiple rows of a single column of a table and returning a single value. **Aggregate functions by default exclude nulls values before working on the data.**

a)Sum Clause:

MYSQL> SELECT SUM (CUSTOMER_ID) FROM CUSTOMER_DIM;

don't limit=0.000 sec

limit 100=0.00sec

limit 500=0.000sec

b)Min Clause:

MYSQL>SELECT MIN (CUSTOMER_ID) FROM CUSTOMER_DIM;

don't limit=0.000 sec

limit 100=0.00sec

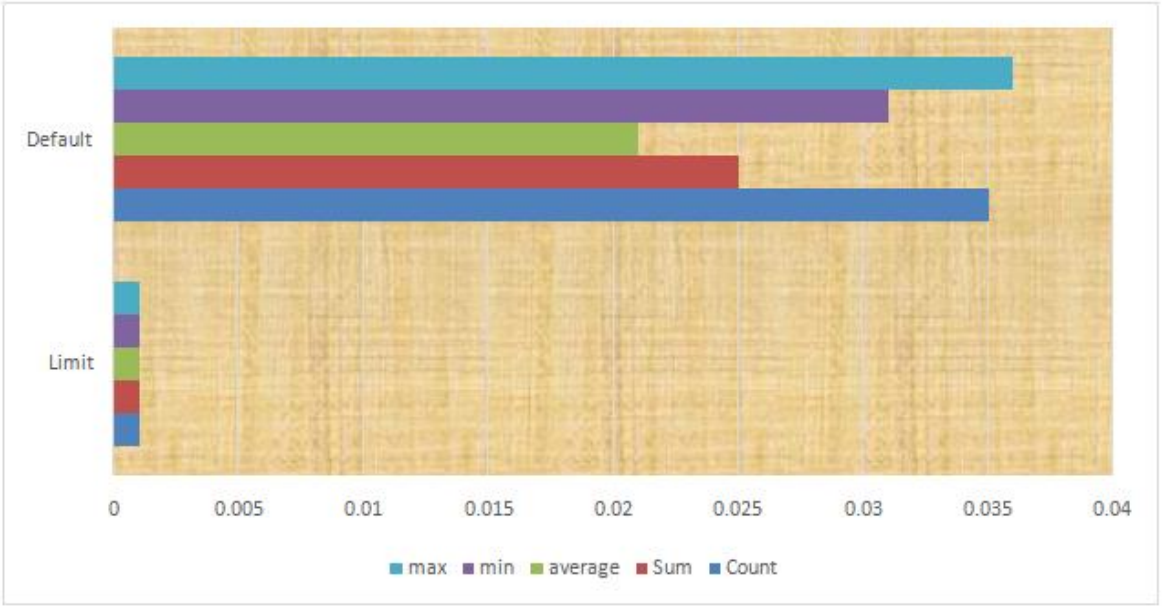
limit 500=0.000sec

c)Max Clause:

MYSQL>SELECT MAX (CUSTOMER_ID) FROM CUSTOMER_DIM;

Don't limit =0.036sec

Limit=0.00sec



VI. CONCLUSION:

The LIMIT clause in a SELECT query sets a maximum number of rows for the result set. Pre-selecting the maximum size of the result set helps to optimize memory usage while processing a query

VII. REFERENCES:

[1].S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD RECORD, 26(1):65-74, March 1997.

[2].P. O'Neil and D. Quass, "Improved Query Performance with Variant Indexes",SIGMOD,1997

[3].https://dev.mysql.com/doc/refman/8.0/en/limit-optimization.html

[4].Inmon, William H. Using the Data Warehouse. New York: Wiley, ©1994.

[5]. Barquin, RC. and H.A.Edelstein, Planning and Designing the Data Warehouse. Upper Saddle River, N.J.: Prentice Hall PTR, ©1997.

[6].Berson, A. and S.J. Smith, Data Warehousing, Data Mining, &OLAP. New York: McGraw-Hill, ©1997.