Query Minimization Methods

S SHANMUGA SRINIVAS, B VENKATRAMANA NAIK,JS ANANDA KUMAR

Abstract— SQL statements can be used to retrieve data from any database. If you've worked with databases for any amount of time retrieving information, it's practically given that you've run into slow running queries. Sometimes the reason for the slow response time is due to the load on the system, and other times it is because the query is not written to perform as efficiently as possible which is the much more common reason. For better performance we need to use best, faster and efficient queries. This paper covers how these SQL queries can be optimized for better performance. Query optimization subject is very wide but we will try to cover the most important points. In this paper I am not focusing on, in- depth analysis of database but simple query tuning tips & tricks which can be applied to gain immediate performance gain.

Index Terms— Introduction, Procedure, General tips for Query Optimization, Query Processing, Steps for Query Optimization, Conclusion.

I. INTRODUCTION

Query optimization is an important skill for SQL developers and database administrators (DBAs). In order to improve the performance of SQL queries, developers and DBAs need to understand the query optimizer and the techniques it uses to select an access path and prepare a query execution plan. Query tuning involves knowledge of techniques such as cost-based and heuristic-based optimizers, plus the tools an SQL platform provides for explaining a query execution plan. The best way to tune performance is to try to write your queries in a number of different ways and compare their reads and execution plans. In this paper I proposed various techniques that you can use to try to optimize your database queries.

- *S Shanmuga Srinivas is currently pursuing Master of Computer Applications in KMM Institute of PG studies in S.V University, Andhra pradesh, PH-9493999689. E-mail: shanmugas888@gmail.com*
- *B.Venkatramana Naik is currently pursuing Master of Computer Applications in KMM Institute of PG studies in S.V University, Andhra pradesh, PH-9618903651. E-mail: venkybukke651@gmail.com.*
- *JS Anand Kumar is currently working as Assistant Professor in KMM Institute of PG studies in S.V University, Andhra pradesh, PH-9441492491. E-mail:jsanandkumar @gmail.com*

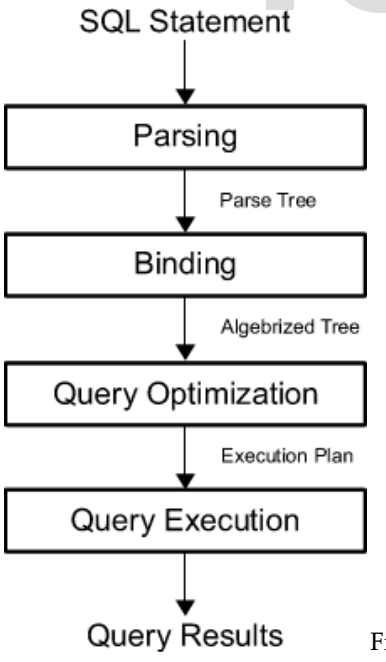
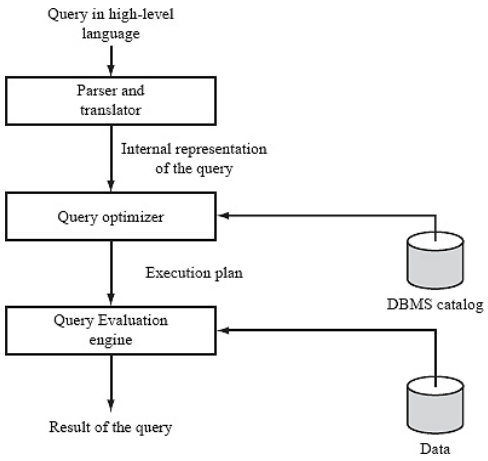


Fig 1.Query Optimizer

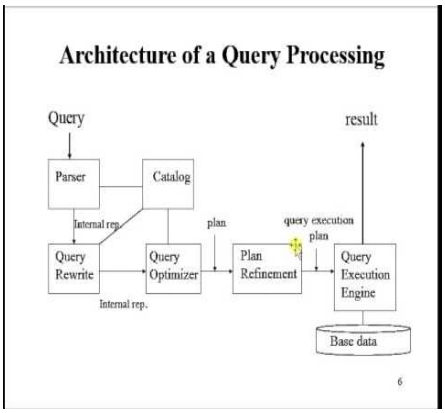
Query optimization is a function of many relational database management systems. The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans. Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints.



II.PROCEDURE:

III.GENERAL TIPS FOR QUERY OPTIMIZATION :

Each tip was tested by running both the original query and improved query while retrieving information from the Oracle 11g sample database especially on Sales schema. I recorded the average time of each query to show the speed increase of using the more efficient query.



IV.QUERY PROCESSING IS DONE WITH THE FOLLOWING AIM :

- Minimization of response time of query (time taken to produce the results to user's query).
- Maximize system throughput (the number of requests that are processed in a given amount of time).
- Reduce the amount of memory and storage required for processing.
- Increase parallelism.

Steps For Query Optimization:

Step 1 - Query Tree Generation

A Query tree is a tree data structure representing a relational algebra expression. The tables of the query are represented as leaf nodes. The relational algebra operations are represented as the internal nodes. The root represents the query as a whole.
During execution, an internal node is executed whenever its operand tables are available. The node is then replaced by the result table. This process continues for all internal nodes until the root nodes is executed and replaced by the result table.

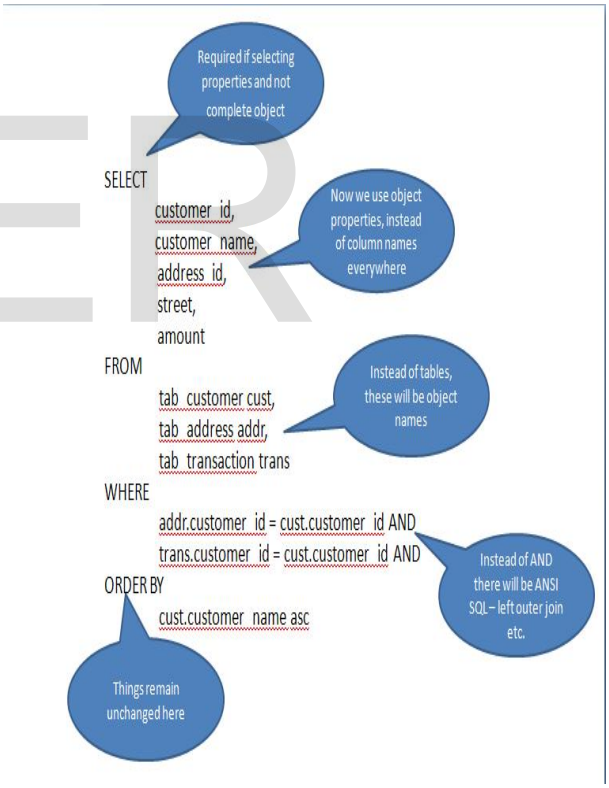
Step 2 – Query Plan Generation

After the query tree is generated, a query plan is made. A query plan is an extended query tree that includes access paths for all operations in the query tree. Access paths specify how the relational operations in the tree should be performed. For example, a selection operation can have an access path that gives details about the use of B+ tree index for selection.

Besides, a query plan also states how the intermediate tables should be passed from one operator to the next, how temporary tables should be used and how operations should be pipelined/combined.

Step 3– Code Generation

Code generation is the final step in query optimization. It is the executable form of the query, whose form depends upon the type of the underlying operating system. Once the query code is generated, the Execution Manager runs it and produce the results



SQL Tuning/SQL Optimization Techniques:

Sql Statements are used to retrieve data from the database. We can get same results by writing different sql queries. But use of the best query is important when performance is considered. So you need to sql query tuning based on the requirement.

V.SQL Tuning/SQL Optimization Techniques:

1) The sql query becomes faster if you use the actual columns names in SELECT statement instead of than '*'.
For Example: Write the query as

SELECT id, first_name, last_name, age, subject FROM student_details;
Instead of:

SELECT * FROM student_details;

2) HAVING clause is used to filter the rows after all the rows are selected. It is just like a filter. Do not use HAVING clause for any other purposes.
For Example: Write the query as

SELECT subject, count(subject) FROM student_details WHERE subject != 'Science' AND subject != 'Maths'
GROUP BY subject;
Instead of:

SELECT subject, count(subject) FROM student_details GROUP BY subject HAVING subject!= 'Vancouver' AND subject!= 'Toronto';

3) Sometimes you may have more than one subqueries in your main query. Try to minimize the number of subquery block in your query.
For Example: Write the query as

SELECT name FROM employee WHERE (salary, age) = (SELECT MAX (salary), MAX (age) FROM employee_details) AND dept = 'Electronics';
Instead of:

SELECT name FROM employeeWHERE salary = (SELECT MAX(salary) FROM employee_details) AND age = (SELECT MAX(age) FROM employee_details) AND emp_dept = 'Electronics';

4) Use operator EXISTS, IN and table joins appropriately in your query.
a) Usually IN has the slowest performance. b) IN is efficient when most of the filter criteria is in the sub-query.
c) EXISTS is efficient when most of the filter criteria is in the main query.

For Example: Write the query as

Select * from product p where EXISTS (select * from order_items o where o.product_id = p.product_id)
Instead of:

Select * from product p where product_id IN (select product_id from order_items);

5) Use EXISTS instead of DISTINCT when using joins which involves tables having one-to-many relationship.
For Example: Write the query as

SELECT d.dept_id, d.dept FROM dept d WHERE EXISTS (SELECT 'X' FROM employee e WHERE e.dept = d.dept);
Instead of:

SELECT DISTINCT d.dept_id, d.dept FROM dept d,employee e WHERE e.dept = e.dept;

6) Try to use UNION ALL in place of UNION.
For Example: Write the query as

SELECT id, first_name UNION ALL SELECT id, first_name FROM sports_team;
Instead of:

SELECT id, first_name, subject FROM student_details_class10 UNION SELECT id, first_name FROM sports_team;

7) Be careful while using conditions in WHERE clause.
For Example: Write the query as

SELECT id, first_name, age FROM student_details WHERE age > 10;
Instead of:

SELECT id, first_name, age FROM student_details WHERE age != 10;

Write the query as

SELECT id, first_name, age FROM student_details WHERE first_name LIKE 'Chan%';
Instead of:

SELECT id, first_name, age FROM student_details WHERE SUBSTR(first_name,1,3) = 'Cha';

Write the query as

SELECT id, first_name, age FROM student_details WHERE first_name LIKE NVL (:name, '%');
Instead of:

SELECT id, first_name, age FROM student_details WHERE first_name = NVL (:name, first_name);

Write the query as

SELECT product_id, product_name FROM product WHERE unit_price BETWEEN MAX(unit_price) and MIN(unit_price)
Instead of:

SELECT product_id, product_name FROM product WHERE unit_price >= MAX(unit_price) and unit_price <= MIN(unit_price)
Write the query as

SELECT id, name, salary FROM employee WHERE dept = 'Electronics' AND location = 'Bangalore';
Instead of:

SELECT id, name, salary FROM employee WHERE dept || location= 'ElectronicsBangalore';

Use non-column expression on one side of the query because it will be processed earlier.

Write the query as

SELECT id, name, salary FROM employee WHERE salary < 25000;
Instead of:

SELECT id, name, salary FROM employee WHERE salary + 10000 < 35000;

Write the query as
SELECT id, first_name, age FROM student_details WHERE age > 10;
Instead of:

SELECT id, first_name, age FROM student_details WHERE age NOT = 10;

8) Use DECODE to avoid the scanning of same rows or joining the same table repetitively. DECODE can also be made used in place of GROUP BY or ORDER BY clause.
For Example: Write the query as

SELECT id FROM employee WHERE name LIKE 'Ramesh%' and location = 'Bangalore';
Instead of:

SELECT DECODE(location,'Bangalore',id,NULL) id FROM employee WHERE name LIKE 'Ramesh%';

VI.CONCLUSION:

Query optimization is a common task performed by database administrators and application designers in order to tune the overall performance of the database system. The purpose of this paper is to provide SQL scenarios to serve as a quick and easy reference guide during the
development phase and maintenance of the database queries. Even if you have a powerful infrastructure, the performance can be significantly degraded by inefficient queries. Query optimization has a very big impact on the performance of a DBMS and it continuously evolves with new, more sophisticated optimization strategies. So, we should try to follow the general tips as mentioned above to get a better performance of queries. Optimization can be achieved with some efforts if we make it a general practice to follow the rules. The main focus was on query optimizations.

VII. REFERENCES

[1] 10 Ways to Improve SQL Query Performance http://www.developer.com/db/10-ways-to-improvesql-query-performance.html

[2] 15 Ways to Optimize Your SQL Queries http://hungred.com/useful information/waysoptimize-sql-queries/

[3] Optimize SQL Server queries with these advanced tuning techniques.
http://www.techrepublic.com/blog/theenterprisecloud/optimize-sql-server-queries-with-theseadvanced-tuning-techniques/.

[4] Making Queries Run Faster
https://sqlschool.modeanalytics.com/advanced/faster-queries.html.

[5] Query Optimization Techniques in Microsoft SQL Server-verhttp://www.dbjournal.ro/archive/16/16_4.pdf

[6] SQL Tuning or SQL Optimization. http://beginnersql-tutorial.com/sqlquery-tuning.htm

[7] SQL Server Optimization
Tips.http://santhoshgudise.weebly.com/uploads/8/5/4/7/8547208/sql_server_optimization_tips-1.doc

[8]Efficient SQL Statements
https://oraclebase.com/articles/misc/efficient-sql-statements

[9] Best Way to Write SQL Query.
http://www.ifadey.com/2010/11/best-way-to-writesql-query/

[10] SQL Tuning Guidelines for Oracle - Simple yet Effective!