IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

Optimizing Query Performance with the Help of Query Optimization Tool

Dr.A.N. Banubakode[1], Ajay Jaiswal[2], Swapnil Magar[3], Jason Vijoy[4], Harshal Kasle[5]

Department of Information Technology. JSPM's Rajarshi Shahu College Of Engineering,

Savitribai Phule Pune University, Pune[1,2,3,4,5]

Abstract: In this paper, we will enroll the procedure of query enhancement in view of Heuristic methodology. Information retrieving and storing are the basic tasks of the relational database. It is regularly found in the database business that a great deal of time is devoured in executing wasteful Queries. Query optimization is used for accessing database in an efficient manner, analyzing and choosing an optimized plan. The primary issue every Data System has is that despite the fact that the DBA realizing that the inquiries read are wasteful, the vast segments of individuals who fire these queries are not able to compose proficient inquiries. Subsequently, the execution of the whole framework corrupts on account of the exceptional fall in the framework throughput i.e. the quantity of exchanges performed per unit time is diminished. Query optimization primarily means selection, followed by sequencing in specific order, of the different clauses to formulate an efficient query from the multiple query plans by drawing a comparison of the query plans based on the cost of the resources involved and the response time. The objective of query optimization is to provide minimum response time and maximum throughput (i.e., the efficient use of resources). A Query Optimizer will acknowledge the inputted client query and naturally produce an identical yet very enhanced query. This will spare a considerable amount of time and effort. This thus enhances the framework throughput and its general execution.

Keywords: Query Optimization, Information Retrieval, Data Base Analyst(DBA), Heuristic Approach.

I. INTRODUCTION

Relational query languages provide a high-level declarative interface to access data stored in relational databases. Structured Query Language is a standard for storing and retrieving data from relational database. Query optimization is the way to reduce execution time of the declared query. Query optimizer is a main part in optimizing process which handles a large input space of complex query. The query optimization process itself is complex task. By reducing execution time end users get response in short time, but apart from getting response today's end users are very much interested in specific results based on their inputs. Ranking query or top-k query plays an essential role in retrieving specific information. Top-k queries intend to provide only the top-k results of a query, according to a user-specified ranking function. The growing significance of top-k queries has caught the attention of the researches. A top-k query only returns the top k results according to a user-specified preference, which generally consists of two components: a selection condition and a ranking function.

II. RELATED WORK

1) Query Optimizer plan Diagram: Production, Reduction and Application, Data Engineering (ICDE)

AUTHORS: Haritsa J.R.

The automated optimization of declarative SQL queries is a classical problem that has been diligently addressed by the database community over several decades. However, due to its inherent complexities and challenges, the topic has largely remained a "black art", and the quality of the query optimizer continues to be a key differentiator between competing database products, with large technical teams involved in their design and implementation. Over the past few years, a fresh perspective on the behavior of modern query optimizers has arisen through the introduction and development of the "plan diagram" concept. A plan diagram is a visual representation of the plan choices made by the optimizer over a space of input parameters, such as relational selectivities. In this tutorial, we provide a detailed walk-through of plan diagrams, their processing, and their applications. We begin by showcasing a variety of plan diagrams that provide intriguing insights into current query optimizer implementations. A suite of techniques for efficiently producing plan diagrams are then outlined. Subsequently, we present a suite of post-processing algorithms that take optimizer plan diagrams as input, and output new diagrams with demonstrably superior query processing characteristics, such as robustness to estimation errors. Following up, we explain how these offline characteristics can be internalized in the query optimizer, resulting in an intrinsically improved optimizer that directly produces high quality plan diagrams. Finally, we enumerate a variety of open technical problems, and promising future research directions. All the plan diagrams in the tutorial are sourced from popular industrial-strength query optimizers operating on benchmark decision-support environments, and will be graphically displayed on the Picasso visualization platform.

2. Testing SQL Server's Query Optimizer: Challenges, Techniques and Experiences

AUTHORS: Leo Giakoumakis, Cesar Galindo-Legaria

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

Query optimization is an inherently complex problem, and validating the correctness and effectiveness of a query optimizer can be a task of comparable complexity. The overall process of measuring query optimization quality becomes increasingly challenging as modern query optimizers provide more advanced optimization strategies and adaptive techniques. In this paper we present a practitioner's account of query optimization testing. We discuss some of the unique issues in testing a query optimizer, and we provide a high-level overview of the testing techniques used to validate the query optimizer of Microsoft's SQL Server.

Methodology

The Query Optimizer in this task is a Heuristic Optimizer. It essentially tries to minimize the quantity of gets to by diminishing the quantity of tuples and number of sections to be sought. Heuristic Optimization is less costly than that of expense based enhancement. It is based on some heuristic guidelines by which analyzer can choose advanced inquiry execution arrangement.

Javacc and Parser Generator:

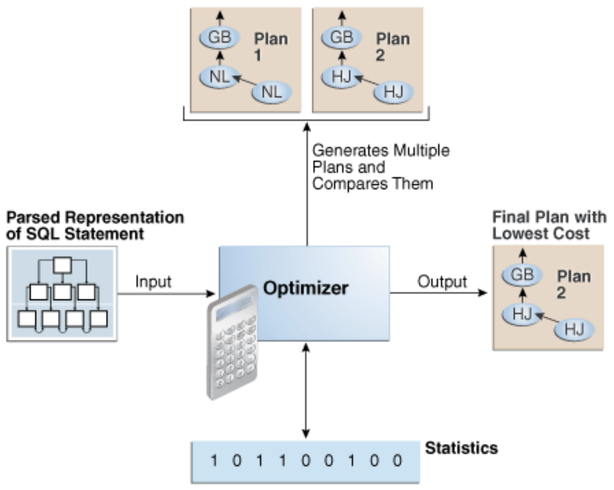
JavaCC is a parser generator and a lexical analyser generator. Parsers and lexical analysers are programming segments for managing information of character arrangements. Compilers and Interpreters consolidate lexical analysers and parsers to translate les containing projects, However lexical analysers and parsers can be utilized as a part of a wide assortment of different applications too. The parser and interpreter is the second module in the undertaking. It has been produced utilizing the parser generator JavaCC. JavaCC develops a recursive drop top down parser when given with important punctuation to SQL.

Optimization:

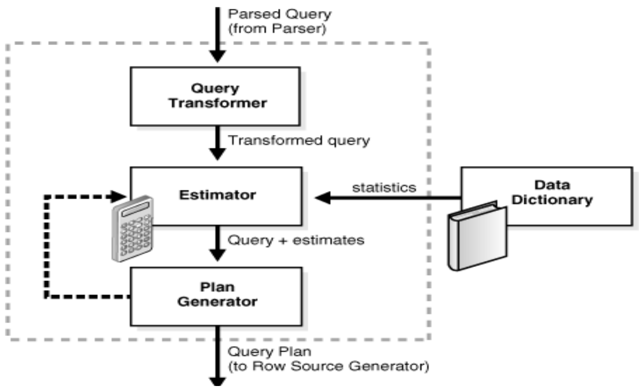
In this stage, the question processor applies tenets to the inside information structures of the inquiry to change these structures into identical, however more e customer representations. The standards can be based upon numerical models of the social variable based math. Expression and tree (heuristics), upon expense evaluations of diverse calculations connected to operations or upon the semantics inside of the question and the relations it includes. Selecting the best possible tenets to apply, when to apply them and how they are connected is the capacity of the question streamlining agent. This stage incorporates utilization of some heuristic guidelines, for example, performing determinations and projection operations as right on time as could reasonably be expected.

III. PROPOSED SYSTEM

The Query Optimizer in this task is a Heuristic Optimizer. It essentially tries to minimize the quantity of gets to by diminishing the quantity of tuples and number of sections to be sought. Heuristic Optimization is less costly than that of expense based enhancement. It is based on some heuristic guidelines by which analyzer can choose advanced inquiry execution arrangement.


GB
Plan 1
NL
NL
GB
Plan 2
HJ
HJ
Generates Multiple Plans and Compares Them
Parsed Representation of SQL Statement
Input
Optimizer
Output
Final Plan with Lowest Cost
GB
Plan 2
HJ
HJ
1 0 1 1 0 0 1 0 0
Statistics

IV. ARCHITECTURE



A set of query blocks represents a parsed query, which is the input to the optimizer. The optimizer performs the following operations:

1. **Query transformer**
   The optimizer determines whether it is helpful to change the form of the query so that the optimizer can generate a better execution plan.
2. **Estimator**
   The optimizer estimates the cost of each plan based on statistics in the data dictionary.
3. **Plan Generator**
   The optimizer compares the costs of plans and chooses the lowest-cost plan, known as the execution plan, to pass to the row source generator.

V. PROCESS FLOW

The main objective of query optimization is to choose effective execution plans for defined query. In this process an optimal execution plan is selected from many alternatives, depending on the optimization parameters, an optimized plan may be based on response time or amount of memory because time and space are the most important parameters in case of data retrieving and data storing. The other parameters of the query optimization process and their are as follows:

1. **Complex Query and Several Execution Plan:** For every entered query, the query optimizer thinks about a

large number of execution strategies, and for every plan there is need to analyze and check the validity. A complex query contains a lot of SQL clauses and filters due to these a large number of alternative execution plans are possible, after a particular limit it is not possible to analyze every possible execution plan. So, this is the main difficult task to select or optimal plan for execution.

2. **Optimization time: "**JOIN" is a keyword in SQL, and it plays an important role in making complex query and optimizing query. If optimizer considers all possible execution plan it may takes more time to find out the optimized plan than the time taking in retrieving data from the data base. The problem of finding the optimal join order in query optimization is NP-hard [1]. Thus, in many cases the query optimizer has to select a plan that is nearly optimized.

3. **Tuple assessment:** An optimal query execution plan is always depend on number of tuples used in query, it means Query optimizer primarily rely on statistical information to make tuple assessment, and query optimizer is always depend on the accurateness of the assessment of the tuples. Increase the qualities of the selection process of an optimal execution plan rely on additional CPU cost and increased memory consumption.

4. **Cost estimation:** Cost estimation models are mathematical algorithms or parametric equations used to estimate the costs of a query execution in terms of time or memory consumption.

VI. FUTURE SCOPE

Majority of research conducted in this area highlights the techniques and models used to enhance the query performance. I intend to propose a technique which will improve the performance issues pointed the in existing research regarding Query Optimizations. My focus will be to improve the query performance through optimization technique in distributed environment which have massive amount of data located at different locations.

VII. CONCLUSION

Query optimization has a exceptionally impact on the performance of a DBMS and it constantly grows with new optimization strategies. The main objective of query optimization is to choose effective execution plans from many alternatives. Query optimization process is a complex process. Top-k queries are leading in many applications such as web databases, multimedia databases, and data mining. Rank query gives ability to database system to efficiently retrieval of information based on user's ranked attribute. Top-k queries work in applications where users have relatively flexible preferences or specifications for certain attributes. The working of top-K query is simply an assignment of target values to the particular attributes of a relation.

REFERENCES

1. Leo Giakoumakis, and Cesar Galindo-Legaria, "*Testing SQL Servers Query Optimizer: Challenges, Techniques and Experiences*", IEEE, 2008.
2. M.A. Kashem, Abu Sayed Chowdhury, Rupam Deb, and Moslema Jahan , "*Query Optimization on Relational Database for Supporting Top-k Query Processing Techniques*", JCIT, 2010.
3. Neha Singh, P.K. Pandey and Anil Kumar Tiwari, "*A Study on Optimization of Top-k Queries in Relational Databases*", IJDE.
4. Dong Xin, Jiawei Han, Hong Cheng and Xiaolei Li, "Answering Top-K Queries with Multi-Dimensional Selections: The Ranking Cube Approach", ACM, 2006
5. Surajit Chaudhuri, "An Overview of Query Optimization in Relational Systems".
6. Alaa Aljanaby1, Emad Abuelrub1, and Mohammed Odeh2, "A Survey of Distributed Query Optimization" The International Arab Journal of Information Technology, Vol. 2, No. 1, January 2005
7. S.K. SINGH, "*DATABASE SYSTEM CONCEPTS, DESIGN AND APPLICATIONS*", PEARSON EDUCATION, FIRST IMPRESSION, 2006 ISBN 81-7758-567-3