SQL QUERY DATABASE MANAGER TOOL FOR DEVELOPER

Abhishiktha.K[1], Pradeep.RS[2], Krithika.S[3], Anandhamala.GS[4]
UG students, Department of Computer Science and Engineering, Easwari Engineering College, Chennai, India
Senior Professor, Department of Computer Science and Engineering, Easwari Engineering College, Chennai, India

Abstract
To design the database and tune SQL Queries. Tuning can be done by reducing the total CPU time and also reducing the I/O taken by the Query. This tool can be used by the developer to tune the queries and also to examine Nested and normalized queries, and information about database objects. The input to the optimizer is a parsed SQL query and statistics about the tables, indexes, and columns named in the query. The optimizer examines parsed and normalized queries, and information about database objects. The input to the optimizer is a parsed SQL query and statistics about the tables, indexes, and columns named in the query. The output from the optimizer is a query plan. The query plan is compiled code that contains the ordered steps to carry out the query, including the access methods (table scan or index scan, type of join to use, join order, and so on) to access each table. Using statistics on tables and indexes, the optimizer predicts the cost of using alternative access methods to resolve a particular query. It finds the best query plan – the plan that is least costly in terms of input.

Keywords: SQL Query, Database Manager tool, SQL tuner, SQL (Structured Query Language), SQL Optimization

1. Introduction:

This project is developed for designing and maintaining SQL database. And also tuning can be done by reducing the total CPU time and also reducing the I/O taken by the Query. This tool can be used by the developer to create database and tune the queries. There is no tool that helps the developer to create and maintain database without the help of DBA (Database Administrator) This SQL query manager tool makes the easiest way to design and maintain the database and it helps in tuning the SQL queries by Cost based method and Execution plan methodology. The optimizer examines parsed and normalized queries, and information about database objects. The input to the optimizer is a parsed SQL query and statistics about the tables, indexes, and columns named in the query. The input to the optimizer is a parsed SQL query and statistics about the tables, indexes, and columns named in the query. The output from the optimizer is a query plan. The query plan is compiled code that contains the ordered steps to carry out the query, including the access methods (table scan or index scan, type of join to use, join order, and so on) to access each table.

2.0 REVIEW OF LITERATURE

In paper [1], the authors M. Zagirnyak, P. Kostenko, paper deals with factors that influence the speed of data receiving in information systems. The method of automatic external SQL-query optimization is described. The method is based on the principle of building a local model of controlled process. It allows one to optimize the SQL-queries regardless of applied database management system and its settings. The structural and functional diagram of adaptive system of an external SQL-query optimization is presented. Possibility of using the proposed method under the conditions of databases structural uncertainty was tested. Electric network architecture.

We study the query optimization problem in declarative crowdsourcing systems. Declarative crowdsourcing is designed to hide the complexities and relieve the user the burden of dealing with the crowd. The user is only required to submit an SQL-like query and the system takes the responsibility of compiling the query,

generating the execution plan and evaluating in the crowdsourcing marketplace. A given query can have many alternative execution plans and the difference in crowdsourcing cost between the best and the worst plans may be several orders of magnitude. Therefore, as in relational database systems, query optimization is important to crowdsourcing systems that provide declarative query interfaces. In this paper, we propose CROWDOP, a cost-based query optimization approach for declarative crowdsourcing systems. CROWDOP considers both cost and latency in the query optimization objectives and generates query plans that provide a good balance between the cost and latency. We develop efficient algorithms in the CROWDOP for optimizing three types of queries: selection queries, join queries and complex selection-join queries. We validate our approach via extensive experiments by simulation as well as with the real crowd on Amazon Mechanical Turk.

In paper [2], the authors Vijayshankar Raman, David Simmen, Guy Lohman, Hamid Pirahesh, Miso Cilimdzic introduce Virtually every commercial query optimizer chooses the best plan for a query using a cost model that relies heavily on accurate cardinality estimation. Cardinality estimation errors can occur due to the use of inaccurate statistics, invalid assumptions about attribute independence, parameter markers, and so on. Cardinality estimation errors may cause the optimizer to choose a sub-optimal plan. We present an approach to query processing that is extremely robust because it is able to detect and recover from cardinality estimation errors. We call this approach "progressive query optimization" (POP). POP validates cardinality estimates against actual values as measured during query execution. If there is significant disagreement between estimated and actual values, execution might be stopped and re-optimization might occur. Oscillation between optimization and execution steps can occur any number of times. A re-optimization step can exploit both the actual cardinality and partial results, computed during a previous execution step. Checkpoint operators (CHECK) validate the optimizer's cardinality estimates against actual cardinalities. Each CHECK has a condition that indicates the cardinality bounds within which a plan is valid. We compute this validity range through a novel sensitivity analysis of query plan operators. If the CHECK condition is violated, CHECK triggers re-optimization. POP has been prototyped in a leading commercial DBMS. An experimental evaluation of POP using TPC-H queries illustrates the robustness POP adds to query processing, while incurring only negligible overhead. A case-study applying POP to a real-world database and workload shows the potential of POP, accelerating complex OLAP queries by almost two orders of magnitude.

In paper [3], the authors Shivnath Babu, Pedro Bizarro state A great deal of work on adaptive query processing has been done over the last few years: Adaptive query processing has been used to detect and correct optimizer errors due to incorrect statistics or simplified cost metrics; it has been applied to long-running continuous queries over data streams whose characteristics vary over time; and routing-based adaptive query processing does away with the optimizer altogether. Despite this large body of interrelated work, no unifying comparison of adaptive query processing techniques or systems has been attempted; we tackle this problem. We identify three families of systems (*plan-based*, *CQ-based*, and *routing based*), and compare them in detail with respect to the most important aspects of adaptive query processing: plan quality, statistics monitoring and re-optimization, plan migration, and scalability. We also suggest two new approaches to adaptive query processing that address some of the shortcomings revealed by our in-depth analysis: (1) *Proactive-optimization*, where the optimizer chooses query plan22s with the expectation of re-optimization; and (2) *Plan logging*, where optimizer decisions under different conditions are logged over time, enabling plan reuse as well as analysis of relevant statistics and benefits of adaptivity.

In paper [4] the authors M. Zagirnyak, P. Kostenko, paper deals with factors that influence the speed of data receiving in information systems. The method of automatic external SQL-query optimization is described. The method is based on the principle of building a local model of controlled process. It allows one to optimize the SQL-queries regardless of applied database management system and its settings. The structural and functional diagram of adaptive

system of an external SQL-query optimization is presented. Possibility of using the proposed method under the conditions of databases structural uncertainty was tested. Electric network architecture.

In paper [5] Hyunjung Park, Jennifer Widom, Deco is a comprehensive system for answering declarative queries posed over stored relational data together with data obtained on demand from the crowd. In this paper we describe Deco's cost based query optimizer, building on Deco's data model, query language, and query execution engine presented earlier. Deco's objective in query optimization is to find the best query plan to answer a query, in terms of estimated monetary cost. Deco's query semantics and plan execution strategies require several fundamental changes to traditional query optimization. Novel techniques incorporated into Deco's query optimizer include a cost model distinguishing between "free" existing data versus paid new data, a cardinality estimation algorithm coping with changes to the database state during query execution, and a plan enumeration algorithm maximizing reuse of common sub-plans in a setting that makes reuse challenging. We experimentally evaluate Deco's query optimizer, focusing on the accuracy of cost estimation and the efficiency of plan enumeration.

In paper [6] Optimization under uncertainty is a challenging topic of practical importance in the Process Systems Engineering. Since the solution of an optimization problem generally exhibits high sensitivity to the parameter variations, the deterministic model which neglects the parametric uncertainties is not suitable for practical applications. This paper provides an overview of the key contributions and recent advances in the field of process optimization under uncertainty over the past ten years and discusses their advantages and limitations thoroughly. The discussion is focused on three specific research areas, namely robust optimization, stochastic programming and chance constrained programming, based on which a systematic analysis of their applications, developments and future directions are presented. It shows that the more recent trend has been to integrate different optimization methods to leverage their respective superiority and compensate for their drawbacks. Moreover, data-driven optimization, which combines mathematical programming methods and machine learning algorithms, has become an emerging and competitive tool to handle optimization problems in the presence of uncertainty based on massive historical data.

In paper [7] Myungcheol Lee, Miyoung Lee, ChangSoo Kim proposed Methodological handling of queries is a crucial requirement in social networks connected to a graph No SQL database that incorporates massive amounts of data. The massive data need to be partitioned across numerous nodes so that the queries when executed can be retrieved from a parallel structure. A novel storage mechanism for effective query processing must to be established in graph databases for minimizing time overhead. This paper proposes a meta-heuristic algorithm for partitioning of graph database across nodes by placement of all related information on same or adjacent nodes. The graph database allocation problem is proved to be NP-Hard. A meta-heuristic algorithm comprising of Best Fit Decreasing with Ant Colony Optimization is proposed for data allocation in a distributed architecture of graph No SQL databases. Lucene index is applied on proposed allocation for faster query processing. The proposed algorithm with Lucene is evaluated based on simulation results obtained from different heuristics available in literature.

In paper [8] Myungcheol Lee, Miyoung Lee, ChangSoo Kim proposed In-memory databases are gaining attention as a solution to efficiently support SQL queries on large volume of data, as main memories are becoming cheaper and grow in size. However, their query performance is not well improved on modern hardware with faster CPUs, registers and caches due to the limitation of the classical iterator style query processing model. We propose a unified SQL query optimization system using JIT compilation of OLTP, OLAP, and Stored Procedure workloads for enhanced performance on modern hardware.

In paper [9] MapReduce is widely acknowledged by both industry and academia as an effective programming model for query processing on big data. It is crucial to design an optimizer which finds the most efficient way to execute an SQL

query using MapReduce. However, existing work in parallel query processing either falls short of optimizing an SQL query using MapReduce or the time complexity of the optimizer it uses is exponential. Also, industry solutions such as HIVE, and YSmart do not optimize the join sequence of an SQL query and cannot guarantee an optimal execution plan. In this paper, we propose a scalable optimizer for SQL queries using MapReduce, named SOSQL. Experiments performed on Google Cloud Platform confirmed the scalability and efficiency of SOSQL over existing work.

In paper [10] Jayant Rajurkar proposed

In today's complex world requires state-of-the-art data analysis over massive data sets. In data warehousing and OLAP applications, scalar-level predicates of set in SQL become highly inadequate which needs to support set-level comparison semantics, i.e., comparing a group of tuples with set of values. Complex queries composed by scalar-level operations are challenging for database engine to optimize, which results in costly evaluation. Bitmap indexing provides an important database capability to accelerate queries. Few database systems have implemented these indexes because of the difficulties of modifying fundamental assumptions in the low-level design of a database system. Bitmap index built one bitmap vector for each attribute value is gaining popularity in both column-oriented and row-oriented databases. It requires less space than the raw data provides opportunities for more efficient query processing. In this paper, we studied the property of bitmap index and developed a very effective bitmap pruning strategy for processing queries. Such index-pruning-based approach eliminates the need of scanning and processing the entire data set and thus speeds up the query processing significantly. Our approach is much more efficient than existing algorithms commonly used in row-oriented and column oriented databases.

Tabulation:

S.NO	Paper	Technique	Result	Issues
1	Crowd-Op: Query Optimization for Declarative Crowdsourcing Systems	Complex query optimization Real Crowd Evaluation	A cost-based query optimization that considers the cost-latency trade-off and supports multiple Crowdsourcing operators. The experiments on both simulated and real crowd. Demonstrate the effectiveness of query optimizer and validates the cost model and latency model.	Privacy issues in relation with crowdsourcing occurs. It has no access or relation with database.
2	Robust Query Processing through Progressive Optimization	Progressive Query Optimization	Progressive Optimization (POP) provides a flexible mechanism to build QEPs that are robust to optimizer misestimated, by inserting CHECK	Outdated statistics or invalid assumptions may cause significant estimation errors in the cardinality, and hence the cost of a plan, causing sub-optimal plans to be chosen. One proposed solution is to continually re-optimize by

			operators into a traditional QEP to test at execution time criteria under which the remainder of a QEP is still optimal.	using progressive query optimization.
3	Adaptive Query Processing in the Looking Glass	Proactive re-optimization Plan logging	Identification of three families of AQP systems (plan-based, CQ-based, and routing-based), and the detailed comparison of these system families with respect to the most important aspects of AQP: plan quality, statistics monitoring and re-optimization, plan migration, and scalability.	It does not provide any update or access to the database. Greedy optimization may not find best plan in routing based method.
4	External optimization of SQL-query under conditions of databases structural uncertainty	Implementation of the local model of controlled process (LMCP) concept	Advantages of application of the external SQL queries optimization based on construction of a local model of controlled process allow one to confirm the correctness and rationality of using this approach.	External optimization of data selection SQL-queries before their execution by DBMS. External optimizer is installed on the server side in client-server architecture, it monitors queries from client to server.
5	Query Optimization over Crowd-sourced Data	Distinguishing between existing vs. new data Estimating cardinality and database state simultaneously Exploiting limited sub-plan reuse opportunities.	The project includes Deco's query optimizer that finds the best plan to answer a query in terms of estimated monetary cost. Several novel techniques into the query optimizer to reflect Deco's query semantics and plan execution strategies are incorporated.	An improvement for a more efficient cost effective estimation is needed. It does not include in any maintenance or accessing of the database.

6	Process Optimization with Consideration of Uncertainties-An Overview	OPTIMIZATION uncertainties	Optimization with consideration of successful uncertainties	In case external optimizer , separate memory required
7	Data allocation optimization for query processing in graph databases using Lucene	Ant Colony Optimization Lucene index	The main goal of this work is to obtain optimal allocation of data considering replication and relation with less number of nodes such that the total allocation space for blocks of data does not exceed the node capacity. It will result in less overhead and efficient query retrieval process. Ant Colony Optimization integrated with BFD is used for selecting the nodes that satisfy this objective. Lucene index is applied to index the data residing in each node.	There is a processing delay present. It does not provide any authorization of the database nor any access to it.
8	A JIT Compilation-based Unified SQL Query Optimization System	JIT Compilation System	SQL queries into machine independent IR at the query plan level or operator level, and then applies various semantic/syntactic optimizations and macro/micro optimizations to the transformed IR. The optimized IR is then JIT compiled into best machine code considering runtime status of heterogeneous	The compile time is small and compilation are less explored for utilizing highly parallel accelerators.

			runtime environments.	
9	Scalable Query Optimization for Efficient Data Processing using MapReduce	Query Optimization MapReduce	Using MapReduce a scalable optimizer is being created for the SQL queries and on which when experiments are performed on Google Cloud Platform confirmed the scalability and efficiency of SOSQL over existing work.	Existing work in parallel query processing either falls short of optimizing an SQL query using MapReduce or the time complexity of the optimizer it uses is exponential. Also, industry solutions such as HIVE, and YSmart do not optimize the join sequence of an SQL query and cannot guarantee an optimal execution plan.
10	Efficient Query Processing and Optimization in SQL using Compressed Bitmap Indexing for Set Predicts	Bitmap Indexing	Scalar-level predicates of set in SQL become highly inadequate which needs to support set-level comparison semantics and Complex queries **composed** by scalar-level operations are challenging for database engine to optimize, which results in costly evaluation	By studying the Bitmap index a very effective bitmap pruning strategy for processing queries is being developed.

3. Conclusion and Future Work:

We could incorporate this Sql Query Database manager tool with other present query optimization techniques to make it a much more successful technology to use.

This project was developed to understand how the SQL Server Optimizer optimizes the queries and reduces the query's CPU time and Input\Output required.

So there are many things for future enhancements of this project. The future enhancements that are possible in the project are as follows:

- To optimize more complex queries i.e. queries which include Joins, Unions, Sub Queries etc.
- To study the database structure and provide the user with suggestions to improve the database structure for best performance.

To optimize the query which is been embedded in the Application, without the efforts of the user or programmer entering the query in the SQL Tuner.

References:

[1] De Capua C, Fulco G, R.Morello (2017), **"External optimization of SQL-query under conditions of databases structural uncertainty"**, *IEEE Journal*, Volume - 17, Issue -23, Pages: 7828 – 7837, 2017.

[2] A.R. Al-Ali, M. Alikarar, R. Gupta, M. Rashid, I. A. Zualkernan, **"Privacy-Preserving Query Processing by Multi-Party Computation**," *IEEE Transactions on Sql query*, Volume - 63, No - 4, pp. 426-434, November 2017.

[3] M. Albu, M. Sănduleac, C. Stănescu , **" Optimisation of Language-Integrated Queries by Query Unnesting,"** *IEEE Transactions* , Volume - 8, No - 1, pp. 485-492, January 2017.

[4] Michael Stonebraker, Sam Madden, Pradeep Dubey **The Optimization for Recurring Queries in Big Data Analysis System with MapReduce and execution plan,** SIGMOD Record 42(1) (2013) 44-49.

[5] S.L. Ho, S. Yang, **A fast robust optimization methodology based on polynomial** chaos and evolutionary algorithm for inverse problems, IEEE Trans. Magn. 48 (2012) 259–262.

[6] **Process Optimization with Consideration of Uncertainties-An Overview** *IEEE Journal*, Volume – 4, No - 5, pp. 1563-1570, Oct. 2017.

[7] Kaiping Xue, Senior Member, IEEE, Shaohua Li, Jianan Hong, **" Two-Cloud Secure Database for Numeric-Related SQL Range Queries with Privacy Preserving,"** *IEEE Sensors Journal*, Volume - 16, No - . 5, pp. 1361-1367, March1, 2016.

[8] S. Dong, S. Duan, G. Li , R. Tao Q. Yang, and J. Zhang, **" Quality-Based SQL: Specifying Information Quality in Relational Database Queries,"** *IEEE Internet of Things Journal*, Volume - 4, No - 5, pp. 1296-1303, Oct. 2017.

[9] L. Lampe, Y. Sun and V. W. S. Wong, **" A Query-Driven Approach to Entity Resolution,"** *IEEE Journal*, vol. 5, no. 1, pp. 69-78, Feb. 2018.

[10] Daniel R. Harris, Darren W. Henderson,**" Using Common Table Expressions to Build a Scalable Boolean Query Generator for Clinical Data Warehouses,"** *IEEE Transactions*, vol. 66, no. 8, pp. 2056-2064, Aug. 201