International Journal on Recent Researches in Science, Engineering & Technology (IJRRSET)
A Journal Established in early 2000 as National journal and upgraded to International journal in 2013 and is in existence for the last 10 years. It is run by Retired Professors from NIT, Trichy. Journal Indexed in JIR, DIIF and SJIF.
Research Paper
Available online at: www.jrrset.com
UGC Approved Journal No: 45483
Volume 8, Issue3 Pages 38-40

ISSN (Print) : 2347-6729
ISSN (Online) : 2348-3105
JIR IF : 2.54
SJIF IF : 4.334
Cosmos: 5.395

PROGRESSIVE PARAMETRIC QUERY OPTIMIZATION

[1]Dr.Muntha Raju,[2]Mr. Kambam Rajeev Reddy,[3]Mr. Mutyala Satish
[1]Professor,[2] Associate Professor, [3]Assist professor
[1,2,3]Dept of CSE ,Shadan College of Engg & Tech

ABSTRACT-In this project we approach a new technique called Progressive Parametric Query Optimization. In the Real world, commercial applications usually rely on precompiled parameterized procedures to interact with a database. Unfortunately, executing a procedure with a set of parameters different from those used at compilation time may be arbitrarily suboptimal. Parametric query optimization (PQO) attempts to solve this problem by exhaustively determining the optimal plans at each point of the parameter space at compile time. However, PQO is likely not cost-effective if the query is executed infrequently or if it is executed with values only within a subset of the parameter space. In this paper, we propose instead to progressively explore the parameter space and build a parametric plan during several executions of the same query. We introduce algorithms that, as parametric plans are populated, are able to frequently bypass the optimizer but still execute optimal or near-optimal plans.

1. INTRODUCTION

In this world of increasing globalization, Stupors moves forward to meet the challenges of the future through the development of R & D projects in various domains. R & D project sector attracts the most prominent thinkers and practitioners in a range of fields that impinge on development. Over the decade, Stupors, a Subsidiary of Spiro Technologies & consultant Pvt. Ltd provides a wide range of R & D project development training. Our uniqueness lies in the exclusive R & D project development. Accordingly, we created a setting that is enabling, dynamic and inspiring for the increase of solutions to global problems by R & D project development. Developing appropriate, responsible, innovative and practical solutions to students, by assisting in R & D project development. All our research is stranded in the need to provide an industry based training for students. Our team consists of more than 300 enthusiastic experts, drawn from a range of disciplines and experience, supported by infrastructure and facilities, which are world class and distinctively state-of-the-art. The strength of the organization lies in not only identifying and articulating intellectual challenges across a number of disciplines of knowledge but also in mounting research, training and demonstration projects leading to development of specific problem-based advanced technologies. The organization growth has been evolutionary, driven by a vision of the future and ingrained in challenges frightening today. The organization continues to grow in size, spread and intensity of work undertaken. Our experts are involved in a wide range of R & D project development training to student wishing to undertake professional development, or just wanting to learn about a new subject or area of study. In many applications, the values of runtime parameters of the system, data, or queries themselves are unknown when queries are originally optimized. In these scenarios, there are typically two trivial alternatives to deal with the optimization and execution of such parameterized queries. One approach, termed here as Optimize-Always, is to call the optimizer and generate a new execution plan every time a new instance of the query is invoked. Another trivial approach, termed Optimize-Once, is to optimize the query just once, with some set of parameter values, and reuse the resulting physical plan for any subsequent set of parameters. Both approaches have clear disadvantages. Optimize- Always requires an optimization call for each execution of a query instance. These optimization calls may be a significant part of the total query execution time, especially for simple queries. In addition, Optimize-Always may limit the number of concurrent queries in the system, as the optimization process itself may consume too much memory. On the other hand, Optimize-Once returns a single plan that is used for all points in] the parameter space. The chosen plan may be arbitrarily suboptimal for parameter values different from those for which the query was originally optimized.

2. LITERATURE SURVEY

Query optimization is a function of many relational database management a good systems in which multiple query plans for satisfying a query are examined and query plan is identified. This may or not be the absolute best strategy because there are many ways of doing plans. There is a tradeoff between the amount of time spent figuring out the best plan and the amount running the plan. Different qualities of database management systems have different ways of balancing these two. Cost based query optimizers evaluate the resource footprint of various query plans and use this as the basis for plan selection.

ISSN (Print) : 2347-6729
ISSN (Online) : 2348-3105
JIR IF : 2.54
SJIF IF : 4.334
Cosmos: 5.395

Typically the resources which are coasted are CPU path length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism. The set of query plans examined is formed by examining possible access paths (e.g., primary index access, secondary index access, full file scan) and various relational table join techniques (e.g., merge join, hash join, product join). The search space can become quite large depending on the complexity of the SQL query. There are two types of optimization. These consist of logical optimization which generates a sequence of relational algebra to solve the query. In addition there is physical optimization which is used to determine the means of carrying out each operation.

The goal is to eliminate as many unneeded tuples, or rows as possible. The following is a look at relational algebra as it eliminates unneeded tuples.

The project operator is straightforward to implement if <attribute list> contains a key to relation R. If it does not include a key of R, it must be eliminated. This must be done by sorting (see sort methods below) and eliminating duplicates. This method can also use hashing to eliminate duplicates Hash table.

SQL command distinct: this does not change the actual data. This just eliminates the duplicates from the results.

Set operations: Database management heavily relies on the mathematical principles of set theory which is key in comprehending these operations.

Union: This displays all that appear in both sets, each listed once. These must be union compatible. This means all sequences of selected columns must designate the same number of columns. The data types of the corresponding columns must thereby comply with the conditions valid for comparability. Each data type can be compared to itself. Columns of data type CHAR with the different ASCII and EBCDIC code attributes can be compared to each other, whereby they are implicitly adapted. Columns with the ASCII code attribute can be compared to date, time, and timestamp specifications. All numbers can be compared to each other. In ANSI SQL mode, it is not enough for the data types and lengths of the specified columns to be compatible: they must match. Moreover, only column specifications or * may be specified in the selected columns of the QUERY specifications linked with UNION. Literals may not be specified (wisc). Intersection: This only lists items whose keys appear in both lists. This too must be union compatible. See above for definition.

Set difference: This lists all items whose keys appear in the first list but not the second. This too must be union compatible.

Cartesian Product: Cartesian products (R * S) takes a lot of memory because its result contains a record for each combination of records from R and S.

3. EXISTING SYSTEM

We propose Progressive Parametric Query Optimization (PPQO), a novel framework to improve the performance of processing parameterized queries.

We also propose the Parametric Plan (PP) interface as a way to incorporate PPQO in DBMS.

The optimization process itself may consume too much memory

4. PROPOSED SYSTEM

We propose instead to progressively explore the parameter space and build a parametric plan during several executions of the same query. We introduce algorithms that, as parametric plans are populated, are able to frequently bypass the optimizer but still execute optimal or near-optimal plans We propose two implementations of PPQO with different goals. On one hand, Bounded has proven optimality guarantees. On the other hand, Ellipse results in higher hit rates and better scalability.

5. IMPLEMENTATION

In addition to SQL commands, the oracle server has a procedural language called PL/SQL. PL/SQL enables the programmer to program SQL statement. It allows you to control the flow of a SQL program, to use variables, and to write error-handling procedures

The Bloom filter, conceived by Burton H. Bloom in 1970, is a space-efficient data structure that is used to test whether an element is a member of a set. False positives are possible, but false negatives are not. Elements can be added to the set, but not removed (though this can be addressed with a counting filter). The more elements that are added to the set, the larger the probability of false positives.

An empty Bloom filter is a bit array of m bits, all set to 0. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions with a uniform random distribution.

To add an element, feed it to each of the k hash functions to get k array positions. Set the bits at all these positions to 1.

To query for an element (test whether it is in the set), feed it to each of the k hash functions to get k array positions.
Adaptive optimization

Adaptive optimization is a technique in computer science that performs dynamic recompilation of portions of a program based on the current execution profile. With a simple implementation, an adaptive optimizer may simply make a trade-off between Just-in-time compilation and interpreting instructions. At another level, adaptive optimization may take advantage of local data conditions to optimize away branches and to use inline expansion to decrease convert char switching.

Consider a hypothetical banking application that handles transactions one after another. These transactions may be checks, deposits, and a large number of more obscure transactions. When the program executes, the actual data may consist of clearing tens of thousands of checks without processing a single deposit and without processing a single check with a fraudulent account number. An adaptive optimizer would compile assembly code to optimize for this common case. If the system then started processing tens of thousands of deposits instead, the adaptive optimizer would recompile the assembly code to optimize the new common case. This optimization may include in lining code or moving error processing code to secondary cache.

6. PROBLEM DEFINATION

Query Optimizer

The query optimizer is the component of a database management system that attempts to determine the most efficient way to execute a query. The optimizer considers the possible query plans for a given input query, and attempts to determine which of those plans will be the most efficient. Cost-based query optimizers assign an estimated "cost" to each possible query plan, and choose the plan with the smallest cost. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, the CPU requirements, and other factors determined from the data dictionary. The set of query plans examined is formed by examining the possible access paths (e.g. index scan, sequential scan) and join algorithms (e.g. sort-merge join, hash join, nested loops). The search space can become quite large depending on the complexity of the SQL query.

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.

Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query. The nodes are arranged as a tree, in which intermediate results flow from the bottom of the tree to the top. Each node has zero or more child nodes -- those are nodes whose output is fed as input to the parent node. For example, a join node will have two child nodes, which represent the two join operands, whereas a sort node would have a single child node (the input to be sorted). The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan.

7. CONCLUSION

The deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner . It is more advantages for
individual users to store their data redundantly across multiple physical servers so as to reduce the data integrity and availability threats. Thus, distributed protocols
for storage correctness assurance will be of most importance in achieving robust and secure cloud storage systems.

BIBLIOGRAPHY

1. S. Babu and P. Bizarro, “Adaptive Query Processing in the Looking Glass,” Proc. Second Biennial Conf. Innovative Data Systems Research (CIDR), 2005.
2. R.L. Cole and G. Graefe, “Optimization of Dynamic Query Evaluation Plans,” Proc. ACM SIGMOD, 1994.
3. D. Harish, P. Darera, and J. Haritsa, “On the Production of Anorexic Plan Diagrams,” Proc. 33rd Int’l Conf. Very Large Data Bases (VLDB), 2007.
4. A. Deshpande, Z. Ives, and V. Raman, “Adaptive Query Processing,” Foundations and Trends in Databases, vol. 1, no. 1, pp. 1-140, 2007.
5. S. Ganguly, “Design and Analysis of Parametric Query Optimization Algorithms,” Proc. 24th Int’l Conf. Very Large Data Bases (VLDB), 1998.
6. A. Ghosh, J. Parikh, V.S. Sengar, and J.R. Haritsa, “Plan Selection Based on Query Clustering,” Proc. 28th Int’l Conf. Very Large Data Bases (VLDB), 2002.
7. G. Graefe and K. Ward, “Dynamic Query Evaluation Plans,” Proc. ACM SIGMOD, 1989.
8. A. Hulgeri and S. Sudarshan, “Parametric Query Optimization for Linear and Piecewise Linear Cost Functions,” Proc. 28th Int’l Conf. Very Large Data Bases (VLDB), 2002