A Model for Building Dynamic Indexes & Storage and Re-use of Optimal Query Plans Generated thru Progressive Optimization (POP)

Sreekumar Vobugari, D. V. L. N. Somayajulu, and B. M. Subraya

***Abstract*—Query Optimization is at the core for contribution towards performance improvements in application systems. A lot of ideas have been proposed towards Query Optimization and there is lot of On-going research happening in this area. Virtually every commercial query optimizer chooses the best plan for a query using a cost model which is based on cardinality estimation. If cardinality estimation is inaccurate, then this may result in optimizer to choose a sub-optimal plan. But once the optimizer chooses an optimal plan for execution based on the approach of POP, the need for generating an optimal plan for subsequent execution of the same query at a later point in time can be minimized/reduced/exempted by storing the execution plan. This paper proposes a Model for building Dynamic Indexes & Storage and Re-Use of Optimal Query plans generated thru Progressive Optimization (POP) for performance gains. This approach is an extension to the work implemented in "Robust Query Processing through Progressive Optimization". This paper proposes a model to build Learning system within the database to analyze the stream of incoming queries and project viable indexes as against the initial indexes created by the Administrator and also store and re-use of Optimal Query Plans generated thru Progressive Query Optimization (POP).**

***Index Terms*—QoS, PoP.**

I. Introduction

Every enterprise relies on good decision-making for its sustainable and predictable growth in the market. Good business decisions are based on the analysis of huge amount of data. Therefore every enterprise (big or small) strives to ensure data integrity, availability of data and also has the need to cater to high degree of concurrent access to data. As organizations continue to evolve by expanding their operations by way of venturing into new business initiatives, small company acquisition that complement to their business capabilities, these result in increase of data volumes, leading for the need of high performance and scalable systems.

A. Motivations

In any automated business process, Performance of software systems is at the core of Customer satisfaction. It is quite obvious that any application system that meets all business requirements but fails to meet required Nonfunctional aspects (QoS parameters) will indeed lead to greater customer dissatisfaction. For non-real time application systems, though the business users expect for the best performance (one of the Non Functional Requirement) which is a nice to have feature, for real time machine critical application systems, Performance of the software system in terms of response time or through put becomes the qualifying criteria for acceptance of the application system by business users for Production roll-out. Adding to these challenges is the lack of standardized approaches for capturing and measurement of Performance requirements in application systems. Hence, given the criticality and limitations, we believe that looking for innovative approaches towards performance gains in the form of totally new ideas or improvements proposed to the existing ideas[1,2,3,4] that can bring in substantial improvements will surely be a differentiating factor to mitigate the "Performance" issue in software systems addressing the needs of larger audiences of IT industry.

B. Performance Engineering Fundamentals

For any mid-sized or larger applications, the Performance of the application systems is dependent on the quality of Architecture definition & implementation aspects. The objective of having more emphasis on Performance aspects is to avoid cost overruns by taking proactive measures than reactive approach. From an Application system perspective, Fig. 1 depicts a scenario of a reactive approach.

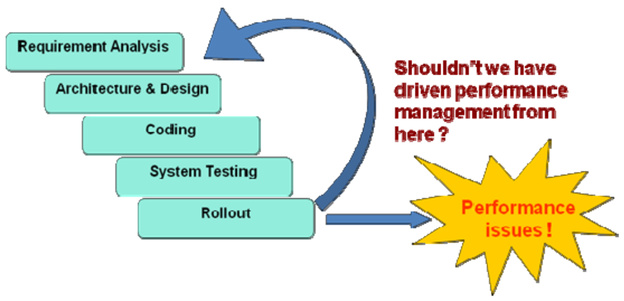

Fig. 1. Performance influence in software life cycle stages

From the above figure, two points are very clear and they are:

- ✓ Absence of proactive approach adds to Costs.
- ✓ Later the reaction, higher the costs.

Clearly, we believe that there is a need for a Performance Management Process to overcome the crisis originating out of performance issues as depicted above. Based on our

Manuscript received May 17, 2012; revised May 30, 2012.

Sreekumar Vobugari and D. V. L. N. Somayajulu are with Department of Computer Science and Engineering at National Institute of Technology, Warangal, India (email: Sreekumar_vobugari@nitw.ac.in; somadvlns@gmail.com)

B. M. Subraya is with Education & Research Unit of Infosys Limited, Mysore, India (email:Subraya@gmail.com)

experiences acquired thru Project executions and Consulting assignments to various customers, following the Performance Management Process at each SDLC phase can surely mitigate the risks originating out of Performance Issues. A Comprehensive view of Performance Management Process is shown in Fig. 2 below.

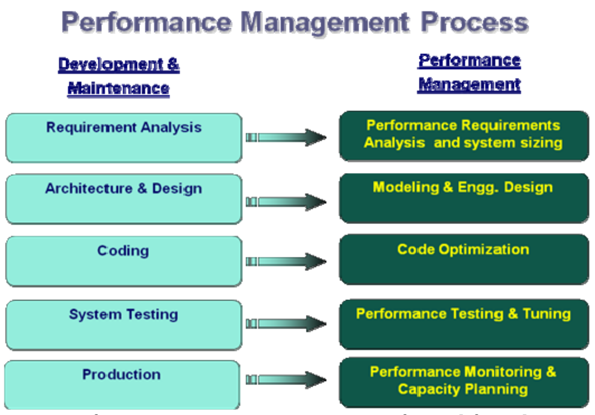
Performance Management Process
Development & Maintenance
Performance Management
Requirement Analysis
Performance Requirements Analysis and system sizing
Architecture & Design
Modeling & Engg. Design
Coding
Code Optimization
System Testing
Performance Testing & Tuning
Production
Performance Monitoring & Capacity Planning

Fig. 2. Performance management process in software life cycle stages

While this is a fact, quiet notably, it is observed that the Performance tuning at the Database level is also an important criterion for success of any application. Since, our emphasis in this paper is dealing with performance aspects at Database side, we tend to limit our discussion to Databases without getting into the Architecture & design of the application systems.

From database perspective, performance can be viewed from multiple perspectives. Two important perspectives of Performance from the end user's need are 'Response Time' & 'Throughput'.

Response Time is the elapsed time to provide the result to the end user since the time a SQL request is triggered. Throughput is the volume of processing done in a stipulated amount of time. Usually, throughput is measured in terms of number of records processes per minute/hour.

C. Database Performance Requirement Analysis Overview

In this section, we attempt to address on three important aspects which should be considered for Database Performance Requirement analysis. They are:

- Types of Applications
- Geographical Dispersion
- Gathering & Analyzing Workload

Types of Applications: One of the important aspects to be dealt with during Performance Requirement Analysis phase is to classify the Application in to one of the following categories:

OLTP	–	Demand for high throughput and Insert/Update intensive.
DSS/OLAP	–	Converts large amount of information into user-defined reports.
Hybrid	–	Demand for high throughput and Insert/Update intensive.

Geographical Dispersion: This is another important aspect to be dealt with for Performance Requirement Analysis phase. This aspect deals with the deployment strategy of the application specially addressing parameters such as Availability and Reliability. The considerations in this aspect are:

- Distributed application
- Centralized application

Though this aspect caters the needs of Availability and/or Reliability, it has profound impact on Performance. Hence, the need to consider this aspect during Performance Requirement Analysis phase.

Gathering & Analyzing workload: This is yet another important aspect to be considered for effective Performance Requirements Analysis phase. The following are some of the important activities to be performed as part of Analyzing Workload patterns:

- Analyze on Transaction volumes & usage patterns
- Analyze on Data volumes

Forecast on estimated growth in transactions & data volumes

- Analysis on Response-time and throughput requirements
- Analysis of total number of databases & applications to be hosted on the server
- Analyze on Load and nature of existing databases & applications
- Analysis on Resource consumption by each database & application.

Above stated activities will be done as part of due-diligence. However, accomplishment of certain activities related to achieving Response-time/Throughput/Optimal resource consumption can be addressed thru Query optimization techniques.

The goal behind Query optimization is to facilitate the Query Optimizer in generation of optimal Query plans in terms of optimal resource utilizations.

D. Our Research Outline

In this paper, we propose an approach that emphasizes on 3 important ideas that should help improve overall database system performance.

- ✓ Enable Database to analyze the stream of incoming queries and project viable indexes as against the initial indexes created by the Administrator [9].
- ✓ Storing and re-using of Optimal Query plans that are generated by POP [1,2]. Here we propose to use the idea/s of existing POP models and extend this model for saving and re-using the Optimal Query plans generated by POP.
- ✓ Once the Query plans are stored, perform certain Pattern's identification using Data mining techniques by applying principles of 'Classification' & 'Association rules'.

II. Related Work

Query optimization is at the core for contribution towards performance improvements in application systems. A lot of ideas have been proposed towards Query optimization and there is a lot of on-going research happening in this area. One of the ideas proposed by Volker Markl, Vijayshankar Raman and team [1] is "Robust Query Processing through Progressive Optimization" which helps in efficient execution of Queries by evaluating cardinality estimations dynamically.

Query Optimizers choose the best Query Execution Plan for a Query using a cost model that relies heavily on accurate Cardinality estimation. Hence, it is important that cardinality estimations should be accurate which will otherwise lead to selection of sub-optimal plans leading to performance issues. Cardinality estimation errors should be avoided to overcome the above stated issue.

Most modern query optimizers determine the best plan for executing a given query by mathematically modeling the execution cost for each of many alternative QEPs and choosing the one with the cheapest estimated cost. The execution cost is largely dependent upon the number of rows (the row cardinality) that will be processed by each operator in the QEP, so the optimizer first estimates this incrementally as each predicate is applied by multiplying the base table's row cardinality by a filter factor or selectivity for each predicate in the Query[5,6,7,8].

POP is a technique to make Query Execution Plan's (QEP) robust, by repeatedly re-optimizing a Query during run-time if the cardinalities estimated during optimization prove to be significantly incorrect. Current Optimizers depend heavily upon the Cardinality estimations. There can be possibility for errors in cardinality estimations. Errors can be due to:

- Inaccurate Statistics
- Invalid assumptions (e.g. Attribute Independence)

The idea behind POP is to "lazily trigger re-optimization during execution if cardinality counts indicate current plan is suboptimal". POP introduces:

- Checkpoint (CHECK) operator to compare actual Vs estimated cardinality.
- Key Idea: Pre-compute cardinality ranges for which plan is Optimal.

The CHECK operator will help in identifying if a plan is sub-optimal.

Progressive Query Optimizer will

- Find out cardinality range for which plan is optimal during Optimization time.
- Check at run-time to ensure cardinality range.
- Stop plan execution and re-optimize if the cardinality range is violated.

When POP encounters violation of CHECK operator, it triggers re-optimization taking observed cardinality into account during previous run. POP also exploits intermediate results where beneficial and limits the number of re optimizations with default = 3.

III. Model for Building Dynamic Indexes & Storage & Reuse of Pop Plans

Our model for generation of indexes dynamically and Storage & Reuse of POP plans is based on a real-time scenario. Basis this scenario, we propose the following:

- ✓ Idea to handle the scenario
- ✓ Problem relevance to IT industry
- ✓ Conceptual Architecture for the proposed idea
- ✓ Objectives to be met thru idea implementation
- ✓ Expected outcomes

A. *Scenario*

Suppose that a particular Transaction Processing system (OLTP) has set of queries say 'n'. Assume that on Day-1, the business users have executed transactions that resulted in running 'n-10' queries.

At a later point in time (on the same day or subsequent day), users have executed transactions that resulted in triggering say 'n-2' queries.

B. *Idea to Handle the Proposed Scenario*

System shall store stream of queries issued by the user and analyze the pattern of the queries [9] and identify for each table the columns on which the queries are frequently issued. Accordingly, DBMS shall drop the unused indexes and create indexes for the frequently used columns used in Query conditions.

Also, for each query that is initiated by the user, POP [1, 2] will identify the best optimal plan (Progressively i.e., during Run time) and execute the Query. We propose to save the optimal plan identified by POP for every query that is executed in the database. For all subsequent initiations of the same queries by the user/s, the DB manager shall pick the stored optimal plan and re-execute.

Doing so, in an ideal scenario (with an assumption that the cardinality of the tables involved in the Joins would not have changed drastically), the proposed idea will reduce the cost of Query execution and also brings in significant performance gains.

Additionally, we also propose to perform certain Pattern's identification using data mining techniques by applying principles of 'Classification' & 'Association rules' etc on the Queries that have been initiated by the user community.

C. *Problem Relevance to IT Industry*

Performance of software systems is at the core of Customer Satisfaction. It is quite obvious that any application system that meets all the business requirements but fails to meet Performance Quality of Service (QoS) parameter will indeed lead to greater customer dissatisfactions.

Performance aspect of any application system will be dealt in multiple ways and they are:

- Application Architecture definition phase
- Database design phase

Hence, for large enterprise application projects, in addition to a set of Application architects, a dedicated Data Architect/s will be assigned whose aim is to focus purely on Database side.

A well designed database, in addition to providing robust data retrieval mechanisms (e.g. Using Java-Oracle Objects to facilitate easy data transmission where data is hierarchical in nature) across layers, should also consider implementing effective Query Optimization techniques. Doing so will lead to complementing towards meeting of overall performance expectations of the target end-user community.

Hence, our proposal to dynamically build indexes and to store & Reuse of Optimal Query plans generated thru Progressive optimizations will contribute towards meeting the Performance QoS parameter in any software system. Further, we are hopeful that this POC will result in a Solution that is generic to most of the Database servers that are available in the market.

D. *Conceptual Architecture*

Fig. 3 below gives the proposed Conceptual architecture for the implementation of ‘Model for building Dynamic Indexes and Storage & Reuse of Optimal Query Plans’ idea.

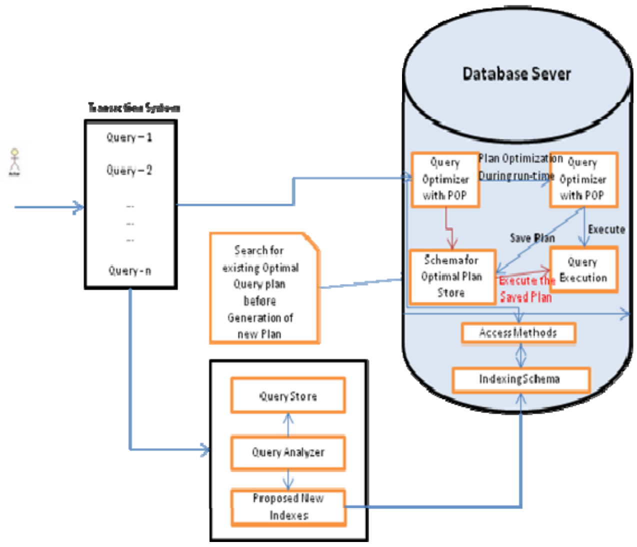

Fig. 3. Conceptual Architecture For Performance Improvement Model

- The lines in RED color indicate the path of re using the already saved optimal plan for execution.
- The lines in BLUE color indicate the path of Query optimizer generating a plan and executing and re-optimizing on the fly if the plan is found to be non-optimal.

E. *Objectives of the Idea Implementation*

This section proposes the objectives of the model to realize the idea. The objectives are very much aligned to Performance gains of the application system. Below are the envisaged objectives:

- Analyzing Stream of input queries and identifying new indexes those are potential for database system performance.
- Creation of newly identified indexes as per the above step.
- Materialization of Optimal Query plans based on Progressive Optimization technique for each initiated query.
- For every optimal plan saved against each query, also record the current cardinality basis which the Optimal Query plan has been chosen by POP.
- Re-use the saved optimal plans for subsequent Query executions, thus resulting in reduction of time by avoiding the Query Optimizer to generate plans and identify the optimal plan for execution.
- Perform certain statistical analysis in identifying the trends in change of optimal plans for each query by POP due to change in cardinality in the tables involved in joins.

F. *Expected Outcomes*

The expected outcomes of this research proposal are to quantify the performance gains by virtue of implementing this model. We wish to run set of sample queries on:

- Database with POP and capture the Cost metrics in terms of CPU utilization and number of I/O block transfers.
- Execute the same set of queries in a similar database (consisting of POP implementation) and with the ability to save POP chosen optimal query plans and collect the Cost metrics.

Once we have captured the Cost metrics for 2 different cases stated above, the difference in the cost metrics should indicate the proposed performance gains.

IV. Conclusion

In this paper we introduce a conceptual architecture that addresses performance improvement for software systems at the database tier in three stages. The first stage emphasizes on tuning of Index objects of the database system based on the analysis of queries submitted by the users. The second stage emphasizes on improvements in Access methods. The third stage emphasizes at storing the shortlisted query plan along with current cardinality in persistent storage and suggesting to reuse the query plan for subsequent execution of the respective query initiated through Online system provided the difference between the stored cardinality and actual cardinality is marginal.

References

[1] V. Markl and V. Raman, “Robust Query Processing through Progressive Optimization,” presented at SIGMOD 2004, France, June 13-18, 2004.

[2] V. Raman, V. Markl, D. Simmen, G. Lohman and H. Pirahesh, “Progressive Optimization in Action,” presented at VLDB Conference, Toronto, Canada, 2004.

[3] P. Bizarro, N. Bruno and D.J. DeWitt, “Progressive Parametric Query Optimization,” presented at Microsoft Research, August 2006.

[4] H. Kache, W. Han, V. Markl, V. Raman and S. Ewen, “Progressive Query Optimization for Federated Queries in DB2,” presented at VLDB Conference, Seoul, Korea, September 12-15, 2006.

[5] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, “Access path selection in a relational DBMS,” presented at SIGMOD'79, Boston, May 1979.

[6] A. V. Gelder. “Multiple Join Size Estimation by Virtual Domains”, presented at 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, Washington, D.C., USA, May 25-28, 1993, pp. 180-189.

[7] A.N. Swami and K.B. Schiefer, “On the Estimation of Join Result Sizes,” presented at EDBT, UK, March 1994.

[8] R. Ahad, K.V.B. Rao, and D. McLeod, “On estimating the cardinality of the projection of a database relation,” *ACM Transactions on Database Systems*, Vol. 14, Issue 1. March 1989, pp. 28 – 40.

[9] Y. E. Ioannidis, T. Saulys and A.J. Whitsitt, “Conceptual Learning in Database Design,” *ACM Transaction on Information System*, vol. 10, No.3, July 1992, pp. 265 – 293.

Sreekumar Vobugari is currently pursuing PhD program in department of computer science and engineering at the National Institute of Technology, Warangal, A.P, India. He holds a Master of Engineering Degree in computer science from Jadavpur University, Kolkata, 1998 and a Bachelor of Engineering in computer science and engineering from Gulbarga University, 1992.

He has over 17 years of IT industry experience and has played various roles such as Programmer, Module Lead, Database Architect, Project manager etc.. for various global customers during his stint working for some of the major Software consulting companies in India. His has published a Conference papers titled ‘An approach for Database Size Estimation’ in IEEE-International Advance Computing Conference, 2009 Patiala India followed by ‘Index Tuning through Query Evaluation Mechanism Based on Indirect Domain Knowledge' in UKSim 14th International Conference on Computer Modeling and Simulation,2012, Cambridge,UK. He has filed a Defensive Publication on 'System and Method for defect tracking in software projects'. His research interest is around Performance Engineering, Software Architectures and Large scale distributed data processing.

D. V. L. N. Somayajulu is a Professor in Department of Computer Science and Engineering at the National Institute of Technology (NIT), Warangal, AP, India. He holds a PhD in Computer Science and Engineering from the Indian Institute of Technology, Delhi, India.

He has over 25 years of Academic and Industry experience. He has headed the Computer Center at NIT Warangal and has been Head, Department of Computer Science for over 4 years. He has been awarded the Best Engineer of the Year in 2007

His area of interest is around Database Performance, Data mining and eLearning. He has carried out various Software projects sponsored by Government of India and has organized various Conferences and workshops. Currently he is guiding and mentoring 6 PhD students in various fields of Databases. Some of his publications are 'Ontology Matching schema integration using node ranking' at International Conference on Semantic Web and Web Services, June, 2006, 'Sentiment Classification of text reviews using novel feature selection with reduced over-fitting' at International al conference on Internet Technologies and Secured Transactions, November, 2010.

B. M. Subraya is a Vice President at Education and Research Unit of Infosys Limited, India. He hold a PhD degree in Computer Science and Engineering from the Indian Institute of Technology, Delhi, India. He has over 25 years of experience in Academic and Research areas. He has spearheaded the inception the Foundation Program for the fresh engineers entering into Infosys at their Global Education Center at Mysore. He has several International publications to his credit and has authored a booked titled 'An Integrated Approach to Web Performance Testing, A Practitioner's Guide'.