International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 10, October 2016)

Hybrid Architecture for Query Optimizers using Checkpoints with Plan Reduction Algorithms

Deepika Malviya[1], Syed Imran Ali[2], Zoha Usmani[3]
Dept. of CSE, Sagar Institute of Science Technology and Research, Ratibad, Bhopal-462044 (M.P), India

***Abstract* – Query Optimization is an important component of all Database Systems. Designing optimizers which takes less search time yet provides the most optimal query execution plan has been a challenge for DBMS research community in the last decade. Most of the optimization techniques focus on the static compilation of selectivity of base relations. The cost of executing a plan depends heavily on selectivity which keeps changing frequently and thus static compilation provides an inconsistent performance. Adaptive query optimization is an excellent method of generating optimal plans consistently. This paper proposes new hybrid architecture for query optimizers which combine features of adaptive query processing and also reduces the search space for re optimization using reduced plan diagrams and cost diagrams. This hybrid architecture is bound to give more efficient performance as compared to any other optimization technique along with increased robustness and a substantial increase in consistency of selecting most optimal execution plan**

***Keywords*– Query Optimization, Selectivity, Plan Cardinality, Plan Diagrams, Checkpoints**

I. INTRODUCTION

Query Optimization is a non trivial task for every commercial database management systems. It is that step in query processing that determines how much time will be consumed for executing a query. Since SQL query is declarative in nature, no information regarding execution sequence is provided by the user. Thus finding out the optimal sequence of execution becomes the overhead of database query optimizers. Query processing is a multi step process. A simple query written in a declarative language such as SQL is first converted to an equivalent relational algebra expression and is then converted into a query tree which is primarily left deep or right deep trees as shown in Fig. 1(11).

Q1: SELECT Lname **FROM** EMPLOYEE, WORKS_ON, PROJECT **WHERE** Pname= 'Aquarious' AND Pnumber=Pno AND Essn=Ssn AND Bdate>'1957-12-31'

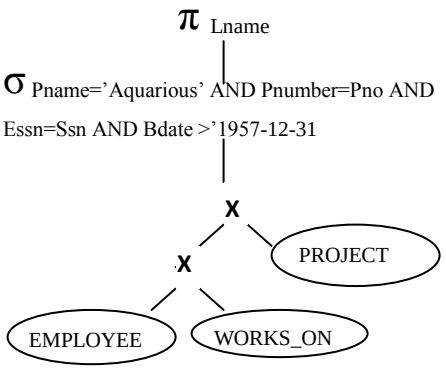

Fig. 1. Query tree for Q1

Problem in query optimization is that many such Query trees can be constructed by shuffling the positions of leaf nodes and non leaf nodes or by changing the relational algebra operations (such as using a hash join instead of nested loop join). Each such query tree corresponds to a specific query execution plan.

The task of a query optimizer is to analyze all the query execution plans and select the optimal plan for executing the query. The selection for best plan is done by applying some rules (heuristics optimization) and by using some cost functions (cost based optimization). The number of query execution plans (hereby referred as QEP) for a given plan or the plan cardinality may be huge which makes it impossible to analyze each and every plan for optimality. If much time is spent in just searching the best plan, its execution won't prove to be much beneficial. Thus a tradeoff between searching time and execution time is necessary. Because of this most query optimizers put efforts to search for the near optimal QEP instead of searching for the most optimal plan.

Another problem faced by optimizers is the selectivity of base relations. The choice for best plan is made on the basis of some complex cost functions whose major parameters are the selectivity of base relation.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 10, October 2016)

Since the selectivity keeps on changing frequently the cost calculation becomes wrong and a sub optimal plan may get selected. Thus a dynamic calculation of selectivity is required for getting correct value of cost functions. Static compilation of selectivity is highly prone to errors.

This paper proposes a new architecture for Query optimizers which combine features from two techniques, one for handling selectivity problem and another for reducing plan cardinality.

II. Related Work

A lot of research work has been done on SQl query optimization in the last decade with none of them addressing both problems of selectivity and plan cardinality. Some major contributors are [4], [1], [12]. All these methods were effective but used static compilation of selectivity. Also search space reduction was not well taken care of by any of these techniques. But [5] suggests a very practical approach of eliminating the problem of wrong selectivity estimation. Another pioneer work suggested in [6] uses a tool named Picasso for reduction of plan diagrams for reducing search space. This paper uses both these techniques to develop new hybrid architecture for query optimizers.

III. Using Plan Diagrams For Query Optimization

Plan diagram reduction is a novel way for query optimization. A plan diagram as defined in [7] is a color coded pictorial enumeration of the QEP choices of the optimizer over the query parameter space. QEP choices are primarily functions of the selectivity of base relations in queries. Using PICASSO tool for QEP analysis we can perform reduction on the number of QEP for a given query. Some threshold value is required for plan cardinality reduction by the process of swallowing [9] which indicates an increase in cost of the QEP. As proven in [7], a 20% cost increase can reduce the number of queries near about 10 which can significantly decrease the searching time of optimizers. For complete description of plan diagrams and PICASSO tool see ([7], [9], [6], [14]).

IV. Progressive Optimization Using Checkpoints

Since selectivity keep on changing frequently, any method based on static selectivity estimates may not deliver a consistent performance. On the contrary we can go for complete dynamic estimation of selectivity as in [1]. But most practical way to eliminate selectivity errors is to apply re optimization along with selectivity estimates at run time [5]. This is done by inserting CHECKS at selected points in the query execution tree (Fig. 2).

Using these checkpoints we can identify whether the selectivity of base relation or the intermediate results is within the expected limits or it has crossed the limits? This information derived at a checkpoint is then used to decide whether to stick on to a QEP or to make some reordering of operators and base relations. As shown in Fig 2, we use a CHECK to determine the selectivity of R1. If the selectivity exceeds the validity limit we replace hash-join with hybrid hash join which gives better performance if the hash file is too large to be saved on the main memory. Since the Checks calculate current selectivity, the chances of errors are very less. But excess of checkpoint can also slow down execution time. So there is a risk and opportunity tradeoff due to which it is important to determine how many CHECKS are to be inserted and where to be inserted.

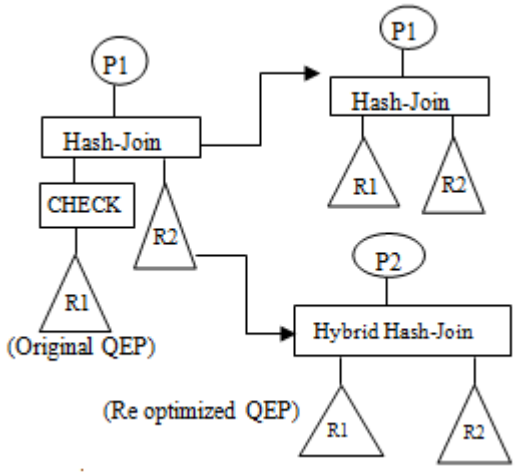

Fig. 2 Using CHECKS for re optimization

V. Our Contribution

This paper proposes hybrid architecture of query optimizers which combine features of adaptive query optimization and reduction of plan cardinality. The problem in using CHECKS is that during re optimization we need to repeatedly search large numbers of structurally equivalent plan (fig.4). Since there is plethora of such additional plans, re optimization time becomes huge which may degrade the performance of optimizer. In any other technique this search is performed just once before the execution of query begins. Also computing the validity ranges for CHECKS is time consuming. Thus there is a need of decreasing the re optimization time by reducing the plan cardinality which is achieved by using reduced plan diagrams.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 10, October 2016)

a. Proposed Architecture

This paper consideres some additional stages apart from the basic stages in query optimizers and also appended some additional features in the basic stages. The overall architecture is shown in fig.3.

b. Benefits of Hybrid Approach

This architecture includes the basic components of a query optimizer with additional features to incorporate CHECKS and one additional block is used to include Plan/Cost Diagrams in the overall process. The benefits of this hybrid architecture are twofold as described:

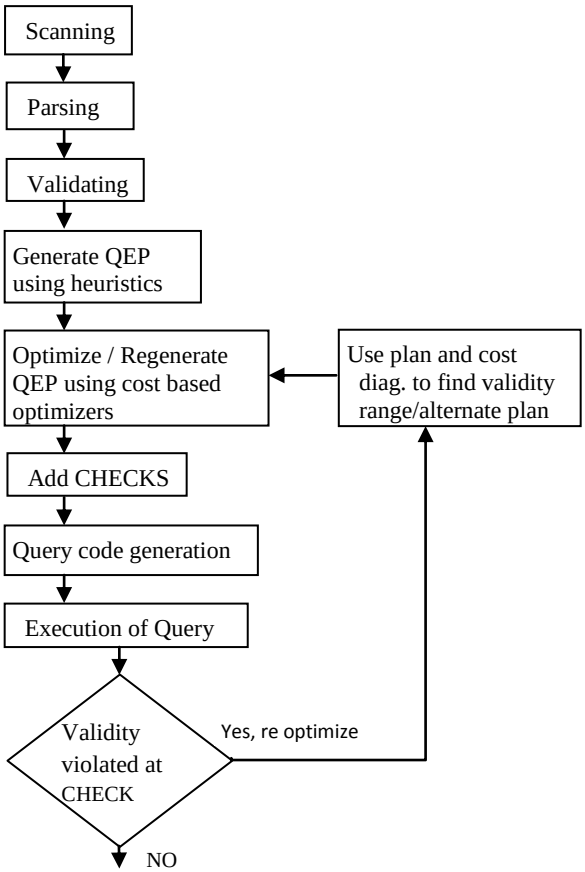

Fig. 3 Hybrid architecture of Query Optimizer

1. The only instance of plan/cost diagrams occur in the 5th stage when a QEP is being generated and before CHECKS are inserted into the QEPs. We use Plan/Cost diagrams at this stage to find near optimal QEP and the validity range for CHECK.

Validity range is computed during plan pruning and enumeration phase. A reduced enumeration of plans can be generated by reducing plans by giving the cost increase tolerance factor. Now, validity computation and plan pruning is carried out concurrently. For pruning sub optimal plans we compare each pair of plans based on their input cardinalities. The plans being compared must be structurally equivalent Fig. 4

For, pruning we compare the cost of P1 and P2 at their root nodes. Suppose that cost of P1 exceeds cost of P2, and then P1 will be pruned. To find the validity range we continue using the modified Newton-Raphson method.

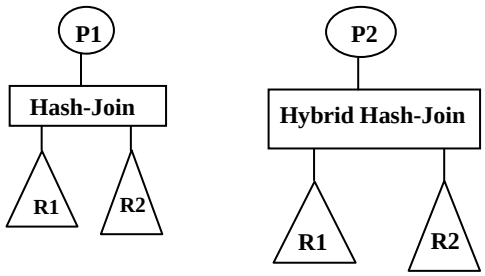

Fig. 4 Two structurally equivalent plans

The benefit we get using reduced plan/cost diagram is that Newton-Raphson method is repeatedly used for evaluation of cost functions for large number of plans. Using plan diagram reduction we can reduce the number of such pairs because of which pruning and validity calculation can be done more quickly. As proved in [7] if cost increase threshold of 20% is tolerable then the total number of reduced plans remains near about 10. This is a substantial decrease in number of plans which can allow us to insert more number of CHECKS in the QEP as validity calculation time is minimized. With more number of CHECKS the robustness of QEP will increase.

2. The same instance of Plan Diagrams in the 5th stage can be used to search for the alternative plan in case the CHECK operator detects the current QEP to be sub optimal as the selectivity of input edges to a relational operator bypasses the validity range.

Suppose as in Fig. 2, the CHECK operator detects the selectivity of left edge in plan P1 is very large so that the hash file cannot be saved on the main memory rather it has to be saved on disk. In this case simple hash-join will become complex and so it becomes important to replace it with either hybrid hash join or partition hash join [11]. Such replacements can be easily done using the plan diagrams by using the cost information of the available additional plans.

Since the CHECK operator has generated the exact selectivity of input edge, using this information we can perform a search for the alternate plan in the reduced search space. We need to search for new plan only in the limited selectivity space using selectivity information form CHECK operator.

VI. Future Work

Implementation of the proposed architecture is the most challenging task. We are in process of implementation of this architecture and hope to discover many other possibilities of optimization during its implementation. Once such architecture is implemented we can go ahead for the performance analysis of this architecture using TPC-H query set.

VII. Conclusion

Static query optimization suffers from the fact that correct estimates of selectivity/cardinalities are hard to be made as they keep changing frequently and thus may result in poor selection of QEP. Progressive query optimization uses a runtime approach for estimating correct selectivity using CHECKS and performs re optimization if required. Integration of Plan/Cost Diagrams with CHECKS further improves the efficiency of query optimizer by reducing the search space and minimizing validity calculation time. Further the overall QEP used for query execution is supposed to be very robust in nature.

Since we are in the initial stages of developing such architecture the information presented is very abstract. As we proceed to implement such architecture we expect many other parameters which will require substantial attention and may open new areas of research.

REFERENCES

[1] R Avnur and J. M. Hellerstein, Eddies: Continuously Adaptive Query Optimization, SIGMOD 2000

[2] G. Graefe and K. Ward, Dynamic Query Evaluation Plans, SIGMOD 1989

[3] R Cole and G. Graefe. Optimization of Dynamic Query Evaluation Plans, SIGMOD 1994

[4] A. Hulgeri and S. Sudarshan. Parametric Query Optimization for Linear and Piecewise Linear Cost Functions, VLDB 2002

[5] V. Markl, V. Raman, D. Simmen, G. Loman, H. Pirahesh, M. Cilimdzic, Robust Query Processing through Progressive Optimization, SIGMOD 2004

[6] N. Reddy, J. R. Haritsa, Analyzing Plan Diagrams of Database Query Optimizers, VLDB 2005

[7] N A. Dey, S. Bhaumik, Harish D. and J. Haritsa, Efficiently Approximating Query Optimizer Plan Diagrams, VLDB, 2008.

[8] A. Deshpande, Z. Ives and V. Raman, Adaptive Query Processing, Foundations and Trends in Databases, 2007.

[9] Harish D., P. Darera and J. Haritsa, Robust Plans through Plan Diagram Reduction, VLDB 2005

[10] R Avnur and J. M. Hellerstein, Eddies: Continuously Adaptive Query Optimization, SIGMOD 2000

[11] R. Elmasri, S. B. Navathe, Fundamentals of Database Systems, 5th edition

[12] M. Stillger, G. Lohman, V. Markl, and M. Kandil, LEO – DB2's Learning Optimizer, VLDB 2001.

[13] V. Poosala, et. al, Improved Histograms for Selectivity Estimation of Range Predicates, SIGMOD 1996

[14] Picasso Database Query Optimizer Visualizer, http://dsl.serc.iisc.ernet.in/projects/PICASSO/picasso.html

[15] Mahajan Chetas Subhash, Producing on the fly anorexic plan diagrams using AniPQO dsl.cds.iisc.ac.in/publications/thesis/chetas.pdf