PICASSO 1.0

Design and Analysis

A Project Report

Submitted in partial fulfilment of the

requirements for the Degree of

Master of Engineering

in

Internet Science and Engineering

by

Tarun Ramsinghani

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

July 2007

Acknowledgements

It is a pleasure to thank all those who made this thesis possible.
I would like to thank my guide Prof Jayant R.Haritsa for giving me the opportunity to work with him. Thank you sir, for your valuable guidance, for the patience with which you taught me, for your support and encouragement.

I would also like to thank my DSL lab-mates Pooja, Rupesh, Shruthi for providing a stimulating and fun environment for work. Life without friends in IISc would have been difficult and dull, so i'll like to thank all my friends in IISc especially Mehul, Megha, Rashmin and Prachee. Thanks, to all those whom I have not mentioned here,but have helped me in some or the other way in completing this work.

Lastly but most importantly, I'll like to thank my parents, my sister and my brother for having faith and confidence in me, for encouraging and supporting me. I dedicate this thesis to them.

Abstract

Picasso is a tool developed in Database Systems Lab, which produces various diagrams, showing the behavior of a database query optimizer. The beta version of the tool was already available. We have improved the tool in may aspects and released the new version Picasso1.0.

In the new version, enhancements have been made on five fronts namely, functionality, presentation, performance, algorithmic and experimental. All these enhancements have resulted in a much more efficient and useful Picasso 1.0. Many new features like foreign plan comparison to compare plans of two different optimizers, command line interface which can be used for producing the batch of diagrams, have been added in the new version, enhancing the functionalities provided by Picasso. The layout of the plan trees have also been changed improving their presentation. To improve the performance of the tool some changes have been made like, the operator level diagrams are now generated on client side without actually going through the process of compiling millions of queries again for producing the diagrams. The plan reduction algorithm have been improved. Now instead of area based comparison, perimeter or border based comparison is used, reducing the complexity of the algorithm.

Many new experiments have been performed using Picasso1.0 which have shown very interesting results about different aspects of the query optimizers. It has been observed that the complexity of the diagrams is not affected even if the plan differentiation is based on operators only. The other observation made is that most of the complexity is

near the origin and axes. If we increase the number of sample points in these areas, the plan cardinality increases by more than 100% leading to even more complex diagrams. It has also been shown that the output cardinality estimates made by the optimizers can be off by large amounts. Experiments have also been performed on the newly released TPC-DS benchmark, which is much more complex and bigger in size than TPC-H benchmark. The initial results of the experiments with TPC-DS benchmark indicate that the current query optimizers are unnecessarily too complex.

Contents

List of Figures

Chapter 1

Introduction

SQL, the standard database query language is declarative in nature. It tells the database system what to do but does not specify how to do. For example, consider the query shown in Figure 1.1 which fetches the name of those employees who earn less than 8000, along with their respective department names. It does not specify how to join the tables `emp` and `dept` i.e. whether to use Nested-Loops join, Sort-Merge join or Hash join.

In a database system it is the responsibility of the module called query optimizer to come up with the strategies known as **execution plans**, for executing the given SQL query. Query optimizer is a vital component of modern database systems because the difference between the cost of the best execution plan and a randomly chosen plan could be in orders of magnitude. Hence, query optimization is a necessary step before the

```
select empName, deptName

from emp, dept

where emp.deptNo = dept.deptNo and

      emp.salary < 8000
```

Figure 1.1: A Example Query

actual execution of the query. The performance of the system is highly dependent on the quality and functionality of the optimizer.

Most of the current query optimizers are cost based, with few rules included. The optimizer takes the given query and with the help of system catalogs and cost model, it comes up with a minimum cost plan. The efficiency of the plan is usually measured in terms of the query response time. An efficient optimizer should come up with a good execution plan without consuming too much time. The need for an efficient optimizer is indubitably evident from the complex queries of TPC-DS[6] decision support benchmark because as the complexity of the query increases the time required to optimize it also increases.

1.1 Picasso Tool

For a given database and system configuration the optimal plan choice given by a cost based query optimizer is mainly the function of the selectivities of the relations participating in the query. The *selectivity* is defined as the estimated number of tuples or rows of a relation that are relevant for producing the result of the query.

Picasso[7] is a tool developed in Database Systems Lab, that is used to analyze the behavior of a query optimizer through the help of various graphs generated by Picasso which illustrate the functioning of various aspects of query optimizers as the selectivities are changed slowly over a range. These graphs reveal various properties of optimizers which otherwise would have been difficult to predict. On the other hand they are also helpful in validating the expected behavior of optimizers by looking at easy to interpret graphs of the optimizer output. There are four basic classes of diagrams that Picasso produces to characterize the behavior of an optimizer for a certain query template. They are:

1. **Plan Diagrams:** A ***plan diagram*** is a color-coded pictorial enumeration of the execution plan choices of a database query optimizer over the relational selectivity space. Plan diagrams are generally 1D or 2D diagrams where the dimensions represent the relations whose selectivities vary along the corresponding dimensions. These diagrams show the optimal plan chosen by the optimizer as the selectivity along various dimensions changes.

2. **Cost Diagrams:** These are similar to Plan diagrams except that the points on the graph shows the magnitude of the cost. These are 2D or 3D diagrams, the extra dimension is for cost. The nature of the cost depends on the diagram being shown i.e., ***compilation cost diagram*** will show the cost of executing the query, as estimated by the optimizer and ***execution cost diagram*** shows the actual cost of executing the query.

3. **Cardinality Diagrams:** These are like the cost diagrams but here the points, instead of referring to the cost, refer to the cardinality of the result for varying selectivities of the relations along the dimensions, based on whether it is a ***compilation cardinality diagram*** or an ***execution cardinality diagram***.

4. **Reduced Plan Diagrams:** Picasso also produces ***plan reduction diagram*** which shows how less complicated the plan space can get while not incurring substantial additional cost, thus providing a metric for quantifying decision granularity of the optimizer and its relative effectiveness. Plan Reduction is actually a process of replacing some of the plans in the plan diagram by some other plan so that the total number of different plans in the plan diagram can be minimized.

Apart from these diagrams, Picasso also shows the plan trees. **Plan Trees** are graphical representations of final plans that are produced by the optimizer for a given combination of selectivities of the relations. It also includes the feature that compares different plans and indicate the points of differences amongst them.

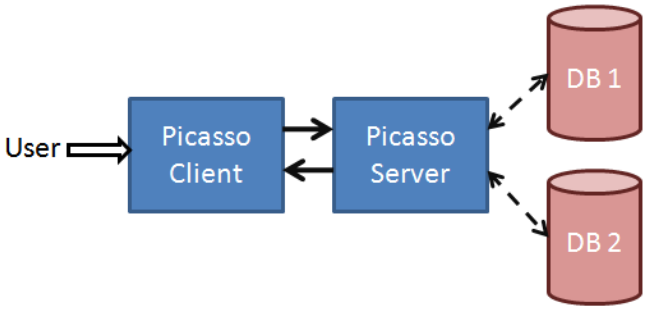

Figure 1.2: Picasso Architecture

Picasso consists of two components namely **Picasso Server** and **Picasso Client**(see Figure 1.2). *Picasso Server* handles the function of producing and serving the diagrams and *Picasso Client* is used for the visualization of the diagrams. Each time when user requests for some diagram, client sends the request to the server and then server sends the diagram to the client. Picasso uses the interfaces provided by the database engines and it does not the affect the behavior of database engine in any way i.e it works completely outside of database engines.

Picasso currently supports DB2, SQL Server, Oracle, Sybase, and Postgres.

Chapter 2

Performance Enhancements

2.1 Client-side Implementation of Operator-level Plan Diagrams

The plans given by the optimizers are tree like structures also known as plan trees. One such plan tree from OPT-A optimizer is shown in Figure 2.1. Each leaf node represents a relations used in the query. The internal nodes in the tree define an operator which indicates what is to be done at that particular node. For example, TBSCAN indicates that engine should do a sequential scan on the relation present as its child node, or HSJOIN indicates that engine should join the two inputs using the Hash Join technique.

Each operator can also have some parameters associated with it. The parameters indicate what resources can be used or how to perform the operation. For example in *HSJOIN*, the parameter *HASHCODE* defines the length of Hash (in bits) to be used. In case of *TBSCAN*, the parameter *SCANDIR* indicates in which direction (FORWARD or BACKWARD) the scan of the table should be performed

While differentiating two plans, one can make use of operators alone, or can also take the parameters into account. Using these two criteria for differentiation two kinds of

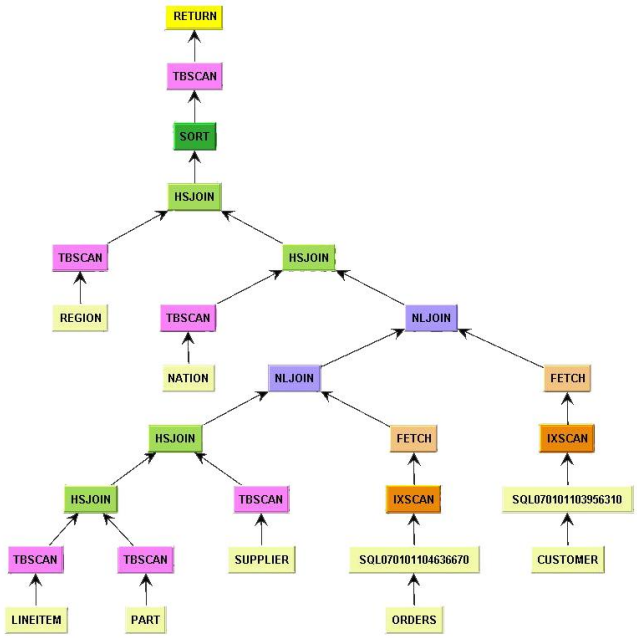

Figure 2.1: Plan Tree for Q8 of TPC-H benchmark

diagrams are obtained: a) Diagrams based on operator level differentiation, b) Diagrams based on parameter level differentiation.

Sometimes the parameters are merely the hints to the database engine and can be ignored by the database engine at runtime based on the availability of resources. So the number of plans in a parameter level diagram gives the upper bound on the plan cardinality of plan diagrams. Whereas if the operator level differentiation is used then the number of plans in an operator level diagram gives the lower bound on the number of different plans that are present in the plan diagram. User may want to know this lower bound on the number of different plans present in the plan diagram ignoring the information regarding the parameters.

It is obvious that the parameter level diagrams contain enough information for generating the operator level diagrams. The beta version of Picasso treated these two diagrams as completely different entities. Hence, for producing either of the two diagrams

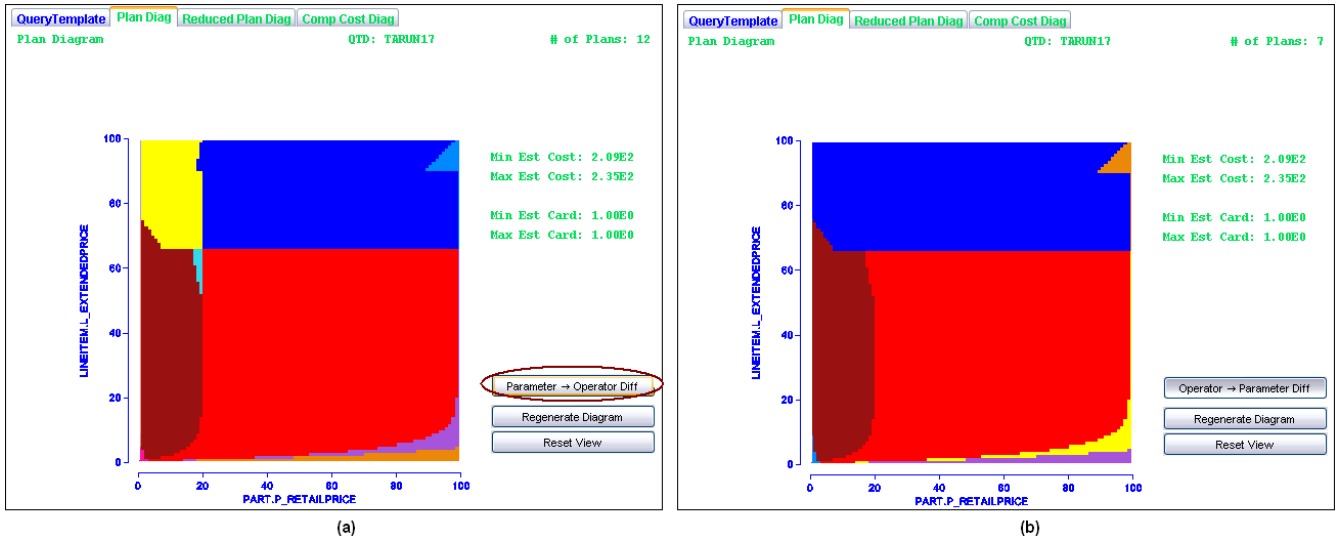

Figure 2.2: Interface for switching between parameter to operator level diagrams

the whole process of diagram generation was performed from beginning. The diagram production is a time consuming process and this production time can be saved for operator level diagrams.

The new version of Picasso, utilizes the fact that the operator level diagrams can be generated from the information contained in the parameter level diagrams. Here, the production of operator level diagrams have been shifted from server side to client side. Picasso server now produces only the parameter level diagrams. If a user wants an operator level diagram then by using the interface provided at Picasso Client(see Figure 2.2a) he can easily switch between the two diagrams. Figure 2.2b shows the operator level diagram that has only 7 plans whereas the corresponding parameter level diagram on left has 12 plans.

The process of switching is simple. Once a parameter level diagram has been obtained at the client side, pairwise comparison of all the plans is performed, after discarding the parameter information. Then a check is performed as which plans are same at the operator level. Once all different operator level plans are known one can recolor the diagram

based on new differentiation scheme. This way, compilation time for lots of queries is saved, which is a substantial part of the diagram production time.

2.1.1 Comparison of plan cardinalities between parameter level vs operator level diagrams

An earlier study done with Picasso[7] has revealed that plan cardinalities of plan diagrams can be very high. This study was based on parameter level diagrams. This implies that optimizers are producing large number of different optimal plans. Here one may expect that most plans would differ only at the parameter level and most of the plans from the plan diagram would disappear in an operator level diagram.

A study was performed to compare the plan cardinalities of operator level diagrams and parameter level diagrams. All the experiments were performed using the TPC-H benchmark. Both uniformly distributed dataset and exponentially distributed dataset were used for the experiments. The numbers shown here are sum of the plan cardinalities for each individual query over both the datasets.

It has been found that even for the operator level diagrams the complexity and cardinality of the plans have not been reduced by much. Figure 2.4 and Figure 2.3, shows the plan cardinalities comparison for OPT-A and OPT-B optimizers between the two type of diagrams over the set of TPC-H query templates. As can be seen, there is not much difference in the cardinalities. Which means that the difference between the plan cardinalities of parameter level diagram and operator level diagram is very less. On an average, only 23% and 13% of plans disappear from parameter to operator level diagrams for OPT-A and OPT-B optimizers respectively.

Same kind of study was performed on OPT-C and OPT-D optimizers. For OPT-C the difference is only of 1% whereas for OPT-D it was found that number of plans at

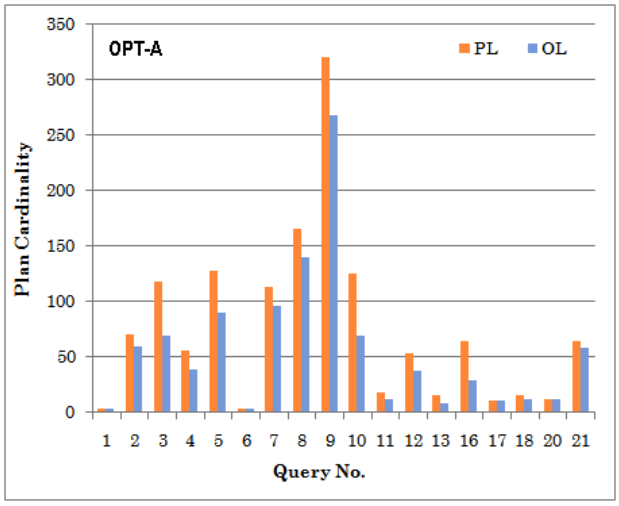

Figure 2.3: Plan Cardinalities comparison (OPT-A)

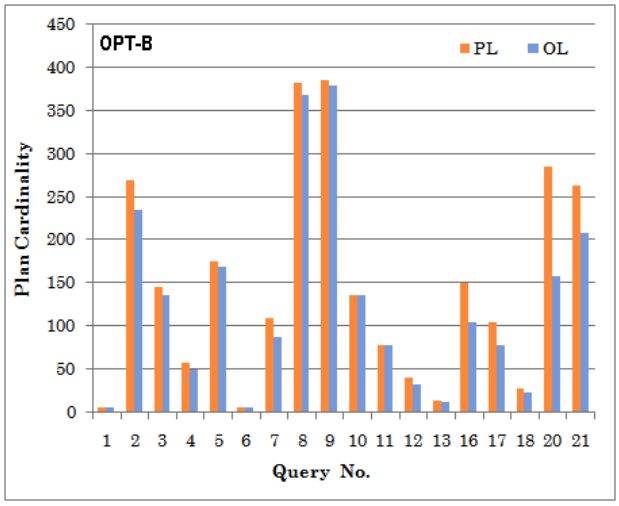

Figure 2.4: Plan Cardinalities comparison (OPT-B)

parameter level diagrams is exactly the same as the number of plans in operator level diagrams i.e 0% increase. These results suggest that even if the lower bound on number of plans is used, the generated plan diagrams still show complex behavior.

2.2 Improvement in *Generation Time Estimator* for Execution Diagrams

Before producing the actual diagrams Picasso estimates their generation times and indicates them to user. So that, user can decide whether to produce the diagrams or not. In case of execution diagrams the estimate was done as follows:

Execute a sample query belonging to center of the selectivity space i.e. the query belonging to the points where the selectivity is 0.5 *for each of the dimensions. Record the execution time for the query. Now interpolate the total generation time by multiplying the execution time for a single query with the total number of points on the grid.*

The assumption here is that the average time to execute each of the query is same. As the selectivity increases, number of rows to be processed also increase and since join nodes have non-linear cost functions, the query at higher selectivities take much longer time to execute than the queries at lower selectivities. Due to this non-linear behavior the above mentioned assumption may or may not hold.

For improving the generation time estimator of the execution diagrams the following two enhancements have been made. First, instead of using the middle point of the space, a point nearest to the origin is chosen. This reduces the estimation time because a query with low selectivity is used, implying that less number of rows have to be processed now and hence, the query can execute faster than the query at the middle point.

Compilation diagrams generated by the optimizer contains the estimated cost of

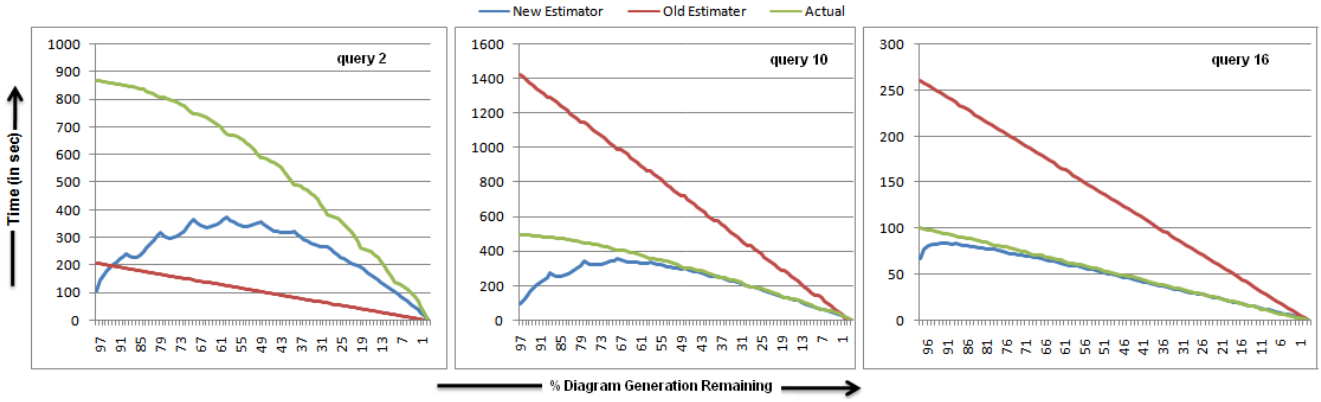

Figure 2.5: Comparison of Old and New Estimator with Actual generation time

executing the query. In the new version of Picasso this estimated cost is used for interpolating the execution time of the sample query over the complete space rather than the linear method used in the beta version. The method is described below.

Let the point nearest to origin be p.

Execute the query corresponding to point p. Let the actual execution time be A_p.

Let the estimated execution time for point p be E_p and the estimated execution times for all the points on the grid be $E_1, E_2....., E_n$ where n is the total number of points on the grid and $E_1 = E_p$

Then the total execution time T_E will be given by the formula:

$$T_E = \frac{A_p}{E_p} \times \sum_{i=1}^{n} E_i \tag{2.1}$$

Further as the diagram is being produced by running all the sample queries one by one, the estimates are progressively improved as now the actual running times for more

queries are known. Suppose m out of n queries are executed then the remaining estimated time R_E is calculated as:

$$R_E = \left(\frac{\sum_{i=1}^{m} Ai}{\sum_{i=1}^{m} Ei} \times \sum_{i=m+1}^{n} E_i \right) \tag{2.2}$$

Figure 2.5 shows the comparison of old estimator and new estimator on three TPC-H queries. The X-axis shows the % of diagram generation remaining. The Y-axis shows the time remaining in seconds for the diagram to be generated. The blue line shows the estimated time of new algorithm for completion of the diagram, red line shows the estimated time of old algorithm and green line shows the actual time to generate the diagram. It can be seen that as more information is available the estimates for new estimator are constantly improving. whereas old algorithm does not adapt and gives out the linear behavior and hence makes wrong estimates.

The results above shows that the new estimator is performing much better than the old estimator.

2.3 Caching of Diagrams at Client Side

To show any of the diagrams supported by Picasso for a single query template, the data needed is fetched from the server. Once this diagram data is available on the client side, all the Picasso diagrams can be produced using it. The beta version of Picasso retrieves the data from the server whenever a new diagram is requested by the user. The fetching of data can be time consuming because Picasso client and server could potentially be running on different systems.

In Picasso 1.0, once the diagram data is retrieved,it is cached at the client side for future use. Now if a request for some other diagram corresponding to the same query template is made, the cached data is used for producing the diagram, instead of fetching

it again from the server. The process of caching saves lot of data transfer time.

Chapter 3

Functionality Enhancements

This chapter deals with the new functionalities added to Picasso 1.0.

3.1 Plan comparison between different optimizers

While executing a query, the plans decide how to execute the query. For the same query different optimizers can give different optimal plans with different costs. User may be interested in comparing the plans given by two different optimizers.

Picasso 1.0, provides the facility for making the comparisons amongst the plans generated by different optimizers or the same optimizer with different optimization levels. The process is simple, one just needs to *CTRL+SHIFT+RIGHT_CLICK* on some point in the plan diagram and a foreign plan request dialog box appears as shown in Figure 3.1. In the window one can choose the foreign optimizer and the optimization level with which one wants to compare the plan. Once the request is submitted the server will generate the plans from both the local and foreign optimizers and then client displays them side by side for easy comparison.

Since different engines use different SQL dialects[10], while comparing the plans between two different optimizers, a problem may arise i.e., the query which is written for

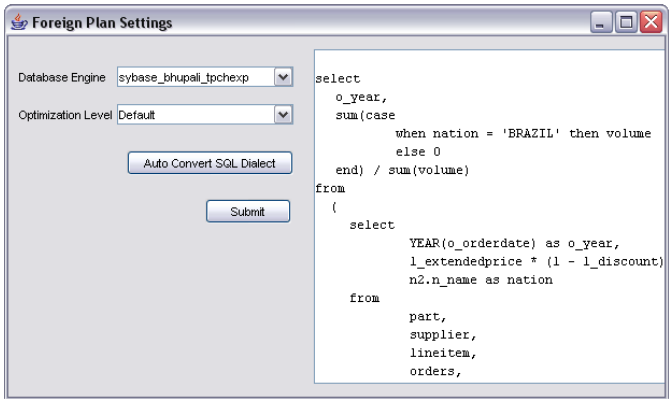

Figure 3.1: Foreign Plan request Dialog window

a particular optimizer say, OPT-A may be incompatible with the foreign optimizer say, OPT-B. To overcome this problem Picasso provides the facility to edit the query before it is given to the server. One can either manually edit the query when one requests for a foreign plan or one can use the auto convert feature provided with Picasso 1.0. It uses commercially available SwisSQL[18] library which does the automatic conversion of SQL Dialects. If the SwisSQL library is present one can simply click on the autoconvert button in the dialog box and the query will be automatically converted for the destination engine.

3.2 Command Line Interface to Picasso

In the beta version of Picasso the only client interface available was graphical interface which did not allow the batch production of the diagrams. For producing a batch of diagrams, user was required to sit at the terminal and generate each of the diagrams individually.

To allow users to separate diagram production from diagram visualization, a command line interface has been added in Picasso1.0. This command can be used to submit

diagram generation request to the Picasso server. The command is known as **PicassoCmd**. User can use the command as follows:

```
PicassoCmd <ServerName> <Port> <DBConnection> <OptLevel> <QTID> <QDist> <Resolutio
    <DiagType> <QTFile>
```

where,

- ServerName : IP of Picasso server
- Port : Port no. of Picasso Server
- DBConnection : Database Descriptor
- OptLevel : Optimization Level
- QTID : Query Template Descriptor
- QDist : Query Distribution
- Resolution : Resolution of Diagram
- DiagType : Diagram Type (Compilation or Execution)
- QTFile : Query Template File

Chapter 4

Presentation Enhancements

4.1 Tree Layout

Various enhancements have been made to plan tree layout. In the beta version of Picasso, all the leaves in a plan tree were placed at the same level. In this layout the width of the tree was quite large, making it quite hard to view the complete tree at once. Another problem with this layout was that it didn't make the structure of the tree clear i.e., it was hard to make out whether the tree is left deep, right deep or bushy tree. Picasso1.0 uses the generic tree layout as shown in figure 2.1.

Sometimes a node can have more children than its default number, which are the sub-queries to be executed and not the actual input. In the beta version of Picasso these extra child nodes were shown as normal children which may lead to wrong interpretations. Now the information about *arity* of operator has been added to Picasso 1.0 so as to distinguish these extra children. In Picasso1.0 these extra child nodes are distinguished by making the dashed link from their parents to them. This makes the plan tree easy to understand.

The other enhancement made is that in the case of OPT-A and OPT-C engines, the index scan node in the plan tree has two children i.e name of the index and name of the relation, and both the nodes are leaf nodes. This is shown in figure 4.1a. The

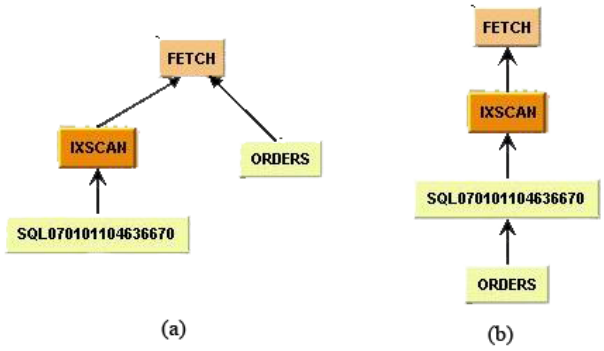

Figure 4.1: Change made in the tree layout for OPT-A and OPT-C

plan difference algorithm used in Picasso to find differences in two plan trees, needs that all the leaf nodes in plan tree should be relations. The older version of *plan difference algorithm* used the workaround by explicitly checking for these kind of index scan nodes while matching the relations in two different plans. In the new version of Picasso, the layout of the plan tree has been changed by making it compulsory for all the leaf nodes to be relations. For OPT-A and OPT-C engines this is done by making the relation node as the child of the corresponding index as shown in figure 4.1b. This does not change the semantics of the plan tree. This makes the plan tree layout for OPT-A and OPT-C to be consistent with other supported engines. After incorporating this new layout in Picasso1.0 these special checks for index nodes have been removed from plan difference algorithm, thus making it simpler.

4.2 Other Enhancements

OPT-B gives the parameter information in the form of a long string, which can also contain some Picasso specific information. This information should be ignored for generating the correct diagrams. While discarding this information the beta version also discarded some other important parameters. In Picasso1.0 this parameter string is carefully checked and only the Picasso related information is discarded and all other parameter

information is retained.

Secondly as OPT-E gives the plan output as ASCII text, Picasso incorporates a parser for displaying the plan diagrams using the ASCII text. The older version of parser sometimes ignored the sub-queries part associated with a node. To correct this error, the parser has been completely rewritten. Now it carefully builds the complete tree taking all the sub-queries in account unlike the beta version.

Chapter 5

Algorithmic Enhancement

5.1 Plan Reduction

Plan Reduction is a way to simplify the plan diagram so as to minimize the total number of different plans in the plan diagram such that the quality of subsequent query processing is not adversely affected.

Plan Reduction Problem is stated as follows:

*Given a 2D grid having N*M points where each point has an associated color(plan) and cost. Let total number of different plans over the complete 2D grid be p. Given a threshold value λ, the aim is to reduce the number of colors(plans) by recoloring the points on the grid without increasing the cost of any recolored point by more than λ%.*

For Q8 of TPC-H benchmark an instance of plan reduction is shown in Figure 5.1. It shows that plan diagram initially has 225 plans but after reduction with threshold of 10%, the number of remaining plans in reduced diagram is only 19.

As is evident from the results that with the help of plan reduction, the plan diagrams can be simplified to a great extent. This shows that the current optimizers are doing

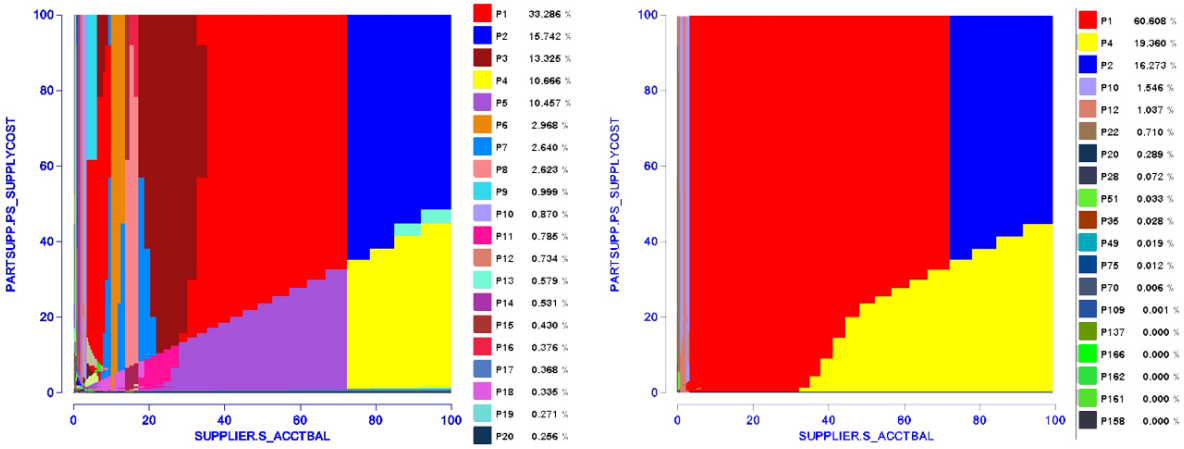

Figure 5.1: TPC-H Q8 Reduction from 225 plans to 19 plans with $\lambda = 10\%$

too good a job of choosing a plan, but instead they may have settled for some suboptimal plan, when it is given that these choices are made using the estimated values for selectivities. Our results shows that these estimates can be off by large amounts. Plan Reduction shows that the optimizers can be simplified to directly produce the reduced plan diagrams, potentially reducing the computational overheads associated with query optimization.

The problem of Plan Reduction is shown to be a NP-Hard problem[3]. The next subsection discusses a greedy approach for reducing the plan diagram. The strategy here is to try removing the color which is used for coloring the smallest number of points. This is helpful as now the replacement for less number of points have to be found.

5.1.1 Area based Approach

The old area based approach for solving the plan reduction problem is described as follows:

1. Create bucket B_i for each different plan i in the selectivity space and put all points

having the same plan in the corresponding bucket.

2. Sort the B_i buckets in ascending order of the size of the buckets. Let this list of buckets be $B_1, B_2, ..., B_n$.

3. for *i=1* to n begin

   (a) for *j=i+1* to n begin

      i. for each point p in B_i find a point q in B_j such that q is in first quadrant w.r.t p and cost(q) is within *(100+x)%* of cost(p).

   end

   (b) if all the points in B_i can be replaced then reassign the point in B_i to corresponding B_j's.

   (c) Empty the bucket B_i

   end

4. Output the points with new plans assignments.

Some inefficiencies found with this algorithm are:

- **All point comparison:** The second problem with, this approach is that it considers all the points within a bucket B_j as a possible replacement for the points in B_i. We will show that, under the cost domination principle, these comparisons can be reduced by a large extent, by considering only the borders of the plan regions as the possible swallowers.

- **One Sided Reduction:** This approach also assumes that a plan can be replaced only by a bigger plan i.e., it checks only bigger buckets as a possible swallower. The results can be improved by allowing smaller buckets to swallow bigger buckets which gives more options for the possible swallowers of bigger buckets, leading to

even more reduction.

5.1.2 Border Based Approach

Before describing the border based algorithm some preliminaries are described.

Cost Domination Principle

Given a pair of distinct query points $q_1(x_1, y_1)$ and $q_2(x_2, y_2)$ in the two-dimensional selectivity space, we say that point q_2 dominates q_1, if and only if $x_2 >= x_1$ and $y_2 >= y_1$ and result cardinality $R_2 >= R_1$. Then if points q_1 and q_2 are associated with distinct plans P_1 and P_2 respectively, in the original space C_1, the cost of executing query q_1 with plan P_2 is upper-bounded by cost C_2 the cost of executing q_2 with P_2, if and only if q_2 dominates q_1.

Intuitively, with cost domination principle we expect optimizers cost functions to be monotonically non-decreasing with the increase in selectivities of the base relation. Equivalently a plan processing a superset of the input and producing a superset of the output, as compared to some other plan, is estimated as more costly to execute.

Assuming that the cost domination principle holds, if a point p within a region X can replace some point q on the grid then there is a point on the border of the region X with cost less than or equal to the cost of p which can replace the point q. Using this fact one can improve the algorithm by considering only the points on the border as the possible swallowers.

Finding Borders in Plan Diagram

Given an image, finding the borders in it is a well known problem in image processing. An existing technique is used to solve this problem. Plan diagrams have an advantage as they have sharp edges between two plans areas unlike the real images where due to shades of the same color it is difficult to find the boundary. Due to this advantage a very simple technique called Robert's Cross edge detection[11] works very fine here. The basic idea behind this algorithm is performing a convolution over the image using two predefined 2X2 convolution kernels. This will compute a 2-D spatial gradient measurement on an image. It thus highlights regions of high spatial frequency which often correspond to edges.

5.1.3 Border Based Algorithm

Another improvement to the older algorithm is that now small plans i.e., plans covering lesser number of points are allowed to swallow bigger plans i.e., plans covering larger number of points.

The algorithm is as follows:

1. Create buckets B_i for each different plan i in the selectivity space and put all points having the same plan in the corresponding bucket.
2. Create Border buckets BB_i for each different plan i in the selectivity space.Using the Edge Detector algorithm, identify the border points of each contiguous plan region and only insert those points into the corresponding bucket.
3. Sort the buckets B_i in the ascending order of the size of the buckets. Let this sorted list be $B_1, B_2, ..., B_n$.
4. for *i=1* to n begin
   (a) Swallow(B_i=true)

(b) for each point in B_i

i. for *j=1* to n and (j not equal to i) begin

A. find, if available, a point q in BB_j such that q is in first quadrant w.r.t p, cost(q) is within [100%, (100+ x)%] of cost(p), and cost(q) is the minimum across all such qualifying points in BB_j.

ii. if one or more q points are identified from the above step, choose the q point with the lowest cost(q), and mark that point p can be assigned to qs bucket

otherwise Swallow(B_i) = false; break;

iii. if Swallow(B_i)= true, move all the points in B_i to their assigned replacement buckets, then delete B_i and BB_i . end

end

5. Output the points with new plans based on the buckets they are shifted to.

The border based algorithm uses the points on the perimeter of the plan regions as the possible swallowers reducing the number of comparisons leading to reduction in running time of the algorithm.

5.1.4 Border Based Algorithm Results

Testbed Environment

For testing, a Pentium-IV 2.4 GHz PC with 1 GB of RAM and 120 GB of Hard disk running Window XP is used. The database used was generated using TPC-H[17] synthetic generator producing the database with over 1 GB of data. Also all the query templates were based on TPC-H benchmark which features a set of 22 queries. Each of the query had different plan cardinalities ranging from single plan to over 200 plans. All the diagrams are produced using OPT-C.

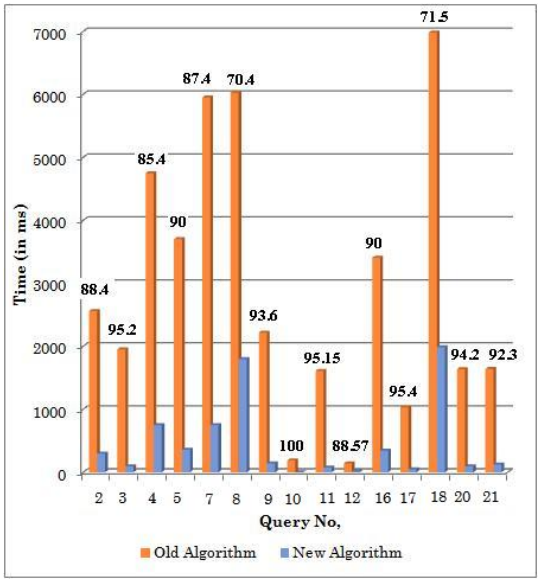

Figure 5.2: Reduction Times for Old And New Algorithms

Query No	**Dim**	**Old** Algo	**New** Algo	% gain
8	300x300	9 min	50 sec	90.7 %
9	300x300	11 min	50 sec	92 %
8	1000x1000	15 hrs	20.5 min	97.7 %
11	1000x1000	52 min	17 sec	99.4 %

Figure 5.3: Reduction times for Old and New Algo for resolution 300 and 1000

Results

The experiments were performed with the queries in the TPC-H benchmark. Figure 5.2, shows the reduction times for both of the algorithms for diagrams with resolution 100x100. The results show the improvements in running time of up to 95% with average improvement of 90% in running time.

The results for 300X300 and 1000X1000 plan diagrams are shown in Figure 5.3. Here also the improvement in running time is more than 90%. The reason for this is that in the area based approach, as the resolution increases the number of possible swallowers are increased quadratically. This is because here the points in the whole area are considered as possible swallowers. But in case of the *Border* based approach the possible number of swallowers increases linearly with the resolution as only the points on perimeter are taken into account. This linear increase is what makes the Border algorithm much much faster. It can be concluded from the results that new algorithm scales well even for bigger inputs unlike the older one, whose performance degrades with the increase in resolution.

Chapter 6

Experimentation

6.1 Analysis of Cardinality Estimators in Optimizers

In this section, a study is performed to measure the quality of the cardinality estimators present in the current optimizers. Current optimizers use these cardinality estimators for choosing the appropriate plan for the given query. The correctness of these estimates severely affect the quality of the plan chosen which has direct impact on the quality of the optimizer itself.

In our experiments, both compilation diagrams and execution diagrams were produced, for a subset of the TPC-H queries. These queries are chosen such that the output cardinality of the queries is high. Most of the queries selected had output cardinality greater than 1000.

Compilation diagrams give the estimated cardinalities while Execution diagrams provide the actual cardinalities. The graph in Figure 6.1 shows the %error in the cardinality estimation for the selected queries. The graph is drawn with log scale with log factor as 2. Positive values indicates the overestimation whereas negative values shows the

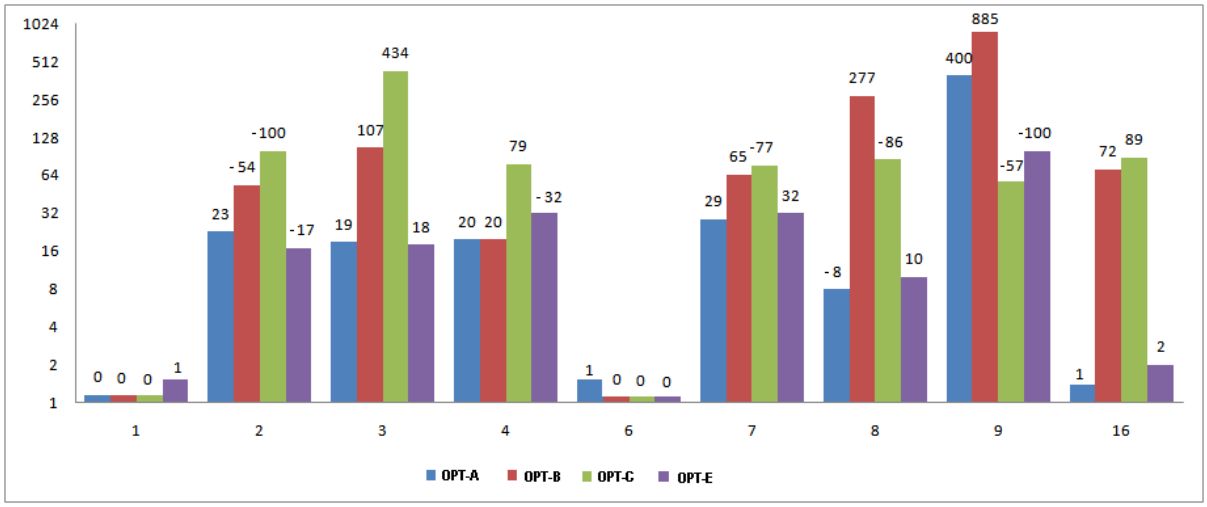

Figure 6.1: Comparison of cardinality estimates

underestimation in the output cardinality values.

The graph shows that estimates of output cardinality can deviate from actual cardinalities by large amount. In case of OPT-B the maximum error is of 885% while in case of OPT-A and OPT-C the maximum error is around 400%. But on average OPT-E performs best with average error of 24% only, while OPT-A makes the average error of 56%. OPT-B and OPT-C makes the average error of 164% and 102% respectively. These results show that sometimes the estimates done by the database engines can be off by large amounts which may lead to inefficient plans.

6.2 Performance of optimizers over uniform and exponential distribution of query points

For producing plan diagrams Picasso generates query points uniformly distributed over the whole selectivity space. Then Picasso executes each of the query to get the plans for each of the points and color them based on their plans. It uses the same color for the

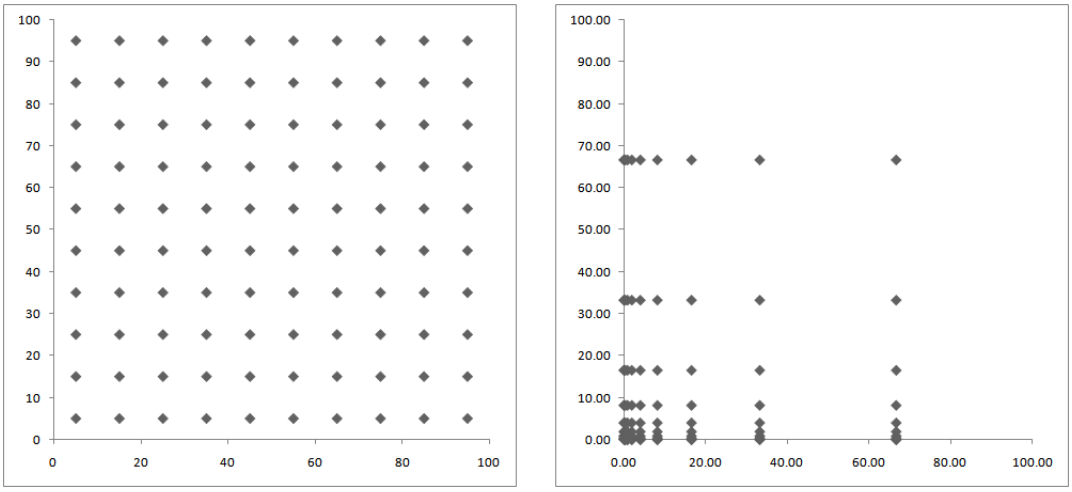

Figure 6.2: Distribution of Query Points. a)Uniform Distribution b)Exponential Distribution

points having the same plan and different colors for the points having different plans. For illustration figure 6.2a shows the uniform distribution of the query points.

A feature noticed in most of the plan diagrams is that the plan density is often high around the origin and along the axes. Hence, these areas are considered interesting.

Picasso1.0 provides two different distributions for the sample points. One can choose between uniform distribution or the exponential distribution of the sample query points. In exponential distribution the query points used are selected in such a way that they follow a 80-20 rule along each axes. In other words along each axes 80% of the points are in initial 20% of the corresponding axes. This will keep more number of sample query points near the origin where the plan density is high. This scheme is shown in figure 6.2b. With this new scheme of distribution of query points one can focus the query workload in these interesting areas.

Experiments have been performed to see the effect of distribution of query points on the plan cardinality. Here, diagrams for all the TPC-H queries for all the 5 engines

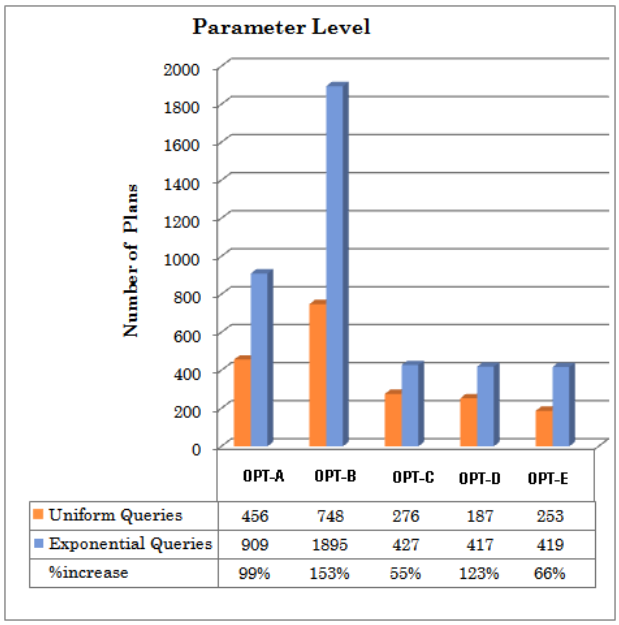

Figure 6.3: Comparison of plan cardinality between uniform and exponential distribution of query points

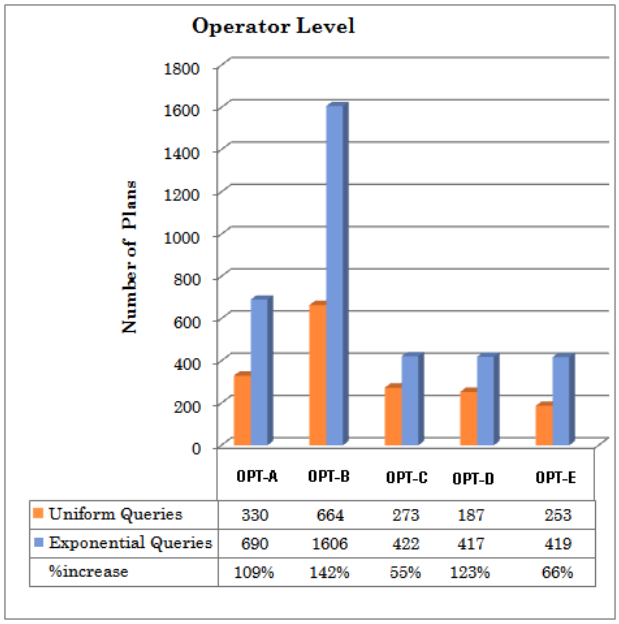

Figure 6.4: Comparison of plan cardinality between uniform and exponential distribution of query points

supported by Picasso are produced, for both TPC-H Uniform and TPC-H Exponential benchmarks and for both uniform and exponential distributions of query points. The numbers here are cumulative sum over all the TPC-H queries for both of the benchmarks.

Figure 6.3 shows the results when experiments are done at parameter level difference between the plans. As seen from the figure, for OPT-A, OPT-B, and OPT-D the plan cardinality for exponential distribution is more than the double of the cardinality for the uniform distribution of query points. For others i.e., OPT-C and OPT-E the increase in the plan cardinality is 66% and 55% respectively which is quite high.

The above results show that the plan cardinalities have been increased by substantial amount. But it may be possible that all the new plans which have appeared for the exponential diagrams are, structurally same as the old plans but have different parameters.

To confirm that the above results are correct even for the operator level differences in the plan same experiment was performed. Figure 6.4, shows the results obtained. Here also the results are same i.e., OPT-A, OPT-B and OPT-D still show more than 100% increase in plan cardinalities.

6.3 TPC-DS Benchmark

The TPC Benchmark TPC-DS is a decision support benchmark that provides a representative evaluation of the SUT's performance as a general purpose decision support system. TPC-DS benchmark has been mapped to a typical business environment.

TPC-DS models the decision support functions of a retail product supplier. The supporting schema contains vital business information, such as customer, order, and product data. The benchmark models the two most important components of any mature decision support system:

- User queries, which convert operational facts into business intelligence.
- Data maintenance, which synchronizes the process of management analysis with the operational external data source on which it relies.

All of the results in [7] were based on TPC-H bechmark. Same experiments have been performed on the TPC-DS to see that the observation made in [7] still holds. The dataset for TPC-DS benchmark has 24 different tables in its schema and the size of dataset is 100GB. The size is 100 times bigger than the size of TPC-H benchmark which was 1GB only. All the experiments were done using OPT-B. The resolution of all the diagrams is 100x100.The queries used are the subset of 99 standard TPC-DS queries.

Figure 6.7 shows the plan cardinalities of the selected TPC-DS queries. The EQP means the queries are ran with exponential distribution of query points. Here, we can observe that the plan cardinalities are very high. Most of the diagrams are found to be very complex in the TPC-DS results. Two of example diagram from TPCDS Benchmark are shown in Figure 6.5 and Figure 6.6.

To see that the other claim made in [7] i.e., in plan reduction process most of the plans disappear from plan diagram still holds, we also ran the reduction algorithm for each of the queries. The threshold used was 10% i.e., the cost of any point can be increased by not more than 10%. The results are shown in Figure 6.7. As seen from the table that all queries show large amounts of drop in plan cardinality. After reduction the absolute number of plans comes down to be less than 20 for all the queries.

All of the above observations made using TPC-DS benchmark support the claims made in [7] which used much smaller TPC-H dataset.

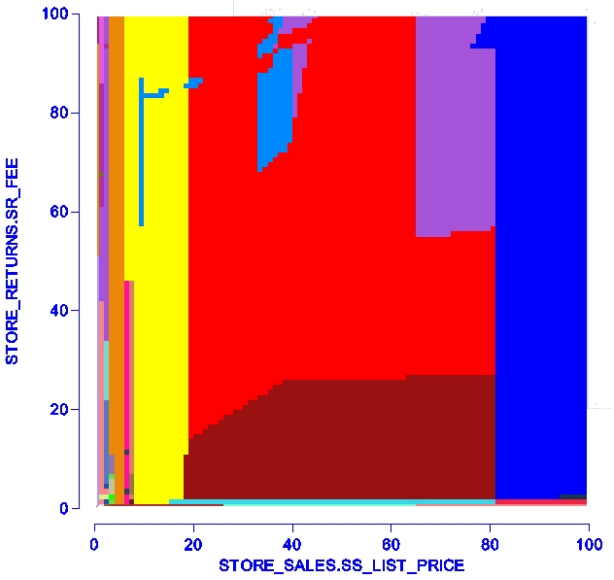

Figure 6.5: Plan Diagram for TPCDS Q19

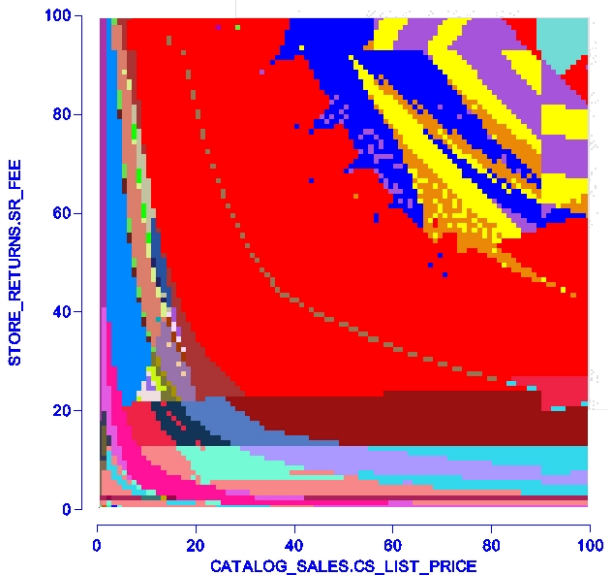

Figure 6.6: Plan Diagram for TPCDS Q25

	# of Plans	
Query	Original	Reduced (λ=10%)
2	3	1
12	13	5
12-EQP	26	9
17	39	6
18	47	11
18-EQP	147	20
19	36	10
19-EQP	76	16
25a	33	12
25a-EQP	136	19
25b	51	13
25c	96	12
25c-EQP	43	5

Figure 6.7: Results for TPC-DS Benchmark

Chapter 7

Conclusion

All the new features and enhancements made to the beta version of Picasso have made it more useful. For example, all the studies performed in this project have been done through command line interface, which has saved a lot of time. Similarly shifting of the generation of operator level diagrams to client side have saved lot of time. The other enhancements like improvement in the estimator, changes made to the plan reduction algorithm have made the tool faster and efficient.

Various studies performed here shows the unnecessary complexities prevailing in the current commercial query optimizers. The comparison of plan cardinalities at parameter level vs operator level supports the claim about complex behavior of optimizers. The analysis of output cardinality estimation errors shows that current optimizer do a very bad job of estimation and these estimates can be off by large amounts.

The study with exponential distribution of query points shows that as the resolution of the diagram is increased the plan cardinality also increases by large numbers. It is also shown that claims made in [7] hold true for TPC-DS also.

References

[1] A. Hulgeri and S. Sudarshan, *"Parametric Query Optimization for Linear and Piecewise Linear Cost Functions"*, Proc. of 28th Intl. Conf on Very Large Databases, August 2002.

[2] A. Hulgeri and S. Sudarshan, *"AniPQO: Almost Non-intrusive Parametric Query Optimization for Nonlinear Cost Functions"*, Proc. of 29th Intl. Conf on Very Large Databases, September 2003.

[3] D. Harish, P. Darera and J.Haritsa, *"Reduction of Query Optimizer Plan Diagrams"*, Technical Report TR-2007-01, DSL/SERC, Indian Institute of Science, March 2007.

[4] D. Mare and E. Hildreth, *"Theory of Edge Detection"*Proc. Royal. Soc., London. Vol. 207, pp. 187-217, 1980.

[5] M. Sharifi, M. Fathy and M.T.Mahmoudi, *"A Classified and Comparative Study of Edge Detection Algorithms"*, Proc. of Intl. Conf on Information Technology: Coding and Computing, 2002

[6] M. Poess, B. Smith, L. Kollar and P. Larson, *"TPC-DS, Taking Decision Support Benchmarking to the Next Level"*, Proc. of ACM SIGMOD Intl. Conf on Management of Data, June 2002

[7] N.Reddy and J.Haritsa, *"Analyzing Plan Diagrams of Databse Query Optimizers"*, Proc of 31st Intl. Conf on Very Large Databases, September 2005.

[8] Y. Ioannidis, R. Ng, K.Shim and T. Sellis, *"Parametric Query Optimization"*, Proc of 18th Intl. Conf on Very Large Databases, August 1992.

[9] *http://dsl.serc.iisc.ernet.in/projects/PICASSO*

[10] *http://en.wikibooks.org/wiki/SQL_dialects_reference*

[11] *http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm*

[12] *http://infocenter.sybase.com*

[13] *http://www.microsoft.com/sql/techinfo/productdoc/2005/books.asp*

[14] *http://www.oracle.com/technology/products/oracle9i/index.html*

[15] *http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp*

[16] *http://postgresql.org*

[17] *http://www.tpc.org/tpch*

[18] *http://www.swissql.com/products/sqlone-apijava/sqlone-apijava.html*