## Chapter 5.5 Introduction of Plan Selection Based on Cost

***Introduction:***

No matter choose the Logical Query Plan or construct the Physical Plan from the Logical Query Plan, the Query Optimizer needs to estimate the cost of the specific Expression.

Just like before, we use Executable Disk I/O to make estimation about the cost of Expression. However, Disk I/O influenced by factors below:

* Select the Realization Operator for Specific Logic Operator, this is what we choose when we choose the Logic Query Plan.
* The Size of Intermediate Relation.
* The Physical Operator that is used to Realize Logic Operator.
* Sort the Similar Operators.
* The Parameters that Passed from Physical Operator to the next Physical Operator.

Here, in order to make Plan effect, we need to solve many questions.

### Chapter 5.5.1 Get Estimation of Size Parameter

***Instruction:***

Before we can solve the issue, we need to know the size of tuples in Relation R and the V(R, a), which is the different number of the attribute a in Relation R. Nowadays, DBMS allows the normal users or managers to collect the statistical information, such as T(R) and V(R, a). These statistical value can be used in the latter Query Optimizer, the statistical number will be updated in the next round that get the collect command.

Through scan the whole Relation R, obviously we can get the count of tuples T(R), and find different number of attribute A, V(R, a). The block number B(R) in Relation R can be used by calculating the real block number or through T(R) divided by tuple numbers that one block can contain.

***Definition:***

* DBMS can calculate one ***Histogram*** by giving values on several attributes.
* If V(R, A) is not too big, then the Histogram will be consisted by each tuple number with attribute A.
* If the attributes exist a lot of different values, then only the most normal value will be recorded, while other value will be included as one group and be calculated together.

The normally used Histogram is called:

1. ***Equal Width:***

* Here, we choose Selected Width w and Normal Value v0. The tuple number v is provided, the range of v will be v0 <= v < v0 + w, v0 + w <= v < v0 + 2w, and so on.
* The value v0 will be the Least Possible Number or Known Lower Bound so far. If it is the latter one, then when meets one new smaller value, then we will decrease w on v0 and add into ***Histogram***.

1. ***Equal Height:***

* They have the common ‘percentage point’. We choose some small number p, and list all numbers that bigger than the least number for about p number, 2\*p, 3\*p and so on, until the final biggest number.

1. ***The Most Frequent Value:***

* We list the most common value and the times they appear. However, this message can be provided with all other values or other than the equal width and equal height, then we list all other values that appears normally.

1. ***Advantage:***

* By using Histogram, we can get the estimation of Join. Especially when Join Attributes appear in the Histogram, then we can get the exact number of tuples that have the value.

***Explanation:***

For those do not show Exact Value on Histogram, then we just need to use the Estimation Method to evaluate the result in Chapter 5.4.

If we use Equal Width Histogram to evaluate Exact Value after Join, then we just need to estimate the Join size on the same Width range and sum all these estimation. This result is the good estimation however and only when Join on the same Width Histogram.

### Chapter 5.5.2 Calculation of Statistical Value

*In the Query Optimization, normally, the Statistical Value can be used to calculate cyclically.* Reasons as below:

1. These statistical value will not change in the short span.
2. Even the unclear statistical value can be helpful, as long as they are used in all plan together.
3. Keep the Statistical Value updated at any time, which makes the design as the “hotpot” in the Database.

*The Recalculation of Statistical Value* would be updated after a while or after some updates. But when Database Manager finds that the bad Query Plan would be selected by Query Optimization, then they would need to recalculate the Statistical Value in order to fix this issue.

Considering that *Recalculation of Statistical Value* would cost a lot, and a normal method would need to get part of the data to recalculate the Statistical Value.

### Chapter 5.5.3 Heuristic Estimation of Decreasing Logical Query Plan

***Definition:***

The important usage of Query and Sub-Query is the Heuristic Changes when applying Query. Some Heuristic, such as Push Selection in Tree, then it will surely improve the cost of Logical Query Plan, and no need to consider the size of Relation.

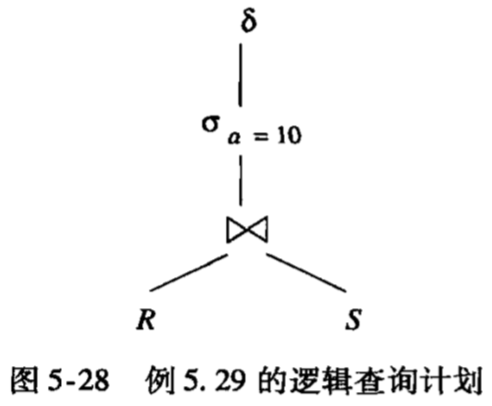
***Example:***

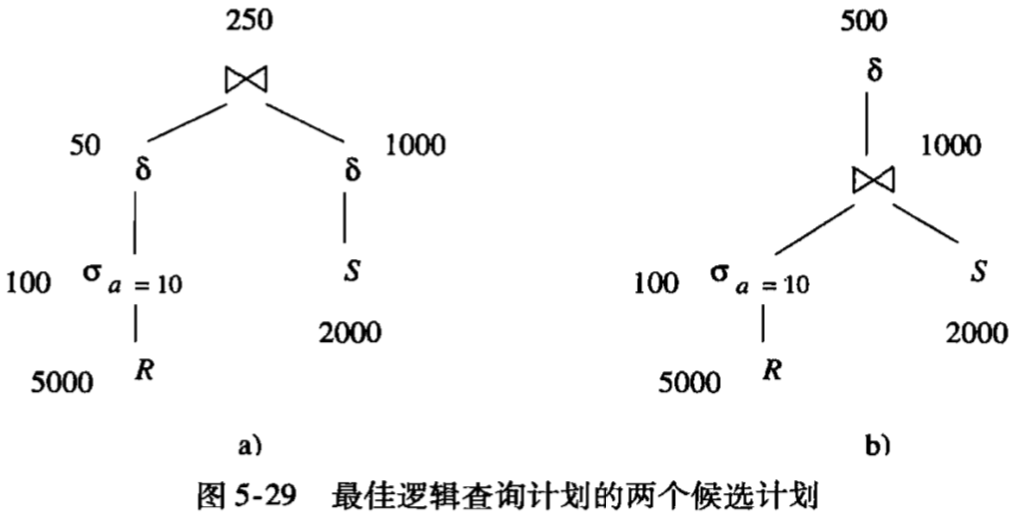
Consider the Logical Query Plan, and make Statistical Value of Relation R and S as below:

|  |  |
| --- | --- |
| ***R(a, b)*** | ***S(b, c)*** |
| T(R) = 5000 | T(S) = 2000 |
| V(R, a) = 50 |  |
| V(R, b) = 100 | V(S, b) = 200 |
|  | V(S, c) = 100 |

Here, we generate the final Logical Query: the difference is that, where the De-Duplication takes place, it just takes place before Join Operator or after Join Operator.

In Plan a, De-Duplication Operator has been pushed into the two branches, if there have no duplications in R and S, then De-Duplication Operator in any branches can be eliminated.





***Analysis - a:***

Here, we already know how to estimate the size of result, we use T(R) / V(R, a) = 5000 / 50 = 1000. Also, we can estimate the size of Join, here we use 5000 \* 2000 / max(100, 200) = 50000.

First of all, let’s estimate the size of De - Duplication(Selection (a = 10) (R)). Since there has only one a value while has 100 value b, and there has 100 tuples in Relation. Because we estimate that the result size of De - Duplication Operator is the half of tuples in Selection (a = 10) (R), so after De-Duplication Operator, then the size turns to 50.

In the same way, after De-Duplication Operator, then the size of Relation S turns to 1000. The final size of Join Operator would be 50 \* 1000 / max(100, 200) = 50 \* 1000 / 200 = 250.

***Analysis - b:***

In the Logical Query Plan b, there has only one a value and the number of b estimates as Max(V(R, b), V(S, b)) = 100 and the value c will be estimated as V(S, c) = 100. The number of tuples after Join, it will be 100 \* 2000 / 200 = 1000. Therefore, the number of tuples after De-Duplication Operator, it will equals to 1000 / 2 = 500.

***Comparison a and b Plan:***

***We add size of all nodes except root node and leaf node, since the size of these nodes would not depend on the selection of plan.***

* For plan a, the cost would equal to the estimation size of inner nodes which would be 100 + 50 + 1000 = 1150.
* For plan b, the cost would equal to the estimation size of inner nodes which would be 1000 + 100 = 1100.

Therefore, we delay De-Duplication Operator to the last Plan would be the better Plan.

### Chapter 5.5.4 The Method to Enumerate Physical Project

Here, let’s consider how to estimate the cost when we convert the Logical Query Plan to the Physical Query Plan. The last method is called ***Exhaustive***, it is used to combine each selection for each question.***(Select the Sequence of Join or Physical Realization of Operator.)***Each Physical Plan can be assigned one estimation cost and choose one plan with the least cost.

There exist a lot of Physical Plans and explanation needs to put on two main methods for Searching the possible Physical Plan.

* ***Top to Bottom:***

Consider starting from the root of Logic Query Plan. For each realization of node, consider calculate each possibility and calculate the cost for each combination, and get the best one.

* ***Bottom to Top:***

For each Sub-Expression of the Logic Query Tree, we need to calculate the cost for each possible method. Consider each Selection for Sub-Expression E, and combine it with the root Operator Realization, calculate the possibility and cost of Sub-Expression E.

Actually, there has no big difference when explain two methods. Here we focus on ***Bottom to Up method***.

1. ***Heuristic Choice***

Apply the method to choose Logic Plan Method on choose Physical Plan: Based on Heuristic Rule to choose a series of Selection.

*Below are common Rules based on Heuristic Choice:*

* If Logical Plan needs to choose Selection A = c (R), and the saved Relation R has Index on Attribute A, then it needs to execute one Index Scan and get tuples that satisfies the condition A = c.
* Generally, if Selection relates with one condition A = c and other conditions, here we can scan the index first and choose from tuples. We use Physical Operator Filter to represent.
* If there has an index on the Join Attribute, then using the Index Join, and this Relation should be in the inner Cycle.
* If the Join Attribute is sorted, then using the Sort Join is better than Hash Join, although it may not be better than Index Join.
* If there have three more Relations that need to Join or Union, then it should be better to combine using the smaller Relations.

1. ***Branch Defined Plan Enumeration***

* This method is normally used in Reality, it finds a good Physical Plan for Logic Query Plan through using Heuristic Rule. Make the cost of this Project C, then we start to consider other Plan for the sub-Query, then we can eliminate all plan for those Cost whose are much bigger than C, since the Plan of sub - Query may not join in our known whole Query.
* Similarly, if we construct one plan for the while Query that satisfies the Cost is less than C, then we will use this Plan to substitute the ole C Cost Query Plan.
* This method can be used to check when we can stop Query and find the Optimized Plan so far.

1. ***Climb Mountain Method***

* This method starts from Physical Plan which selected by Heuristic Rule, and actually we search from one canyon with ‘Physical Plan’ and ‘Cost’. We can make tiny modifications on Plan and find lower cost ‘neighbor’ Plan.
* When we find one plan, then little change will not get one more lower cost Plan, then we choose the current one as the Physical Query Plan.

1. ***Dynamic Programming***

In the Dynamic Programming, for each Sub-Expression, we just keep Plan with the least cost. When we process the tree from Bottom to Top, assume that each expression has used the best Plan and however we consider every possible Realizations.

1. ***Selinger Styled Optimization***

This method improves *Dynamic Programming*, it not only records the plan with the smallest cost for each Sub-Expression, it also records those plans with high cost while the sequence of generated result does help for higher level of the Expression Tree.

***These kind of Sorted Sequence is:***

1. The Sorted Attribute in Root Node.
2. The Grouping Attribute in Grouping Operator.
3. The Join Attribute in Join Operator.

Then we see the cost of a Plan as Sum of all intermediate Relation, and it seems that there has not any advantage on the Sort of one parameter.

But If we are using the more precise measurement, such as Disk I/O, therefore we can sort for some parameter, so we can save the first trip work for already sorted parameters.