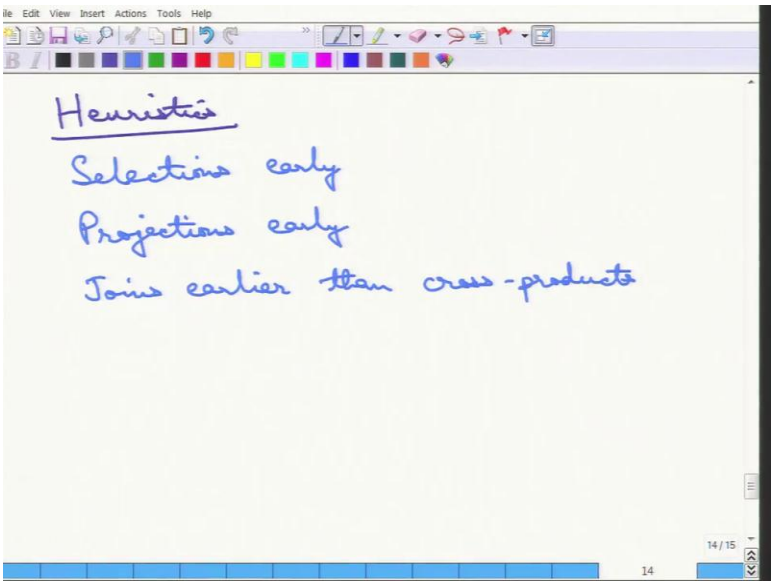
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 27
Query Optimizarion: Heuristics and Sizes

(Refer Slide Time: 00:10)



Heuristics are implied not just in sort order and join tree, it has been used in many such cases. So, some of the Heuristics that the database engine performs to do better query optimization is the following is that, first thing, the one of the thing it does is that, selections are performed early. Why is that? Because, once the selection is done, the size of the relation is reduced considerably.

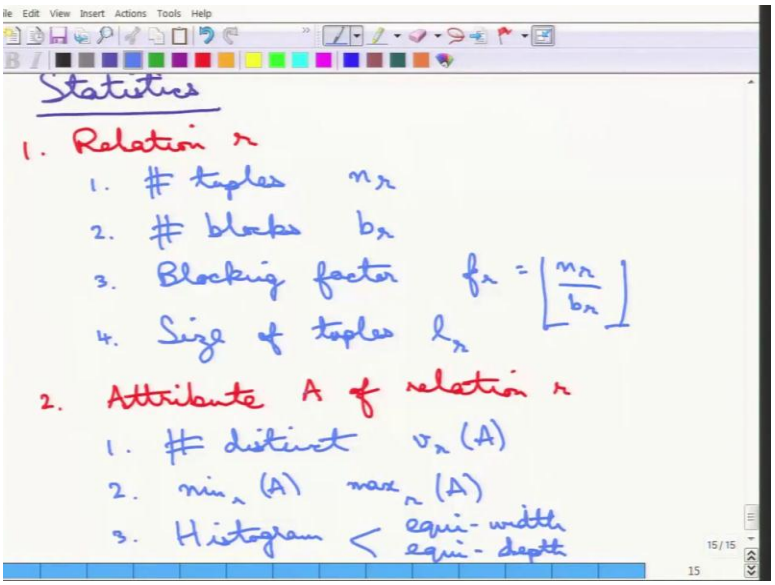
And so, whatever operation that is done on the relation, it can be done in a faster manner; that is one thing. Projections are also, if it they can be done early, they are done early, again projections can be done early, because that reduces the number of attribute; that reduces the size of the relation in general and again, that can be beneficial later, again that is mostly beneficial later.

Then, joins are generally done earlier than cross products, if there is a cross product versus a join, then join is generally done earlier. The reason is easy to understand, joins are essentially cross products and then, a selection. So, joins can early produce smaller relations than the cross products. This is one kind of thing, these are all kind of heuristics

that do not depend on the semantics of the query, this is just using the equivalence rules under doing it.

The optimizer can also use certain semantic optimizations. For example, suppose the query is find all employees, who on more than their manager. Now, a database engine can simply go ahead and try to solve this query by joining the employee salary table with the manager salary table, etcetera, etcetera. But, if a semantic knowledge is being build in, that the employee can never get salary more than her manager, then this query uses that semantic rule and simply returns a null set, simply returns an empty set. So, that is an example of a semantic optimization. So, it uses essentially the domain knowledge and the constraints that are built into the database directly and not this equivalence rules.

(Refer Slide Time: 02:27)



So, the query optimizer to do all of these things to choose which algorithm is better, etcetera, stores a lot of statistics about the relations, about the attributes, etcetera, etcetera. So, what are the statistics that is stored? So, for each relation r it stores the following statistics, so this is about relation r. First it stores the number of tuples, so number of tuples in r; that is being stored, then the number of blocks that r is required.

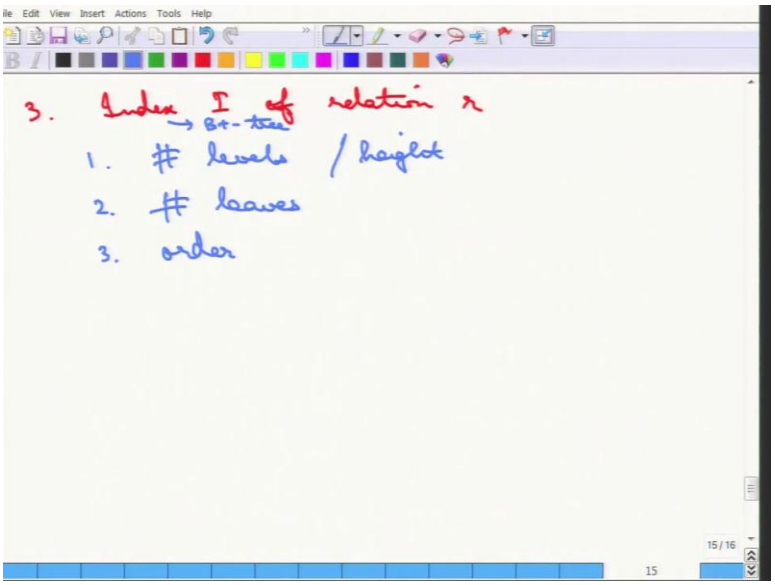
So, this we have seen example, this is n r and this is b r; that it store, it also store something called a blocking factor. So, the blocking factor is essentially the number of tuples that fit in a block. So, this is the blocking factor, this is the derived statistics, one can say, this is n r by b r, but it stores it explicitly, because that helps in a certain things.

And the size of a tuple, again the size of a tuple is essentially the block size divided by the blocking factor. So, that can be a, I can derive, but these are the statistics that the query optimizer maintains, this is for each relation. Now, for each attribute A of a relation, this is an attribute column, attribute A of relation r. Again, certain statistics are maintained, the first statistic is the number of distinct values.

So, this is kind of an interesting statistic, this is essentially the size of your… If the projection on A is done on relation r, what is the size? This is the v r, this is denoted by v r of A. This is useful for many cases, while duplicate elimination and all those things, this can be useful. Then, the minimum across A and the minimum value and the maximum value and in some cases the standard deviation and mean, etcetera can also be stored.

Then, genuinely the histogram of values is being maintained. So, how are the values for that attribute A are distributed? So, the histogram can be either equi-width or an equi-depth. So, in an equi-width histogram, the range of value that is put in one histogram is same and in an equi-depth, the number of values that it put in one histogram been from the next is same.

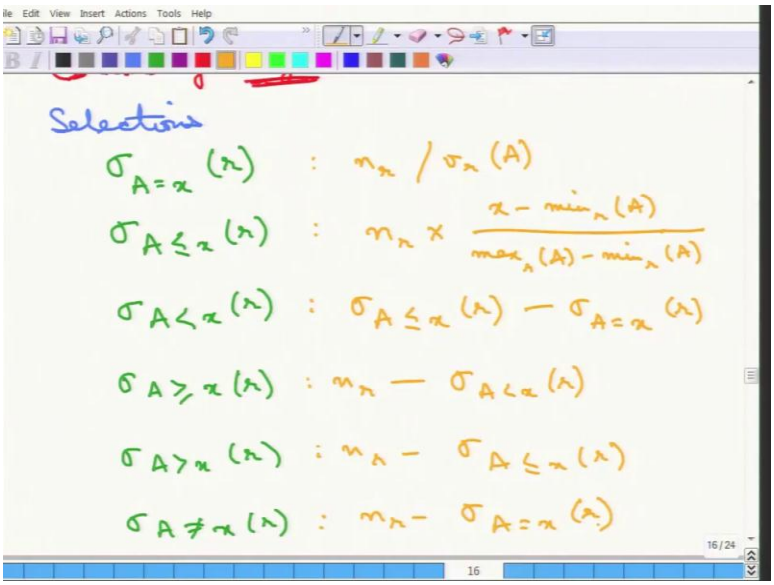
(Refer Slide Time: 05:07)



So, histogram are can be maintained, then for each index, for an index, say index I of a relation r, it also again maintains certain statistics and the statistics are of course, the number of level. So, this is when I say index I mean the B plus tree of course. So, this is

the number of levels of the tree; that is the height of the tree and then, number of leaves; that is in that, there and some others, it says the order of the tree and the average branching factor and all those things. Again, this helps to decide, whether to use the index for hash, join and etcetera, etcetera.

(Refer Slide Time: 05:48)



So, after this, let us see how does the query optimizer estimates the size. So, what we are, the next topic that we are going to study is about estimating size of different operations, estimating size. So, for selections, let us first handle selections, so suppose the selection condition is sigma, the attribute value A is equal to, so particular value x on the relation r.

Now, what is this? There are couple of ways of solving it. Suppose A is the super key of this relation r, if there is a case, then one knows that the size, the answer said size of this is going to be either 1 or 0. Because, if A is a key, if that value occurs, it occurs, then it is 1, if it does not occur, it is 0. Otherwise, if it is not the key, then what is the answer, then the answer can be got from the previous things, this is roughly n r by v r by A.

Why is that? Because, n r is the total number of tuples and v r is the number of distinct values and we are use the number of distinct values. So, roughly on an average for every value, this is n r by v r, if it exist, otherwise it is 0. So, this is for the equality selection. What is on an non-equality selection? Suppose it is a less than query, now once more, if x is less than the minimum value that is maintained, then this the answer to this is 0.

And if again x is greater than the maximum value that is stored, then the answer is a entire relation, the size of the relation is the answer. Otherwise, this is estimated in a following manner, this is n r times x minus the minimum value that is stored and this is max minus v minimum value; that is stored. Note that, these are all rough major; this is not going to be exactly the equal, so this formula assumes that the values are distributed uniformly from the minimum value to the maximum value.

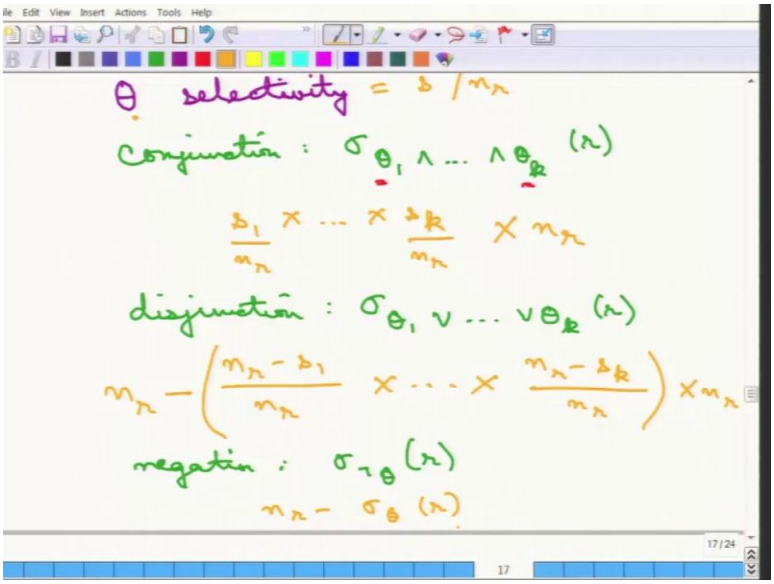
So, wherever x occurs, the x minus minimum gives you roughly the percentage of tuples that are going to fall below x and then, if you multiply that with n r that gives the actual number of tuples; that is an estimate. Now, this estimate does not use the histogram actually, this is just uses the min and max x values and when the histograms are available, the equi-width or the equi-height histograms are available.

Then, the estimates are going to be better, because what the query optimizer needs to do is to simply count the number of histogram means, which below it and the histogram mean that it over laps, it needs to estimated somehow using this formula. But, otherwise it gets an actual count of the histogram means that are below that are less than x. So, the next query is this A less than x, the previous one was exactly equal to x and this is a way of solving it is.

If one can solve A less than x equal to x r and then, if when gets read of A equal to x and that solves the A less than x. So, that is the easy, the next that we will study is A greater than equal to x of r and this is can be again solved very easily. Note that what is the A greater than equal to x r is the compliment of A less than x r. So, one can use that estimate to solve it.

So, this is simply n r minus sigma A less than r, the size for that and similarly, A greater than x r can be estimated in two ways. So, this is n r minus sigma A less than equal to x r and finally, there is only one thing left, which is sigma A not equal to x r and by this time, we have surely figure out how to solve this, this is essentially n r minus sigma A equal to x r. So, that completes the estimation of size of selections, simple selections on one attributes.

(Refer Slide Time: 10:05)



Next, how to do complex selections, so that is not on one attribute, but on multiple attributes; that is what I mean by complex selections. So, the selectivity, so there are multiple conditions. So, something is defined as for the condition theta, the selectivity is defined is that, how many tuples pass only that condition theta; that is called the selectivity of the condition theta.

So, it is essentially the probability the tuple in r satisfies theta, now if so the way to estimate is that, if s number of tulpes passed the theta condition, then s by n r is the selectivity. So, selectivity essentially s by the total number of tuples which is s by n r. Now, if the condition is conjunction of these things, then using this selectivity, so the conjunction the condition let me write on a little more explicitly, the selection is on theta 1, 2, some theta k, this is on the r.

If the conditions are independence, so that means, theta 1, theta 2, etcetera independent of each other, then the estimate is very simply, so far theta s 1 is the selectivity, for theta 2, s 2 is the selectivity so on and so forth, one can write down this selectivity conditions. So, the number of tuples that is selected for theta 1 conditions is your s 1 by n r and then, for s 2 and n r so on and so forth, this is s 2 by m r.

This is the total selectivity of the consumption condition that when it is multiplied by n r gives the total number of tuples at are going to the selectivity. So, that is the, if these are independent, we are properly going to assume, this is an independent. Again, if the

independent is violated, then this does not work out, but that is fine. The next one is disjunction, so the disjunction condition is essentially sigma theta 1 or theta 2 or up to theta k.

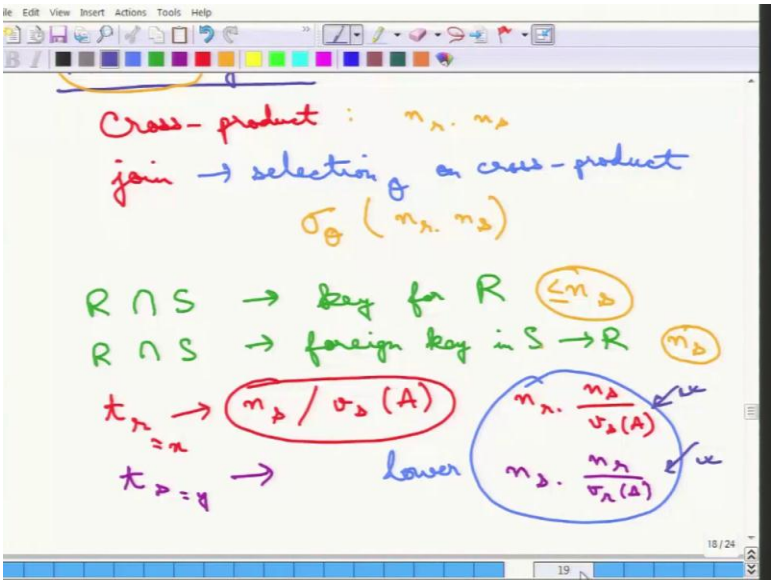
And how can one solve this is a following manner. So, what is disjunction. So, what exactly is the theta 1 and theta k is that is that, when is the tuple selected on the theta 1 and theta k is that, let us negate this condition, disjunction can be think of as the negation of conjunction. And then, negation of those are the theta 1 values, so our tuple is selected for this disjunction, if it is not selected under any of the negative conjunction.

So, other way of saying that at tuple is not selected for this disjunction, if it is selected for the negation of those conjunction. So, using this strategy, one can write down the formula in the following manner is that, so n r minus s i by n r, this is the selectivity of the negation of the condition of theta 1 and so one can right down that and when we are again assuming the all theta 1 to theta k are in dependent.

So, the negation of also in dependent this gives the on the selectivity of the negation of this conjunction that times in our gives the number of tuples, where the conjunction is not correct and one points needs to find the negation of that, because is the disjunction. This is n r minus this that gives the formula for the disjunction and if one goes ahead and tries to solve it for the next thing, which is negation the just a negation.

So, the negation of the condition is simply, this is not theta of r; that is very easy and that we have already handle. Although, one can say this is the complex selection, but this is essentially as the n r minus the estimate for sigma theta of r, this we have already see. So, that is the estimating the size for selection. So, we have seen simple selection and we have seen complex selections.

(Refer Slide Time: 14:10)



Next we will estimate the size of joins. So, let us first start of with natural join you estimate the size is for natural joins. Now, the simplest case is size for cross product and you will come to natural in status. So, cross product, what is the size of cross product, it is simply n r time n i. So, we are assuming it is r and s are two relation. So, just as double join also the join with only two relation, it is a single join operation that we done.

So, cross product is simply n r by n s. So, what is the join? So, if you do not worry join so for normal join is essentially just a selection of theta on cross product. So, essentially the estimate of the size of the these join is the estimate of theta on the cross product size. So, that is simply whatever is the estimate on this n r dot n s, so that is a way to do it, essentially just assuming the track.

Now, for we come back to the case of natural joins, now the following is to going, if r intersection s is not there, then the size of this natural join is essentially just the size of the cross product itself. Now, if r intersection of s is a key for r, then each tuple of s will join at most one tuples of r. So, let us worry about the schemer's of r and s and this is the schema of r schema of s.

Suppose, is schema of r and s is the key for r. So, which means that at most one tuples of r will join with at most one tuples of s. So, the size of this can be at most number of tuples in s, so that is n s. On the other hand, if r intersection s is a foreign key in s that references r, then what happens is that this is exactly equal to be n s, because in a foreign

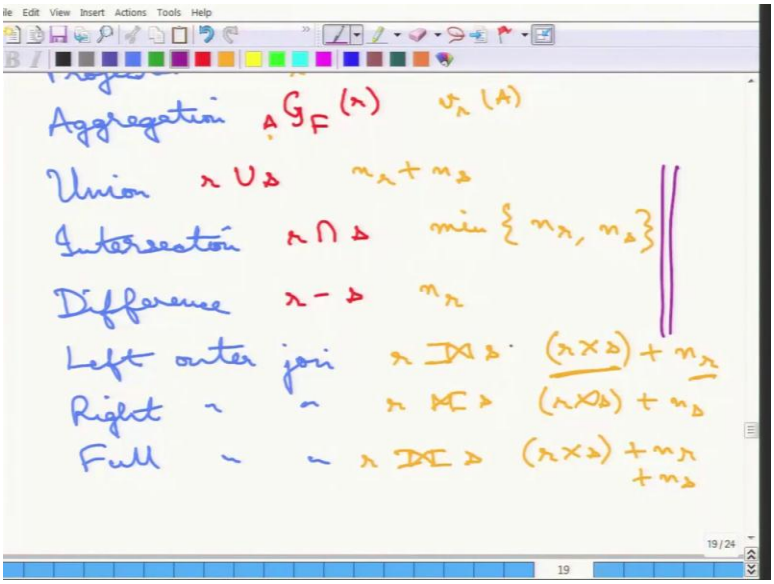
key n s; that means, all of these s is, so all the values in s must one have a call is thing is r.

So, all the tuples in must join and so all in a is a result, this is at most n s, this is less than equal to n s, because if it is not a foreign key it is not clear whether everything will sure everything s will have a finding tuples in r, r not. And otherwise general case is following, so every tuples in r suppose the tuples t r in r can join with suppose n s by v s by A.

So, why is that, because tuple in r as a particular value of x in s estimate for the selection of x in a this n is v s by s every tuples in r can join with n is by v s by A. So, the total number of join tuples that can found is n r times n s by v s by A, this is if one thing for point of view of r. However, one thing if only point of v of s, then the same thing can be done and the estimate that one gets is in is times n r by v r by A.

Now, the question is which one is better etcetera, etcetera. So, this is the two estimate generally that the find out that the lower size is a better estimate. So, out these two, the lower is the better estimate, this is the observed. And once more, this is the, if there is histograms, if one us the histograms then the estimate can be of course, improved. So, the histograms are used in both of these, both of these estimate can be improved. So, the actual estimate is also improved. So, this is the sizes of joins etcetera, now size of some other operations, we can of see, so suppose size of projection, this we have already seen the answer.

(Refer Slide Time: 18:17)



This is the number of distinct values that we have seen. So, this simply v r by A, then we can see the this estimate the size of aggregation is estimating size of aggregation, see suppose this is the aggregation function and on this… So, this is the function that is right, this is essentially again just the v r of A, why is that, because aggregation assumes particular attribute. So, which essentially means, it is a projection and that attributes. So, this aggregation function is projecting is this aggregation is being done by so each distinct value of A, which is this essentially saying that the number of v r by A; that is the number of distinct value is of a that has been maintained.

Union of r and s or union s, the estimate is simply n r plus n s, this is of course, in upper bound and it can be less than that. similarly, intersection then we found out, so intersection of r, intersection s, the estimate is the minimum of n r and n s, again this is conservative estimate. The next one is set difference, which is r minus s, the estimate for this is simply n r, this is again upper bound, because is assuming that none of the tuples s is common, because so all n r may be return.

Then, this all of these are essentially all of these are upper bounds fine. So, the next thing is comes is that left outer join. So, the estimate of left outer join is the following is that the, so the estimate of r left outer join with s is the size for r joined s and the it does not cover everything in r. So, this is the r to size of that is so all the tuples that are join in r

cross s plus all that tuples that we not, because the worst case is that all tuples in r have been left out.

Right outer join is similar and let me just write it down for the correctness. So, r joint with s is r join with s plus and s. The more interesting case is the full outer join of r and s is that size of r joined with s plus n r plus n s, because where the worst case is that r and s none of the can be join. So, again these are upper bound. So, that finishes the topic on quarry optimization, next we will move on to one very important topic about data bases, which are the data base transactions.