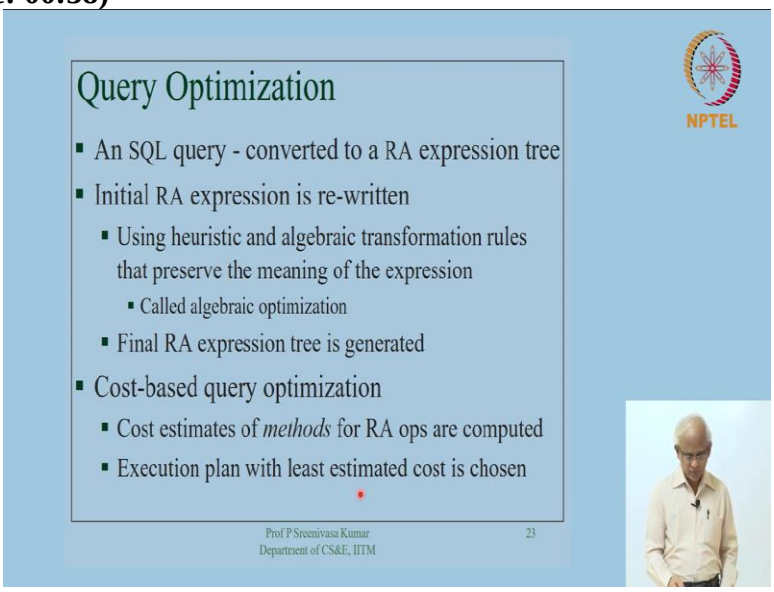
Database Systems
Prof. Dr. Sreenivasa Kumar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture-35
Query Optimization

(Refer Slide Time: 00:38)



So today's lecture, we are basically want to focus on query execution and optimization. So, we will discuss few; discuss issues and then few fundamental ideas in this in the screen. This is a vast field and, you know, requires a lot of study. So, in this lecture I will basically give you an exposure to the basic issues that are more than this query execution. So, we know that an SQL query when it is specified, it needs be converted into a relational algebraic expression tree.

And then we have been discussing various algorithms for executing these relational algebra operators. So, we need to choose those operators and then execute the query actually. So but before we actually you know, go ahead with the actual execution, what is normally done is to take this initial relational algebra expression and then rewrite it in a manner that it is meaning preserving rewrite it and that rewriting basically takes place using certain heuristics and as well as algebraic transformation goes.

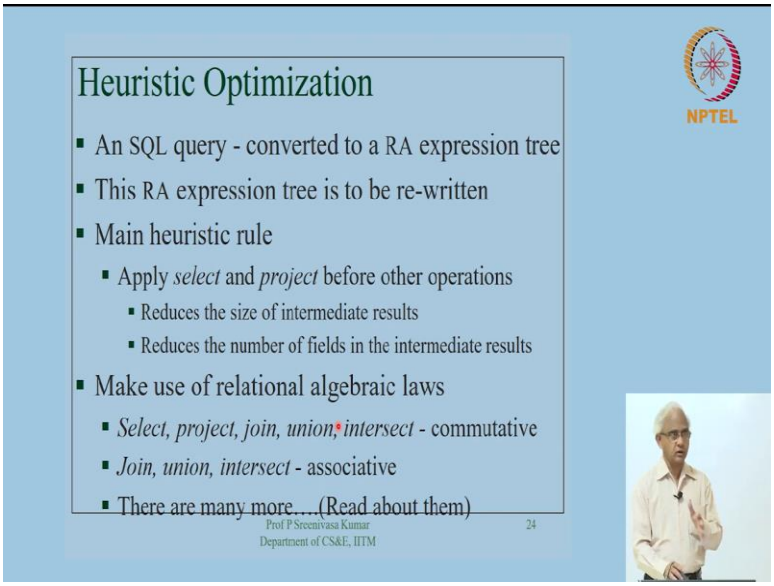
So, I will not go deep into this algebraic transformation goes but you can imagine we will mention some of them. And in addition to the algebraic transformation goes, we also use certain heuristics and then we write this relational algebra expression into an expression which is amenable for execution. So, this can be called as heuristic or algebraic optimization. Now, once we generate the tree, the final relational algebra expression tree, we will also do one more step which is called the cost based query optimization.

Now, for various operators that are there in the relational algebra tree, we have choices of algorithms, choices of what algorithm to use, if it is a join, you know what is appropriate algorithm to use in that particular relational algebra expression. So, and if there are multiple algorithms that are applicable for that particular operator then what is the estimates of the costs of those methods?

So, taking these estimates of the costs of the relational algebra operations and actually if you do one of these operations in one way, some other operation may have to be done in only in a certain way they are interdependent the operators are interdependent in this in the tree. So, taking all these things into consideration, we have to actually formulate that as an optimization problem and then try and get a minimal solution of choices of these methods for all of those operators in a combined manner.

So, and then you generate what is called an execution plan, where you fix as to what operates What method is being used for operator and then go ahead executing a query. So in I will illustrate this algebraic optimization with an elaborate example so we will see that.

(Refer Slide Time: 04:16)



So let us focus a little bit on this heuristic optimization. So, the main heuristic rule that we will use is apply the selected project operations before other operations are being applied. That means in from a relational algebra, tree point of view, if you find a select operator, try and push it to the leaves see what is the relational algebra operator a tree and the base or the leaf levels we will have the actual base relations that are available on the relational in the database.

And then you 2 of those have 2 of those relations will be combined using a join and then the result will go you know, with a projection to some other operator and that operator will get another operand from somewhere else. And then, you know, there is some maybe union is applied and things like that. So, it all goes like that. So at the leaves, you have relations basically. Now, one of the heuristics that we will apply is that push these selections on project operators, all the way towards the leaves.

So if you do that, then right at the initial stage because you are executing the, the expression in a bottom up manner, you are executing the relation expression and the bottom of manner. So right at the beginning, you would reduce the number of suppose that go into these operators. So, with that idea, so, we try and push this select and project operators all the way to the leaves are in other words, you can you need to apply them before you can apply other operators. So, if it is possible to do that.

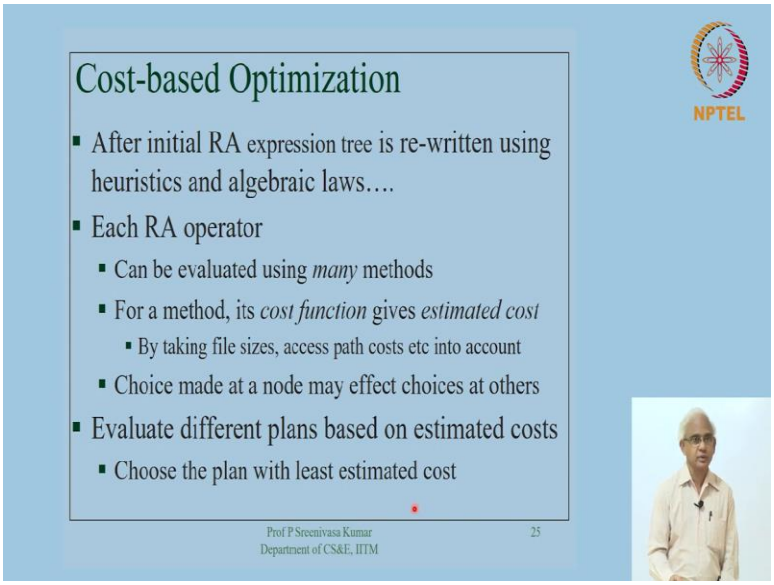
So, that we will see in through an example so, the, the effect of this heuristic, applying this heuristic is basically that we will be able to reduce the sizes of these intermediate results, there are a lot of intermediate results, the internal nodes of the relational algebra query we are essentially intermediate results. And because projection also is being pushed as close to the leaves as possible it also reduces the number of fields that are there in intermediate results.

So, this is one of the main heuristic rules that will apply and then make use of these relational algebra operators, all of them satisfy certain algebraic rules like for example, select project join union intersect all of them are commutative operations you can split them around they will give the same result when you so, 2 selects have to be applied then you can apply them in any order 2 projects have to be applied, you can do them in any order. So, A join B same as B join A.

So, all those properties are there. So, you make use of them and then adjust these operators in such a way and then union join intersect they are also associative. So, A join B join C can be written as A join B and then join C or A join B join C, you can write it either way. So, use make use of these associations also make use of these algebraic rules and do some rewriting if necessary. So, there are many more of these, you know, how do these operators interact with each other, there are a lot more algebraic laws that we can express.

Now, I am not going to cover them in the lecture based please read about them, they are very interesting. So, you can make use of them they are not very difficult to understand also. So, basically we will make use of these relational algebraic laws and the heuristic rules and then do a rewriting of the expression.

(Refer Slide Time: 08:52)



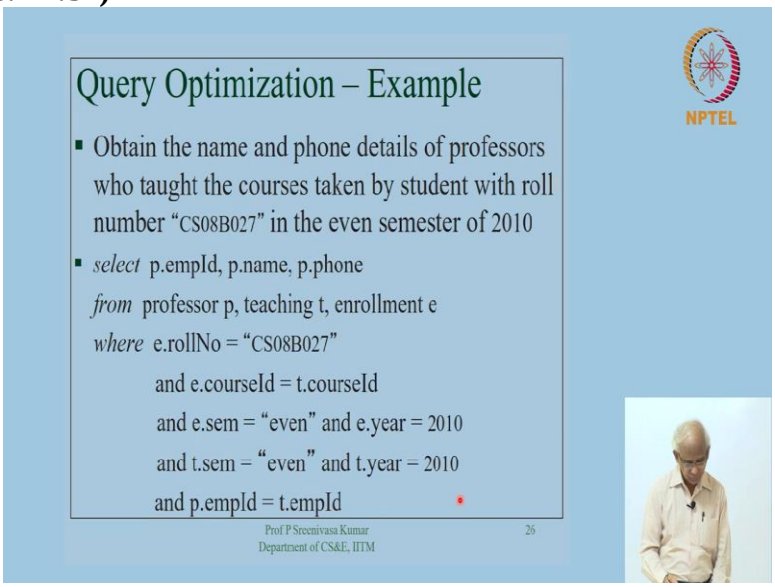
So, and then the cost based optimization will come into picture so the issue here is that the relational algebra operator that is there in the middle of the tree somewhere can be evaluated using many methods we have discussed several possibilities for doing joints and making use of indexes if they are available like that there are so many possibilities there. So, for each of these methods, we need to have some kind of an estimation of what is the cost, it is we are going to incurred if we use that method. So that is what is the cost function.

So for all these methods, you can essentially use those formulas that we were discussing as to how many number of block accesses it will do and things like that and convert them into some kind of cost function. So that cost function will take the various parameters like the size of the files, and then the availability of indexes all these things into consideration, essentially the size of the sizes of the 2 operands, and then it will give you an estimated cost.

And one important thing that we should note here is that these choices have to be made in some sense simultaneously, if we make 1 choice here that might affect the choice of another method somewhere else. So, there are interdependencies between these methods. So, one has to cast this as a as an optimization problem, and then use some optimizers for doing that. So, it is actually an intractable problem. It can be shown that it is an intractable problem.

So, you basically have to use a lot of heuristics and then you have these different plans of execution. So, evaluate them, based on these. The individual cost estimates and then overall plan cost can be, what is the cost of the whole plan with all these choices made at each of these, all the properties, and then choose the plan with the based estimated cost, that is the cost based and these optimizers can actually sometimes go horribly wrong also, because they are all basing decision on estimates. We do not really know how much or how the data.

(Refer Slide Time: 11:57)

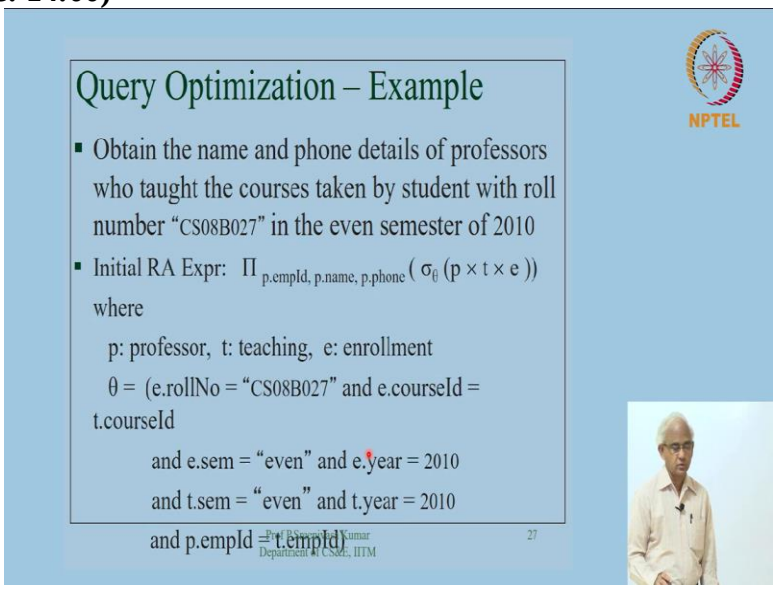


Now let us take an example and illustrate these essentially the heuristic and algebraic optimization ideas with this example. So read this query. The query asks for names and phone details of professors who taught the course are taken by some student with roll number. I am in the even semester of from here. So it is a simple query. So here is the SQL query for that. So, remember that we have professor details in professor relation, and who is teaching what in teaching relation and then who is enrolled for what course in enrollment relation.

And so, we have up various conditions now so, the roll number is given. So, e e dot rollNo is CS08B207 and the e dot courseId it should be the one that is being taught by the professor t e dot courseId and the semester is even a year is 2010 in both enrollment as well as teaching relation things and then the person who is teaching is who we are interested in. So, we will project employee Id name and phone off that particular professor.

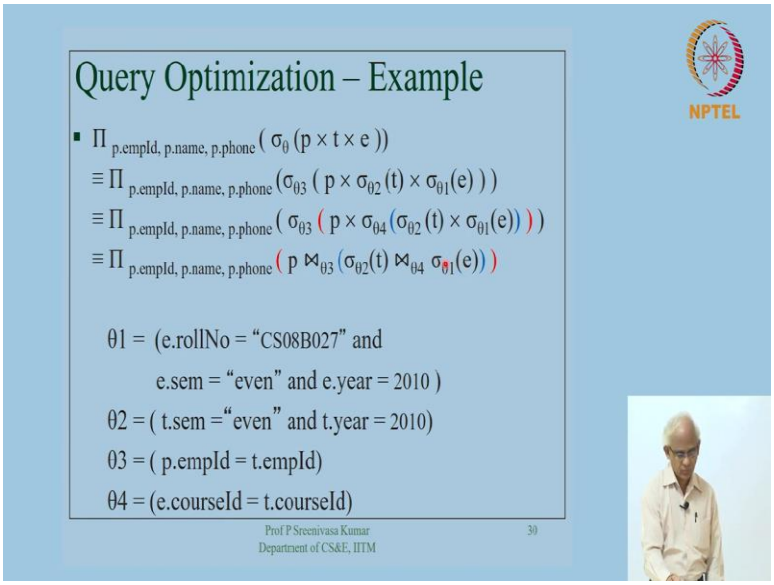
So, this is how typically the where condition is specified so, we do not distinguish between which of these things are, you know, join functions and things like that, but we one can actually figure them out figure them off. So, the initial relational algebra expressions.

(Refer Slide Time: 14:00)



That we will write will be simply like this you know that this is equivalent to be SQL query pi p. empId p e dot name p e dot phone sigma theta, p cross t cross e we the 3 the 3 relations involved in the from clause so p cross t cross e and though this p what is theta, theta is this entire stuff that we have seen where. So I have reproduced this entire where condition here. So that is how the initial relational algebra expression will look like. Now we start a series of equivalences for this relational algebra expression, when we will show how exactly this entire condition has to be placed appropriately inside and how we convert this operating system.

(Refer Slide Time: 15:10)



So, let us see how it works. So, p dot there. So, this is the projection list and then sigma theta or p dot, p cross t cross e where theta is this entire. Now, what I have done here is that this theta that was in the previous slide, I have you knows the big group predicates. Luckily for us, it is all conjunctions. So, what I have done is to group together the things that involve enrollment, all the conditions are all on the enrollment here.

The conditions on the teaching relation here e e dot sem equal to even e dot equals and then the other conditions the remaining conditions are put it with me. So, theta 1 theta 2 theta 3 are broken down the same thing into this one. Now, after having done that, we know that it is sigma you know sigma. So, if this is a conjunction then I can apply them in any order. So, I can now, write this as theta 1 and theta 2 and theta 3, then having realized that this you know this theta 1 is actually applicable to enrollment.

I can now kind of push it all the way towards this is what it is. So, this theta 1 is completely working on the e enrollment. So, I can now so, this I can take this theta 1 that is part of this theta and then push it all the way here, you know give you that and so similarly, I can push theta 2 to t that is completely applicable to the relation here even if you keep it here, we will actually form the cross product and then again do the selection so, instead, I can now do the selection and then do the cross for the cross product and selection can be use that way.

So, this is what we do. So, theta 3 of course, involves p t e and all that. So, it involves all the 3 so, I cannot do anything about it, I will keep it for the moment. Now, in the next slide actually, I will actually break this into once that involve p t and the ones that involve e t. So, I can then the one that is involving e and t, I can actually push up to this let me see we are now let us break this theta 3 of the previous slide into 2 which is p e dot empId = t e dot empId and e e dot courseId = t e dot courseId.

I have broken down that into for the 2 predicates. So, having done that, I now realize that this portion of the expression has e and t. And so any condition that is involving e and t can be pushed all the way up to this sub expression. So, that is what I will do theta 4 is having to do with e and t. So, I will take the theta 4 and then push it all the way up to here. So, I have a new bracket now, sigma theta 4 here and then I applied to this one.

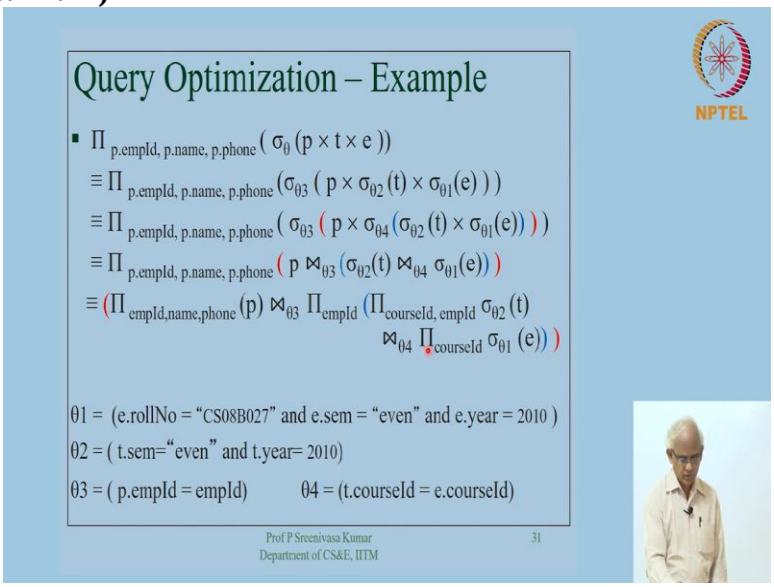
So now you can see that when this selection conditions are going, they are going down the tree. Now this theta 3, which has p and t has to remain here, because this is involving p and then something else so p and t, and so it has to do. So I hope it is clear up to this. Now, actually we can use this transformation that the cross product followed by a cross product followed by selection and has to be replaced by joins, basically.

So now we will do that. So the cross product of 2 relations followed by a join can be replaced by a is followed by a selection can be replaced by a join. So we will take this as a join condition. So put the join operator here. So, theta 2 a sigma theta 2 t sigma theta 1 e will come as it is now, we have in between them we have the join can join this theta 4 similarly here for the result of this and the p I know apply the join and then write this as join theta 3.

So, theta 3 p e dot empId = t e dot empId which is applicable here. And then theta 4 is e e dot courseId = t dot for courseId so it is appropriately position. So did you get that so, this transaction is actually in simple algebra so, the cross product followed by selection can be replaced by joining it at whatever. So, now that we have it, we will love focus on the projections, we will focus on the projections.

The projections in fact what is it that is required off enrollment in theta 4 what are the conditions that are there courseId and courseId here also so, you basically need courseId and from what is theta 1 theta 1 has conditions on enrollment and semester year so after you do this condition, after you do apply the selection condition, what should it be? What should we project here? Because the only thing that is being used here is courseId. So that is what needs to be projected after doing this selection.

(Refer Slide Time: 22:21)



So, we will work that out. So, now you see, I have introduced projections; basically with the idea of saying that do not carry any attribute that is not going to be necessary, upward in the tree the simple principle. If you are at some lead and it is producing some intermediate results, do not carry any attributes upwards in the tree unless it is going to be useful for tree, of course. So theta 1 is all this. So you applied it to enrollment so the join connection here requires courseId so only project courseId that is all you need.

And here from the teaching you require not only courseId which is being used here in theta 4 but you also need employee id because that is what is being used t e dot empId is coming somewhere theta 3 is coming us employee ids system so you project courseId and empId from teaching. And then you have that appropriate intermediate result. And then from there, actually, you after doing this join, you only need for the employee id because that is what is being used in theta 3.

So and then from, for p, you can keep all the other things because the empId, name on all of them are applicable to professor relation keep these here. So, this is how we will transform and push the projections also down as much as possible into the individually towards the leaves so of course I am illustrating the algorithm through the through an example only but I just want to get you get the spirit behind the process.

So, basically we will avoid carrying unnecessary attributes upwards in the tree and then use projections to reduce the sizes, the widths of the tuples that we are going to carry. So this is how define this is now, you cannot do anything more on this particular algebraic expression. So, this is the final expression that we have. Now, we have to find out how this joint has can be evaluated, and how this away and joint can be evaluated, find out the methods for evaluating these things.

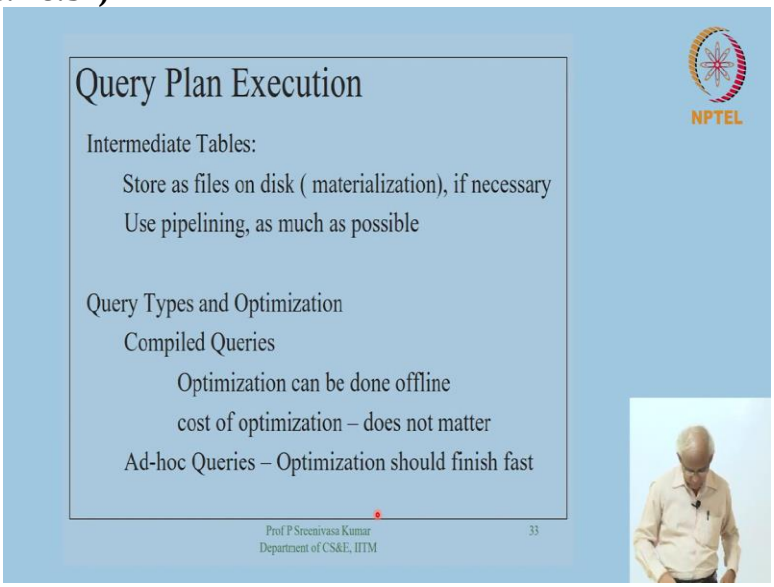
And then so what you basically also do is while you are trying to prepare the input for the join, you will read in this relation. So, you have to read this relation in order to prepare. So, actually what you do is you combine all these things into an operation where as you are reading the records, you apply the selection function, pick up the attribute and then prepare the results.

So, you combine and then pipelines the obviously it does not make sense to first you know, apply the selection condition write it back to the list and then you know, and then later on do so, what you do here is you also you make use of pipelining idea. So, once you start reading the records, you see what are all things that can need to be done with the records and then do them all together and then prepare the results.

So, once you are prepared this results probably in as an intimated relation, then you also do the same thing then you have to essentially 2 joins to perform and then you figure out what are the costs or what are the applicable methods and make use of this formula or any estimates that are available in the catalog bigger or what are the cause for the different kinds of plans. So, the plan here could be that I will you know, I will apply this join first and then the other join will actually in this case, I do not think we have much choice.

Because we have to apply this join and then apply this but in general in a query you may have options of how to execute joins fine. So, this is what we will do is this whole process kind of clear. So, essentially what we are doing is to initially put this entire where clause as the theta condition here, and then break it down into pieces. Have conditions that are to be satisfied by appropriate components of the expression. And then push them towards the operators towards the operands lower down the tree and then use all the bright transformation rules to replace them by joints and then go ahead and then finally also possibly projections.

(Refer Slide Time: 28:32)



So, finally, the query plan execution is, like this. It also kind of depends on the what kind of query that you are handling. But before that, so, you have to take a call on how to handle this intermediate tables. You have to take the intermediate tables will come up. So, depending on the size of those intermediate tables, if you can handle them in memory you better handle them in memory if necessary only you can materialize and then actually push them to this to the extent possible, keep it inside and then do what is called pipelining.

So, try and apply multiple operators on to be stream of records that are going to come in a pipeline (())(29:32). Now, so, there are actually 2 kinds of queries, the compiled queries and the ad-hoc queries. So the compiled queries are the ones that actually sit inside the transaction are pieces of work that you want to do in order to carry out certain logical request organize the database.

So, it is a program that involves you knows, shooting several queries to the efficient database getting reserves and then probably shooting one more query. So, you know the queries that you are going to shoot ahead of time. So, since you know them, you can actually afford to take that query and then do a lot of whatever optimization that you can do and then actually fix a nice query plan and then go ahead, use that query plan.

But as again as this kind of queries, if you get ad-hoc queries that means from the terminal use you analyst is sitting there and when you if he or 3 types a brand new query, which you have never seen them before, because he has some data analysis me and then you know, and he is expecting results fast. So, you may not be able to do well and operate optimization in such context, you will have to do some a little bit quick optimization and figure out.

How, fast he can execute that, because there is some, performance penalty that you pay while you are doing optimization of a query also. So, you have to work out you know learn some algorithms and figure out and then choose the query plan and all that so, it takes thing. So, you, also have to have some lighter versions of this optimization, so that you do not, where it matters, you may have to you know, apply some lighter versions of optimization. Whereas, for compiled queries since we are, you know, doing them kind of offline.

The optimization can be done offline, the cost of optimization actually it may not matter much. So and since this compiled queries are going to be executed some 100s of times, so you are better off in spending some time and figuring out as to how best to evaluate that way. So that is how this whole so; it is indeed a vast subject and a lot of research is actually has gone on and is continuing to be done by relational databases, researchers on how to improve this query object, because they are in some sense the heart of the whole system. So I think with that, we conclude this lecture.