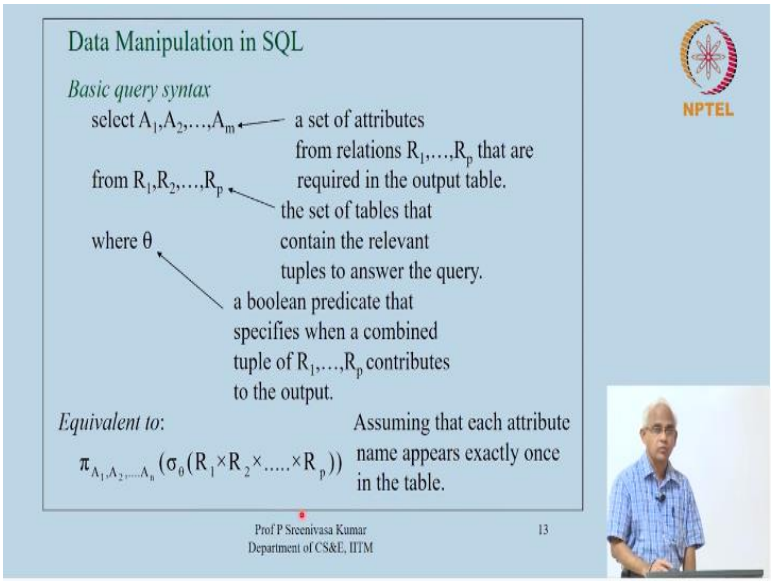
Database Systems
Dr. Sreenivasa Kumar
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture – 14
Basic SQC Query Block and Sub Queries

In the last lecture, we started looking at SQL, so SQL is going to be a very important component for you. So we are going to touch upon the fundamental ideas and some of the fundamental ideas behind SQL.

(Refer Slide Time: 00:37)



And in the last lecture we looked at how a relation scheme can be defined using constructs that are available in SQL. So in this lecture we will focus on the one of the most important aspects of SQL, which is query, querying for information. Now with this introduction of this you are now looking at the third way of expressing relational queries we have started we looked at relational algebra is one way of expressing queries then tuple relational calculus now we are going to see SQL and you can see that they the influence of tuple relational calculus exists on SQL.

Both tuple relational calculus and the query expression in SQL are thought of as the declarative ways of specifying queries whereas the relational algebra way of specifying queries is more procedural okay. So with that little introduction let us get started with looking at what is how we

can express queries in SQL. So the basic query syntax is probably familiar to you by now because you some of you would have seen this clausal structure of queries in SQL earlier.

So it has a select clause a from clause a where clause and a few other clauses we will see later so this is the basic query syntax. So select keyword followed by a bunch of attribute names for the moment we will assume that no attribute name no attribute name appears in more than one relational scheme okay. All the relational schemes that we are talking about or in the database have distinct attribute names.

No attribute name appears in more than one relation we will just make a simplifying assumption we will later relax that assumption it is not a important assumption we will make relax it for the moment we will make such an assumption so here a bunch of attributes a set of attributes from these relations R1 through RP that are required in the output okay and then the from clause is where we specify R1 through RP the set of tables that contain the relevant tuples to answer the query okay and then the most important thing is the where condition.

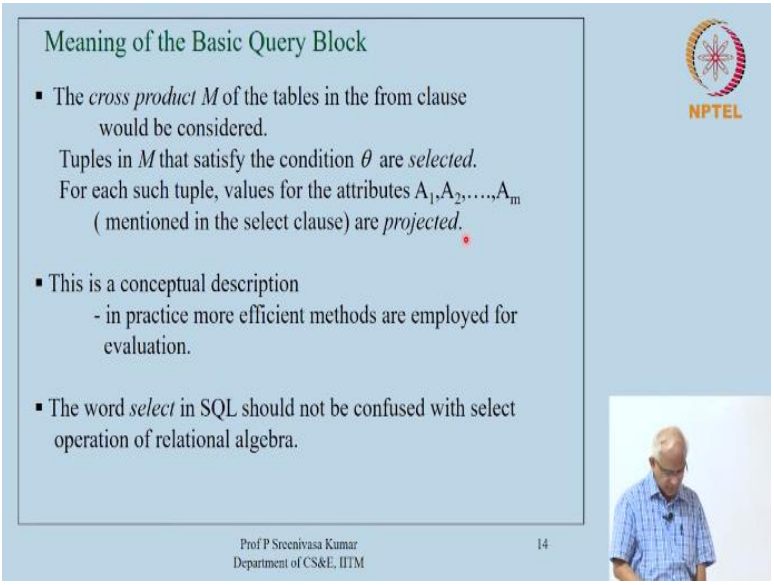
So the where condition theta is a Boolean predicate that specifies as to when a combined tuple of all these R1 through RP okay a combined tuple. We will see exactly how the combination works out a combined tuple or R1 through RP contributes to the output. So if it satisfies this predicate then it contributes to the output. So what is the condition Boolean predicate that has to be satisfied by the combined the tuple of R1 through RP so that its at this it can contribute to the output.

Now you can see that this because of this assumption that we made that all the really no 2 relations have any common attribute name okay attribute name when you talk about a combined tuple you can think of tuple from R1 and followed by concatenated by values from a tuple from R2 concatenated by a tuple from R3 so etc. You can so that is what is combined tuple is and there is no issue in combining them because all these relations have disjoint attribute names okay.

So such a combination we are looking at and then it satisfies the condition to be so what is the condition it should satisfy we will specify here and now under this assumption that each attribute appears exactly once in the table this entire query is actually equivalent to the following relational algebra expression.

So you can imagine that we have R1 cross R2 cross RP main cross product of all relations that are mentioned such that and then you select tuples from that combination that satisfy this wear condition and project A1 through Am that is the. So it is logically equivalent to this relational algebra expression and with that with this equivalence written down I think it kind of becomes very clear as to what the select from where query achieves. So but we will see it in elaboration now how exactly what is the interpretation now what is the meaning of this basic query block.

(Refer Slide Time: 06:34)



So the cross product M of the tables in the from class would be considered and so in that this this let us call this cross product as M. So tuples in this M that satisfy the condition data or selected okay and then from these selector tuples we choose the attributes A1 through Am we pick up the values of the attributes A1 through Am and mention them so and in some sense project them to generate the output.

So the output will have as many rows as there are combined tuples that satisfy the condition theta okay and this schema would be schema, or the output is essentially decided by the attributes

mentioned in the select clause of SQL these projected attributes. So in some sense this is a conceptual description this is how we can think of it as being produced the output being produced.

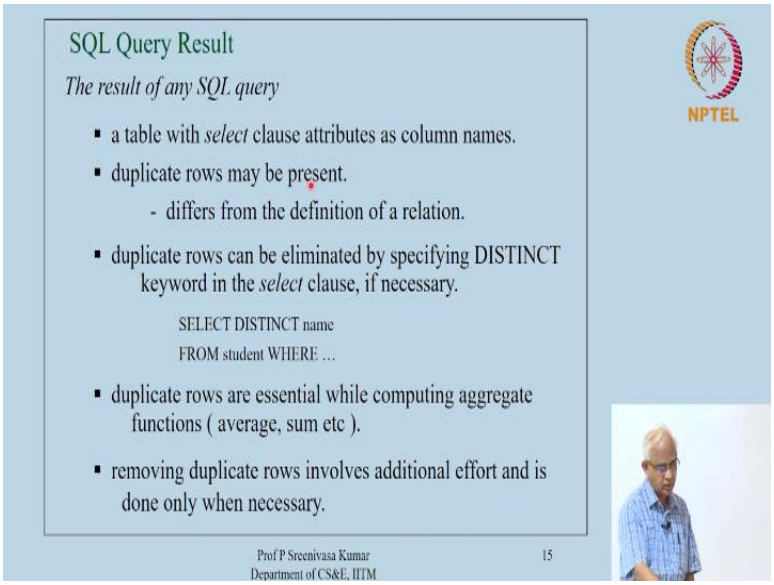
Of course in practice it is very inefficient to construct the cross-product of all the tables that are mentioned in the from class so you would not actually do it that way, but we will find some efficient ways of you know considering the combinations without actually producing those combinations okay. So that we will see when we come to the stage of query evaluation and things like that.

Now of course it will be very clear that the word select in SQL should not be confused with the select operator of relational algebra it is exactly I mean it is serving a completely different purpose here in fact it serves more in the purpose of a projection actually. So that is the meaning of a basic query block. So you can observe the similarities between this and what we had done in the tuple relational calculus.

In the tuple relational calculus situation also we had considered all possible tuple assignments to the free variables in the tuple that expression right and then in that combination if some of the combinations have satisfied the condition that is there on the right hand side of the vertical bar then that combination produce the output and what is the output? Output is indicated by the attributes that are present on the left-hand side of the vertical bar right.

So in a very similar to what we have been doing in tuple relation calculus we execute mean we can imagine this SQL queries being executed okay now let us start looking at various other issues.

(Refer Slide Time: 10:11)



First of all let us see what is the result. So the result of any SQL query is a table with select clause attributes as the column names and duplicate rows may be present in the result of SQL query this in some sense differs from the definition of a relation because in the relation, we never allowed duplicate rows to be present because we defined relations as sets of tuples earlier. This issue of having duplicate rows is important and first thing is in case you do not want duplicate rows SQL gives you a way of specifying that okay.

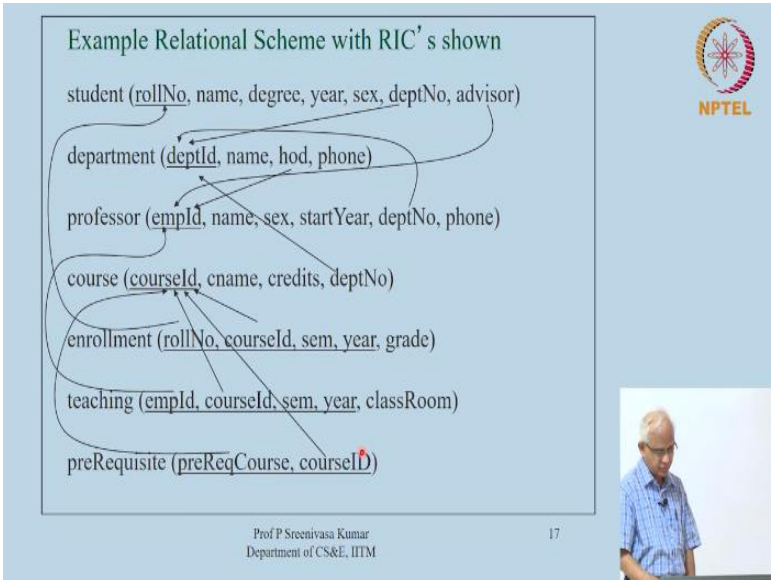
So duplicate rows can actually be eliminated from the output by specifying what is called a keyword called distinct in the select clause. So if you include the keyword select keyword distinct in the select clause select distinct name etc. If you write it like that then duplicates would be removed. Now there is a reason for doing you know keeping the duplicates and by default in the result.

Mainly you will require duplicate rows duplicate you know tuples to be present especially when you are applying some aggregate functions over the result of these generated results okay over these computed set of tuples if you want to apply some aggregation function like for example computing the sum of all the salary values or average of all the age values or things like that if you want to do some computation then you should not be doing it on distinct values in the output you should consider all the values that are present in the instance and then do that.

So these are essential for computing aggregate functions and so the default start strategy that is that is adopted by SQL is to give you the duplicate rows. So in that sense the tables that SQL produces as into our results or intermediate results are not really relations and that is why we sometimes reserve the word table when we are talking about results of SQL then relation.

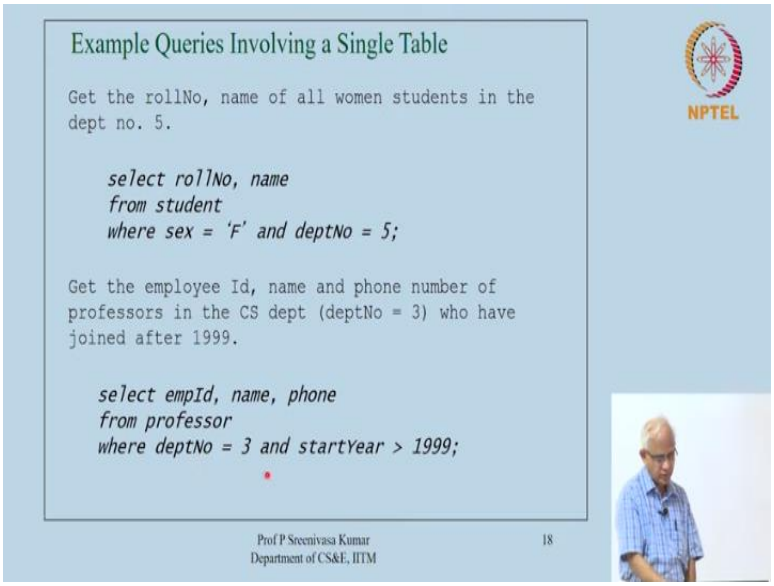
And removing duplicates is also a little bit of a costly operation because you one in order to remove duplicate tuples you can imagine that you have to sort those tuples and then find out where the duplicates are and then you know remove one copy of them right so it cost some effort so the idea is to put the traditional effort only when it is necessary and is requested by the end user okay so that is the output SQL query result.

(Refer Slide Time: 13:42)



So now we will start looking at lots of examples and then to illustrate various points about so this schema is the same scheme as we have been using so please keep it handy for you so that we can start looking at simple looking at various features of SQL through examples actually SQL query in portion okay.

(Refer Slide Time: 14:13)



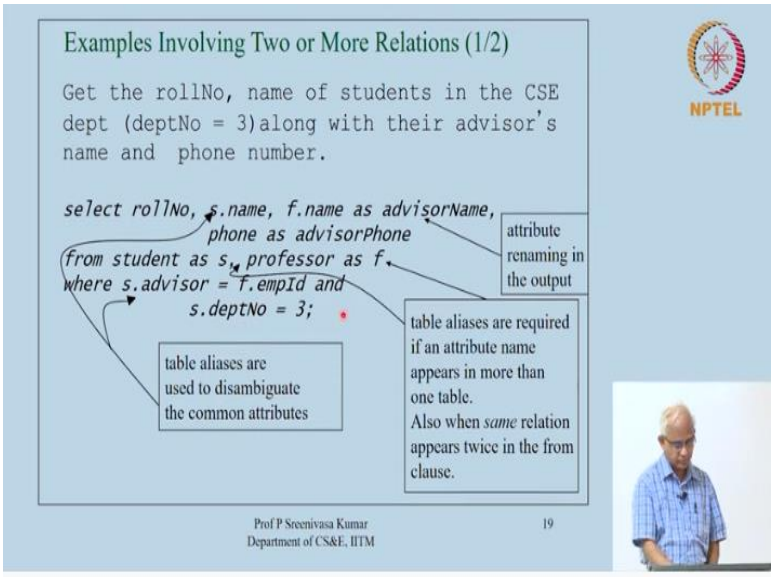
So let us look at probably this is the same kind of query that we started off with when we were looking at relational algebra tuple relational calculus get the roll number name of all girl students in the department number 5. This involves a single table all that you need is student in that student the attributes the gender attribute and the department number attribute have to be restricted and then you can get hold of the roll numbers.

So the way you write it is essentially select roll number name because that is what is required ask for where does the information come from it obviously comes from the student table. Now comes this where clause so the tuples should satisfy this condition in order to produce output in order to be used for producing output. So what is the condition it should be the gender should be F because we are looking for women students and students should be in the department number 5 so it is very simple.

It is also very simple so you can this information also comes from 1 relation which is the professor relation, so we are restrict putting conditional such that the professor is CS department and has joined after. So we have a joint date join year as an attribute so we can use them so essentially attribute names and comparison operators with appropriate constants or other attribute names that is how the basic conditions are to be formulated in these predicate I said there is a Boolean predicate right.

So the Boolean predicate is essentially constructed as we did it in tuple relational calculus but little bit some a few restrictions are there in the sense that we cannot use universal qualification. So as we see more and more examples you will realize as to how what are the various kind of constructs that are allowed in the Boolean predicate. Okay so this is easy select employee ID name phone that is what is asked for from professor where department number equals 3 and start year is joined after 1999 okay.

(Refer Slide Time: 16:51)



Now here is another query where a lot of points need to be illustrated so we have put lots of boxes around so let us read the query let roll number name of students in the CSE department, department over 3 along with their advisors name and phone number for some reason okay we want to get this information. So where is this information student information is there in the student and then professor information because we want advisors or professors, so we want name of the professors and phone numbers.

So now you will see that the student relation now here we are relaxing the assumption that we made earlier that the relations do not have common attributes right. We will henceforth assume that they may have common attributes and we will handle them and the way we handle them is actually by using the our familiar dot notation okay.

So in order to use the dot notation we need something before the dot what is that yes, we introduce what is called in the from clause its possible for us to not only just mention the relation name but we can give an alias for that a name for that. So in case you are you know you are using students say 2 times for some reason student as s1 and student as s2 because you want to do some comparison among students.

So you definitely have to you know distinguish between the 2 users of the same relation okay so that is when you will definitely need table aliases and if table if 2 relations share attributes share the same attribute name then also in order to disambiguate as to which attribute whose attribute are you talking about you will need table aliases. So these s and f are called table aliases.

So student as s professor as f and now we need roll number, roll number does not need to be you know s dot roll number you do not have to mention because only students have roll numbers then s dot name to pick up the name of the student f dot name to pick up the faculty's name and we are also doing renaming of attributes here in the select clause itself its possible for us to do renaming using the word as and giving a new name here and then phone as advisor phone so that you know in the output it is clear that it is advisor phone or the students phone.

So we are doing in the Select Clause roll number s dot name f dot name has advised the name phone as advisor phone and then in the from class we introduced this table aliases. So table aliases are required if the attribute name appears in more than one table also when same relation appears twice in the from clause. For example you want to compare for some reason you want to do comparison of tuples of the same relation right in 2 different rows.

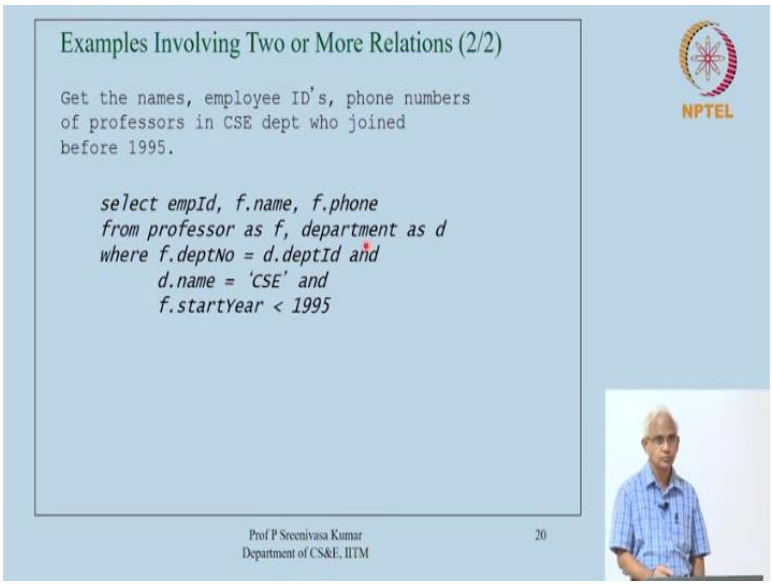
For example when you are trying to compute prerequisites of a course or something like that then you will need to use the course as a course playing the course playing the role of a course and playing the role of a prerequisite all right. So it is appropriate to introduce table aliases for that now what is the appropriate condition here that the combination of student and professors should satisfy.

First thing is we want the student to be in CSE department so s dot department number equals 3 the other one is these combination of the student and the professor should be such that, that professor is the adviser of the student we do not want random combinations right. So that is how do we enforce that you just simply enforce the same by s dot advisor is f dot employee okay. Now notice one thing here that I am very careful not to use the word join condition here unless you know.

So you can say that the combination has to satisfy this condition we do not care whether it is a join condition or a selection condition or whatever condition it is all part of this the conditions that it should satisfy the conjunction, so conjunction is there. Okay so this is where it you know in some sense, we are not telling that okay you compute the join of s and f with this join condition and then do something etc. Right as we used to do in relational algebra.

But instead what we are doing here is that consider the combination of these students and professor tuples and then ensure that this condition is satisfied by the combination and then use it for generating the clause. So that is the essential difference between procedural way of expressing queries and declarative way of expressing queries okay. So I suppose all these points are clear about from this slide as to what we are doing here we talked about renaming attributes using the as keyword and the same as keyword is also used for introducing table names or table aliases alternate names for the same tables and then the conditions okay.

(Refer Slide Time: 23:24)

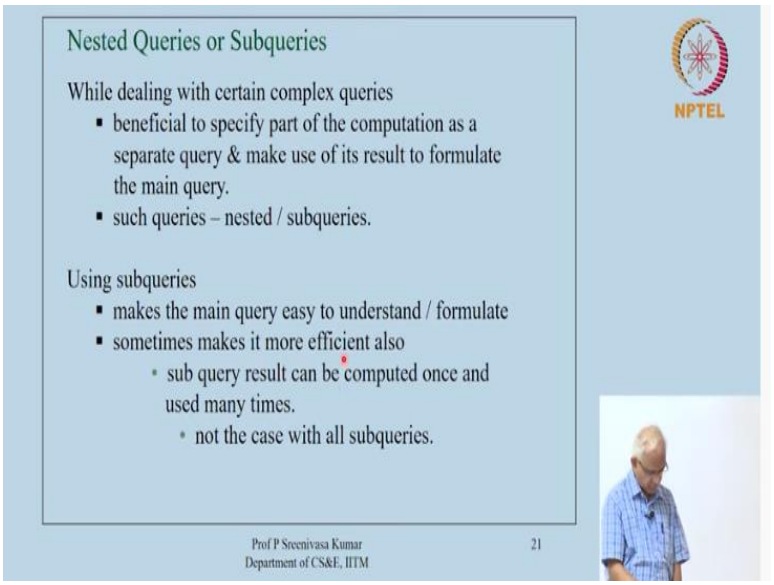


So let us go ahead and look at another query this is simple get the names and employee IDs of 4 numbers of professors here who joined before 1995 I think similar only thing was earlier, we said we gave the department number for the CSE. Now we are saying just CSE Department is there so the information is available in professor as well as department tables why do you need department table that is because name of the department does not exist in professor table and we have to act as though we do not know the number of CSE Department right.

So we do not know this number a department ID of CSE Department put it as condition here so here professor as f department as d .The combination of professor and department is meaningful if the department number of the professor is same as the department ID of the department tuple then then we are considering the appropriate combination of professor and department and then we are putting condition saying that the department should be computer science department and the professor must have started his surveys before 1995 is that clear.

Now you may say that probably we do not need the table aliases here because I think the can you please check whether the department and professor have any common attributes phone is their common name and phone are common okay, so you do need okay.

(Refer Slide Time: 26:05)

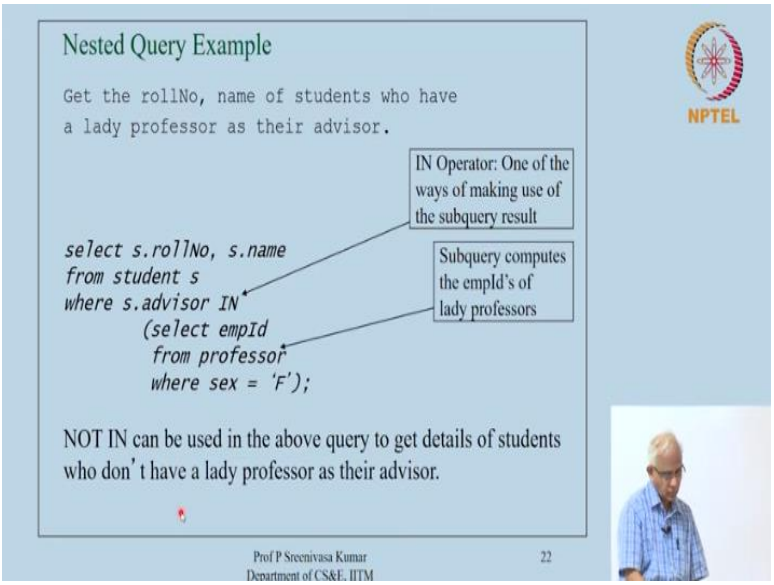


So now a somewhat unique feature of SQL is that it allows you to kind of introduce sub queries as part of 1 query you can start using a you know to express a complex kind of a condition you may sometimes want to you know break it down and specify the one part of the condition by another SQL query and then somehow make use of the results of that SQL query in the outer SQL or outer query block.

So while leading with some strong kind of complex queries its sometimes beneficial to specify part of the computation as a separate query and make use of its scissors to formulate the main query. So we call such queries as nested or using sub queries they use sub queries. So one of the main advantages of sub queries is that it kind of makes the main query easy to understand and sometimes easy to formulate.

And sometimes it also can make it efficient to run the main query because it may be the case that the sub query result can be computed once and then can be kind of used many times while you are running the outer query. Though it may not be the case for all sub queries but no these are the reasons why we use sub queries so let us see how sub queries can be introduced.

(Refer Slide Time: 28:03)



Okay get the roll number names of students who have a lady professor as their advisor. Now you could you know imagine a query an independent kind of a query who figures out who was lady professor's first is going to be a little small set and then imagine that you are getting the wrong number names of students whose advisor happens to be in this set of a small set of lady professors okay.

So one can structure the you know computation or specification of the query we do not say computation because we can only specify the query right. So how we can imagine these structuring the specification in such a way that we can imagine that there is a small independent query that can that will produce employee IDs of lady professors and then we can run a query in the main query we will only examine students and then check whether their advisors are in the result of the sub query okay.

So now actually this brings in a crucial point into consideration that we need some operators that kind of connect these 2 queries we need operators in some sense to connect this queries. So we use some kind of set kind of operators for doing that so let me start a list rating that so here is how we could do that using the sub query feature. So this is the sub query and the we are using the IN operator to make use of the sub query results so what is the sub query doing select employee ID from Professor where sex equals f, so you are getting hold of all lady professors employee ID.

Now what is this table this table is a single column table with employee ID as a temporary table with single column with employee ID. Now so in the outer query we are saying get the roll number name from student s where s dot advisor in this set. Those of you who are little keen eyes will observe that there is a type mismatch here subtle type mismatch is there right s dot advisor as far as we are concerned is actually a single value whereas here this produces tuples okay.

So let us be conscious of that type mismatch SQL allows you to kind of be a little bit type liberal okay it says okay fine we will ignore that type mismatch. The value here is s dot advisor, s dot advisory the employee ID so it is a atomic value and that atomic value now we are checking whether it is in this set. So this set is actually set of tuples each tuple containing 1 value okay. So technically speaking very theoretically speaking the value will not match this because there is a type mismatch there is a single value there here is a tuple of single value here right.

So let us ignore that so let us imagine typecasting this as a tuple s dot advisor we will convert that into a tuple. So that it can now check whether that tuple exists in this set of tuples. So this IN is one of the operators we can use in to connect up the sub-query results with the main query. So how does the main query how the whole or overall query work again student tuples will be considered because that is the only there in the from clause only relation in the from clause.

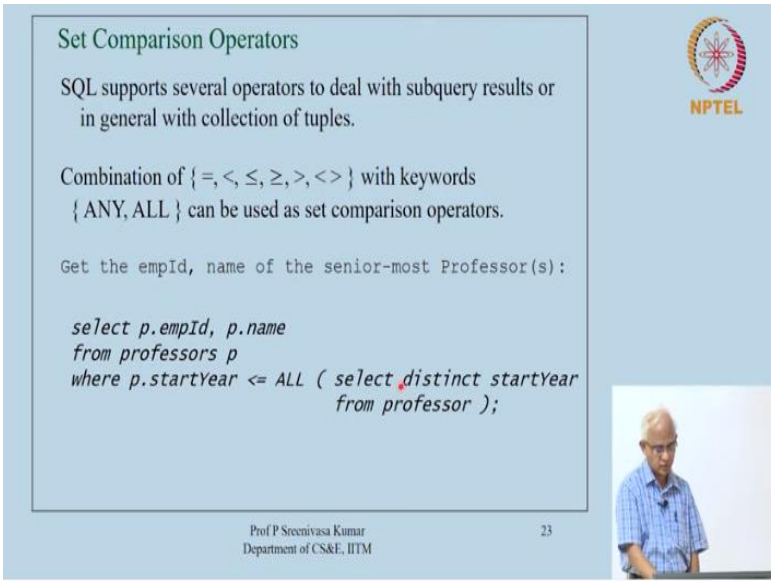
And for each student tuple we are checking whether this particular condition is satisfied and that satisfy making the satisfaction or otherwise of this condition involves computing another set of tuples. So in this case you can see that this set of tuples can be computed once and then reused several times will this always be the case can this always be the case is an interesting question. So the answer is of course the fact that I am asking such a question the answer is not right.

So it depends actually whether when it will be can be computed independently and when it may not be computed independently you know there is certain there is something else that is probably involved. So but you can imagine that in case this particular thing so there it actually the idea of a sub-query raises several issues can we make use of these table aliases introduced in the main query here that is a question right.

Should this this has a no from clause but no are these the relations they are being considering the outer from class are they accessible here that is an important question. Indeed they are actually accessible so you could use off in case it is really needed you can also use yes here okay we will see how exactly such a query you know can be formulated. But right now first of all let us assume let us settle this issue of how to make use of the result of a sub query in the main query.

So something else okay so IN is there in a similar way NOT IN also is there.

(Refer Slide Time: 35:30)



So in order to so let me go to the detail about what are the kind of operators that are available to do this. So this they are essentially like set comparison operators because you have one set of tuples and how it will be compared. So to deal with the sub-query result SQL gives you several operators. So the combination of these comparison operators with the keyword ANY and ALL can be used as set comparison operators in the SQL okay.

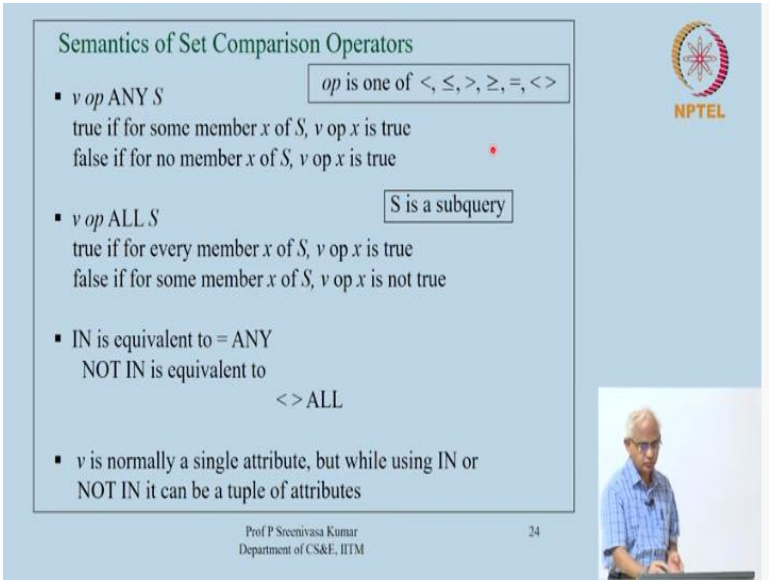
‘

So let us look at this query get the employee ID name of senior-most professors. This is the query that you have tried in the quiz. Similar query was there for you in the quiz. Now we can see that with the with the provision with the this operators being present you can actually express this query very easily. So who is the senior most professor whose start year is less than the start year of everybody else that is it all right.

So now less than is there less than ALL the combination of less than and ALL can be used. So professors from professor p,p dot start year is less than equal to all select distinct start years from professors, so this professor is different from that professor. So this is another instance of the professor. So this sub-query sorry this sub-query what it is doing is simply taking all the start years of all professors and generating a list of start years.

Because now what is this doing it is just checking that the for an individual professor the start year is less than or equal to all the values here. So if the start year of a particular professor is less than everybody else start year then he is the senior most right. So this is the set comparison operator that we are talking about.

(Refer Slide Time: 38:30)



So in general what are all the various comparison operators are there and what is the meaning of them this is the general comparison operators v some value of operator op is one of these 6 comparison operators and S. So ANY and ALL is possible so let us start with ANY v op ANY S is true if for some member x of S for some tuple v op x is true because its ANY S and false if for no member of x of S v of this true where S is a obviously sub-query of appropriate type it of course if the sub-query produces a wider tuple then it is an issue right.

Right now we are assuming that the sub tuple produces values tuples with single value in each of them v op ALL S is true if for every member x of S every tuple x of S v op x is true one of these

operators false if for some member v of x is not true. So that is the general meaning of this set comparison operators and you can see actually what we have in tuple of slides back we have used IN IN.

But that IN is actually equivalent to is a short form for equal to ANY. So instead of writing equal to ANY we will write IN and what do you think is NOT IN equivalent to equal to something right not equal to ANY or not equal to ALL. There are only 2 combinations ANY or ALL nobody wants to say anything wake up that guy how are you doing is my voice so okay let us see not equal to ANY will serve the purpose of NOT IN.

So it will be true if some members of x v op v not equal to is true. So if not equal to one of them even if it is equal to something else it will pass right it is a dangerous thing whereas we do not the meaning we are expecting from NOT IN is that it is not there in that particular set so be careful. So not equal to ANY will not serve this purpose so not equal to ALL serves the purpose.

Okay so normally most cases we use v as a single attribute but while using this IN or NOT IN it can in fact be a tuple of attributes also. Obviously if it is a tuple then tuple equivalences can be carried out the tuple is equivalent to the other tuple or not we can figure it out there is no of course we cannot you know apply other operators for tuples like greater than less than and all that it does not make sense though you can technically probably define that it does not make sense to do that.

So well using IN and NOT IN it is possible that we the outer query is actually generating a small tuple and then it is asking whether that tuple is present in the results generated by the sub-query okay. So okay so to summarize what we have done is to kind of look at the basic structure of SQL queries which has this clausal kind of structure select from where and where condition is the most important thing.

So the condition is a Boolean predicate that has to be satisfied by the combination of tuples that are of relations mentioned or tables mentioned in the from clause and those combinations will be

used to produce the output and we are now also looking at whether sub queries can be used and things like that. Okay so we will stop here today.