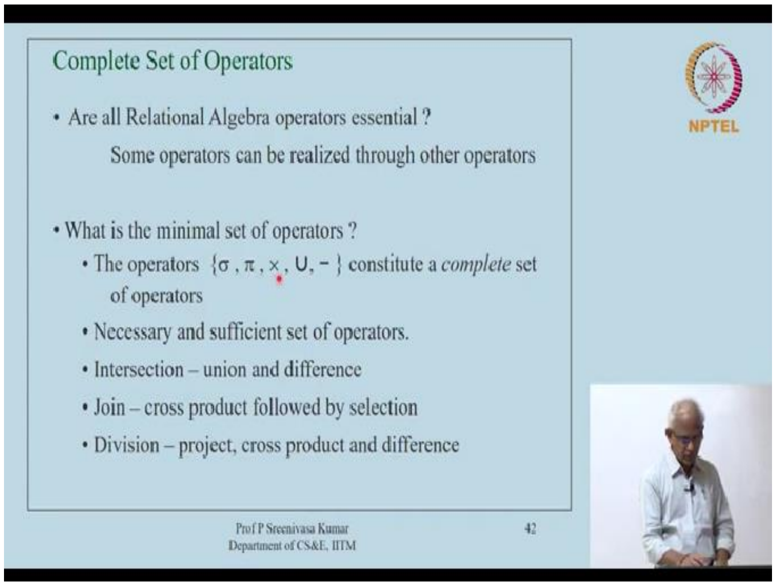
Database Systems

Prof. Sreenivasa Kumar
Computer Science and Engineering Department
Indian Institute of Technology-Madras

Lecture-11
Example Queries in Relation Model and Outer Join Operation

In the last few lectures I have talked about various relational algebra operators and in the towards the end of last lecture, I was asking you this question whether all these relational algebra operators are essential and then we discussed this point that.

(Refer Slide Time: 00:51)



Complete Set of Operators

- Are all Relational Algebra operators essential ?
  Some operators can be realized through other operators
- What is the minimal set of operators ?
  - The operators $\{\sigma, \pi, \times, \cup, -\}$ constitute a *complete* set of operators
  - Necessary and sufficient set of operators.
  - Intersection – union and difference
  - Join – cross product followed by selection
  - Division – project, cross product and difference

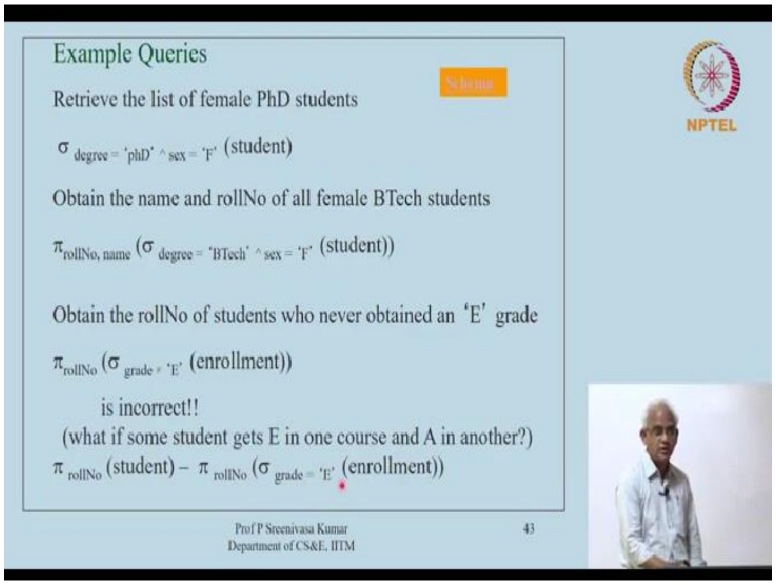
Prof P Sreenivasa Kumar
Department of CS&E, IITM
42

The few just about the 5 operators the select, project, cross product, union and difference are the constitute the complete set of operators in the sense that we can these are the necessary and sufficient set of operators, we can realize other operators through these operators, but in practice in addition to sigma, the selection projection, we use join operations and then of course also use intersection, union and difference.

So, that at this stage we will not really bother about how they get internally translated to each other, but this is from a theoretical interest at least 5 are necessary okay. So, in this class in this lecture, basically I want you to be a little bit active and then you know, solve a few of these queries. So, and then I want you then we will realize one particular feature of relational algebra and see that certain things cannot be done in relational algebra.

So, I will present to you a series of queries and I encourage you to kind of keep the schema diagram ready with you. So that you know you can try and formulate relational algebra queries for these examples, and then we will see how exactly you will do okay.

(Refer Slide Time: 02:26)



Example Queries

Schema

Retrieve the list of female PhD students

$\sigma_{degree = 'phD' \wedge sex = 'F'}$ (student)

Obtain the name and rollNo of all female BTech students

$\pi_{rollNo, name}$ ($\sigma_{degree = 'BTech' \wedge sex = 'F'}$ (student))

Obtain the rollNo of students who never obtained an 'E' grade

π_{rollNo} ($\sigma_{grade \neq 'E'}$ (enrollment))

is incorrect!!

(what if some student gets E in one course and A in another?)

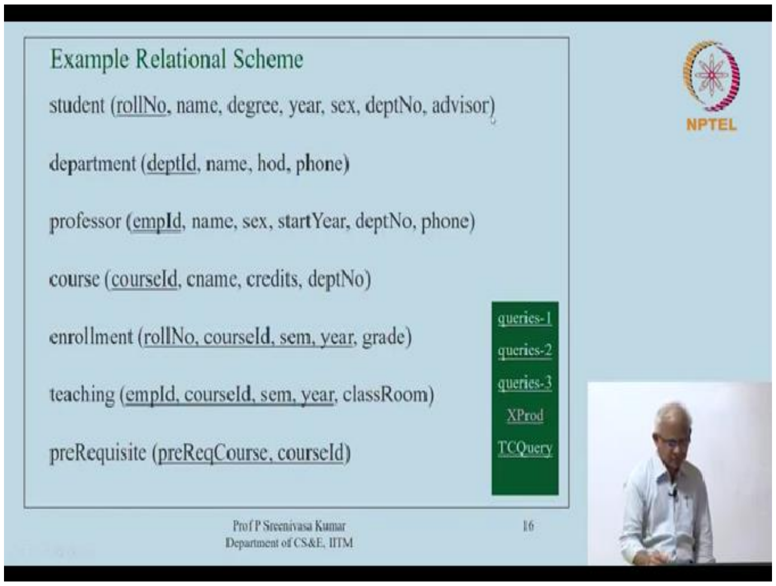
π_{rollNo} (student) – π_{rollNo} ($\sigma_{grade = 'E'}$ (enrollment))

Prof P Sreenivasa Kumar
Department of CS&E, IITM

43

So, here is very simple query. So, please try to do this, we try the list of female PhD student. So, if you want to see the schema, it is here.

(Refer Slide Time: 02:41)



Example Relational Scheme

student (rollNo, name, degree, year, sex, deptNo, advisor)

department (deptId, name, hod, phone)

professor (empId, name, sex, startYear, deptNo, phone)

course (courseId, cname, credits, deptNo)

enrollment (rollNo, courseId, sem, year, grade)

teaching (empId, courseId, sem, year, classRoom)

preRequisite (preReqCourse, courseId)

queries-1
queries-2
queries-3
XProd
TCQuery

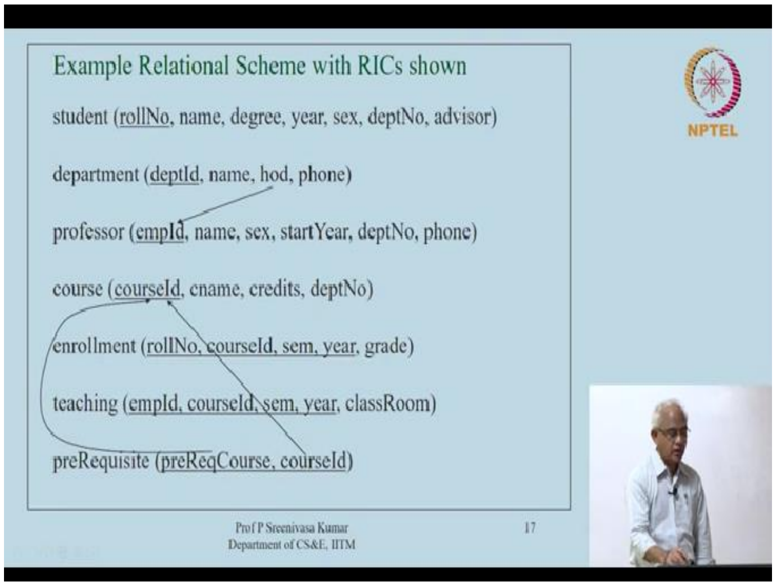
Prof P Sreenivasa Kumar
Department of CS&E, IITM

16

This relational schema is here and the query was retrieve the information about female PhD students. So, under normal circumstances I will go around and then look as to what you are doing in your own book but I do not know whether. So have this schema with you on one page of the notebook, so that you can refer to it. So, any way the general information is here and then the degree information is here.

So, all that you the information that you need for this particular query happens to be there in one this one.

(Refer Slide Time: 03:16)



Example Relational Scheme with RICs shown

student (rollNo, name, degree, year, sex, deptNo, advisor)

department (deptId, name, hod, phone)

professor (empId, name, sex, startYear, deptNo, phone)

course (courseId, cname, credits, deptNo)

enrollment (rollNo, courseId, sem, year, grade)

teaching (empId, courseId, sem, year, classRoom)

preRequisite (preReqCourse, courseId)

Prof P Sreenivasa Kumar
Department of CS&E, IITM

17

So, let me go back to that, retrieve the list of female PhD students. So, this can be done as opposed to would have done it. So by just doing a selection on appropriate attributes, so selection with so in the student relation, we have all this information. So all that you have to do is to impose the conditions that the degree should be PhD and sex should be F, this is also similar, obtained the name and roll number of all female BTech students.

Where is the so it is also there in the student relation name and roll number of all female BTech students. So, this is also there in the student relation. So, only thing is here we did not. So since the query was not specific saying that you know get only this information about students, we just got we you know finished this with just the selection without any projection okay.

Whereas here since this query is very specific saying that get the name and roll numbers of all female BTech students we are also what they need to project the appropriate attributes pi roll number, name and then degree course BTech sex equals F. Now obtain the roll number of students who have never obtained in E grade. To try this out, where is this information about. So we are asking for roll numbers of students.

And who have never obtained a E grade. So this information is there in what is the relation in which this information is there enrollment, right. In enrollment it is there. Try it out, the query is asking for roll number of students who have never obtained an E grade, please write the query in your notes, I hope you have been reading along. So, what does it mean to say that they have never obtained any E grade. I think it is simple right.

So, in whatever courses they have done so far information is there, they have never obtained in E grade anything is everything is other than E and the enrollment has roll number, course Ids semester year and grade. And recall that you know for the current semester courses the grade will be null and for all the completed courses the appropriate grade will be there. Now, let me see what you have done.

How many of you have done this, just raise your hands if you have written it like this, raise your hands one. So, you have not written anything, is that the case or have you written something in your notes okay, how many of you think that this is solving the query, what is it saying grade is not equal to E and from the enrollment and project the roll number right.

The roll number of students who never obtained any grade at that is what we want, how many of you think that this will solve this problem, how many of you think that it will not solve the problem 1 2 3 okay, several of you know that it does not solve the problem. Why does not it solve the problem here is the roll number of students who have obtained. So, what if the student has done several courses and one of them he has obtained in E grade.

And in some in another of the courses is obtained a non E grade, if he has not obtained non E grade at least one non E grade the roll of the student will turn up here in the result of this query, and this is not what we are looking for. We are looking for students who have never obtained an E grade. So if a student has obtained at least one grade which is not E that students roll number will come here.

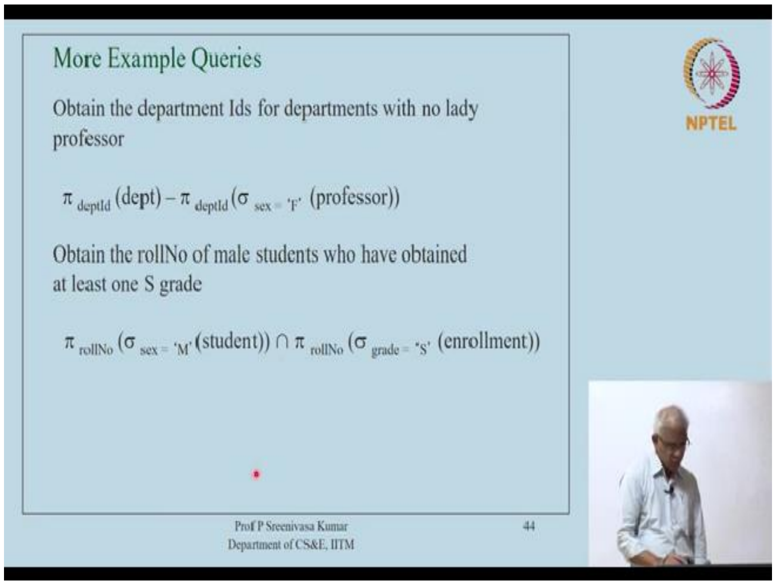
Whether or not he has obtained an E grade or not right. So, this is not solving the query. So, the roll number of students who are never obtained an E grade is not being so. So, how do you proceed with this. This is not correct, this is incorrect right. So any ideas of how to correctly solve this. So, basically we have to get hold of the people who have obtained an E grade in at least one course that is easy.

Because we can do a great equal E, and then get all of the students who obtain in E grade in at least one course, then we know that these are the people's we do not want. These are the people who we do not want. So we know whom we do not want we can set up the people whom we want by doing what set difference, we can do a set difference. So, you can get the set of students who have obtained E grade in at least one course doing grade equals E on enrollment.

And get the roll numbers and then from the set of all students, so we give the benefit of doubt for the students who are currently registered and all that so null grade and all that. So okay, we ignore that. So we from the set of whole roll numbers, all the roll numbers we subtract this roll number of students who have obtained an E grade then we will get the correct roll numbers, the roll numbers of people who have never obtained an E grade okay.

So even though sometimes the statement might look very simple, it might you know, a simple solution you are not currently think about it and then get the answer. Good okay let me proceed with some more.

(Refer Slide Time: 11:39)



More Example Queries

Obtain the department Ids for departments with no lady professor

$\pi_{deptId}(dept) - \pi_{deptId}(\sigma_{sex = 'F'}(professor))$

Obtain the rollNo of male students who have obtained at least one S grade

$\pi_{rollNo}(\sigma_{sex = 'M'}(student)) \cap \pi_{rollNo}(\sigma_{grade = 'S'}(enrollment))$

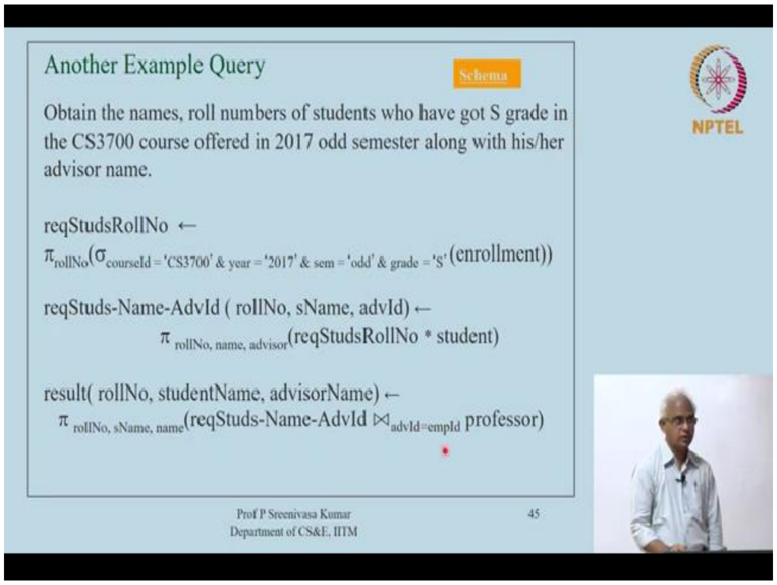
Prof P Sreenivasa Kumar
Department of CS&E, IITM 44

This is simple. So you can basically do the obtain the department IDs for departments who have no lady professors. This is very similar. Also try it out. Very similar to the existing one, the previous one. You can find out the departments that have a lady professor and then do a subtraction again. So, this is the set of departments that have at least one lady professor and here is the set of all departments and you can subtract.

This is easy try it obtained roll number of male students who have obtained at least one S grade. This kind of at least one, this kind of an existential check, right. So, whether that student satisfies some condition or not. So, this is easy to enforce. So, you will be able to handle that by just okay in this case I have wanted to demonstrate that we can use intersection to do this okay, the information about grades is in enrollment, whereas the information about gender, gender is not the student.

And so you have to do selection there, selection here and project roll number so that they are union compatible and then to the intersection okay.

(Refer Slide Time: 13:38)



Another Example Query

Schema

Obtain the names, roll numbers of students who have got S grade in the CS3700 course offered in 2017 odd semester along with his/her advisor name.

reqStudsRollNo $\leftarrow$

$\pi_{rollNo}(\sigma_{courseId = \text{'CS3700'} \,\&\, year = \text{'2017'} \,\&\, sem = \text{'odd'} \,\&\, grade = \text{'S'}}(enrollment))$

reqStuds-Name-AdvId (rollNo, sName, advId) $\leftarrow$

$\pi_{rollNo,\, name,\, advisor}(reqStudsRollNo * student)$

result(rollNo, studentName, advisorName) $\leftarrow$

$\pi_{rollNo,\, sName,\, name}(\text{reqStuds-Name-AdvId} \bowtie_{advId=empId} professor)$

Prof P Sreenivasa Kumar
Department of CS&E, IITM

45

So, here is another interesting query okay. This you can try it out actually this I just wanted to illustrate that, in some situations like this, you need to get information from multiple relations and then you know, obtain our step by step way of writing the relational algebra query. So obtain names and roll numbers of students who have got S grade in CS3700 course offered in there is notice here all the only thing is that it is the information that we need is spread over multiple relations. So you will have to pull it from multiple relations.

And then appropriately take care of renaming etc. offered in 2017 on semester along with his or her advisor name .So, what are all the relations that we need to answer this question. Since we need names and roll numbers of students we need of course the student relation and who have got S grade and so on so of course, and so we need enrollment relation and along with his or her advisor name.

So advisor is a professor and advisor name is required okay, advisor Id is present in the student relation which is a foreign key. So, advisor name is not present in student relation and so again so you will need the professor relation to pull the advisor name into the advisor name okay. So, let me illustrate the process of you know generating these temporary relations and then naming them appropriately.

Again we have done this in an example of here. So, required students, roll numbers is from the enrollment, I will find out the set of students who have obtained an S grade in the CS3700 course in the year 2017 and semester odd. Project the roll numbers do a selection on enrollment and project programs, so this is the required roll number students, but then we know need their names.

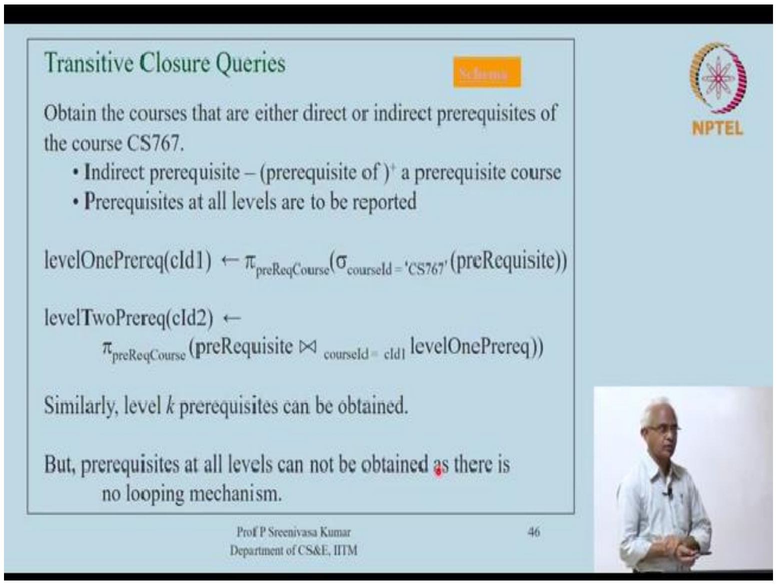
We only have roll numbers, we need their names and we also need their advisors names. So require students name and advisor Id. So I appropriately named it this temporary relation. So roll number, we rename this roll number, name and advisor okay, I will take this required student roll numbers and then do a natural join a student because there is a common attribute, which is his roll number.

Here there is only one attribute. The schema has only one attribute whether the student has a lot of attributes and roll number is the common attribute. So we can do a natural join, to join and then pick up the information about students that matches the required students roll numbers and project name of the student and advisor and the advisor is an advisor Id right. And so, we now we rename these things because so essentially the attribute name will contract with the attribute name in employee and then the professor relation.

And so we will rename it as sName. So finally, result has roll number, student name as required and advisor name right along with his advisor name and so we have advisor name, and then how do we get that this required student advisor Id join with professor on the join condition advisor Id is equal to employee Id right. So if you join with an advisor Id equals employee Id we will get appropriate combinations from this roll number name advisor.

So the advisor details we need now, so as well as details are available in professor relation and so we do a join with on this advisor which is need renamed as advisor Id. So advisor Id is employee Id that is a joint condition and then finally we project the SName which comes from this and name which comes from the professor relation. Finally get me things. So this style of writing is something which I want you to adopt okay.

(Refer Slide Time: 18:57)



Transitive Closure Queries

Schema

Obtain the courses that are either direct or indirect prerequisites of the course CS767.

- Indirect prerequisite – (prerequisite of)$^+$ a prerequisite course
- Prerequisites at all levels are to be reported

$$\text{levelOnePrereq}(\text{cId1}) \leftarrow \pi_{\text{preReqCourse}}(\sigma_{\text{courseId} = \text{'CS767'}}(\text{preRequisite}))$$

$$\text{levelTwoPrereq}(\text{cId2}) \leftarrow \pi_{\text{preReqCourse}}(\text{preRequisite} \bowtie_{\text{courseId} = \text{cId1}} \text{levelOnePrereq}))$$

Similarly, level *k* prerequisites can be obtained.

But, prerequisites at all levels can not be obtained as there is no looping mechanism.

Prof P Sreenivasa Kumar
Department of CS&E, IITM 46

So let us proceed with one more interesting query, this is I call the transitive closure kind of queries. So, let me show you how what this query is. So, we will focus on the prerequisite relation, there is one prerequisite relation if you prerequisite it has prerequisite course and course Id. So if data structures is the prerequisite for databases systems so there will be data structures course Id here and then databases course Id as a tuple okay.

So now the query here asks for the direct or indirect prerequisites of the course CS767. What are direct, so you would understand this direct all those which are available as directly prerequisites of CS767 okay, and now those courses might have some other prerequisites. So those are the indirect prerequisites of 767. So the indirect prerequisites or prerequisites of a prerequisite course for CS767 okay.

Now, so we want all these kind of prerequisites at all levels to be reported. You can imagine that **7** 767 is one course and its prerequisites are there at one level and the prerequisites of that those courses are there at the second level, etc. So, we want all these prerequisites to be reported that is what is in the queries asking for obtain the courses that are either direct or indirect prerequisites of the course 767.

So, let us attempt how to solve this. So, level one prerequisites, how do we get level one prerequisites. So, you take the course IDs 767 is given to us and then do a selection on prerequisite relation, then all the rhos correspondent, which have 767 as the course Id will be selected and then you project the prerequisite courses from there. So, those are the first level prerequisites for the course 767 say 767.

So now we are renaming this as level one prerequisites and with cId 1, rename this attribute as cId 1. I hope you get the idea now. So our idea now is to take this and then how do you get the second level ones. How do you get the second level prerequisites, can you get it from of course this will be useful, right but then what else is needed, my question is how do you get the next level prerequisites using relational algebra opera, where is that in which relation has that information.

There is only one relation right here. We have concerned about all information is there in only one relation, the prerequisite relation, that is all. So how do you use the prerequisite relation now, again, is the question. So how to use the prerequisite relation along with this now join. okay, good. So you have to join this with the prerequisite again what is the joint condition. Let us try and figure out, what is the joint condition.

What should cId 1 be compared to course Id that is good. So, if you combine again do another do a join now with prerequisites and then enforce this join condition that course it is now equal to cId 1 then basically what we are doing now is to and then project the prerequisites, we are now obtaining the level 2 prerequisites right. So, level 2 prerequisites are like this. So, take the prerequisite and then join it with course Id equals cId 1 with this level one prerequisites.

So, the level 1 prerequisites is a small bunch of courses that are prerequisites of **7** 767. So now the course it should match with one of these things, if it matches, then those rhos will contain the prerequisites of the prerequisites of 767 right. Now, that is good, but then obviously, you can do it for level 3 also in a similar way you can do it for ;level 3 also, but then how many levels are there, how do you know how many levels okay in this particular example, you might say that okay.

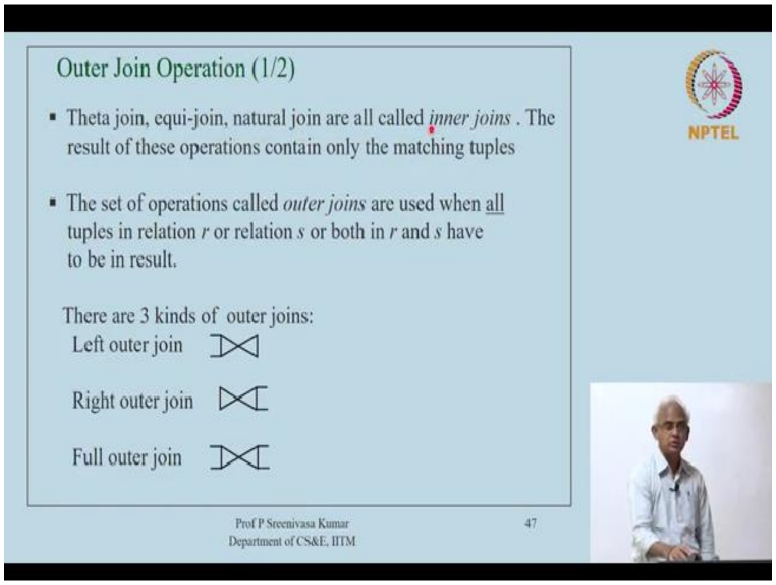
There are only 8 semesters in BTech course then probably there cannot be more than 8 levels. And so I will go and then do this 8 times in them can be, but in general the spirit of this question is that for example, to illustrate the same concept, in Elmasri and Navathe book, they use the employee and supervisor. They give one particular person from John and ask for all the people who are either directly reporting to John or indirectly reporting to John supervised by John or directly supervised by John or indirectly supervised by John.

So, in such cases like this, it is not normally possible to put a appearing we may not know the number of levels. So, if you know the number of levels as k, level k with prerequisites can be obtained. Repeat this k number of times you will get level k prerequisites but in general, we can solve this query, because this is in some sense is asking for the transitive closure of this particular binary relation called prerequisite of.

Something is a prerequisite of something else and it is the transitive relation. And so, you would have studied transitive closures in the script, biometrics right. So, this query is indirectly asking for the transitive closure of the binary relation. So, that kind of a thing is not really possible for relational algebra, because it does not have any looping mechanism basically, the prerequisites at all levels be cannot be obtained as there is no looping kind of mechanism okay.

So, this is a limitation of relational algebra framework and will later on see when we discuss SQL whether we can get over this okay. So I suppose the concept is clear that we cannot do transitive closure kind of curious okay.

(Refer Slide Time: 27:19)


Outer Join Operation (1/2)

- Theta join, equi-join, natural join are all called *inner joins* . The result of these operations contain only the matching tuples
- The set of operations called *outer joins* are used when all tuples in relation *r* or relation *s* or both in *r* and *s* have to be in result.

There are 3 kinds of outer joins:

Left outer join $⟕$

Right outer join $⟖$

Full outer join $⟗$

Prof P Sreenivasa Kumar
Department of CS&E, IITM 47

So moving on I want to briefly talk about other kinds of joints that are also kind of relevant in practice and we will see. So, these joints that I have discussed so far theta join, equi joint, natural joint, they are all called the inner joins. Because the result of these operations contain only the matching tuples okay. Now sometimes matching means the condition the 2 tuples that satisfy the joint condition from the left operand, the right operand, the combination that satisfies the joint condition.

The only those will be available in this choice, in some situations, you know, we want to take a relation and retain all the tuples in that relation whether or not they satisfy the joint condition, you know are in combination with some tuple or not, we want to retain all the tuples in one left hand side, let us say and, you know in case a particular tuple joins with the tuples on the right hand side relation and the joints means it satisfies the joint condition.

Then we want to kind of paired up with the attributes of the appropriate tuple okay, and if it does not, we will only the keep null values for all the other attributes in the joint, so, that kind of joints are what are called outer joints. So, when all tuples in relation r or relation s are both are needed in the result. So, they are these 3 kinds of outer joints. So, what is called left outer join which has the joint symbol with these little lines on the left hand side.

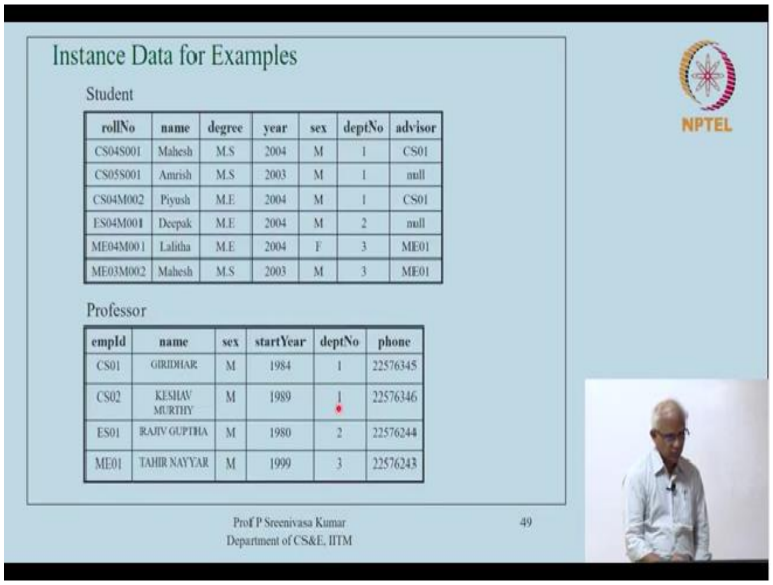
Whereas the right outer join has the joint symbol with these things in the right hand side and the full outer join has a funny looking symbol like this okay, so, let us look at some examples of this okay. So to define left outer join, so basically r left outer join s what it does is keeps all tuples in the first or the left relation r, this relation in the result and for some tuple t in r if no matching tuple is found in s, then all the s attributes of the tuple t are made null in the result.

So, what is the scheme of the result, it is the scheme of the left hand side relation and the scheme of the right hand side relation concatenated together right. So, there will be some s attributes, there will be r attributes followed by s attributes. So, every tuple in the result there will be r attributes followed s attributes. So, if the one tuple from the relation instance r matches say one or more tuples from s then those combinations will be there in the result okay.

And if some tuple does not match with any tuple in s, then what we do is we simply paired up with null values and then keep the tuple there anyway. So, that is what they left outer join is so, it keeps all the tuples in the first irrespective of whether they combined with the tuple in s or r, in a similar spirit the right of the joint what it does is to do the same kind of thing for the right hand side at things.

So, if necessary they all attributes are all made null in this case, in a full outer join, what happens is tuples from both relations r and s are there in the result, but then, you know, in case the they do not match with appropriate tuples from s, then they will be simply padded with null values, the appropriate attributes will be paired with will be simply made as null values okay.

(Refer Slide Time: 31:57)



Instance Data for Examples

Student

rollNo	name	degree	year	sex	deptNo	advisor
CS04S001	Mahesh	M.S	2004	M	1	CS01
CS05S001	Amrish	M.S	2003	M	1	null
CS04M002	Piyush	M.E	2004	M	1	CS01
ES04M001	Deepak	M.E	2004	M	2	null
ME04M001	Lalitha	M.E	2004	F	3	ME01
ME03M002	Mahesh	M.S	2003	M	3	ME01

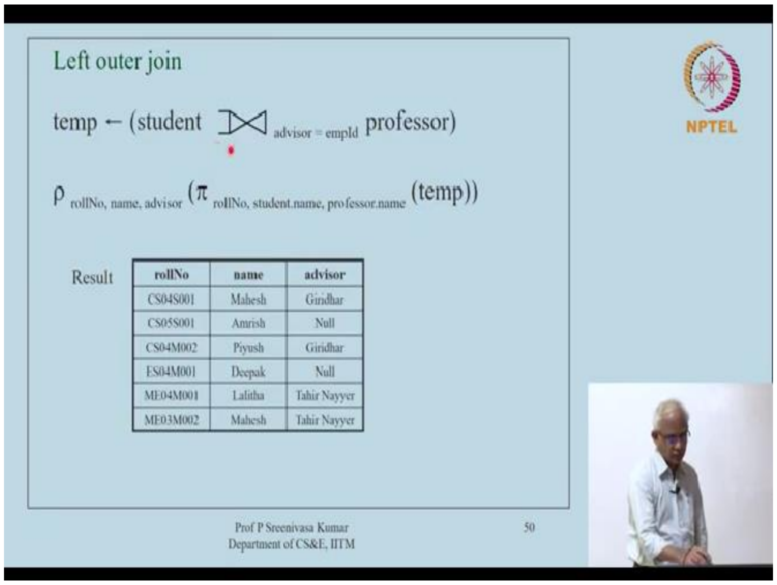
Professor

empId	name	sex	startYear	deptNo	phone
CS01	GIRIDHAR	M	1984	1	22576345
CS02	KESHAV MURTHY	M	1989	1	22576346
ES01	RAJIV GUPTHA	M	1980	2	22576244
ME01	TAHIR NAYYAR	M	1999	3	22576243

Prof P Sreenivasa Kumar
Department of CS&E, IITM 49

Here is some instance data for the examples, so has the same bunch of students, Mahesh, Amrish, Oiyush, Deepak, couple of Maheshes and all Giridhar same of okay, now look at this situation. Some students do not have advisors okay.

(Refer Slide Time: 32:36)



Left outer join

$$\text{temp} \leftarrow (\text{student} ⟕_{\text{advisor} = \text{empId}} \text{professor})$$

$$\rho_{\text{rollNo, name, advisor}}(\pi_{\text{rollNo, student.name, professor.name}}(\text{temp}))$$

Result

rollNo	name	advisor
CS04S001	Mahesh	Giridhar
CS05S001	Amrish	Null
CS04M002	Piyush	Giridhar
ES04M001	Deepak	Null
ME04M001	Lalitha	Tahir Nayyer
ME03M002	Mahesh	Tahir Nayyer

Prof P Sreenivasa Kumar
Department of CS&E, IITM 50

Now so student left outer join, advisor equals employee Id. So we want to keep all students, we do not want to throw away students and in case the advisor is present we want to get hold of the name of the advisor, roll number, name and name of the student and advisors name. That is what we want. So professor name, student name, the roll number. So in some situations, it is not matching.

And so if you go back to the data, so for this Amrish there is no matching tuple, the null does not match with anything. There is no matching tuple, the same is the case with Deepak, there are no matching tuple. So we still want to report all the students and if it matches, if, for example, CS01 Giridhar is the advisor for 2 students. Mahesh and Piyush and so we get him here like that, okay. So I suppose in some situations where we want to keep the information in one relation intact.

And then attach the appropriate information from the other relation, this is kind of very useful okay, so moving on we have the same case. Now we want to do with professors, I want to keep all the advisor names. And in case they are advising some students, we would like to get the details of the students, in case they are not advising any students will simply put null.

So that at least the head of the department knows that who are all not advising students. So they will try to assign some students to those faculty. Good so, and in a similar way you can go through these things, the full outer join will basically what the left outer join was doing for the student and the right outer join was doing for the professor it will do both okay. So we get all students and listed out here and all advisors listed out here.

So in case the student does not have any advisor, it will be null value, in case the advisor faculty member does not have any student it will be null values here okay. So this is called full outer join okay. So okay. So to kind of sum up, what we have done in the relational algebra is to look at the various operators. We have introduced selection projection, cross product okay. Since cross product we have introduced but then we realized that cross product is almost always followed up by selection.

And so there is a combined operation called it join and then we have various kinds of join equi join, natural join equi join all that the division operator and now the outer joins. So with this we have kind of completed the set of algebra operations, what I want you to do is to practice expressing queries using these operators okay. And we will conduct a tutorial one of these days maybe this week or early next week will conduct a tutorial.

But we should not wait for the tutorial. Please practice how to express queries using relational algebra, operators is very, very much important and in some textbooks they also introduce what are called little extended relational algebra operators which involves you know, aggregating data okay. So, that means like for example, on a particular column which has some numerical values.

You want to report the average of that column that is called computing in aggregate on that particular column okay. So, there may be so such kind of things of course, are present in SQL whether we should discuss them in relational algebra context or not is more like a, you know, pedagogic kind of decision. I have chosen not to introduce those operators right now okay. When we come to SQL, we will discuss how to actually do aggregations, and all that.

So, for fundamental understanding of the relational algebra model, those are not very essential. So, we will go ahead with this set of relational algebra operators. So, an important thing that we have realized through examples today is that there are certain kinds of queries the trans to closure kind of closure which cannot be expressed easily let us cannot be expressed in relational algebra operated.

Because it is a it is an expression language. It is not a full-fledged programming language. It is an expression. It is a okay, so, please, but it is wonderful, and it is a very nice that we can indeed represent almost all the queries that we want to express using just about half a dozen kind of operators.