

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

Lecture-09
Relational Data Model and Notion of Keys

Okay, so in the last lecture we started discussing the relational algebra, right.

(Refer Slide Time: 00:27)

Examples of *project* expressions

student

rollNo	name	degree	year	sex	deptNo	advisor
CS04S001	Mahesh	M.S	2004	M	1	CS01
CS03S001	Rajesh	M.S	2003	M	1	CS02
CS04M002	Priyush	M.E	2004	M	1	CS01
ES04M001	Deepak	M.E	2004	M	2	ES01
ME04M001	Lalitha	M.E	2004	F	3	ME01
ME03M002	Mahesh	M.S	2003	M	3	ME01


$\pi_{rollNo, name}(student)$

rollNo	name
CS04S001	Mahesh
CS03S001	Rajesh
CS04M002	Priyush
ES04M001	Deepak
ME04M001	Lalitha
ME03M002	Mahesh

$\pi_{name}(\sigma_{degree = "M.S"}(student))$


name
Mahesh
Rajesh

Note: Mahesh is displayed only once because project operation results in a set.



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

23



So, relational algebra is a central component of the relational model and offers certain operators which operate on relational instances and produce relational instance. And then we talked about 2 important 2 unary operators, they select and project operators okay, a couple of important points before I move on to other algebra operators. See, one of the common mistakes that you know people do when they are in the early stages of learning is you know put some in the condition here in a typical algebraic expression.

In the selection condition, they try to put another selection condition because they produce, they think that that other selection condition produces a value or something like that. So, that will be in a okay. So, for example, let us just consider in this particular data there is only one Mahesh or whatever is that let us say that this particular result, you know, produces only one name okay.

Let us consider the case that this result produces only one name. Even so, even if it produces just one name, it is important to recognize that what it produces is a relation right. Even if it is just a single value, it is actually a relation, which has 1 column and 1 row and 1 value. It is still a relation, it is not to be confused with the just the value okay. So, if this confusion is removed, then you will not have the tendency to kind of treat it as though it is a value.


And then try and put it in the condition here. So the conditions syntax is very fixed in a very much to be. I mean, we discussed this condition syntax for this particular account selection condition, it can only involve atomic operators that have attribute names and comparison operators with values or other attribute names. So, that is the atomic kind of thing. So, we cannot try and put another selection condition I mean equal to sigma something you cannot right here for example, okay.

So, these are certain. So, this point about even though the algebra expression produces a relation with single column and a single row, it is still a relation that has to be clearly understood. It is not to be confused with the value that is present inside that okay. Now with these called points, let us proceed to understand what are the binary operators that are present in relational algebra.

(Refer Slide Time: 04:16)


Set Operators on Relations

- As relations are sets of tuples, set operations are applicable to them; but not in all cases.
- Union Compatibility** : Consider two schemes R_1, R_2 where
 $R_1 = (A_1, A_2, \dots, A_k); R_2 = (B_1, B_2, \dots, B_m)$
- R_1 and R_2 are called *union-compatible* if
 - $k = m$ and
 - $\text{dom}(A_i) = \text{dom}(B_i)$ for $1 \leq i \leq k$
- Set operations** – *union, intersection, difference*
 - Applicable to two relations if their schemes are union-compatible
- If $r_3 = r_1 \cup r_2$, scheme of r_3 is R_1 (as a convention)



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

25



One of the first things that comes to our mind is that since relations are sets of tuples, what about set operations, the familiar set operations, will they be useful for us with a familiar set operations

are the union, intersection and difference right, these are the set operations, but how can we make use of these set operations. One issue that we find here in these relations is that each of these relations, relations instances has a different set number of attributes okay, one relation may have 3 attributes, the other relation may have 5 attributes right.

So, in one relationship tuples are of length 3. In the other relationship tuples are of length 5. So it is not obvious how you can take simply union of those 2 things because if you take a such a union, then the very concept of in producing a relation does not, you know, any more work than it is the union will be union of 3 tuples and 5 tuples. So it is not a relation instance, is just some union, right. It is not a use for us.

We want our algebra operations to produce relations, take relations and produce relations. So it is kind of required that we must put some kind of condition before we can make use of these operators, the union difference and intersection operators. So, that condition is what is called union compatibility. We call this condition is it union compatibility. So basically we are just checking whether 2 relations are you know compatible for applying this union difference and intersection operations.

So, what is the compatibility that we are asking for, it is basically that the number of attributes in so let us consider these 2 relations R_1 with A_1 through A_k R_2 with B_1 through B_m attributes. So, union compatibility basically says that the K should be equal to m . That means that they should be equal number of attributes in both relations and not only that, the domains of corresponding attributes should be same.

So, the domain of A_i should be domain of A_1 should be equal to domain of B_1 , domain of A_2 should be equal to the domain of B_2 , etc. So, if you want to combine take a union of tuples from R_1 and R_2 these attribute names may not matter, but the underlying values should come from the same set, if you take, so, if you take a tuple and consider its first value suppose this is integer and if this is string, then the union does not work out that well right.

So, you want both first tuple to be integer in both this relations and so on. So, you want domain of corresponding attributes to be the same and the number of attributes should itself be same. So, under these 2 circumstance, conditions, if these 2 conditions are satisfied, then we will call that pair of relations as union compatible and then we can talk about producing unions and differences and all that okay.

So, set operations union intersection difference, these are applicable to relations, if 2 relations if their schemes are union compatible, otherwise we will not be able to apply union intersection and difference in our context, because our operators are relational algebra operators, we want them to take relations and then produce relations. So, another issue that comes up is supposing $r_3 = r_1 \cup r_2$ where r_1 or r_2 are union compatible and we are producing.

So, what should be the schema for r_3 , or should it be let us say the schema of r_1 or should be the schema for r_2 right. So, that question comes up. So, what we do is we just simply follow a convention saying that the left operates schema is what we take as a scheme of the result operation okay. See notice that these schemas are compatible, but the attribute names may be different.

So, one set of attribute names we have to take which set is to be decided. So what we do is follow a convention saying that the first operates schema is what we use for the result okay, is that point clear. So just as a convention okay, now let us look at some examples of application of this union practice okay the definitions are very simple.

(Refer Slide Time: 10:04)

Set Operations


r_1 - relation with scheme R_1
 r_2 - relation with scheme R_2 - union compatible with R_1

$$r_1 \cup r_2 = \{t \mid t \in r_1 \text{ or } t \in r_2\}$$


$$r_1 \cap r_2 = \{t \mid t \in r_1 \text{ and } t \in r_2\}$$

$$r_1 - r_2 = \{t \mid t \in r_1 \text{ and } t \notin r_2\}$$

By convention, in all the cases, the scheme of the result is that of the first operand i.e. r_1 .



Prof P. Sreenivasa Kumar
Department of CSE, IITM



So, r_1 r_2 relation schemes with capital R_1 capital R_2 and r_2 is union compatible with r_1 then the operators are defined like this $r_1 \cup r_2$ is the set of all tuples t such that t either belongs to r_1 r belongs to take the tuples that belong to either one of them and then simply take the whole set together. So, it is possible that some tuple exists both in r_1 as well r_2 in which case obviously, only one instance because it is a set right it should not contain duplicates.

So only one tuple will exist. Now, the intersection is different in a similar way, in the usual way and so $r_1 \cap r_2$ is the set of all tuples that belong to both r_1 as well as r_2 and r_1 difference r_2 is the set of all tuples that belong to r_1 and do not belong to r_2 . So, these are the usual definitions of union intersection. So, and by convention in all these cases, the scheme of the result so this is the result relation right. So, is the scheme of the first opera and that is r_1 okay. Now, let us look at another.

(Refer Slide Time: 11:56)

Cross product Operation

r_1	A_1	A_2	...	A_m
a_{11}	a_{12}	...	a_{1m}	
a_{21}	a_{22}	...	a_{2m}	
...				
a_{i1}	a_{i2}	...	a_{im}	
...				
a_{j1}	a_{j2}	...	a_{jm}	
...				
a_{s1}	a_{s2}	...	a_{sm}	

$r_1 \times r_2$	
A_1	A_2 ... A_m B_1 B_2 ... B_n
a_{11}	a_{12} ... a_{1m} b_{11} b_{12} ... b_{1n}
a_{21}	a_{22} ... a_{2m} b_{21} b_{22} ... b_{2n}
...	...
a_{i1}	a_{i2} ... a_{im} b_{i1} b_{i2} ... b_{in}
...	...
a_{j1}	a_{j2} ... a_{jm} b_{j1} b_{j2} ... b_{jn}
...	...
a_{s1}	a_{s2} ... a_{sm} b_{s1} b_{s2} ... b_{sn}
...	...
a_{t1}	a_{t2} ... a_{tm} b_{t1} b_{t2} ... b_{tn}
...	...
$a_{s+1,1}$	$a_{s+1,2}$... $a_{s+1,m}$ $b_{s+1,1}$ $b_{s+1,2}$... $b_{s+1,n}$
...	...
$a_{s+t,1}$	$a_{s+t,2}$... $a_{s+t,m}$ $b_{s+t,1}$ $b_{s+t,2}$... $b_{s+t,n}$
...	...
$a_{s \times t,1}$	$a_{s \times t,2}$... $a_{s \times t,m}$ $b_{s \times t,1}$ $b_{s \times t,2}$... $b_{s \times t,n}$
...	...
$a_{s \times t+1,1}$	$a_{s \times t+1,2}$... $a_{s \times t+1,m}$ $b_{s \times t+1,1}$ $b_{s \times t+1,2}$... $b_{s \times t+1,n}$
...	...
$a_{s \times t+1,1}$	$a_{s \times t+1,2}$... $a_{s \times t+1,m}$ $b_{s \times t+1,1}$ $b_{s \times t+1,2}$... $b_{s \times t+1,n}$

$r_1 : s \text{ tuples}$

$r_2 : t \text{ tuples}$

$r_1 \times r_2 : s \times t \text{ tuples}$

Prof P Sreenivasa Kumar
Department of CS&E, IITM

So, I am actually skipping giving a specific examples for this r_1 because they are very familiar operate as far as the union intersection and difference. We will see that coming up later in other examples okay. Now, moving for the we oftentimes actually want to combine information from one relation with information in other relation irrespective of whether they are actually union compatible or otherwise. Sometimes we need to do that.

Whereas the union allows us to combine information from one relation to and another relation provided their union compatible right. But even if when the not incompatible also, we would like to combine but the different kind of a combination is that actually I want to take a tuple from here and concatenate the tuple from the other relation. Right now what we are doing is taking tuple from one relation and taking tuple from the other relation.

And then simply adding them together into one long list. But instead, let us say I want to take one tuple from here, take another tuple from there and then concatenate the tuples and produce a new tuple, sometimes I may need that. So, we will see an operator that actually allows us to do that. And this is also very familiar operator for us actually, the cross product of sets will recall the definition of a cross product.

So in the context of relations, the let us look at this carefully now, r_1 is a relation on the schema A_1 through A_m okay, and it has certain s number of tuples. So, I have written as $A_1 A_2 \dots A_m$.

So, all the first tuple has the first subscript as 1, the second tuple has the first subscript as 2 and then the remaining thing says 1 2 3 up to m and the last tuple is $A_{s1} A_{s2}$. So, there s number of tuples in the first relation in a similar way r_2 has B_1 through B_m , n number of attributes and it has some t number of tuples.

So, s and t so, r_1 has s tuple as r_2 has t tuple and so, we were allowed to find this cross product, the cross product of r_1 cross r_2 $r_1 \times r_2$ denoted as r_1 you know with this symbol cross r_2 is this relation okay. So what is this relation the scheme of this relation is the concatenation of the scheme of this and the scheme of that. So A_1 through A_m and B_1 through B_m all attributes are present.

And what we do is, you might actually wonder what is this guy doing this is going to produce huge amount of data. Suppose this relation has 1000 tuples, and this original has another 1000 tuples, then you are going to produce a million tuple, tuple in the output. Is it really useful yes, we will see that will come to that point a little later. So, what it basically does is to combine this tuple with every tuple here and produce a tuple here.

This operator, so $a_1 a_2 a_m$ is combining with $b_1 b_2 b_m$ and also $b_1 b_2$ and then this second tuple and the teeth tuple like that. So this produces some t number of tuples, this produces t number of tuples etc, you will get so many number of tuples so. So the operator itself is clear, but then, you know, we might actually wonder how useful is this operator is going to be because it is going to actually produce a huge amount of data okay.

(Refer Slide Time: 16:37)


Example Query using cross product

Obtain the list of professors (Id and Name) along with the name of their respective departments


- Info is present in two relations – professor, department

Schema

- $\text{profDetail}(\text{eId}, \text{pname}, \text{deptno}) \leftarrow \pi_{\text{empId}, \text{name}, \text{deptNo}}(\text{professor})$
- $\text{deptDetail}(\text{dId}, \text{dname}) \leftarrow \pi_{\text{deptId}, \text{name}}(\text{department})$
- $\text{profDept} \leftarrow \text{profDetail} \times \text{deptDetail}$
- $\text{desiredProfDept} \leftarrow \sigma_{\text{deptno} = \text{dId}}(\text{profDept})$
- $\text{result} \leftarrow \pi_{\text{eId}, \text{pname}, \text{dname}}(\text{desiredProfDept})$



Prof P Sreenivasa Kumar
Department of CS&E, IITM



So let us look at an example situation where we will need this cross product. But then we also will notice how exactly the cross product is going to be used in data retrieval queries okay. So obtained the list of professors Id and name, along with the name of their respective departments okay, here is the data retrieval and a query request or query, applying the list of professors Id and name of the professor.

Along with the name of their respective departments okay. So where is this information just I wanted to look at your schema diagram and then tell me where what are all the relations in which this particular information is present. Is it just there in the professor relation itself. In the professor relation, do we have the name of the departments, we do not have the name of the department, we have the name of the department inside in a another relation which is the department relation.

And there is a foreign key in professor that refers was to department relations. And so, you need to actually you know, combine this information and then obtain the name of the department. So the required information is present in 2 relations, the professor and the department with if you want we can go back to the schema and have a look at it but I suppose you have the schema with you.

But let me try whether this works. And then we will have to apparently go to the join then and there we are.

(Refer Slide Time: 18:40)

The slide displays an 'Example Relational Scheme' with the following tables and their attributes:

- 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)

On the right side of the slide, there is a vertical menu with the following items: queries-1, queries-2, queries-3, TCQuery, and XProd. The NPTEL logo is visible in the top right corner. At the bottom of the slide, it says 'Prof P Sreenivasa Kumar, Department of CS&E, IITM' and the number '16'. A video inset in the bottom right corner shows a man in a white shirt speaking.

So, if you just want to have a look at the relation scheme, we have it here. So, you can see that professor has name and department number alone. So, if you want to know the name of the department, it is present in the department. So that is why we need to combine these 2 things and so and you can see that these are different relations and so there is no question of taking any union or anything like that it does not work that way.

So how it works is that I should consider a across product of these 2 things. That is the only way I can combine information from this to that, I can take a tuple from here and then concatenate the tuple from there and then see what does I can do with that. You can see that issue. So that is where as the cross product, but you also see that we are doing unnecessary work.

Because we are combining a professor with all the 10 departments and 14 departments, which are actually unnecessary okay. So now, let us proceed. So I will kind of solve this. Let us see how we can do that. So let me project another thing is that okay. So let me illustrate several points, you know that come up here with this thing notation also. So we are taking the professor, table, I mean relation, sorry.

And then projecting employee Id, name and department number because these are the only things that were relevant for us right now, in this particular query. We want ID and name and department number of from the professor. So let us, so this particular thing where I on the left hand side, I given new relation name, and then give us a bunch of exactly the same number of attributes as I am projecting here is an important construct, you know.

This is called the renaming of attributes, we are renaming the attributes like this. So, this particular operator the project, the project operator basically takes the employee Id, name, employee Id name and department number of professor relation and projects them out right and discard the rest of the attributes. So, it produces a sub tuple for every tuple produces the sub tuple consisting of only values from these components values from these attributes right.

So you get since it contains the employee Id which is a key for professor, you will get as many tuple as there are in the professor relation okay, so you get all of them, but only thing is here, I am renaming them like this, we will see why exactly we do that, but then I want to illustrate I want to exactly you know, define the process of renaming okay. So, we put a new relation name here and give you even sometimes the same names and then you know use this operator.

This operator is can be thought of as assignment operator, assignment operator in my view should be always a symmetric operator should not be looking symmetric like equal to and things like that. So, I use this arrow like this to say that result will be assigned to this new relation. And in the process the attributes will be renamed as these entrepreneurs okay, so that is what we are about.

In some textbooks, they also introduce a new, they introduce an operator for doing this. It is called a rho operator, okay. And they use that, but instead we will follow this convention in our slides that whenever we want to remain, we will. And also this kind of this relation you know stands for some kind of an intermediate result in our formulation of the solution for this query.

So, okay, let us proceed and then I will come back to this if necessary. So and in a similar way, I am producing a new relation called department detail. And they were since I am interested in the

name of the departments. So I am projecting department Id and name from the department relation and renaming them as the dId and dname. So, the name is getting renamed as dname here, the names of these attributes are important. So, remember that.

So, the name is getting replaced by d and then we have a now we can, now that we got the, you know, relevant information from both professor and department, we can actually combine them and produce a new relation called prof department using the cross product okay. So once we do this, we have tuple from this and tuple from this you know concatenated. So a tuple from this will get concocted with tuple from the other relation.

And we have the bunch of us, obviously, we are not interested in all those combinations. We are interested in meaningful combinations of this particular cross product. What are meaningful combinations in this case. In this case, if for a combination to be meaningful a tuple concocted with another tuple here for it to be, you know, meaningful in our context, what is the condition it should satisfy. Both of them should have the should refer to the same department right.

If I take a professor from computer science department and then take a mechanical engineering department tuple and then simply combine, that is not going to be useful for me at all right. So, for this combination of tuple from this to this to be useful, we would like both the tuple to refer to the same department and how do we ensure that that simple because we have the department number here and department Id here.

So, we can impose a condition saying that they should be equal to each other. If we impose that condition, then we will get meaningful combinations of these tuples. So that is what we will do immediately. So, desire the professor department tuple or those so, we can now use the selection operator to get the appropriate combinations. So department number equals dId. So department number exists in this particular.

So what is the scheme of this prof department schema, the prof department is eId, pname, department number dId, dname, the concatenation of these 2 schemes. And so in that scheme, these are both attributes. And that is why I had to do the renaming. Because if I do not do the

renaming, this name attribute is common to both relations, right. So if you combine the information from this and this.

Then we should ensure that the attribute names are different, right. So you cannot have the same attribute name occurring 2 times in a relation scheme. That is not allowed. So, we did the appropriate rename, we call it pname here and then we call it dname here. So that when you take the concatenation of these schemas, we get a proper schema. And then now we are applying the selection operator to get to impose this condition to.

So that we get meaningful combinations of tuples from the cross product. So this particular phenomenon, of you know taking the cross product of 2 relations and selecting the required combinations occurs very often occurs almost always, this is the case where we will use the cross product. And so there is a strong motivation to actually combine these 2 kind of things into 1 operator.

So, we will see that actually in a short while now okay. Now finally, to get the result of this particular to get the result obtained the list of professors along with the name of their respective departments, all that we have to now do is to take this desired professor department project eId, p name and dname right. So exactly as to how we done the, now notice one thing that if you have the operator for the let me try okay.

First let me take any questions on this one. Is this clear, do you want me to go over this again. Is this clear. First thing is, we are renaming the we are projecting desired attributes from professor and renaming them and then getting an intermediate relation called professor detail. And then we are taking relevant attributes from department and projecting the relevant attributes from for department and it giving, producing a intermediate relation called department detail with these 2 names.

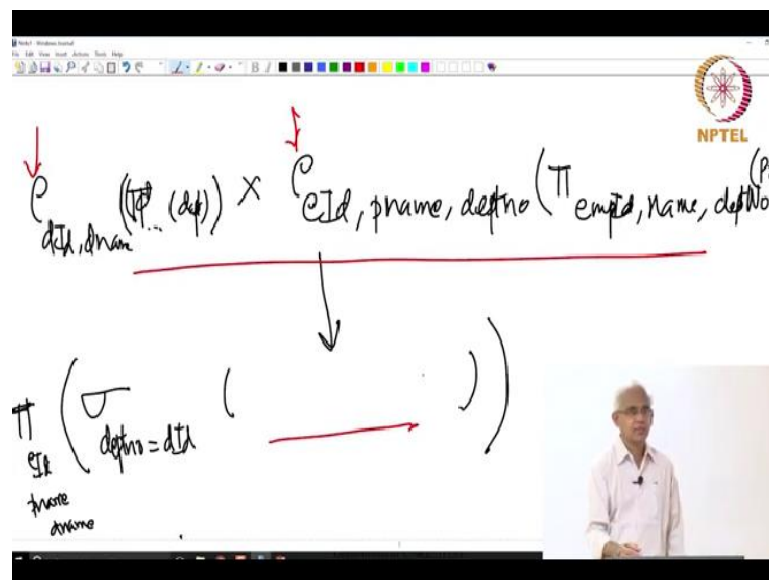
And then we are doing a cross product. And since the cross produces a lot of combinations of tuples, which are not really does a meaningful. We are going to impose this condition that department number should be equal to department Id and only such combinations are meaningful

for us and we are calling that as desired professor department and then from there we are projecting what are the attributes that we actually require the eId, pname and dname.

That is how we will design. Now, this style of writing a query is kind of desirable because, first thing is that we are able to organize our thinking and then produce some meaningful names for the intimate relations. These are the intermediate relations and we are giving some meaningful names for those intimate relations. And then finally producing one relational algebra expression okay.

Supposing we adopt that particular operator called the rename operator okay. If you have the rename operator how exactly this gets returned I will show you. Let us see, let me try and write that okay.

(Refer Slide Time: 31:15)



If so the rename operator is called rho operator okay and then whatever is the new names we should right here okay, what are those new names eId, pname, and department number. Now so what is written inside is basically this entire expression pi employee nameId, name, department number, professor. So you can imagine that being written by pi the employee Id, name and department number this.

And then we can have professor is the way to move this okay. So, you can imagine this kind of situation. So, this stands for this relation now that expression that I wrote there it is rho operator stands for this particular relation prof detail, so, we did not name it as prof detail okay. Instead what we did is to simply produce it as a expression. Now, this is a rename operator.

So, what it takes is takes a relation and then renames the attributes like this and then gives a new relation okay. So, what we did in this slide in the actual the slides that are showing is to kind of name this entire relation as prop detail, okay. So it is an intermediate thing. In a similar way you can assume you can see that I will produce this department detail in a similar way by using a rho operator rho dId, dname of this particular thing.

Then that will give you an the intermediate relation, then I can do a cross product and so on. So, I can basically take this and then do a cross product and then give that other thing that we had dId, dname from okay, we have pi here and so on. The point I am trying to illustrate here is that we have so inside this there is a pi expression projection expression and here is the rename operator.

And now we can see that this is the cross product okay, the entire cross product. Now after the cross product, we basically lead the selection operator, selection on that cross product okay. So we can you can imagine that this entire thing is inside like this, and then put a selection with the selection condition department number equal to dId and then actually as a final step, we are taking this entire thing and then producing and then finally giving a pi with eId, pname and dname okay.

So, this is actually the relational algebra expression that produces the, that gives us the result for this particular query, obtain the list of professors, along with the name of their respective departments okay. Are you getting the point that this entire thing in fact can actually be expressed as one single expression like this, which has inside this is the cross product, this entire cross product thing okay, let us not bother about that.

So basically pen color I can choose I think right. So this is the entire cross product expression. So I can put that here and then apply the selection, and then finally do the production. So, I get an expression, but as you can see this big expression is kind of difficult to understand. So, it is a good style to write the expression in this way, where we give meaningful names for the intermediate relations, okay.

And then combine and then use so many steps to kind of finally produce the result of that we want okay. So I want you to follow this kind of style of writing where we put sub expressions on the right hand side, assign it to meaningful intermediate relations with appropriate naming for the attributes if necessary. And then combine them and then you know, use one line for each of the operators that we are producing, we are required to use.

And then finally produce the result. So it is kind of easy for you to develop and then, you know, understand the query. And it is also easy for anybody who is looking at this query and understand that, that yes, this query is really producing exactly the kind of result we want okay, so even though theoretically, you can express that entire thing as one big relational algebra expression like this with the help of the rho operator.

The rho operator I did not introduce formally, but there is a rho operator which basically does the renaming okay. So is there any questions now in this particular thing. So, we listed several points here of renaming and then giving meaningful names for intermediate relations and then making use of those intimate relations on the right hand side again and produce new relations and then finally produce the results that we want.

So, with that, let me see whether I can take you a little further, okay. The other point that we notice here is that after the cross product in general you know, producer huge number of tuples and a lot of combinations of tuples from one operator and the other operator and it is generally seen that it is that all the combinations are not meaningful. And so, we most often follow it up by having a selection operator.

And since the combination of cross product followed by selection is such an often occurring phenomenon, what we will actually do is to introduce a new operator and that operator is actually you would have heard about it, it is the join operator. So, we will discuss the join operators and the various variations of join operators in the next lecture.