


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

Lecture-10
Uses of Renaming, Join and Division in Relation Algebra

So in the last lecture I was discussing this query and then I introduced several ideas. So, I kind of added couple of slides I thought I would repeat some of this detail before I proceed further.

(Refer Slide Time: 00:35)



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

Schemas

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

Prof P Sreenivasa Kumar
Department of CSE&E, IITM

28




So, basically I illustrated this issue of you know using renaming and introducing temporary relations while doing the specification of the query.

(Refer Slide Time: 00:51)


Query using *cross product* – use of renaming

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

- $\text{profDetail}(\text{eId}, \text{pname}, \text{deptno}) \leftarrow \pi_{\text{empld}, \text{name}, \text{deptNo}}(\text{professor})$
 - this is a temporary relation to hold the intermediate result
 - “empld, name, deptNo” are being renamed as “eId, pname, deptno”
 - creating such relations helps us understand/formulate the query
 - we use “←” to indicate assignment operation.
- $\text{deptDetail}(\text{dId}, \text{dname}) \leftarrow \pi_{\text{deptId}, \text{name}}(\text{department})$
 - another temporary relation
- Renaming is necessary to ensure that the cross product has distinct attribute names.



Prof P Sreenivas Kumar
Department of CS&E, IITM



So, the use of renaming I have kind of specifically written here that this is a temporary relation to hold the intermediate results. And all these attributes are being renamed as eId, pname and department number. So, we discussed this already and creating such relations basically helps us and understanding and formulating the query make easier manner. And in a similar way we have the other department detail also as a temporary relation.

And we are using this left arrow as a to indicate assignment operation okay this is also another temporary relation. Now, this renaming is necessary, because we want to ensure that the cross product has distinct attribute names, this also I mentioned in the last class.

(Refer Slide Time: 01:51)


Use of renaming operator ρ

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


- One can use the rename operator ρ and write the whole query as one big expression (as an alternative to using temporary relations)

$$\pi_{\text{eId}, \text{pname}, \text{dname}} \left(\sigma_{\text{deptno} = \text{dId}} \left(\rho_{\text{eId}, \text{pname}, \text{deptno}} \left(\pi_{\text{empld}, \text{name}, \text{deptNo}}(\text{professor}) \right) \times \rho_{\text{dId}, \text{dname}} \left(\pi_{\text{deptId}, \text{name}}(\text{department}) \right) \right) \right)$$

- It is easier to understand and formulate the query with *meaningfully named* temporary relations as shown earlier.
- Students are encouraged to use temporary relations.



Prof P Sreenivas Kumar
Department of CS&E, IITM



Now, the operator rho is a renaming operator and I tried to write this big expression using a the journal here right, but I do not know whether you have been comfortable seeing that thing. So, I read under written in this slide here. So, one can use the rename operator and write the whole query as one big expression as an alternative to using this temporary relations.


So, how it looks is like this okay. So you can now make use of, so what we are doing here is that, so, this is the professor relation that we have and then we are projecting the required attributes from here and then renaming them using this rho operator renaming them and then doing a cross product with the other. This is what we call us department detail in the in the previous slide, whereas this is the one which we call us professor detail right.

And so we are doing a cross product of these 2 things, and then selecting the relevant combinations by giving the selection condition department number equals dId, and then finally projecting eId, pname and dname attributes from this entire relation. So, these brackets seem to have modes here and there in this slide, but this bracket closes this whereas this one closes that right.

So, one can write it as a big relational algebra expression like this, but then you can see that it is easier for us to understand the and formulate the query if we use this meaningfully named temporary relations. So I would encourage you to do that students are encouraged to use the notation of introducing the temporary relations and then writing the sub expressions and then formulate the query okay.

So, I hope these points are clear. Now, let us move on to the need for other operator which is called the joint operator I just mentioned in the last class.

(Refer Slide Time: 04:21)




Join Operation

- *Cross product* : produces all combinations of tuples
 - often only certain combinations are meaningful
 - cross product is usually followed by selection
- *Join* : combines tuples from two relations provided they satisfy a specified condition (join condition)
 - equivalent to performing *cross product* followed by *selection*
 - a very useful operation
- Depending on the type of condition we have
 - *theta join*
 - *equi join*

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

31



This is a very important operator, mainly we noticed that whenever we use cross product, we are generating you know combinations of tuples from 2 relations and most of the time all the combinations will not make sense. Only certain combinations will make sense and so we need to more often than not follow it up immediately with a selection condition to select the appropriate combinations of tuples from the 2 relations.

So, if you are doing it very often, why do not we introduce an operator for that that is the thinking before for this joint operation. So, since cross product produces all combinations of tuples, and since only certain combinations are meaningful, we have to follow it with using selection. And so we know introduce a new operator called join, what it does is combined tuples from 2 relations combines meaning concatenate.

In this case we will take 1 tuple from here take and another tuple and concatenate the tuples, concatenate tuples provided they satisfy a specified condition, they call the joint condition. So, this is kind of equivalent to performing the cross product followed by selection and it is very useful operation in practice okay. So, depending on the type of the condition that we specify, there are various kinds of joints that we that are kind of talked about. So we will talk about these things called theta join, equi join and natural join various different joints.

And then later on of course, when we go to you know the implementation details about these operators, we will talk about specific algorithms for realizing these joint operations okay.

(Refer Slide Time: 06:28)

Theta join


- Let r_1 - relation with scheme $R_1 = (A_1, A_2, \dots, A_m)$
- r_2 - relation with scheme $R_2 = (B_1, B_2, \dots, B_n)$
- where w.l.o.g we assume $R_1 \cap R_2 = \emptyset$
- Notation for join expression : $r = r_1 \bowtie_{\theta} r_2$
- θ - the join condition - is of the form : $C_1 \wedge C_2 \wedge \dots \wedge C_s$
- C_i is of the form : $A_j \langle \text{CompOp} \rangle B_k$
- where $\langle \text{CompOp} \rangle$ is one of $\{ =, \neq, <, \leq, >, \geq \}$
- Scheme of the result relation r is:

$(A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n)$

$r = \{(a_1, a_2, \dots, a_m, b_1, b_2, \dots, b_n) \mid (a_1, a_2, \dots, a_m) \in r_1,$


$\bullet (b_1, b_2, \dots, b_n) \in r_2$

$\text{and } (a_1, a_2, \dots, a_m, b_1, b_2, \dots, b_n) \text{ satisfies } \theta\}$



Prof P. Sreenivas Kumar
Department of CSE&E, IITM

32



So what is theta join, r_1 is a relation with scheme r_1 equals A_1 through A_m some m number of attributes, r_2 is another relation with scheme B_1 through B_n . And let us assume that without loss of generality we can make this assumption that the intersection of this r_1 and r_2 is null set. There are no common attribute names across the 2, because if there are there, we can always rename it right.

So, we can make this assumption now, there is no loss of generality in making this assumption that this A_1 set is disjoint from this B_1 through B_n , there no common attributes. So r_1 intersection r_2 is phi. So, we will make that assumption, we will actually also as sometimes relax this assumption in the case of natural join as we see it later okay. So, now, we make this assumption that the attribute sets the schemas are disjoint for these 2 relations.

And we express this join with this symbol the join, so the subscript is the joint condition, so, r_1 join r_2 with the joint condition and what is the structure of the joint condition, how should it be. So, theta is the joint condition is of the form we can assume that it is a conjunction of certain atoms and where each of the C_i is of this form that whereas the A_j compared with some B_k , notice

that there is no need for having a condition in which there are attributes of the same relation of compound.

Because we are talking about joint relation, joint condition here. So, attributes from one relation have to be compared with attributes of the other relation. If you are comparing attributes of the same relation, you might as well do it in the selection conditions right, you do not have to do it in the join. So the in the joint conditions, the form will be like, we take an attribute from the left side relation and take an attribute from the right hand side relation and then we do a comparison.


So the comparisons is one of these 6 possible comparison operators. So, we make comparisons and then we can do conjunction. And then and this is a general form of the theta, notice that we do not need to really bother about the form where we are having r_1 and things like that, because a every logical you know, expression can be rendered into conjunctive normal form. So, we know that, so we basically make use of that thing. So, these are conjunctions.

And what is the scheme of the result relation r , the scheme of the result relation r is basically the concatenation of these 2 schemes right. So, $A_1 A_2 \dots A_m$ followed by B_1 to B_n of the scheme and the result relation itself is defined like this. So, the tuples even $A_1 A_2 \dots A_m b_1 b_2 \dots b_n$ we are using here to indicate the values. So, such a tuple such that A_1 through A_m data tuple belongs to r_1 b_1 through b_n belongs to r_2 .

And this combination satisfies the joint condition provided satisfies the joint condition. So, that is the results such that this part of the tuple belongs to r_1 , this part of the tuple belongs to r_2 and it together satisfies the joint condition and what is the joint condition is in terms of these values of these. So, it says A_j equals B_k . So, you pick up the A_j value B_k value and then check the condition and there was a bunch of conditions to be checked. You can do that okay.

So that is what a joint is. So, this is very useful operation in practice and so the most general form of that is called the theta join. And we will oftentimes do not use this very general form, but we will use some specific forms of join and so we have names for those kinds of joints okay.

(Refer Slide Time: 11:18)



For each department, find its name and the name, sex and phone number of the head of the department

Professor

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

Department


deptId	name	hod	phone
1	Computer Science	CS01	22576235
2	Electrical Engg.	ES01	22576234
3	Mechanical Engg.	ME01	22576233

Courses

courseId	ename	credits	deptNo
CS635	Algorithms	3	1
CS636	A.I.	4	1
ES456	D.S.P	3	2
ME650	Aero Dynamics	3	3

Prof P Sreenivas Kumar
Department of CS&E, IITM

33



Okay here is a set of some data that I will use in order to illustrate a join. Let us see. So, we have professor relation and department relation. And the courses relation and professor we have a couple of computer science professors and in electrical engineering and mechanical engineering professor and so on. And so we have here computer science, electrical engineering, mechanical engineering departments.

And then some courses like algorithm A.I, D.S.P and aerodynamics all that okay, so some data is there, so we will just use this data in order to illustrate couple of queries. So now let us consider this for each department find its name, find its name, the department's name, and the name, sex and phone number of the head of the department okay. So, where is that information that information is obviously here in the department relation.

And as in the professor relation, because they hod is an employee, hod is a professor and professors details are here. So you need to join department with professor with an appropriate joint condition and what is the joint condition here. The joint condition is basically that here department is mentioning the hods employee Id. So this employee Id should match the employee Id of the professor tuple only then that particular combination gives us the details about a particular department.

So the matching the joint condition here is this hid should be equal to the employee Id. And so with that we will be able to formulate this query, so we will see how exactly we will do that.

(Refer Slide Time: 13:19)

Example


For each department, find its name and the name, sex and phone number of the head of the department.

prof (emplid, p-name, sex, deptNo, prof-phone)

$\leftarrow \pi_{\text{emplid, name, sex, deptNo, phone}}^*$ (professor)


result $\leftarrow \pi_{\text{deptid, name, hod, p-name, sex, prof-phone}} (\text{department} \bowtie_{\text{hod} = \text{emplid}} \text{prof})$

deptid	name	hod	p-name	sex	prof-phone
1	Computer Science	CS01	Giridher	M	22576235
2	Electrical Engg.	EE01	Rajiv Gupta	M	22576234
3	Mechanical Engg.	ME01	Tahir Nayyar	M	22576233



Prof P Sreenivas Kumar
Department of CS&E, IITM

34



So, before we do that some renaming is necessary because there are some common attribute names between these 2 things. So name is common. The attribute name is common here okay. And what else is common, phone is common. Phone is also a common attribute. So we have to do renaming because we assume that these attribute lists have to be disjoint. So we will do appropriate renaming.

And then so I will introduce a short form of professor as prof and then do the renaming here, employee Id, pname instead of name, I just made it pname and the rest of the things are as they are, and then I rename phone as prof-phone, because then this will be different from the department phone. So it is a prof-phone. So that is how I take professor and then project the required attributes from the professor and then rename them here like this. And then similarly, then I can go straight for the join.

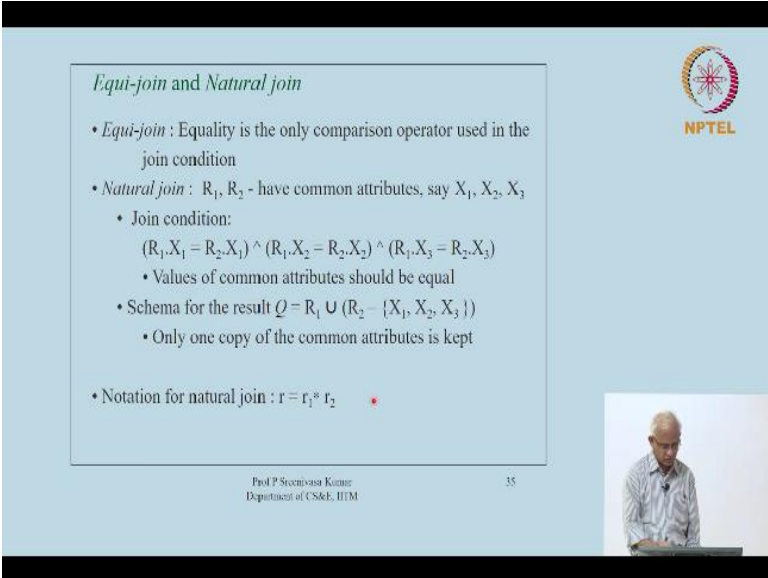
So here I perform department join process prof, which is this temporary relation with the joint condition hod equals employeeId right. And so once you do that, we get the relevant combinations and from the relevant combinations we have been asked to find name or the

department name and other details of the head of the department. So we will project the so even though we just been asked name, we will know project department Id also.

So that, you know, we know that the key of the department is the result. So, department Id name hod, pname for the name of the professor and other details. So, that is the result how exactly this gets computed and all that we will not go into right now we just look at them as mathematical operators right now. But then of course, we have to worry about how exactly to implement these operators at an appropriate time later.

Good. So I suppose this should, so if you do not have join operation, then we just have to we have to do a cross product and then get all possible combinations of department prof and then followed it by a selection condition with this selection thing, right, that is how we would have done without the join operation with the existing the join save some you know, expression here, it simplifies the expression, the presence of joint okay.

(Refer Slide Time: 16:30)



The slide is titled "Equi-join and Natural join". It contains the following content:

- *Equi-join* : Equality is the only comparison operator used in the join condition
- *Natural join* : R_1, R_2 - have common attributes, say X_1, X_2, X_3
 - Join condition:
 $(R_1.X_1 = R_2.X_1) \wedge (R_1.X_2 = R_2.X_2) \wedge (R_1.X_3 = R_2.X_3)$
 - Values of common attributes should be equal
 - Schema for the result $Q = R_1 \cup (R_2 - \{X_1, X_2, X_3\})$
 - Only one copy of the common attributes is kept
- Notation for natural join : $r = r_1 * r_2$

At the bottom left, it says "Prof P Sreenivas Kumar, Department of CS&E, IITM". At the bottom right, it says "35". There is an NPTEL logo in the top right corner and a small video inset of a man in the bottom right corner.

Now moving on, we have other kinds of joints called the equi join and natural joint. So equi join is basically where the in the joint condition is not a generic kind of a joint condition, but it is a very specific kind of joint condition where equality is the only component operative used in the joint condition, that is the case, we call it an equi joint. That is all. Sometimes it is useful because

most of the time we are only doing about value matching. And so, equi joints would be enough for us.

And natural join is where we make a specific assumption saying that r_1 and r_2 have indeed common attributes like say $X_1 X_2 X_3$ is let us say these are the 3 common attributes, okay. And then we do not even have to specify the join condition, so this particular natural joint has its own small operator, star operator asterisk operator. So, where we assume that whatever be the common attributes between the 2 schemas will be simply equated.

That is the joint condition. So the joint condition is always assumed to be like this $R_1.X_1 = R_2.X_1$, $R_1.X_2 = R_2.X_2$, $R_1.X_3 = R_2.X_3$, $X_1 X_2 X_3$ are the only common attributes between R_1 and R_2 and we will put the equality for them and then so values of common attribute should be equal is the join condition. And the results came of a Q will be $R_1 \cup R_2$ minus these 3 attributes because you only need one copy of $X_1 X_2 X_3$.

$X_1 X_2 X_3$ are present in both R_1 and R_2 but we need only one copy because it does not make sense to keep both copies because anyway they will be equal because we have putting this condition right here saying that there has to be equal. So in the result they will be equal. You get burned in the result, the result of a join the values of the X_1 from R_1 and the value of X_1 from R_2 in the tuple will be the same where you are imposing the condition.

So and the notation for the natural join is simply like this, $r_1 \star r_2$ this is asterisk symbol. So, you do not have to specify any condition, the condition is assumed to be this always. If there are no common attributes, then there is almost like you know, you are not specifying any condition. So, it will actually, you know, come down to cross product. There are no things it will be equal to prosper.

(Refer Slide Time: 19:53)


Example – Equi-join


Find courses offered by each department

$$\pi_{deptId, name, courseId, cname, credits} (Department \bowtie_{(deptId = deptNo)} Courses)$$

deptId	name	courseId	cname	credits
1	Computer Science	CS635	Algorithms	3
1	Computer Science	CS636	AI	4
2	Electrical Engg.	ES436	D.S.P	3
3	Mechanical Engg.	ME650	Auto Dynamics	3

Prof. P. Srinivas Kumar
Department of CSE&E, IITM






So, here is an example of equity joined, find courses offered by each department. So, we want to find the list of courses offered by each department. So, each department offers several courses in our example data, we just had only 4 courses and so, 2 of them are offered by computer science department with other offered by electrical and then the third one is offered by mechanical engineering department.

And so when you do a department so this is an example where equi join, so I just want to say that is the quality condition. You cannot perform natural joint here because these attributes are different, you know, they are not the same. So, they even though they are conceptually same, but their actual, you know, the strings are different. So they will be treated as different attributes.

(Refer Slide Time: 20:52)



Teaching


empId	courseId	sem	year	classRoom
CS01	CS635	1	2005	BSD361
CS02	CS636	1	2005	BSD632
ES01	ES456	2	2004	ESD650
ME01	ME650	1	2004	MSD331

To find the courses handled by each professor
Professor * Teaching
result

empId	name	sex	startYear	deptNo	phone	courseId	sem	year	classRoom
CS01	Gindhar	M	1984	1	22576345	CS635	1	2005	BSD361
CS02	Keshav Murthy	M	1989	1	22576346	CS636	1	2005	BSD632
ES01	Rajiv Gupta	M	1989	2	22576244	ES456	2	2004	ESD650
ME01	Tahir Nayyar	M	1999	3	22576243	ME650	1	2004	MSD331

Prof P Sreenivas Kumar
Department of CS&E, IITM

37



So, for example, involving natural join let us look at this. Let us say apart from the professor relation, we also have the teaching relation, which has these details like CS01 employee is offering some courses CS635, 636 like that. And these are the courses being offered in sounds of semesters and what are the classrooms etc. So now to find courses handled by each professor or taught by each professor, what we do is to simply do a natural join between professor and teaching.

Because employee Id is the common attribute. And so we want we can go back to that professor table that had employee Id and of course it has okay, I think it is almost there here right. So professor, employee Id, name, sex, start year, department number, phone. These are the attributes of professor and now in the teaching we have employee Id. So, there will be a natural join that occurs and so we get details about.

So, in case a particular professor is accurate teaching more than one course then the entire details about the professor will be repeated right. In this case, it is not like that so, we get this lesson. So this is natural join okay. Moving on, here is another interesting kind of operator. I mean, called the division operator.


(Refer Slide Time: 22:43)

Division operator

- The necessary condition to determine $r \div s$ on instances $r(R)$ and $s(S)$ is $S \subset R$
- The relation $r \div s$ is a relation on schema $R - S$.
A tuple t is in $r \div s$ if and only if
 - 1) t is in $\pi_{R-S}(r)$
 - 2) For every tuple t_s in s , there is t_r in r satisfying both
 - a) $t_r[S] = t_s$
 - b) $t_r[R - S] = t$


// $t_r[S]$ – the sub-tuple of t_r consisting of values of attributes in S

- Another Definition $r = r_1 \div r_2$
Division operator produces a relation $R(X)$ that includes all tuples $t[X]$ that appear in r_1 in combination with every tuple from r_2 where $R_1 = Z$ and $R_2 = Y$ and $Z = X \cup Y$



Prof P Sreenivas Kumar
Department of CSE&E, IITM

38



It kind of gets used in certain may be specific kind of situations. So here is the details of the division operator, the necessary condition for us to determine the odd divided by s on instances R and S is that the scheme is should be a subset of R proper subset of R ok. So the relation r divided by s is a relation on the scheme R minus S . R minus S okay here is the actual definition before we understand the definition let me introduce this little bit of notation here.

We take a tuple, say t r whatever it is, and then write a subset of attributes are actually subsequence of attributes in this okay, a bunch of attributes. Let us say $A_1 A_2 A_3$, then what that means is these sub tuple of that particular tuple, consisting of values of these attributes, in S okay, so that is the notation, we introduce a notation for that because we oftentimes have to talk about that sub tuple.

So, there is a wide tuple and then we have a subset of those attributes from that tuple. So, we want to talk about that. So, we introduced a notation for that $t_r[S]$, S is a subset of the scheme is the sub tuple of that tuple t_r consisting of values of the attributes in that particular sets are the sequences okay. Now, so to define r developed by s , there are 2 conditions here that it will consist of all the tuples t which $R - S$ part of R okay.

$R - S$ part of the r the first left hand operator, but then they have this very unique property that you know. So, the combination of this tuple which is a tuple of $R - S$ okay. It combines with

each of the tuples that are there in S and exists in R. So, that is actually expressed here that for every tuple t_s in s, there is a tuple t_r in the relation r satisfying that the sub tuple for S is this a sub tuple of t_r for this particular attributes s is indeed this t_s .

And then the $R \text{ minus } S$ is the t that we are looking for okay, t is going to be in the result. Actually, the sub tuple t is the one that is we are looking at we are looking for. So, this is the tuple sub tuple that exists in the result, but this one will have the property that no it combines with every tuple t_s okay. So, if you focus on some tuple t_s . So, we are guaranteed that some tuple t_r exists in r such that the $R - S$ portion of it is this particular t that we have.

And the S portion is the tuple t_s in s okay. So, basically it kind of you know we can also restarted it in this way maybe to help us to understand the definition of have a look at here okay. So the division operator produces a relation R X, now here I am slightly again renaming the schemas that includes all tuples T X that appear in r_1 in combination with a every tuple r_2 of r_2 every tuple from r_2 okay where this you know schemas are appropriately worked out.


So r_1 is some Z is R_2 is this Y, and Z is the union of these 2 things okay, so, I think this needs an example we can have a look at the example and then come back to this definition, so that we can understand this definition better. So, let me show you an example first.

(Refer Slide Time: 27:55)

$R = (A, B, C, D), S = (A, B), X = (C, D)$
 $X = r \div s$

A	B
a_1	b_1
a_2	b_2

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_3	d_3
a_2	b_2	c_3	d_3




C	D
c_1	d_1
c_3	d_3

(c_2, d_2) is not present in the result of division as it does not appear in combination with all the tuples of s in r

Prof P Sreenivas Kumar
Department of CS&E, IITM

59



Let us look at this okay, so let us say r is having attributes $A B C D$ and S is $A B$ and X is $C D$. Okay and now looking at r divided by s . And the scheme for r divided by s is $r - s$ okay. So that means the set of all attributes that are there in r but not in s that means $C D$ right. So, $X C D$ is the output schema of the r divided by s okay, now how exactly this result gets computed, you can have a look at it.

Just let us look at this instance . And so s having 2 tuples $a_1 b_1$ $a_2 b_2$. Now, what are those tuples in the $C D$ part of the relation r that combined with every tuple here and exist here that is the question we are asking. So, get hold of all those tuples in the $C D$ part of this data, which has this interesting property that they combined with every tuple of s and exist here okay.

So, now we can see that among so if you focus on the $C D$ part, C and D part of the relation r , you can see that if you project is here, you will get $C_1, D_1 C_2 D_2 C_3 D_3$ right if you project this relation on $C D$, you will get 3 tuples $C_1 D_1, C_2 D_2, C_3 D_3$, right. So among the $C_1 D_1, C_2 D_2, C_3 D_3$ which of those things have the property that they combined with $a_1 b_1$ and $a_2 b_2$ and exist here.

That is the question we are asking okay, so now we can see that $C_2 D_2$ does not have that property, $C_2 D_2$ combines it only $a_1 b_1$, but does not combined with $a_2 b_2$, $a_2 b_2 c_1 c_2 c_2 d_2$ does not exist in the data here. Whereas $c_1 d_1$ combines with both $a_1 b_1$ and $a_2 b_2$ exist here, whereas $c_3 d_3$ also combines with $a_1 b_1$ and $a_2 b_2$ and exists here only $c_2 d_2$ does not have that property. So in the result we get only $c_1 d_1$ and $c_3 d_3$ okay. So is the process I mean is the semantics of division clear and basically what we are asking for let me know take it back to this definition.

So, the division operator produces a relation $R \div S$ that includes all tuples t in R where t is that you know the difference $R - S$ kind of that appear in R in combination with every tuple from S okay, S is the other side the right hand side. So, now you can reread this definition t belongs to the $R \div S$ portion of r and for every tuple s in S , there is a $t \cdot s$ in r such that the $t \cdot s$ portion is t and S portion is this particular s .

Basically we are saying that t combines with all t 's which are tuples in s . Well, you might wonder actually is this operator it still looks quite complicated when do we really need it do you use it you know that kind of yes in some situations, it becomes useful. Any questions here. Have a look at the example again if you want. So the result is $R \div S$. So which is $C \cap D$ and so. So $c \cap d$ is not present in the result of the division as it does not appear in combination with all the tuples.

So to kind of illustrate this, let me take an example of a set of students, you know, who have this unique property. They are probably, you know, graduating soon or something like that, and they have registered for enroll for all the courses that are offered by computer science department right. And some time ago, we used to have this interesting situation that the number of faculty was less, and so the number of electives were less.

So the dual degree students ended up doing almost all the courses that are available in the department, you know, like that.

(Refer Slide Time: 33:54)

Query using division operation

Find those students who have enrolled for *all* courses offered in the dept of Computer Science.


Step1: Get the course enrollment information for all students
 $\text{studEnroll} \leftarrow \pi_{\text{rollNo}, \text{dept}, \text{courseId}} (\text{student} \bowtie \text{enrollment})$

Step2: Get the course Ids of all courses offered by CS dept
 $\text{csCourse} \leftarrow \pi_{\text{courseId}} (\sigma_{\text{dept} = \text{"Computer Science"}} (\text{courses} \bowtie_{\text{deptId} = \text{deptNo}} \text{dept}))$

Result : $\text{studEnroll} \div \text{csCourse}$


•

Schema



Prof P Sreenivas Kumar
Department of CS&E, IITM

-49



So, of course nowadays, it is not really possible for many of you to do all the courses in the department. So find those students who have enrolled for all courses offered in the department of computer science department. So what is it that we are looking for here. So this is an apt example for this is an app situation for applying division operator, because we are now looking at the

enrollment data and then, you know, we are asking for who are those students who combined with every course that is there in the set of courses offered by the computer science department right.

That is what we are asking for. Who are those students, roll numbers that combined with every course that is existing in the set of courses that offer by computer science department and exist in the enrollment relation, right. So that is how this the division operator will be used here. So get the courses enrollment information for all students. Where do you get this. You get it from the enrollment relation okay.

So I want to take a natural join between student and enrollment. In fact, I do not actually need this because enrollment itself has roll number and roll number is there but if you want the name details, then you will have to combine with student let us assume that, so we are doing that okay. So we are doing a natural joining student and enrollment and getting the student enrollment data, roll number, name and the course ID.

And then in the second step, we are getting the course Ids of all courses offered by the computer science department. How can you get that, it is very simple. So we take the courses and the department and do it join saying that department Id should be same as department number. In the courses we have department Id attribute as what is the offering department. And in the department we have department number.


So we take the appropriate the combinations. So this will produce information about courses along with the details about the department that is offering. And then we do a selection based on computer science, the name of the department should be computer science. We are assuming that we do not know the idea of the computer science department. And so we have a selection here. We in fact, know that it is department number 1 right from the data.

But then anyway, we have to go by what is given in the query. So gets course Ids of all courses offered by CS department, so we get you can project the course Ids, after doing a selection of computer science of this joint, so we will get this CS courses. Now all that we had to do is to

simply divide the student enrolled relation instance with the CS courses to get the desired result. So, result a student enrolled division CS course set of all.

So, this will me give me so what will be the scheme of this, what will be the scheme of the roll number, name, right is not this minus this right. So, roll number name is the scheme for this So, these students are these special kinds of students who have enrolled for all courses that are here in the computer science department they have nothing else to do, they have to leave okay.

(Refer Slide Time: 37:57)



Suppose result of step 1
(we skip roll number for simplicity)

name	courseid
Mahesh	CS635
Mahesh	CS636
Rajesh	CS635
Piyush	CS636
Piyush	CS635
Deepak	ES456
Lalitha	ME650
Mahesh	ME650

result of step 2

courseid
CS635
CS636

Let's assume for a moment that student names are unique!


$\text{studEnroll} \div \text{csCourse}$

result

name
Mahesh
Piyush

Prof P Sreenivasan Kumar
Department of CS&E, IITM

41




So, this can illustrate with data, so suppose the we skip the roll numbers for simplicity, let us assume that for the moment that the student names are unique okay, so we have name and course Id. Let us say this is the student enrollment data. And this is the course Ids just ridiculously small I mean just 2 courses being offered by computer science department which is not the case and then we do a division.

So you can verify that this is indeed only this Mahesh and Piyush should have this oh, okay. The only man left out is Rajesh okay he is not done all the courses okay. So you get the point right, I suppose. So this is for in some special situations like where you want to check whether a data set of data combines with every tuple in another relation. You basically use the division operator okay.

So, study this examples and also the definition so that you will get the you know the way we have defined it whether okay.

(Refer Slide Time: 39:23)




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 Sreenivasan Kumar
Department of CS&E, IITM

42



Now let us come to this complete set of operators question are all the relational. So, what are all the relational algebra expressions we have introduced so far, we have introduced selection, projection, union, intersection, difference, cross product, join and division, rename, of course, rename is a not a computational object but its computational operator but it is the convenient operator. So, some operators can be realized through other operators we have already seen that.

So, that leads to a question as to are all algebra essential. The answers of course is not, that they are all not essential, the minimal set of operators we can show it theoretically which we are not going to do it now in this course, but one can show this in theoretically saying that sigma selection projection cross product union, difference, these are the complete set of operators that means, these are necessary and sufficient.

So, you can realize all the other operators through these operators. So, the intersection can be realized through union and difference, work it out as an exercise as to how you can introduce, you know, realize intersection using union and difference. The join and as I was already mentioned, can basically be made use of can be done by cross product followed by selection. The

division also you may take it as an exercise and work it out as to how you can do the division without using the other operators.

Division is not an essential operator actually, it is you can do it without with the basic operators project costs products and because I am just giving a hint here saying that you can do it with using project cross product in difference. But you have to work it out as to how exactly to do that. So, it is very interesting and that all the various kind of queries that we may have against information system can all be expressed using this less than half dozen operators right.

So, of course in practice, so are these the operators that are going to be used in practice. Well, that is a different question, cross product is not the recommended operation to be used in practice because it generates lots of things instead and the appropriate joints will be implemented okay crossword is almost never implemented. So the joints will be upward, so sigma pi join union, difference you can take it as a practical set of operators.