

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

Lecture-08
Introduction to Relational Algebra

So in order to talk about referential integrity constraints we introduced the notion of a foreign key.


(Refer Slide Time: 00:22)

Referential Integrity Constraint (RIC) - Example

courseId	name	credits	deptNo
CS635	ALGORITHMS	3	1
CS636	AI	4	1
ES456	D.S.P	3	2
ME650	AERO DYNAMICS	3	3
CE751	MASS TRANSFER	3	4


deptId	name	hod	phone
1	COMPUTER SCIENCE	CS01	22576235
2	ELECTRICAL ENGG.	ES01	22576234
3	MECHANICAL ENGG.	ME01	22576233

The new course refers to a non-existent department and thus violates the RIC



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

13



A foreign key is something in a scheme is a group of attributes which would be used to refer to tuples in other relations. That is what we call a foreign key. And the referential integrity constraint basically says that this kind of referencing you know, should be to existing tuples in the database instance, and they should not be dangling references okay. So if there are any questions in that, we can take it up.


(Refer Slide Time: 01:02)

Example Relational Scheme

student (rollNo, name, degree, year, sex, deptNo, advisor)
degree is the program (B Tech, M Tech, M S, Ph D etc)
for which the student has joined.
year is the year of admission and
advisor is the EmpId of a faculty member identified as
the student's advisor.


department (deptId, name, hod, phone)
phone is that of the department's office.

professor (empId, name, sex, startYear, deptNo, phone)
startYear is the year when the faculty member has
joined the department *deptNo*.



Prof P Sreenivasan Kumar
Department of CS&E, IITM

14



In this class, I am going to give you a set of an example database, an example relational database. And I am going to kind of continue using this example throughout this module and also in the SQL and other later modules. So it is going to be important to clearly understand this relational database scheme and we will keep it. I suggest that you write down this the just the relational and names of the attributes separately in a sheet of paper.

So that it is accessible to you when we are discussing queries and SQL and things like that okay. So okay so let us go ahead with this. So we have database is to capture the academic details about a in institute which has a number of departments and students studying in the departments for various degrees. So, we have student relation which has these are the following attributes for the student relation.

We have roll number, name, degree, years, sex, department number and advisor. So, here the conventional relational schemes when you write down relational schemes is to underline the key attribute. So when an attribute or a group of articles written together are underlined, what it means is that it is a key okay. So roll number is obviously the key for this particular. So there are multiple case then there will be multiple groups of attributes which are underlined in the relational scheme.

So roll number is a key for this student relation because attribute the value of roll numbers can be used to uniquely identify a particular row or a particular tuple in the student relation, right. And so, name these are obvious, you know, degree is the program, the value of the degree attribute is the strings that stand for the various programs that the student might have been enrolled for.

So, what are the degree for which he has joined the institute and the year is the year of admission and advisor attribute is the employee Id, employee ID of a faculty member identified as the student's advisor okay. Also all students required to have an advisor, so there is a advisor attribute for student which will take values that are from the employee Id domain of employee Id of a faculty member.

So that way you will be able to identify as to who is the advisor for the student. And of course, if you want to know more details about the student and advisor, then you have to look up the appropriate other relation. So, and here is the other relations, there are actually some more in the next slide. So department is another relation. It has department Id, name, hod, and phone.

So phone is the department's office phone number, and hod is a faculty member who is the head of that particular department who is kind of academically responsible for the running of the department. And what is the name of the department and what is an ID. So, we give some numbers for the department so that they are easily identifiable and serve as a key. In this case, even the name could serve as a key.

But we choose to have a department Id which is a number. Now, professor, employee Id because professors are all employees of the institute. So, there is an employee Id and name, sex start year. So, start year is the year when the faculty member has kind of joined that particular department. So, all faculty members work in departments. So, what is the department in which particular person is employed is given by the department number and then phone number of the particular professor. So, these are the various attributes we have for professor relation okay.

(Refer Slide Time: 05:47)

Example Relational Scheme


course (courseId, cname, credits, deptNo)
deptNo indicates the department that offers the course.


enrollment (rollNo, courseId, sem, year, grade)
sem can be either "odd" or "even" indicating the two semesters of an academic year.
The value of *grade* will be null for the current semester and non-null for past semesters.

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

preRequisite (preReqCourse, courseId)
Here, if (c1, c2) is a tuple, it indicates that c1 should be successfully completed before enrolling for c2.

15





Prof P Sreenivasa Kumar
Department of CS&E, IITM

Now, next one is the other important things that we need to keep track of course, a course is a subject of study, right and it has been named, it has been given a name c name, we call it to just to not confuse it with other names. So C name is the course name. And each course has certain credits, right the number of credits that you earn if you complete that course and then department number indicates the department that offers that particular course.

And, then the course Id. So we use a course Id to kind of identify the courses so to like, act like a key for the course relation. And then we have the issue of roll numbers. Sorry, the issue of enrollments. So students enroll for courses. Now in this case, we see that there is a need for you know taking a key, which consists of 4 attributes, the reason being that you know, course Id alone cannot be, you know, roll number and course Id.

A student typically does several courses in a particular semester. And so and of course gets offered multiple times across the years, right, say so for example, you are now doing CS3700. The same course would have been offered last year and it will get offered next year and so on so forth. And so the year in which it is offered, and the semester in which it is offered, that is the one that is you know, in which a particular student is enrolling and obtaining a particular grade.

So because of this reason, you need roll number, course Id, semester in which this course is running, and the year all of them to be part of the key for this relation. And what we will assume

here is that the same attribute stands for semester, and it has the value either odd or even standing for the August semester and the January semester okay. So, we are in the odd semester now, though there is actually nothing odd about the semester.

Anyway, so the value ah grade will assume that the value grade will have null for the current semester. And then we will have appropriate values after the semester gets over, okay. And so for all the previous semesters, there will be some value for the grade. So that is how this is. Now so you can see that the issue of how students, you know, enroll into courses and then applying grades is all captured by this enrollment relation.

Then we have the from the faculty side we have, we want to keep track of who teaches a course. So we have teaching relation, which basically captures that a particular employee, a particular professor is teaching a particular course, in a particular semester and a year okay. For example, so, I am teaching the CS3700 course of the odd semester of 2019 in the classroom, CS25 like so that is one.

So naturally you can also see that the need for all of the attributes to be the key here because no subset of this will be able to uniquely identify a rho in this particular relation and employee Id typically employee are typically teachers several courses across the that is you know employment and so, obviously, that itself is not a key for this and course can be taught by several professors.

And also so, that itself cannot be the and then you know, obviously, you need to have this semester and year only then you will see that a particular edition of the course is being talked about and so you have the unique way of capturing that particular topic okay. So, the identifying the keys are depends on the domain that we are modeling and so in this case these are the keys. Now, finally we also have prerequisite relation which basically has 2 attributes called preReqCourse and course Id.

And here so, if some tuple C1C2 exist in this particular relation, what it basically indicates is that C1 is a prerequisite for C2, C1 should be successfully completed before enrolling for C2. So, that

is the meaning of a prerequisite course. So, for example, data structures is a prerequisite for database systems course. So you should have completed data structures before you take database systems course. So there will be a tuple like that here okay.

So I hope the example relations scheme is kind of clear. So we will use this relation scheme in this module and also in the SQL module. Now, I will show this entire relation scheme in one slide.

(Refer Slide Time: 12:12)

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

- student (rollNo, name, degree, year, sex, deptNo, advisor)
- department (deptId, name, hod, phone)
- professor (empld, name, sex, startYear, deptNo, phone)
- course (courseId, ename, credits, deptNo)
- enrollment (rollNo, courseId, sem, year, grade)
- teaching (empld, courseId, sem, year, classRoom)
- preRequisite (preReqCourse, courseId)

A red asterisk is placed between the 'teaching' and 'preRequisite' relations. 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 in the top right corner. At the bottom left, it says 'Prof P Sreenivasa Kumar, Department of CS&E, IITM'. At the bottom right, it says '16'. A video inset in the bottom right corner shows a man in a light blue shirt speaking.

So that you know we have it all in one place and now actually we will show what are the foreign keys here, and where are they, what are they referring to okay, well let us identify the referential integrity constraints and the foreign key and then how with what a foreign key is refer to. So you can start identifying lots of these in each of these relations there are foreign keys. So for example, here department number is a foreign key right.

Because this is the one that we will use in the student tuples to kind of identify what is the department in which that particular student is available is present. And then, you know, we can use this to kind of link up with the appropriate department tuple okay. So to refer to the computer science department tuple, so we will use the computer science department Id and then put it here, right. That is how we refer to the.

(Refer Slide Time: 13:28)

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



So to indicate this we have. So, those things will come up now. So, I will now indicate them by using these rhos. So for example, hod is a foreign key in the department tuple department scheme. And it refers to the employee Id of the professor. So, one of the professors is the hod, and so you have the employee Id of that particular person sitting here. So, and that can be used to refer to the actual professor who is the hod for that particular department.

So, you can see that this attribute has the you know, the qualities that we are looking for a foreign key, which basically is a bunch of attributes that we use to refer to tuples in other relation and you know and the constraint here is that it either it should be null or it should have values that are existing in the other column of the other referring the relation that we are referring to.

So likewise now we can identify, so prerequisite course and course Id are both foreign keys referring to the course Id right. So, there are some more one more one thing that you will notice here is that the same the same attribute name might occur in multiple relations okay. And if there is a ambiguity as to which course Id attribute are we referring to we will use the conventional dot notation to disambiguate that name.

So, if you say course dot course Id then we are referring to this attribute, if you say employment dot course Id then we are referring to this particular attribute like that and also the same you know kind of quantity is actually being referred to by 2 different names like for example, here


department number and department Id, you know, we know that they are, in some sense the same kind of thing.

So, but then we give the freedom saying that you can use whatever name you want, but then this is a foreign key that refers to the primary key of the other relation okay. So, even in such situations it is not compulsory that now this particular attribute name should be the same as this particular attribute name. That is what I am trying to emphasize okay. Now, let us identify some of this employee Id, obviously is a foreign key refers to professors.

And then roll number obviously, is a foreign key in the enrollment relation, referring to student. So even though these are part of some keys and all that they can independently be foreign keys department number is a foreign key refers to department Id. Then this department number here in the professor is also a foreign key reference department. Course Id here is a foreign key referring to that. Obviously here also course it referred to.

So this is also as a foreign key referring to that and then here advisor is a foreign key that refers to the employment Id. And I suppose that is all, okay. Department number here in the course refers to the department Id remove, okay, that is it okay.

(Refer Slide Time: 17:43)




Relational Algebra

- A set of operators (unary and binary) that take relation instances as arguments and return new relations.
- Gives a procedural method of specifying a retrieval query.
- Forms the core component of a relational query engine.
- SQL queries are internally translated into RA expressions.
- Provides a framework for query optimization.

RA operations: *select* (σ), *project* (π), *cross product* (\times), *union* (\cup), *intersection* (\cap), *difference* ($-$), *join* (\bowtie)

Prof P Sreenivasa Kumar
Department of CS&E, IITM

18



So this diagram is what we traditionally call as a relational schema diagram, in which we have the relation names, attribute names, keys are underlined and then we have arrows to indicate the foreign keys and what are the foreign keys referring to, what are the relations they are referring to. So, this particular slide, I want you to you know, note down in a separate page of your book.

So that you can refer to this particular diagram when you are talking about queries in either relational algebra or SQL, have it handy so that we can discuss this okay, any questions about this. So, I think, feel clear okay. Now, let me introduce, start introducing the next major topic in this module, which is the topic of relational algebra. Now we have seen the relational model relational algebra is part of this the whole relational approach to databases.

And is a basically a set of operators that take relation instances as arguments and return new relation instances, new relations okay. So obviously it is an algebra, so it works on some domain of things. So the things, it works on our relations. That is why it is a relational algebra. So it takes relations and produces new relations. So each of the operators here are like that.

So we have a bunch of operators, both unary and binary that take relations since this is arguments and return new relations. Now, what we will see here is that using these options operators will be able to express the retrieval request will retrieval see database has a lot of information. And we want to retry certain information from the database right. So how do we express your requirement, how do we express your query, it is called query in the database parlance, we call it a query.

A query is basically a data retrieval request, we want this kind of data, this is what we want from the database to be retired. So how do they express that. So, relational algebra operators give us one way of expressing what we want actually. So but interestingly you can also see as we go along that this is a procedural way of expressing what we want okay, a procedural way of expressing what we want, that means it kind of says that okay.

You do this operation here on this relation, do another operation on other relation, combine this like this and then combine it with something else and then give me this okay. So, when you have

an algebraic expression, you can think of it as a tree right. And so, you can think of information flowing up like that finally, the output comes at the root in some sense, we will understand this as we go along and then discuss the queries .

So, relational algebra forms the core component of the relational query engine, which is the heart of the relational database system, uses these relational algebraic operators and efficient implementation of these operators. Right now in this module we will not really bother others with the implementation details about these algebra operators, we will think of them as logical operators.

And will confine ourselves to the conceptual understanding of what is operators, we will just think of them as mathematical operators and then see how we can make use of this operators, later on of course, we need to study as to how to exactly implement these operators, how do you write a program to realize the operator okay, and those programs will go into will all be available in the in the query engine okay.

So SQL queries so we will study SQL in detail and SQL a declarative way of specifying queries, a largely declarative way of specifying queries, we will see that later. And that is internally translated into relational algebraic expressions as I was mentioning earlier also. And because of that it kind of provides us an opportunity for optimizing that relation, optimizing the way that particular query gets actually executed okay.

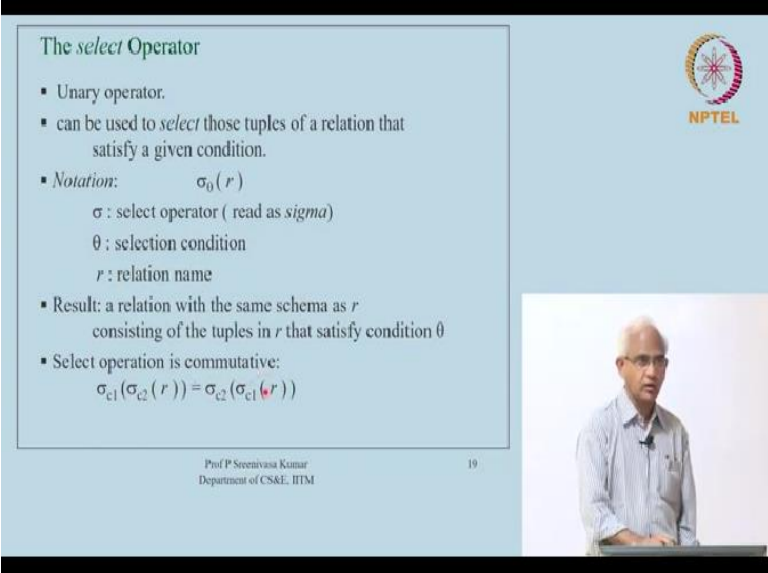
So we will see all this in this particular set of slides. So what are the operators. It is surprising that we have just about half a dozen you know operators, actually, we will see that the minimum set of operators is you know, we will come to that little later. But right now we just have 1 2 3 4 5 6, just about 6 7 operators. And, if you feel understand these operators, this kind of sufficient for expressing all kinds of queries on relational databases.

So, so that is the power of this particular up the system of the algebra. So we have select project which are unary operators and cross product operation, the union intersection the difference and you can also see that actually not all of them are really required, you can express some of them in

terms of the other. So let us say union intersection and difference and then join operator. We will later discuss as to what is the minimum set of operators and things like that okay.

So let us start looking at these operators one by one and understand the meaning of what are those operators do, for it is not necessarily that you should be familiar with SQL at this stage of the course. But if you are familiar with SQL, do not confuse the select operator if you are going to talk about here with the select clause of SQL and the select clause of SQL actually corresponds to the project operator okay. So but even if you do not know SQL at this stage, it does not matter okay.

(Refer Slide Time: 25:19)



The slide is titled "The select Operator" and contains the following text:

- Unary operator.
- can be used to *select* those tuples of a relation that satisfy a given condition.
- Notation: $\sigma_{\theta}(r)$
 - σ : select operator (read as *sigma*)
 - θ : selection condition
 - r : relation name
- Result: a relation with the same schema as r consisting of the tuples in r that satisfy condition θ
- Select operation is commutative:
$$\sigma_{c_1}(\sigma_{c_2}(r)) = \sigma_{c_2}(\sigma_{c_1}(r))$$

At the bottom of the slide, it says "Prof P Sreenivasa Kumar, Department of CS&E, IITM" and "19". In the top right corner, there is an NPTEL logo. On the right side of the slide, there is a video inset showing a man in a light blue shirt speaking.

So let us start with the select operator, it is a unary operator. And basically it can be used to select those tuples in the relation instance that satisfy a particular given condition okay. So it has a parameter which is the condition and what it does is to select all those tuples in the relation instance with satisfy a certain condition, the given condition. And the notation here is to write it as sigma theta and the relation operator the input religion.

So theta is the selection condition, sigma is the select operator and r is the relation name. So it is a very simple operator. So, it basically takes a condition and then returns all the tuples in this relation instance r that satisfy that particular condition. So, it gives a new relation instance and

then it is easy to see that the scheme of the new relation instance is the same as the scheme of the input relation right.

So, result is a relation with the same scheme as r , the same set of attributes, same sequence of attributes, same order of attributes as r consisting of those tuples in r that satisfy the condition θ and the operation is commutative. So, if you write $\sigma_{c_1}(\sigma_{c_2}(r))$, okay notice that we can compose the operators and then get algebraic expressions right. And that is the whole idea of algebra right.

So, you could $\sigma_{c_2}(r)$ is now a new relation. So I can now use that relation and then make another selection operation, right. So σ_{c_1} is and then inside that I am giving this new relation, which is a result of this particular doing selection with c_2 and r . And you will notice that this is actually same as doing σ_{c_2} on $\sigma_{c_1}(r)$. So if you are first selecting those tuples that satisfy c_1 okay.


And then from there you are selecting the tuples that said c_2 is the same as doing first the tuples as c_2 and then choose from among them. So, basically the tuple sets were both c_1 and c_2 and so the order does not really matter and so, this operation is actually commutative. So, it is this kind of identities involving this algebra operators that will later help us I know in rewriting these algebraic expressions.

So that it will be advantages for us to actually run the query in a efficient manner okay. So that is the simplest operator select.

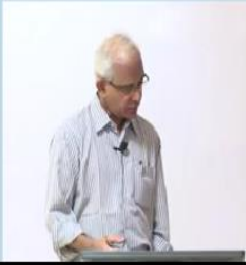
(Refer Slide Time: 28:59)

Selection Condition

- *Select condition:*
Basic condition or Composite condition
- *Basic condition:*
Either $A_i <compOp> A_j$ or $A_i <compOp> c$
- *Composite condition:*
Basic conditions combined with logical operators AND, OR and NOT appropriately.
- *Notation:*
 $<compOp>$: one of $<, \leq, >, \geq, =, \neq$
 A_i, A_j : attributes in the scheme R of r
 c : constant of appropriate data type



Prof P Sreenivas Kumar
Department of CS&E, IITM



Here are some okay, let us go into the details about what exactly is the selection condition. So, the selection condition can be either a basic condition or a composite kind of condition, the basic condition can be either of these 2 forms. So, basically when you take a attribute and compare it with another attribute or what are these attributes, where are these attributes from, obviously they are from the relation scheme that is on which it is being applied, the relation on which it is being applied.

So, these A_i, A_j are the attributes of the relation well, that is the input for this operator. And so, you compare those 2 attributes, somewhat a pair of attributes or you compare an attribute with a constant appropriate constant each of these attributes come from a certain domain. And so if you take a particular string you cannot probably compare it to the integer, but you will compare it with strings right.

So, appropriate things and appropriate constants, constants of appropriate type. So, these are the kind of atomic kind of conditions, simplest of the conditions, the simplest of the conditions will involve comparing an attribute with another attribute and or comparing an attribute with a constant. So, these are the simple conditions and then you can combine these conditions with the logical operators, the logical operators are and or not are all allowed.

And then make a complex condition or composite condition, I am calling it as a composite condition. So, basic conditions combined with logical operators, and these are familiar to us, we have been using them. So, earlier courses also, so and or and not that we used and so. So, all of them of course, are Boolean value right. So, this is a comparison operator is A_1 greater than A_2 , either it is true or false right.

So, these are Booleans so, they can be combined with Boolean operators and or not and then you can now get a composite condition and that is how you will use. So, this θ is of this form, the θ the selection condition is either a basic condition or a composite condition. And if it is composite, it basically uses logical operators and then combines this atomic conditions and the what are the comparison operators.

The common notice the operators are the obvious ones here are the less than less than equal to greater than greater than equal to and equal to not equal, the six possible comparison operators which compare values basically they compare values and of course, we will. So, in order for the we will assume that if you are applying this comparison operators the corresponding domains are you know linearly ordered sets like that.

The order relations do exist on those attributes only then it will make sense for us okay. So, c is a constraint of appropriate data right okay. So let us start using this condition and then express a few data requests.


(Refer Slide Time: 32:52)

Examples of *select* expressions

1. Obtain information about a professor with name "Giridhar"


$$\sigma_{name = \text{"Giridhar"}}(professor)$$
2. Obtain information about professors who joined the university between 1980 and 1985, both inclusive

$$\sigma_{startYear \geq 1980 \wedge startYear \leq 1985}(professor)$$



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

21



So let us look at some examples select expression. So obtain information about a professor with name Giridhar start with that, there may be several Giridhars in the institute. So now we are saying seek name is a attribute of professor relation. So, this information is there in professor relation, the details of professor obviously or the professor relation. And so what we are saying and we want information that tuples that have Giridhar as the name.

So we are putting a simple condition here, name should be equal to the string Giridhar in this relation professor and then obtain all those tuples one. So if there are multiple Giridhars, they will all be listed as. So the entire information about the professor tuples will be, so now this is now stands for a relation, which has the same scheme as professor okay, but contains only those tuples in which the name is exactly Giridhar this thing okay.

So it is a subset of the original set of tuples. Now, we have not had given a name for that particular result right. It can, we have to give a name in case we are using this relation that has been produced by this particular expression later on in some other context, we may actually want to give a name for it okay, so we will see that later. So obtain information about professors who joined the university between 80 and 85. Both inclusive, some numbers okay.

So there is a start year attribute in the professor which says that when the professor has started working in that particular institute or department to be specific. So, here we are saying so, I use


this short way of you know expressing 9 and start here is either get an equal to 1980 or less and it is actually less than equal to 85 between exactly between these 2 things. So, this will contain the all the professor tuples, whose start here lays exactly between 1980 and 85 inclusive.

So now, these are typical data retrieval requests. And we can see that we can express them using this particular operator.

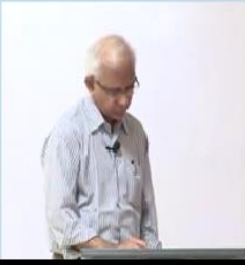
(Refer Slide Time: 35:52)

The *project* Operator

- Unary operator.
- Can be used to keep only the required attributes of a relation instance and throw away others.
- *Notation:* $\pi_{A_1, A_2, \dots, A_k}(r)$ where A_1, A_2, \dots, A_k is a list L of desired attributes in the scheme of r .
- Result = $\{ (v_1, v_2, \dots, v_k) \mid v_i \in \text{dom}(A_i), 1 \leq i \leq k \text{ and there is some tuple } t \text{ in } r \text{ s.t. } t.A_1 = v_1, t.A_2 = v_2, \dots, t.A_k = v_k \}$
- If $r_1 = \pi_L(r_2)$ then scheme of r_1 is L



Prof P Sreenivasa Kumar
Department of CS&E, IITM



So, moving on let me also introduce a project operator, the project operator while the select operator who actually working on the tuples and then choosing a subset of these tuples, what the project operator will do is to help you to kind of get rid of unwanted columns in relation. Supposing you want a relation is pretty wide, and you want to focus on a only particular attributes.

And you do not want to really look at all the attributes of the relation, then this project operator will be useful to you okay, so what project operator does is to keep only the required attributes of a relation instance and throw away others in some sense. So I do not want to, you know, other information so I will just throw it away okay, so here is the notation pi is the symbol that we use for this operator pi.

And here is a list of attributes that we you want. So obviously this and then this is the relation on which we want to apply that. So, obviously each of these A_1 through A_k is part of the scheme of r okay. So they need not be consecutive anything like that. So, some set of attributes from the original scheme which were interested in. So, the this L list L is the desired attributes in the scheme r and exactly what is the output or what is the result of this particular expression is given here.

The result is the set of tuples v_1 through v_k matching all these A_1 through A_k such that v_1 belongs to the domain of A_i okay for each of these case, and there is some tuple t in r . Such that $t.A_1 = v_1$ $t.A_2 = v_2$ and $t.A_k = v_k$ okay. There may be multiple tuples actually. But as long as there is some tuple in t such that this condition is satisfied, we take that bunch of values v_1 through v_k and then put them in the output okay.

So, basically remember that the result of a particular expression is a relation and what is our definition of a relation, it is a set of tuples, okay. And so it will not have duplicates. It is a set. So there is no scope or duplicate elements okay. So that is why even if you let you know, there are multiple tuples that have these particular corresponding values v_1 through v_k for these attributes A_1 through A_k only one sub tuple of this v_1 through v_k will exist in the relationship okay.

Let me illustrate that. So before we go further, so if r_1 is the new, you know, relation name we are introducing and then assigning it as this one $\pi_{l_1} r_2$, then basically we are calling this $\pi_{l_1} r_2$ as this new relation r_1 , then the scheme of r_1 is l . The projection list, what is this scheme of the output, the scheme of the output is this projection list, right. Because that is what you are projecting.

These are the attributes that you have chosen. The remaining attributes around being thrown away. So it is kind of obvious that the, you know, schema for the output is this projection list.

(Refer Slide Time: 40:13)

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	Piyush	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

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

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


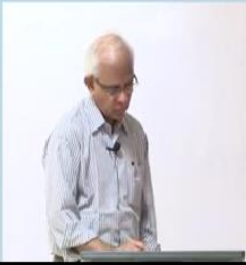
$\pi_{\text{name}}(\sigma_{\text{degree} = \text{"M.S"}}(\text{student}))$

name
Mahesh
Rajesh

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

Prof P Sreenivas Kumar
Department of CS&E, IITM

23

So, here is the example, just look at this slide student. So, we have Mahesh in computer science department and there is a Mahesh in the mechanical engineer department, both of them are doing M.S etc. And so now you do a sigma degree M.S degree equals M.S on student. And project mean okay. So now degree M.S. So, sigma degree M.S on student will give me this tuple, this tuple and the last tuple the 3 tuples.


Because all of them are enrolling the M.S degree. Now, if you do a project name alone without projecting roll number, what it means is that I only need the column name or attribute name information of this particular relation and do not want anything else. So now if you do that, then you will actually lose information because only one Mahesh will exist, and the other tuple contribution Rajesh.

And that is it, you would have actually lost some information. They had to be careful about how to use it, but this just illustrates the idea that if you have a desired set of attributes and you project that you get information corresponding to that. If you do roll number name from the student. Supposing I just want to get only the names of the students I am not really bothered about degree and all the other details, then you would do pi roll number name on student.

And then you will see that you will get in fact you will get exactly the same number of tuples that are there in the student because there are no there cannot be any duplicates in this particular

combination of roll number and name because roll number is actually a key. Since roll number is a key you will get all these tuples okay. So, in this basically in this lecture I have been able to introduce just these 2 operators, the selection operator and the project operator.

(Refer Slide Time: 42:59)




Size of *project* expression result

- If $r_1 = \pi_{L_1}(r_2)$ then scheme of r_1 is L_1
- What about the number of tuples in r_1 ?
- Two cases arise:
 - Projection List L_1 contains some key of r_2
 - Then $|r_1| = |r_2|$
 - Projection List L_1 does not contain any key of r_2
 - Then $|r_1| \leq |r_2|$

Prof P Sreenivasa Kumar
Department of CS&E, IITM

24



We will just close with one particular point that if r_1 is $\pi_{L_1} r_2$, then what about the number of tuples in r_1 , the number of tuples will depend whether the projection list contains the key or does not contain the key. If the projection list contains the key, then the output will be the same, the size of the output be the same exactly the same as the input, otherwise it can be less. I wanted to list, we in the previous example also, it is clear. So, with this will stop today, I will proceed with the rest of the operators in the next lecture okay.