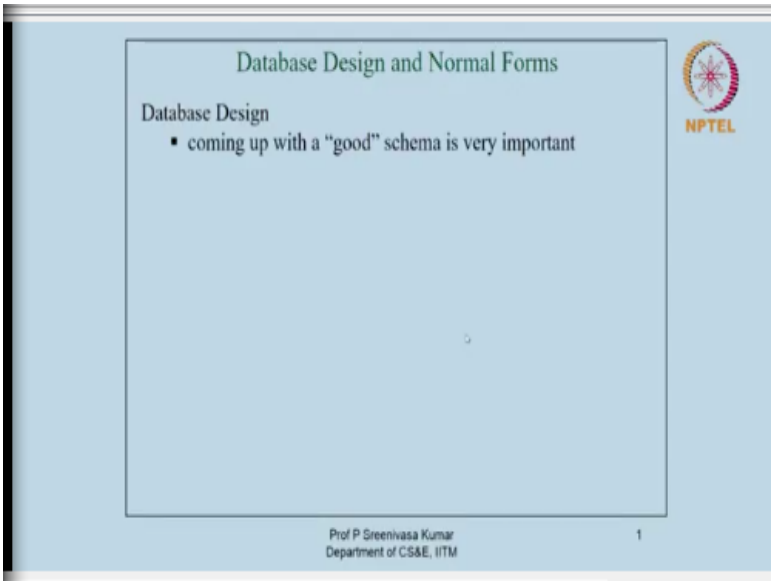
Database Systems
Dr. Sreenivasa Kumar
Department of Computer Science and Engineering
Indian Institute of Technology-Madras

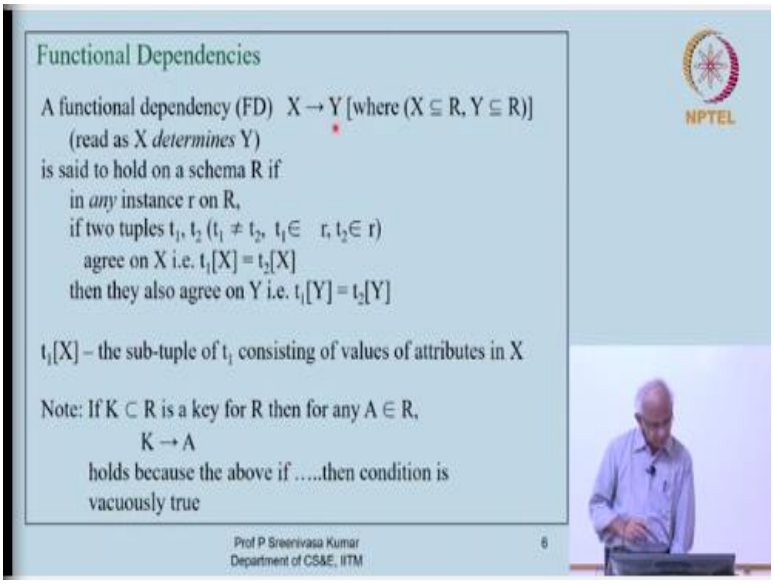
Lecture-20
Deriving new Functional Dependencies

(Refer Slide Time: 00:16)



Ok, so let us continue our discussion on normal forms.

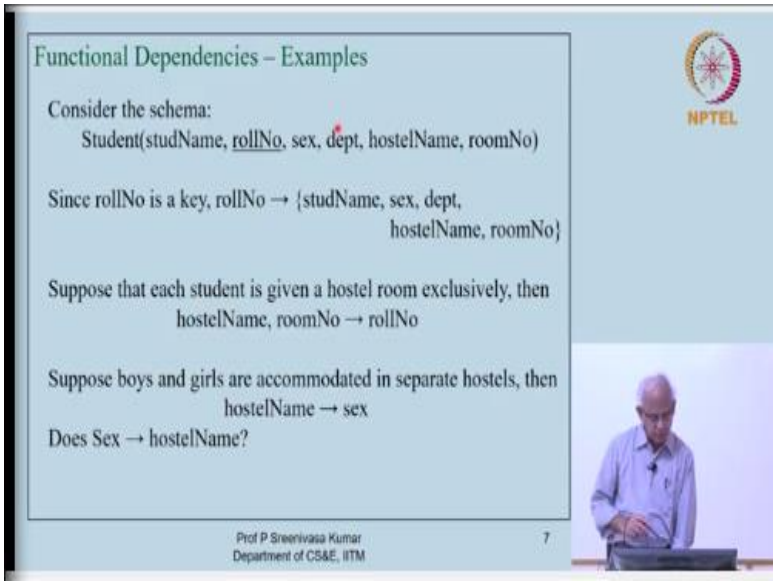
(Refer Slide Time: 00:28)



So in the last class I have defined in the last lecture we have seen the definition of the functional dependencies. Let me briefly recall this important definition because we are going to spend considerable amount of time understanding this notion of functional dependencies. So it basically says that one set of attributes is actually functionally dependent on X that means it is actually a function of X, so you all know what a way function.

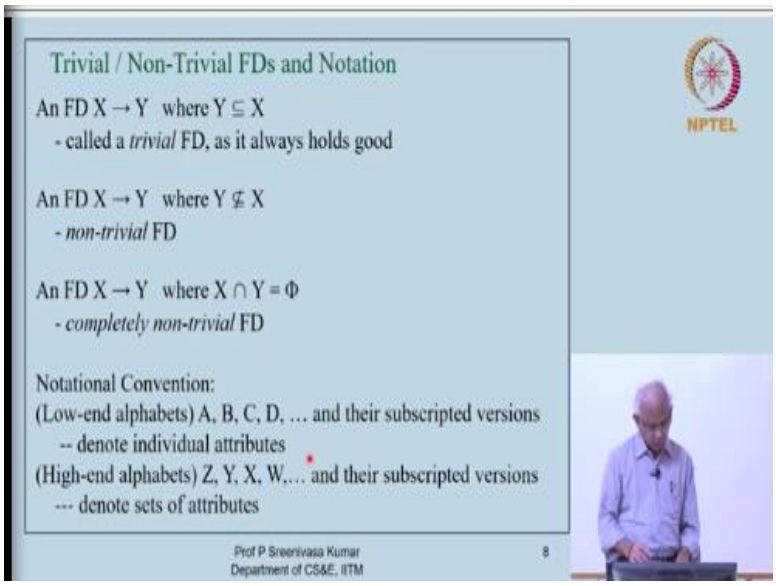
So the value of these attributes would uniquely determine the values of these attributes ok. So that is all we are saying but we are saying it in elaborate manner here that if any instance r of R if 2 tuples agree on these attributes of X then they will also agree on the attributes of Y which is a different way of saying that it is a function ok.

(Refer Slide Time: 01:26)



So having said that we would look at several examples.

(Refer Slide Time: 01:28)



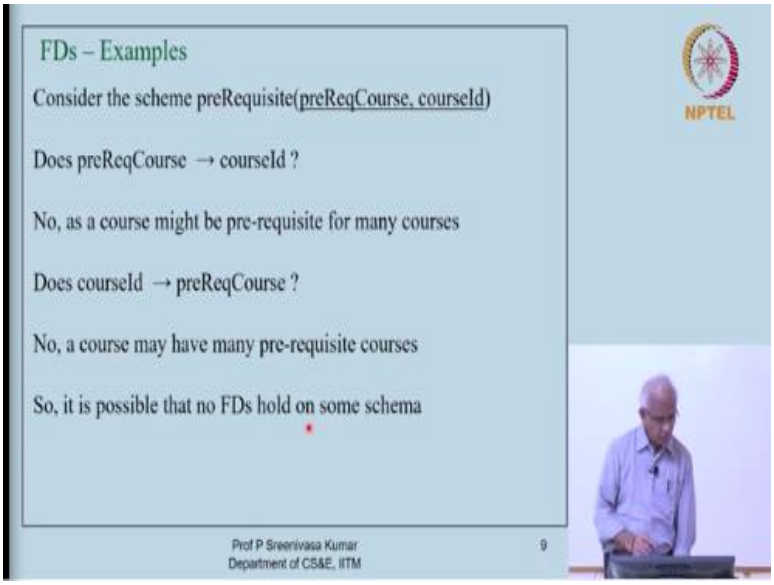
And in this class I am going to continue giving a couple of more examples and then we will start looking at these functional dependencies in a little bit abstract way and then see what are it is implications ok. So in the last class we also talked about what are called trivial and non trivial functional dependencies. So you notice that if the right hand side attributes are a subset of the left hand side attributes.

Then this kind of a functional dependency X determines Y will always fall on any scheme, what it says here is that if 2 tuples agree on X, then they will agree on Y. But if Y is a subset of X, so they are already agreeing on Y ok so there is nothing. So we call such kind of dependencies as trivial functional dependencies as they always hold. Now if Y is not a subset of X we will call that as a non trivial functional dependency.

And then if X and Y are actually completely they are disjoint then we will call that as a completely non trivial functional dependencies. So most of the time we will be worried about non trivial functional dependencies and we will see how exactly what are the other various other things we will do with functional dependencies ok. Before we proceed here is a bit of a notational convention, we will use this low end alphabets A, B, C, D along with their subscripted versions like A subscript 1, B 1, C 1 like that.

To denote individual attributes and then the alphabets on the high end the X, Y, Z, Z, Y, X, W, U, V etc those alphabets we will use four sets of attributes. So these letters we will use to stand for sets of attributes. That is why in the functional dependency we have always been using X determines Y, so X is a set of attributes, Y is a set of attributes. And if we want to talk about some individual attributes, then we will usually put it as A 1, A 2, B 1 B 2 like C 1, C 2 we use letters from a lower end of the occupants just to make it easier for us to remember.

(Refer Slide Time: 04:08)



Now let us look at some more examples, consider this scheme prerequisite we already have this in our scheme of institute scheme. We also have a lots of other relations in the scheme, if I leave it to you as an exercise to look at the functional dependencies those things. Actually most of the other relations the functional dependency that holds would be key on the left hand side and the remaining attributes on the right hand side.

If you can verify that but this one, what about this one does prerequisite functionally determine course Id. So prerequisite recall that prerequisite course is a course number, course Id is also a course number. And we will have a row if some particular course is a prerequisite of some other course, right. And so, there will be all such tuples in this particular instance.

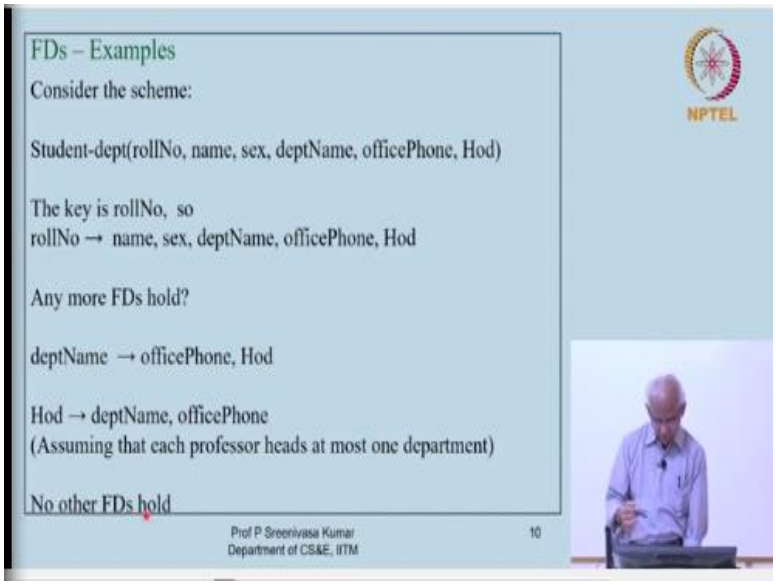
Now the question is, if 2 rows agree on prerequisite course should they also agree on course Id in this relation. So if they always agree in all instances of prerequisite relation then we say that

prerequisite course determines course Id. So from the logical understanding of the domain, what do you feel about this, this is not true.

So a course might be prerequisite for many courses and so it might appear along with many other courses as tuples here. So given the value of prerequisite we will not be able to uniquely determine what is the course Id in general ok. What about the other way around, does course Id determine prerequisite it is also not true because again a course may have many prerequisites actually.

So it is possible that no FDs hold on some schemes, so do not be looking for functional dependencies in all possible schemes. It is possible that there are cases where no functional dependencies hold at all ok.

(Refer Slide Time: 07:00)



Let us proceed what about here student, remember that this is the example I told you that this is a example of a bad scheme. And then we started all our discussion with this example. So let us reconsider that example where we have roll number, name, sex, department name, so and office phone and HOD. So what are all the assumptions here, the assumptions are that each student belongs to a single department.

And each department is headed by a professor and the department has one office phone, so these are the assumptions. So under these assumptions what is the key for this scheme, what is the hesitation for telling me the key here. It is simple right, we just chose to attach some more information with each student. So that does not you know affect the you know property that roll number is the key or the relation.

Roll number continues to be the key for this here, because given the roll number we will be able to uniquely identify 1 row in this particular dataset. Even the roll number will be able to identify, whether there would not be 2 rows with same roll number except relation. The roll number continues to be key, so roll number since roll number is a key. So roll number determines all the other attributes roll number attributes.

Notice one more thing here that we could actually write roll number determines department name as a separate FD and it will still be valid ok. And because you know instead of writing roll number name determines name, roll number determines department Id etc. We now actually have one FD which says roll number determines all the other attributes. And this kind of a thing as we have mentioned right in the beginning that this kind of thing always happens.

If the the left hand side is a key, key will always determine all the other attributes ok. So that is a functional dependency that always holds true in any scheme, of course if this key itself is all the attributes, then there is no possibility for the right hand side, that is what happened for the prerequisite relations, good. So that is the what about any other FDs that are existing here anymore FDs exist.

Let us pick up some attributes and then see whether they determine any other attribute. That means if 2 rows agree on that attribute then they have to agree on some other attribute right. Can you see that something happening in this set, roll number, name, sex department name, office phone and HOD. Particularly let us look at this department name**,** department names are unique in an institute we will not have 2 departments of computer science ok.

So Computer Science and engineering with title of the name of the department is only one department, so like that. So with this values for this if you focus on the values of this, let us say if 2 rows of this particular table agree on the value of the department name. Then what should they agree on what other attributes they should agree on, what is one more attribute that they should agree on at least a HOD right.

You cannot say that in one tuple in computer science department has he escaped from a Chandra is HOD. And in some other tuple you cannot say that Krishna Sivalingam is the right, it will be inconsistent defect. So if 2 rows have department of computer science in as under this column then under the column HOD they have exactly one person, they cannot have 2 different values there, they have to agree on.

And so is the case with Office phone, if you mentioned office phone as one phone here and then some other phone in somewhere else we have doubt actually which is the actual phone, so right. So department name indeed functionally determines office phone and HOD well good. Is there anything else everything in one more thing or anything more let us look at this, they just half a dozen of these attributes, so ok.

Let me give you again what about HOD on the left hand side, what is the policy of the institute of regarding HODs, can some professor be a HOD for 2 departments will any professor will agree for that, one department itself is a big headache. So I cannot be head of this department for 2 head department, so nobody will agree generally for being a HOD for 2 departments, so in general, a person is HOD for exactly one department.

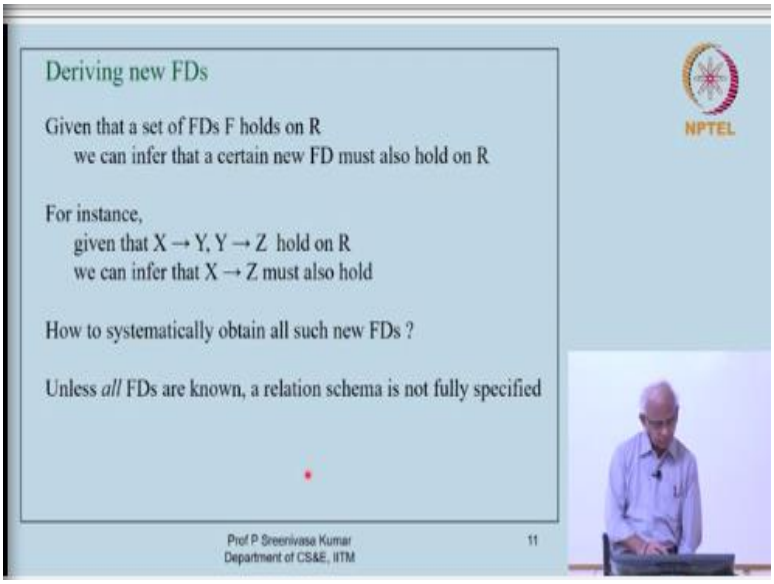
That is a reasonable assumption to make in this context. So given that so if you have a HOD on the left hand side, or do you have on the right hand side. If 2 tuples agree on HOD then they will agree on department name and obviously phone. So HOD determines department name and office phone. In some places they may violate this policy and then make some poor professor as HOD of 2 departments.

So it kind of depends on the policy for the institute, luckily we have a policy that each professor had only one department. So again it brings to focus the fact that these functional dependencies are actually knowledge about the domain that you are modeling. They are not they do not come from some thin air from somewhere, they are knowledge about the domain that you are modeling.

And particularly they cannot be inferred by just looking at it you have look at the model, you have to look at the domain. Now no other FDs hold and so we can does office phone determine department name, actually that, I think is. So given that if the exact each you know department has, so even office phone can be part of a some kind of a determiner for good, so we can add that.

Office phone now determines department name and also determines HOD, good so some other FDs also. So this actually again brings to focus ok this is one thing this kind of you know determining the functional dependencies that hold on the domain. By you know understanding the domain figuring out the key properties in the domain, knowledge about**.** But there are some kind of functional dependencies which can actually be you know derived sort of mechanically from the given functional dependencies. That is another situation that arises in this context and we will look at that

(Refer Slide Time: 17:01)



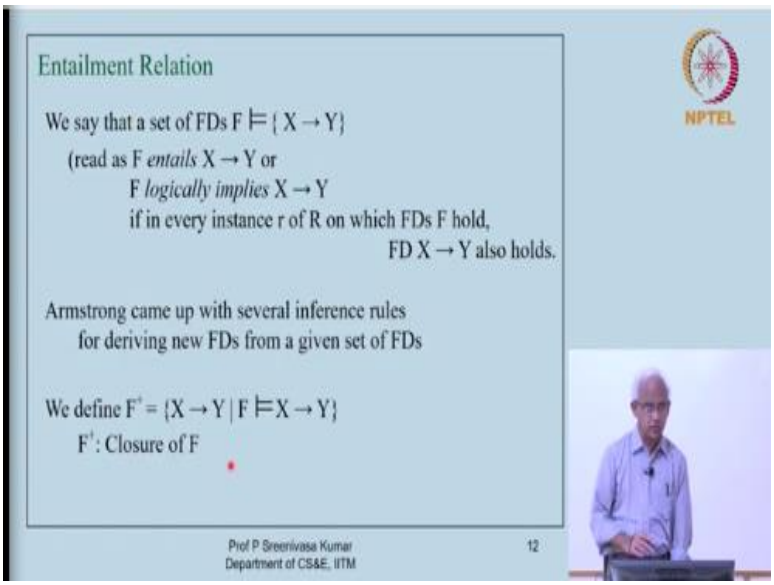
Given that some set of functional dependencies F hold on some scheme R, it turns out that we can actually do some logical inferences. And then logically we can conclude that some other functional dependencies should always hold ok. I will give you example for this, say for instance given that X determines Y and Y determines Z hold on some relations scheme. We can actually infer that X determines Z must also hold, why is this.

If 2 rows agree on X and because X determines Y holds they agree on Y attribute values. And because Y determines at holds in the given scheme. Since they are agreeing on Y attributes they must also agree on Z attributes. So that means basically if we assume that they agree on X attrbutes, we will conclude that they agree on Z attributes this is a simple you know we can later of course given nice name for this.

We can actually call this as some kind of a transitivity property of the function that determines symbol, well. Then they are going to be a lot more of these functional dependencies, which are kind of you know mechanically can be figured out given some set of functional dependencies. We can actually figure out a lot more functional dependencies by this process of deriving them. So then the question that comes up is how do we systematically obtain you know all such new functional dependencies.

And obviously we as designers are you know how precious time, so we will not be able to sit down and list on lots of things. We will list the crucial things and let is there some algorithmic process by which we can actually obtain F a lot of dependencies, that is one question that comes up. Of course under the we assume that unless all FDs are known a relation scheme is not kind of fully specified, so ok.

(Refer Slide Time: 19:46)



In this context, let me define a new kind of a thing we call it the entailment relation, it is a new name and bringing in a new symbol I will bringing in. This symbol actually is called the turnstile symbol. So we say that a set of functional dependencies would entail some other functional dependency X determines Y. So it is read multiple ways we say F entails X determines Y or F logically implies X determines Y like that we will read it.

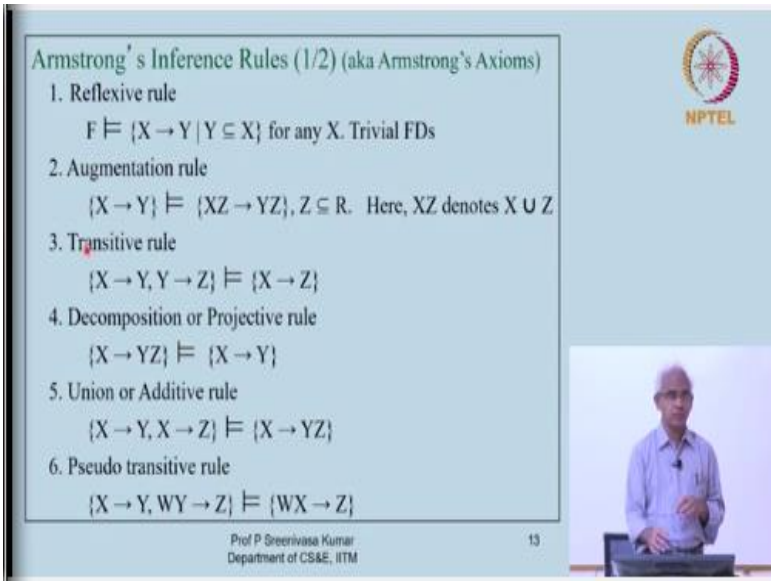
Now the defining condition is that if in every instance r of some relation scheme R capital R. On which these functional dependencies F hold, so if the left hand side functional dependency is hold. Then it is always the case that the right hand side functional dependency also holds ok. If that is the case we say that then this right hand side functional dependency is logically implied by the left hand side functional dependency.

Or left hand side functional dependencies entail the right hand side functional dependency ok. Now we have already seen such an example, so in the previous slide we have seen that if X determines Y, Y determines X hold on some scheme. Then they will logically imply that X determines Z holds ok. So are there more such things, so there was a researcher by name Armstrong who investigated this issue in the early you know when the relational theory was getting developed in the 80s 90s.

And then he came up with it several inference rows for deriving new functional dependencies from a given set of functional dependency ok. So we will study them ok, before we go one more notation we call this F + F superscript + as the closure of F which is the set of all functional dependencies X determines Y. That are logically implied by the given set of functional dependencies here ok.

The set of all functional dependencies that are entailed by a given set of functional dependency, that is F +. So given F, we talk about F + which is the in some sense analogical closure of F ok. Now so these logical the inference rules that Armstrong came up with.

(Refer Slide Time: 23:09)



They are called Armstrong's inference rules and they are also known as Armstrong's axioms we will look at them off ok. The first one is called the reflexive rule which actually is you know always holds. So if F entails X determines Y all X determines Y such as Y is a subset of X these are actually trivial function dependencies. So what basically saying here is that whatever we is the given set of functional dependency we can always derive trivial function dependencies out of it ok.

So these trivial functional dependencies are those which have their right hand side as a subset of the left hand side so trivial FDs. Next one is what is called the augmentation rule, what this is **is** that, suppose you are given a functional dependency X determines Y. Then that would logically

imply in the a new functional dependency where you take some bunch of attributes and add them both to the left hand side and to the right hand side ok.

So we write them as XZ determines YZ where exactly some set of attributes from R. Now so we use this notational convenience saying that instead of writing X union Z. We simply write it as XZ, ok you can also in our context of schemas it kind of makes sense because we are just going to write these additional attributes concatenated with the old attributes ok. So XZ determines YZ for some Z which is a subset of R.

So this will given that on all any instance if X determines Y holds, then what we are saying is that XZ determines YZ also false. But it is not very difficult to see this actually, because anyway we are adding the same set of attributes on the left hand side and right hand side. So if now additionally let us say to argue that this is correct. So on 2 rows if XZ attributes agree then because we know that X determines Y holds ok they have to agree on Y attributes.

And since you already know that they agree on Z attributes because in the left hand side that Z is also there. So it is automatic that why is that the agree on why is that attributes nothing ok. Have any questions please pause me because I am using a little bit new terms here. By now I think you are comfortable with this term called the 2 tuples agree on attributes that means they have the same corresponding values on a attributes.

Because they agree on XZ attributes, they will agree on YZ attribute, given that X determines Y, so that is called the augmentation rule. Then the next one is actually it will transitive rule and we have already seen I have given you this example. So if X determines Y holds Y determines Z together hold on some scheme then we can actually argue that they are logically implied X determines Z also holds ok.

And the appropriate name to do for this is obviously transitive call it transitive rule. Now in addition to this there are some other there are actually 3 more rules which are exactly I will show you in the next slide that they are not exactly essential. But they are nice to have them you know

so let us look at that actually. So one of this called the decomposition or projective rule, what this says is that.

Given that X determines YZ see if I want to now distinguish between 2 sets of attributes, that is why I have written 2 symbols here Y and Z, Y is a set of attributes, Z is another set of attributes. X determines YZ is given then it logically implies X determines Y goes ok. So what we have basically saying here is that we are only choosing some subset of the right hand side and then claiming that function dependency holds which is actually very obvious.

Because if 2 given that this happens that means any 2 rows if they agree on X they will agree on YZ it is given. They will obviously agree on the subset of YZ that they already are agreeing on the entire YZ. So we will agree on some subset of, this Y is a subset of YZ remember this we will continue to use this notation the 2 sets write them together it is unique ok. So this is called decomposition rule which basically allows us to kind of drop some attributes on the right hand side and then derive new functional dependencies from the given functional dependency.

Now the reverse of this is what is called the union rule if you are able to do that, then obviously we can do this. So if X determines Y holds, X determines Z holds, then we might as well combine that together and then write that X determines YZ ok. Given that X determines Y and X determines Z whole that means if 2 tuples agree on X. They are going to agree on Y, they are going to agree on Z.

So we might as well write that X determines YZ which is in some sense the inverse of the about rule the projective rule, so that is why it is called union or additive rule. Now here is a one more rule called pseudo transitive rule, what this says is that if X determines Y and some WY determines Z is given. Then you can derive WX determines Z it is not very actually difficult to see this. So if you start off with WX determines WX if they agree on WX then you know since they agree on X they will be agree on Y and since W is already there.

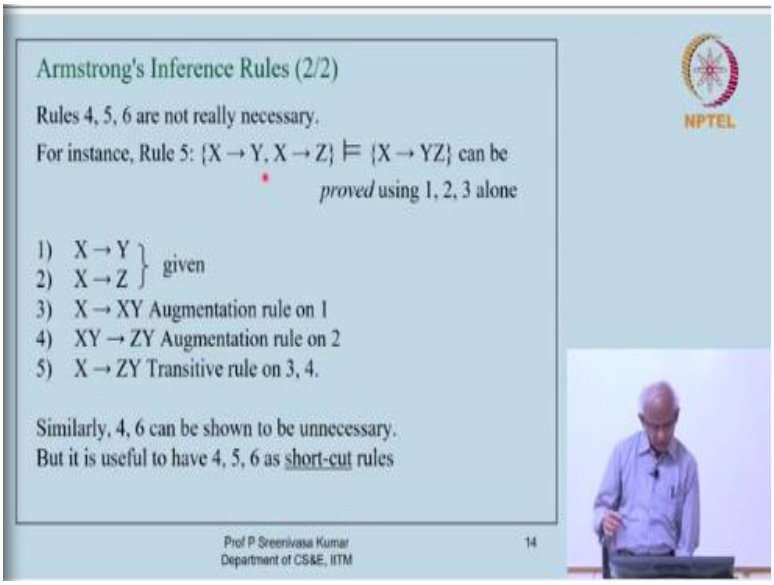
So if they agree on WY and it is given that they agree on Z, so you can see that they agree on WX they will agree on Z. We can actually formally prove by using, I will show you I will take

one of these examples and then show how exactly we can so called derived then using. In fact what we can show is that all these 4, 5, 6 are not really necessary. These that this is logically implied by this can in fact be proved by using 1, 2, 3 itself ok.

We will see that I think I will show you for this additive rule and I will leave it as an exercise for you to show the other proof ok. So basically what is happening here is that there are some functional dependencies that can be in some sense logically derived from the given some functional dependencies. And now we are finding that this researcher Armstrong has postulated given certain rules.

And what we actually find interesting about this whole thing is that these first 3 rows or AT or in fact necessary. And also sufficient to get hold of all the functional dependencies logically hold on a given a particular set of functional dependencies. In order to get hold of F +, these R A T this reflexive rule augmentation rule, transitive rule or necessary and sufficient. We will actually make that statement along with it, we will elaborate on next slide good.

(Refer Slide Time: 32:59)



Before we proceed to that let us take rule 5 rule 4, 5, 6 actually are not really necessary. We will take rule 5 and then see that we can actually prove that using 1, 2, 3 alone. So let me take this as a small exercise as to also illustrate to you as to how you if you are given some entailment

relation like this, how do you actually prove it. So you prove it by writing down these given things on into different lines as given.

And then start writing all the inferences you know what can be, so now given X determines Y extra X determines Z. We can use the augmentation rule on this 1 and then simply augment X on both sides augment it with X on both sides. So X union X is X itself and X union Y is XY, so we will get a new thing called X determines XY. Now take the other one X determines Z and augmented Y, XY determines YZ ZY.

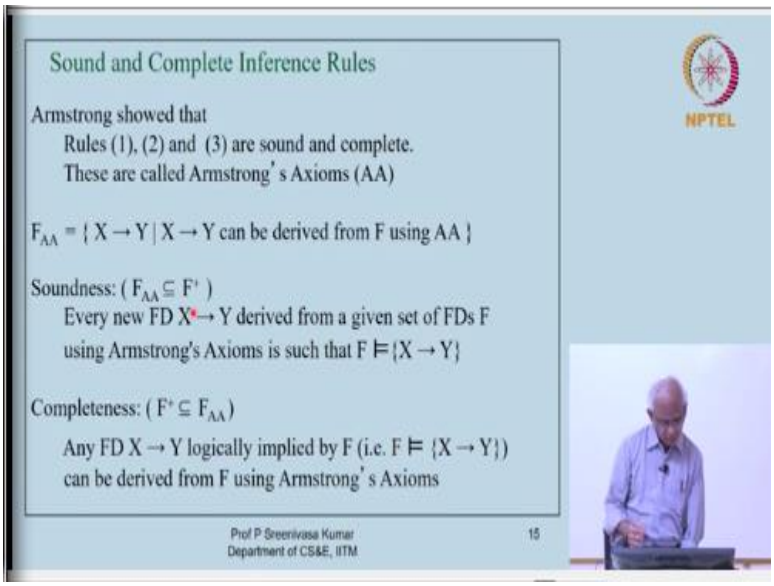
Now that you have X determines XY, XY determines ZY use the transitive rule and then you will get this X determines ZY or YZ. It does not matter whether you write it YZ or ZY because union is committed. So what is the rule here saying X determines Y, X determines Z logically implies X determines YZ we will be able to prove that X determines by making use of rule 1, 2, 3 reflects the sorry**.**

Rule augmentation rule 2 times and the transitive rule once, rules (()) (35:07) we have been able to be sometimes produce. So try this as an exercise for the other 2 rules also 4, 6 also you will be able to prove them is in just want to (()) (35:21) ok. But it is kind of useful to have this 4, 5, 6 as you know shortcut rules. Because every time you want to use this nice you know rule called the projective rule or union rule, you do not have to again you know use 1, 2, 3 depend first derive.

So well let us keep them as 4, 5, 6 as useful shortcuts but theoretically speaking they are not really necessary ok. So what is this story so far, the story so far is that normally functional dependencies have to be determined by the designers ok. But there are some kind of functional dependencies which can be you know logically derived from the given set of functional dependencies.

In order to understand this we have defined a notion called logical implication among the sets of functional dependencies ok and we call that has entitlement ok. And in that after we define this entitlement relation we have now seen that there are 3 rules called the reflexive rule, augmentation rule and transitive rule using which we can derive other functional dependencies.

(Refer Slide Time: 36:58)



Now the interesting question that comes up is that are these rules arbitrarily chosen or they you know good first thing is are they good and are they enough. So this also can be called as sound and complete inferences. So what Armstrong has not shown is that, he not only postulated these 3 rules but he is also shown that these rules are what are called sound and complete. In the sense that supposing I define this F I mean subscript double A for standing for Armstrong's Axioms.

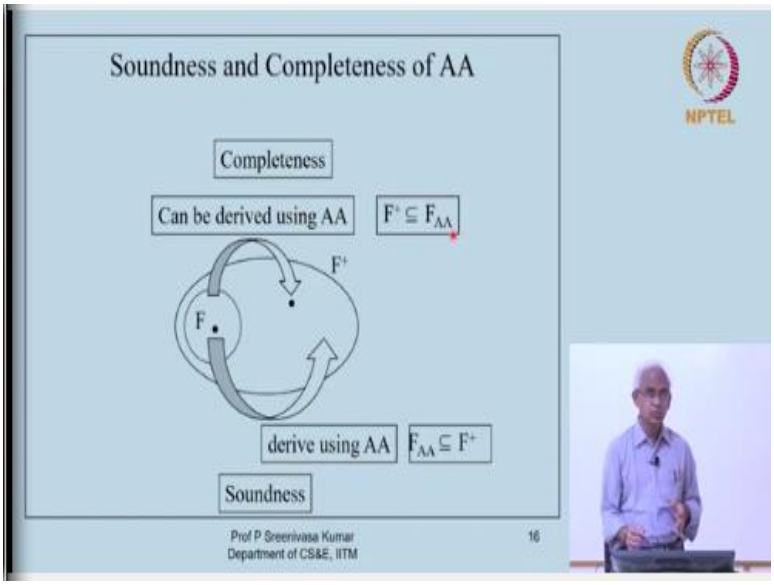
As the set of all functional dependencies, X determines Y that can actually be derived from F using Armstrong's Axioms. Now that we have Armstrong axiom we clear understood them, so you know also how to derive a new functional dependency given a set of functional dependencies. So let us call that set of all functional dependencies that can be derived from using the Armstrong as let us give it a new name F underscore double AA.

Now for soundness, what we have claiming here is that F double A is in fact subset of F +, what is F +, F + is a closure that is the set of all functional dependencies that can be logically derived from. So whatever the Armstrong's axioms derive or in fact correct ones. That mean they are in fact members of F +, so every new functional dependency X determines Y derived from a given set of functional dependencies F using the Armstrong's axiom is such that F indeed entails that.

So this is if you show this, this is called soundness and or correctness of these rules. The other one is called the completeness, what this says is that given any functional dependencies that is so what actually what this says is F + is a subset of F underscore double A. That means given any functional dependency X determines Y that is logically implied by F. That means is a member of F + it can in fact be derived from here using Armstrong's Axioms, this is surprising fact actually right.

So if you are able to show these 2 things then we have made really very strong statement about the Armstrong axiom is that. These rules are enough, do not look around for new rules these rules are enough ok. So that is very nice property of this Armstrong's Axioms. So in the next lecture, I will actually prove that the soundness and completeness of this Armstrong's Axioms indeed hold ok.

(Refer Slide Time: 40:13)



Now there is a nice picture that I have here to illustrate this thing, that this is F and this is F + ok. So basically derive using double A, so what we have soundness what claims is that whatever you derive using Armstrong's axioms falls within F + it does not go outside that is the sound. Now the other one is that taken something from here you take some functional dependency that is a member of F + just it can in fact be derived using F.

That is what the completeness F, F + is subset of so it is little bit interesting to prove these 2 facts. We will see them in the next lecture ok, I suggest you reflect on these 2 interesting properties of F transactions before we come to the next class. Actually try out a some schemes in fact you have all you know some derived some schemes. So on them, you try to figure out what are the functional dependencies that hold between attributes. Look at them from this angle ok, good.