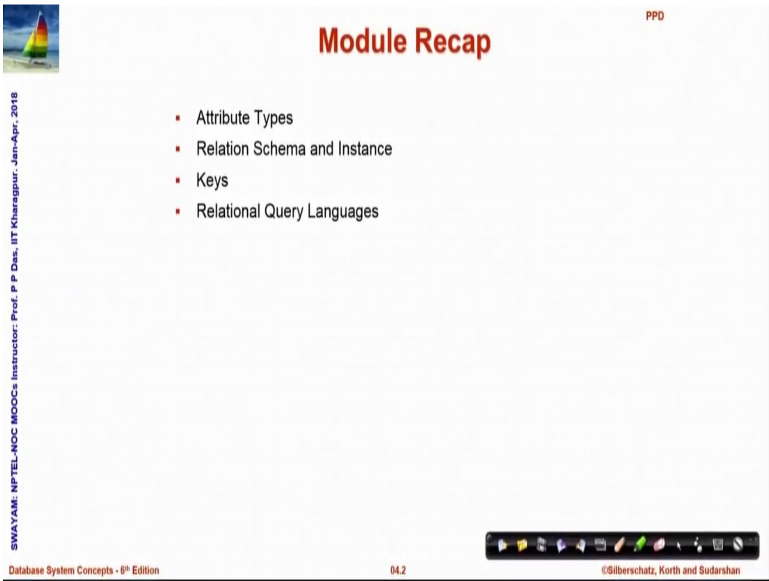
Database Management System
Prof. Partha Pratim Das
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture -04
Introduction to Relational Model/2

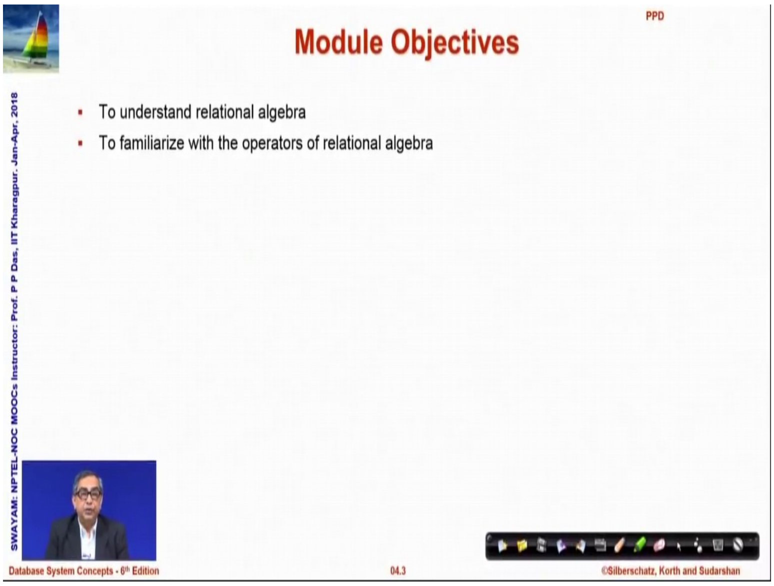
Welcome to module 5 of database management systems. In the previous module, we started discussions on introducing relational model we will conclude that in this module.

(Refer Slide Time: 00:36)



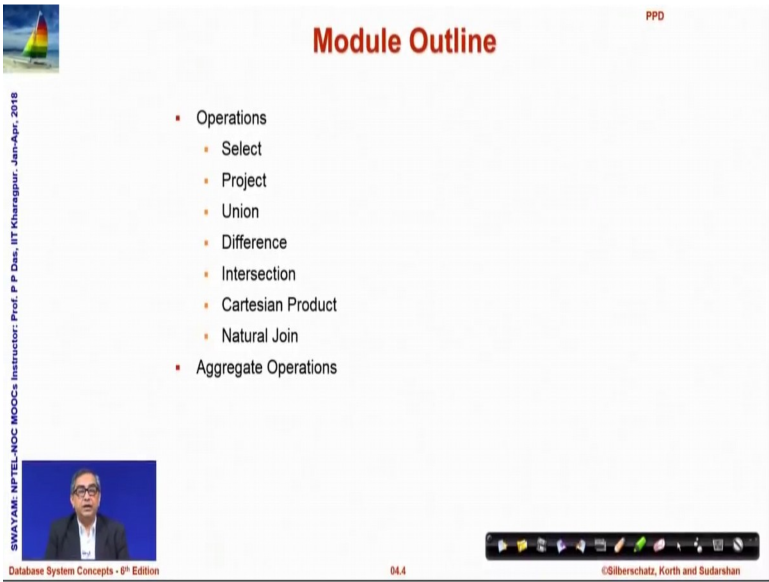
So, in the last module we have talked about attributes relational schemas and instances in mathematical form and very importantly we have tried to introduce discuss about the concept of keys.

(Refer Slide Time: 00:53)



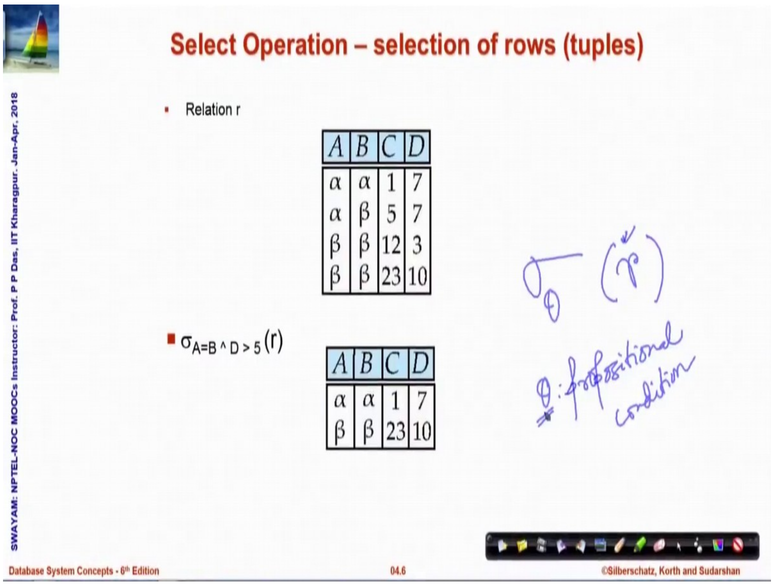
In this module we, will try to understand more on the relational algebra and familiarize with the operations of relational algebra.

(Refer Slide Time: 01:00)



So, these are the different operations that we will look at select project union and so on. Some of them are simple set theoretic operations some are newly defined operations and we look at the aggregators.

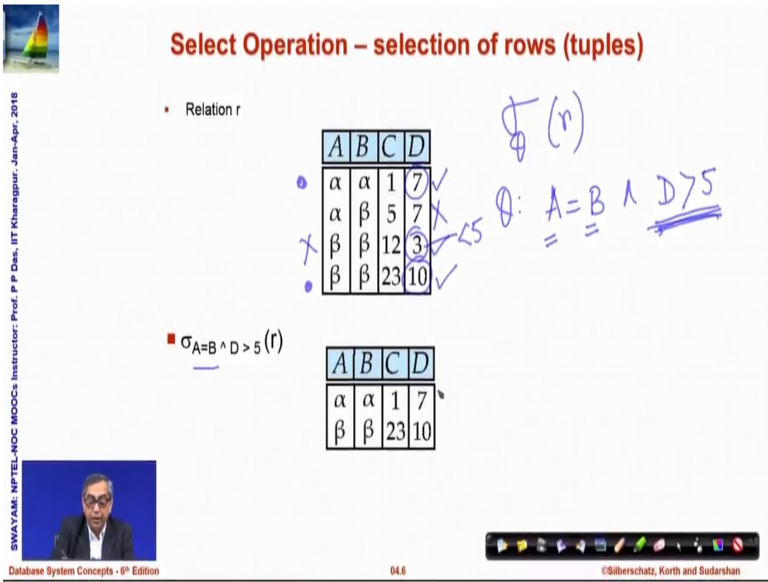
(Refer Slide Time: 01:18)



So, relational operators, so what is a relation as we have seen already? A relation is nothing but a table. It has a set of columns and it has a set of rows or records that fill up data according to those columns. Select is an operation, which chooses a subset of rows from a relation based on a certain condition. So, it is written in terms of in relational algebra we write it with a notion of notation of **σ** and following a parenthesis we put the name of the relation.

So, we say we are selecting from the relation r and then we put a condition here, which is a propositional condition. So, if for this all rows of r will be checked if a row will satisfy this condition. Then it will be included in the result, if it does not satisfy the condition, then it will not be included in the result.

(Refer Slide Time: 02:53).

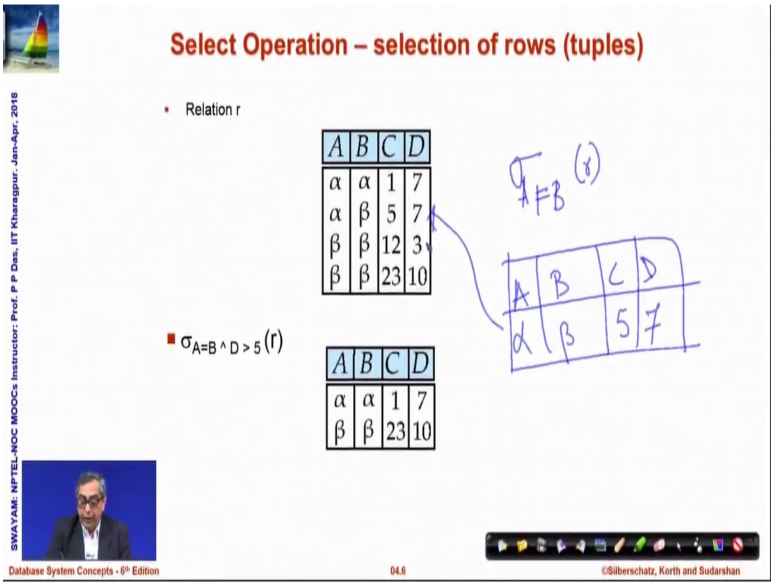


So, let us look at this example. So, our condition is sorry let this put this back. So, our condition **ϴ** is **A=B ^ D>5**. So, you are saying that that any row to be selected, the value of it is A attribute should equal the value of it is B attribute and when that happens the value of it is **D >5**. So, you can easily if we look through this we can easily say by the first condition **A=B** you can say that this row does not satisfy this condition.

Because, A is **α** and B is **β** whereas, these 3 rows satisfy because **A= β**. Then you again look at D we find that this **D<5**. We say this also does not satisfy, because it fails the second condition both have to hold. So, we finally, come to that this record and this record r the selected record in the result **α α** 1 7, **β β** 23 10. In both of these records **A=B** in both of these records **D>5**.

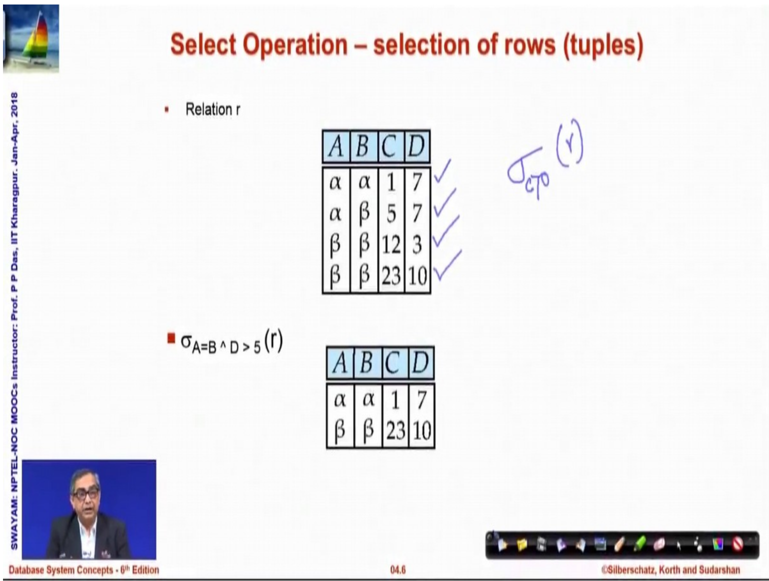
So, selection is a process is the operation which selects a subset of rows from a table, from a relation and creates a new relation. Based on a selection condition that must be true for all the rows for all the records that have been selected, but the set of columns do not change they remain the same.

(Refer Slide Time: 04:57)



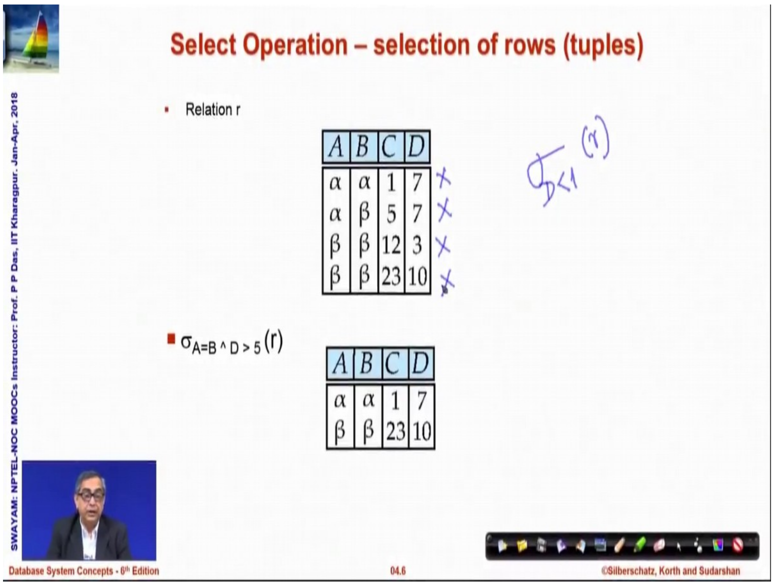
So, for example, if I if in the contrary if we say, that $\mathbf{\sigma_{A \neq B}(r)}$ then naturally, I will have a relation with fields A B C D and that relation will be only this row. Where that is the only row where **A≠B** we can.

(Refer Slide Time: 05:26)



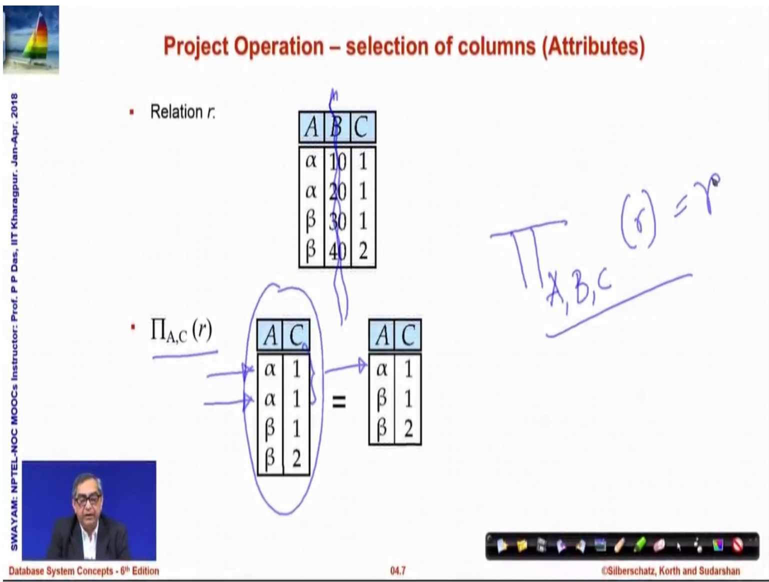
I can have a selection saying r where **C > 0**. Naturally, this will satisfy, this will satisfy, this will satisfy, this will satisfy. So, this whole relation would be the result of the selection. So, it is possible that it is not necessary that some rows will have to get eliminated in the result.

(Refer Slide Time: 05:54)



I say that this is the selection is **D <1** this will fail, this will fail, this will fail this will fail. So, all of them will fail so, it is possible that the result of an operation could be either the whole relation as we saw last time or a null relation which has where none of the record with feature because, none of the record satisfy the condition. So, that is a basic select operation.

(Refer Slide Time: 06:38)



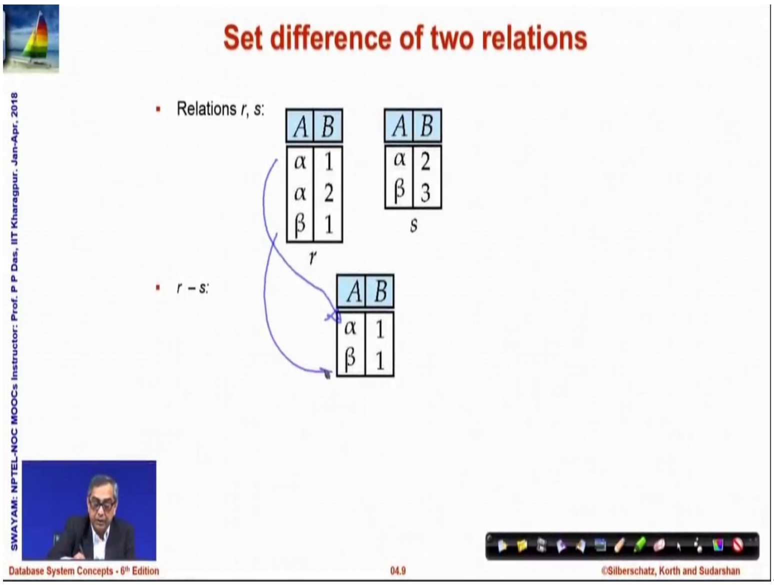
Let us move on look at the next one is called a projection operation. So, select chooses a subset of the rows. Projection necessarily chooses projects a set of columns from the original relation. So, this is a quite straightforward to see, it is written in terms of this notation **Π** and then you write the columns that you want in the result of projection. So, we say this is $\Pi_{A,C}$. So, which means basically the column, which is not selected in the projection, you can simply forget about that simply erase it. If you erase that you get this relation and once you get that, please recall that a relation is a set and in a set every element has to be distinct. So, after erasing B this first and the second row have become identical.

So, naturally both of them cannot be there, it will have to be made distinct by erasing any one of them. And hence, they become one row in the result; obviously, I can project on any of the infield single or all the fields also I can do a $\Pi_{A,\,B,\,C}(r)$of course, that means, in this case that will mean that it is a set which is equal to r anyway, but; obviously, I must have at least 1 column at least one attribute to project on. I cannot project on a null set of attributes because that does not give me a schema.

So, there will have to be some attribute one or more attribute on which I project. So, selection and projection, selection has given me the set of rows to retain and projection has given me what are the columns to retain in the result. And combining them I can do several different operations in a database table which can give me several interesting results. Before proceeding further,

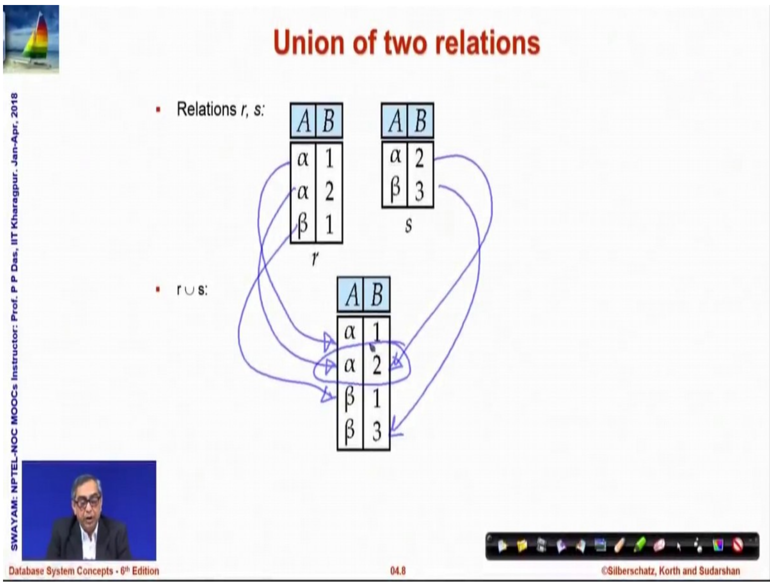
Let us look into some of the typical other operations that relational algebra allows. The next one is **∪** given two relations I can take a **∪** this is nothing but a set theoretic union. The two relations r and s must have the same set of columns A and B because, if the columns are not same, then the **∪** does not make sense. They because certainly if the columns are different attributes are different their types and type of data values would be different. So, they cannot be put to a same table so when two relations have the same set of attributes then their instances can be taken a union of.

(Refer Slide Time: 09:49)



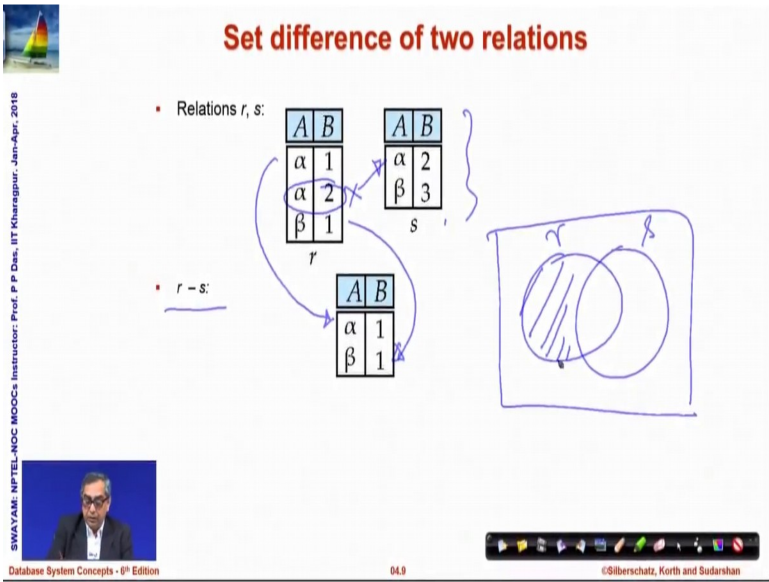
So, all records that exist in both these relations will be put together into a single table. So, here α1 is coming here β 1 is coming here.

(Refer Slide Time: 10:09)



In terms of relation, α 1 is coming here, α 2 is coming here, β 1 is coming here, β 3 is coming here and α 2 is coming here. You can see that this record α 2 exists in both the relations. And by the set theoretic notion of uniqueness, in the **U** they have to be unitified. So, one of them will be removed, it does not matter because they are identical anyway. So, for relations having the same set of attributes we can simply make a U of all it is records.

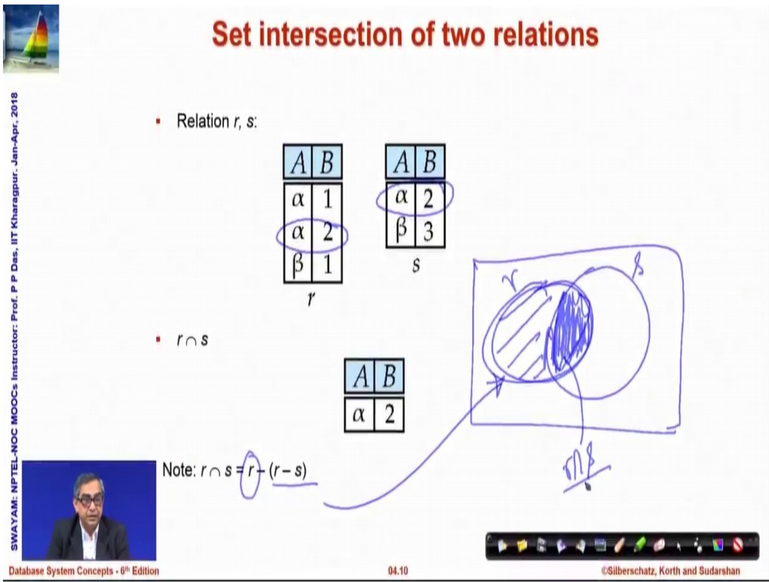
(Refer Slide Time: 10:57)



So, other the 3rd operation is the a 4th operation is doing a set difference it is works simply as set theoretic difference. Again, the two relations must have the same set of attributes and I can do a difference of **r-s** which mean that all tuples which exist in r, but do not exist in s will be included. So, this is included because, this is not here, but this is not included because it is in s this is included.

Because, this is this does not exist in the set s so it is the set, so you take the set r and then erase all the records like this which exist in s and you get **r-s**. So, if I look into just to recap if I look into the Venn diagram, then this is these are set **r-s ϵ r**, but **ɇ s**. So, there is a 4th operation that one can do with the in relational algebra.

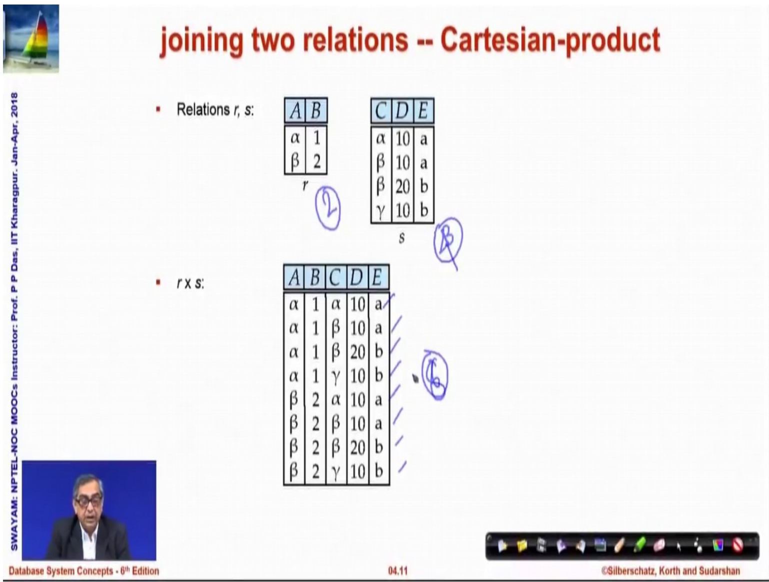
(Refer Slide Time: 12:28)



Fifth is a set intersection(∩) of 2 relations. So, again the two relations need to have the same set of attributes. You can take their intersection, which is the record which belongs to both. It is the record that belongs to both of them and as you are aware now, set intersection actually is not a new operation. It is not a fundamental operation because, if I have r, if I have s, then this is **r -s**.

Now, if I subtract this **r- s** which is this set from r which is this bigger set, then what will be remaining? This is what will be remaining? So, if I subtract **r- s** from r then what will remain? Is necessarily the intersection of this is **r ∩ s**. So, set intersection is not a fundamental operation of relational algebra. But, can be used because it can be expressed in terms of set difference.

(Refer Slide Time: 14:03)



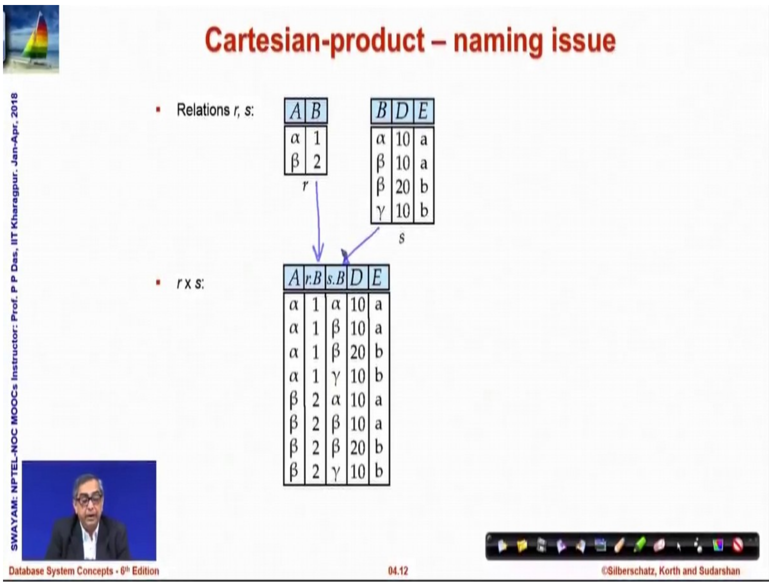
A	B
α	1
β	2

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Next comes how can we join two different relations which have different set of attributes? So, relation r has A B and relation s has C D E. So, we can take a Cartesian product(×). So, taking Cartesian product is making all possible combinations. So, necessarily if since this has two relations and this has 3 relations. So, this will have 1 2 3 4 5 6 7 8 this has 4 relations. So, these are 8, 8 total all possible pairing of relations of r or records of r and records of s are included. So, that is the Cartesian product all possible combinations. This is this is how we can join two relations, but certainly what is a important is something which we will discuss shortly.

(Refer Slide Time: 15:00)



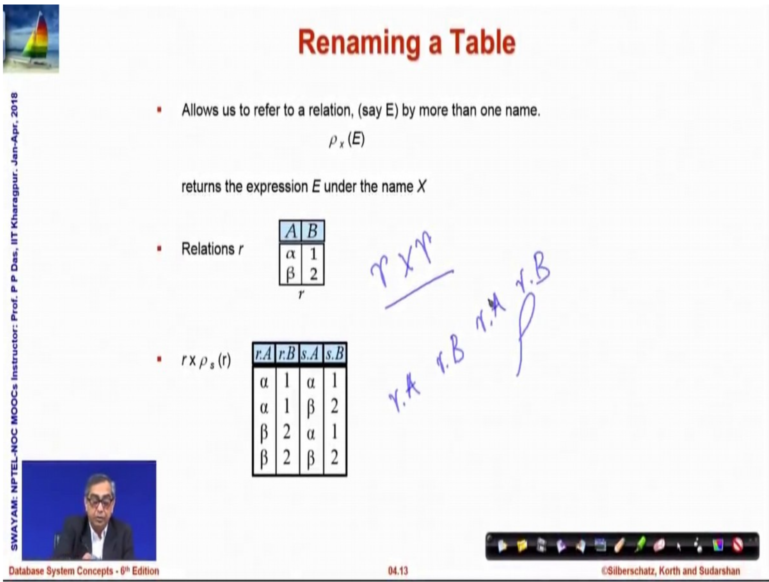
A	B
α	1
β	2

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

A	r.B	s.B	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

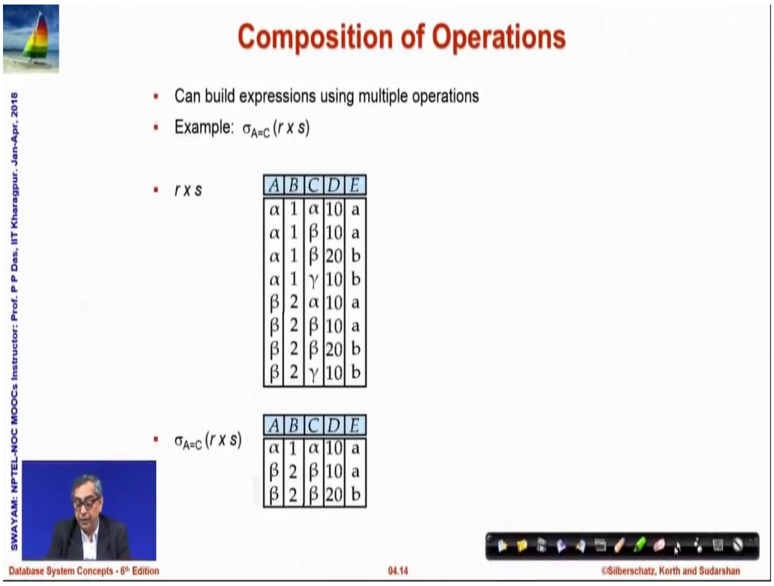
Now, in the Cartesian product then issue may happen because there could be attributes which are common between two relations. So, if you if two attributes are common when you take Cartesian product how do you put their name? Because as with the example show here between r and s the attribute b is common so, how do you take care of that? So, when the, such common names happen then we actually change the name of the attribute with the name of the relation. So, b coming from r will be called **r.b** and b coming from s will be called **s.b**.

(Refer Slide Time: 15:46)



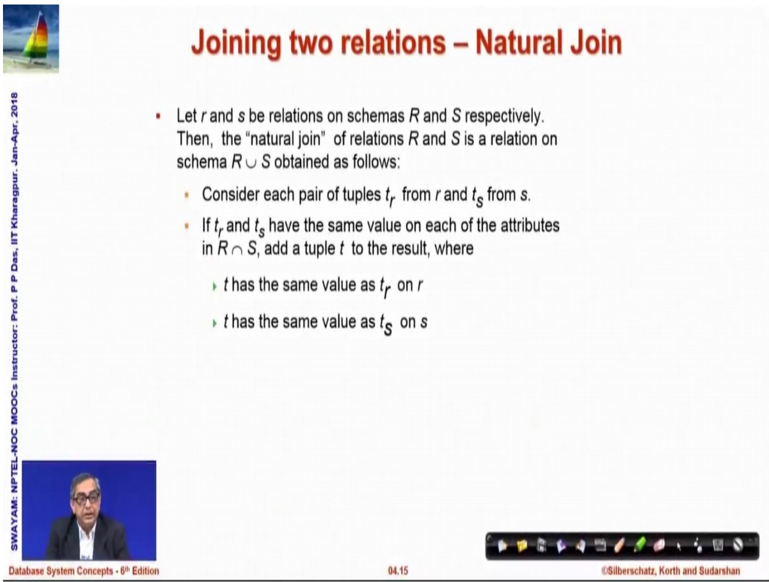
And accordingly, the relational algebra, gives you a way to rename a table and put it is name differently. So, this is given a this is given by this relation is given by this symbol **P**(rho). So, you can using a relation r you can actually give it a different name s and do that in terms of. So, with that you can actually because otherwise you cannot compute **r** × **r**. Because, if you try to do compute **r** × **r** then you will have **r. a r .b** and you will again have **r. a r .b**. So, you are using this to rename r to s and then compute this. So, renaming a table is another feature which is provided of course, it is not a fundamental operation of the algebra, but this is what makes the any kind of Cartesian product possible.

(Refer Slide Time: 16:55)



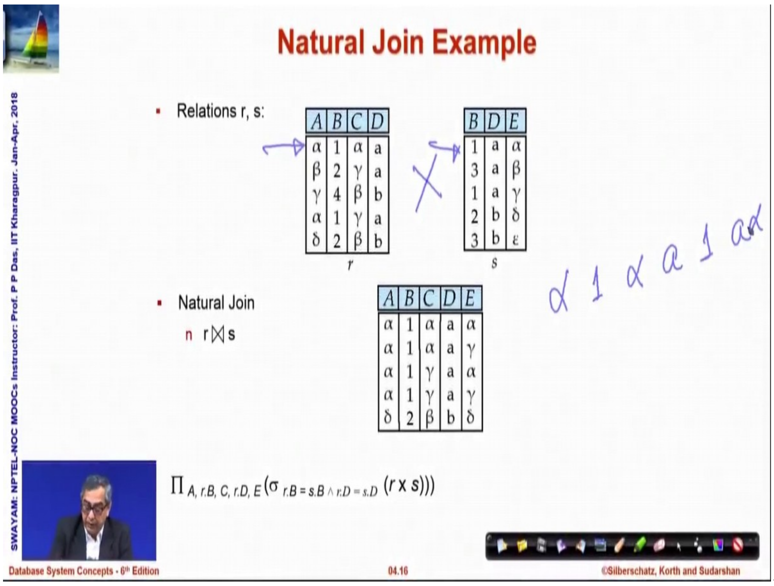
Finally, we can make composition of operations that if for example, what we show here is we have two relations r and s you have taken a Cartesian product and then we have taken a selection. So, taken a Cartesian product of r and s to produce the table as you can see the process($\mathbf{r} \times \mathbf{s}$) and then $\boldsymbol{\sigma}_{\mathbf{A=C}}$ $(\mathbf{r} \times \mathbf{s})$ based on that condition. So, all these operations can be combined in multiple different ways to give you really complex relational algebra operations.

(Refer Slide Time: 17:33)



There is a nice operation which is a derived one which can be written in terms of other operations which is called a natural join(⋈) which we will use very heavily.

(Refer Slide Time: 17:48)

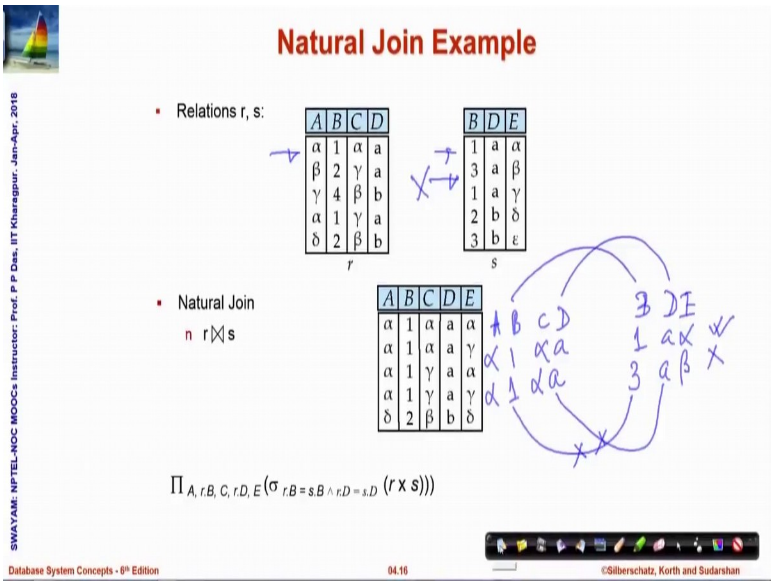


Let me first show you an example of that, suppose I have two relations r and s and what is important is there are some attributes which are common between them. Now, we saw earlier that in Cartesian product, in terms of common attributes. We basically the attributes got renamed in terms of the table name, but this is not what we are looking at in relation natural joint.

What we want to say is, if an attribute is common between two tables then, while you join them, the records from two fields can be joined if their value on that common attribute is same. So, it is what it tries to do is it tries to make a Cartesian product of this two tables ($\mathbf{r} \times \mathbf{s}$)first take all possible combinations, but then you select only those rows where the values are identical between columns having the same name. So, for example, if you if you if you look into this row and this row. So, what will happen in the Cartesian product I will have this **α 1 α a 1 a alpha**.

Let me write it in smaller. So, I am doing a Cartesian product I am looking at this row I am looking at this row.

(Refer Slide Time: 19:39)

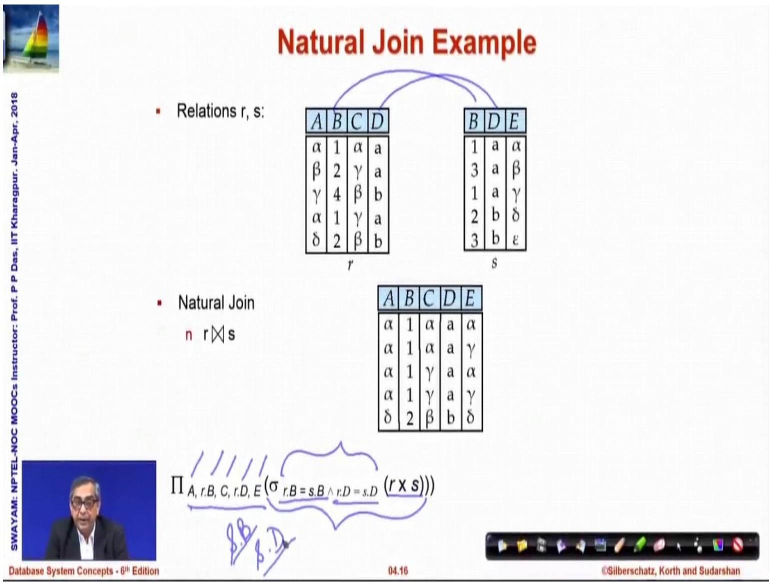


So, I have A B C D I have B D E and **α α** a 1 a **α**. Now, they match on B they match on D. So, I will say this is this will get retained, but in the Cartesian product I will also have the first row of r going with the second row of s **α** 1 **α** a 3 a β. Here, the B does not match D does not does match, but the B does not match. So, this particular entry will not go in the final result. To take the Cartesian product and you only retain those rows where the values match for the identically named attribute. That is why, so you take the Cartesian product now look at the expression the attributes common attributes are B and C, B and D.

So, the B attribute is **r.B** and **s.B**. So, we say that in the Cartesian product **r.B** = **s.B**. The name is common; the value will have to be same similarly D is a common attribute. So, r.D value in the **r.D** and the value in the **s.D** has to be same. So, based on the Cartesian product you do a selection for equality of values on attributes which are identical between the two column between the two relations between the two tables.

This is a final selection, as you do that you get a table where there are two Bs r.B s.B there are two Ds r.D s.D, but according to this selection for all at all records for all rows the value on r.B and value on s.B are same value on r.D and value on r s.B are same. Because that is how we have done the selection. So, there is no reason to keep two B columns or two D columns.

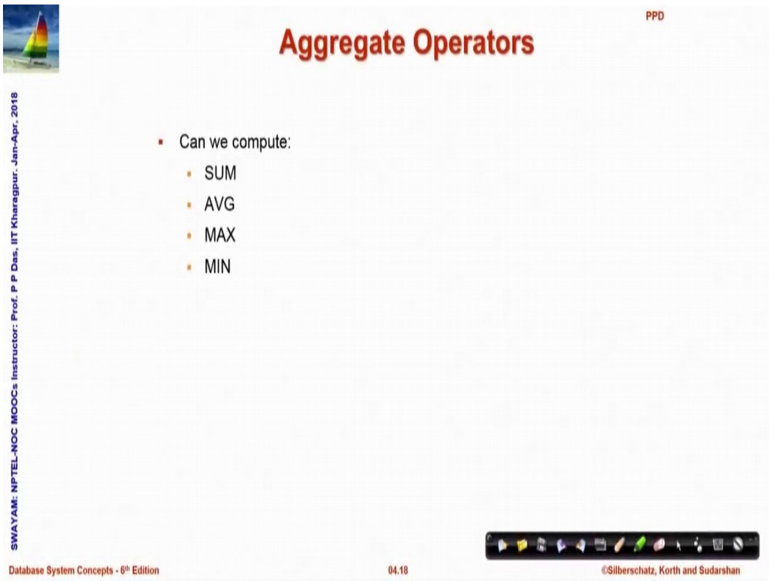
(Refer Slide Time: 22:33)



So now, you project based on **A r .B C r.D and E** which means that s.B and s.D are left out you do not project them. So, after you have done this projection, you get the final result of the natural join, which has a union of all the attributes that the two relations at A B C D and B D E union is A B C D E and you have all those records whose values matched on the common attributes between relation r and relation D. So, you can say that if I now do a selection, if I now do a projection on A B C or rather A B C D.

I will get a subset of r if I do a projection on B D E I will get a subset of s. So, this is the natural join operation in relational algebra as you can see this is a derived operation because we could use the Cartesian product selection and projection to get this, but as I tell you we will see more of this when we look at the look at all these different aj query coding, but natural join is one of the most widely used most fundamental relational algebra operation that you will often need beyond selection and projection. So, these were the 6 operations and the important derived operations of relational algebra. Besides that relational algebra has some aggregation operators.

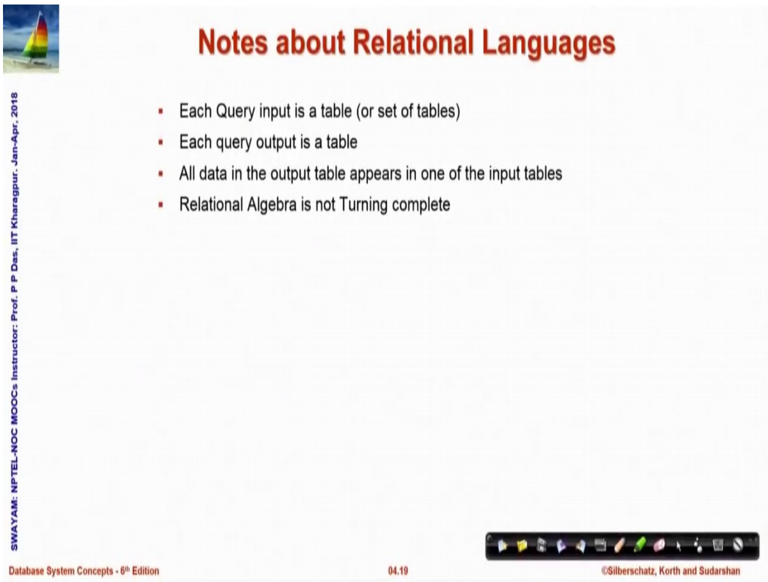
(Refer Slide Time: 24:13)



For example, given a table we could compute the sum of values on a column we could compute average of values, max of values, mean of values. So, these somehow aggregate values of multiple rows on a particular column. And therefore, these are called aggregate operators we will see when we talk about SQL. We will see how this really can be coded in SQL and used, but these are this become very convenient to use because often we will need to know.

If this is I mean these are the instructors. And so, let us see which instructor has a maximum load of courses? How many based on hours or which instructor has? What is the average load on the different instructors and so on? So, in every possible context different aggregate operators are frequently required and they are also available as operators in most of the pure as well as commercial query languages.

(Refer Slide Time: 25:18)

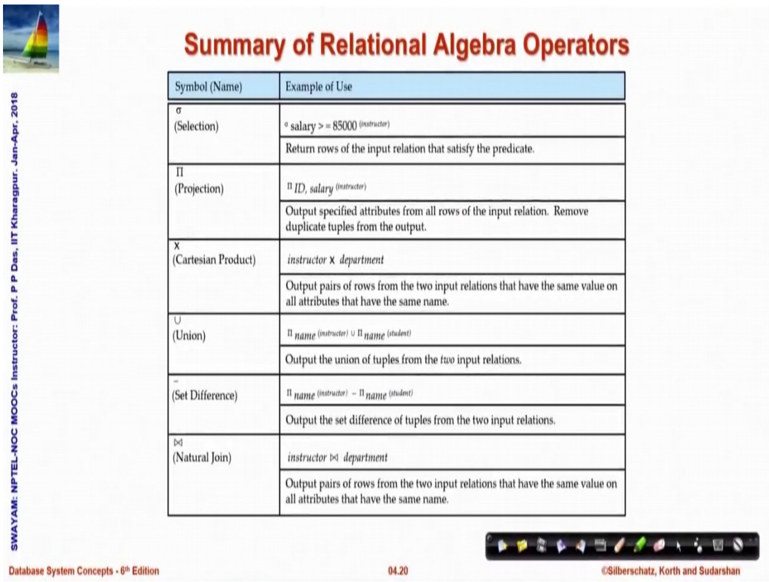


Finally, to note that relational algebra in relational average every query input is a table. And the output is also a table, it is always manipulating one or more tables into a single table that is what we are, I mean in very simple terms that is the way you can look at it. And all data in the output table appears in one of the input tables that is no new data gets generated. It is basically taking, selecting, combining data from different input tables. It does not generate a new data that is that has to be that is if I see that a, we an attribute for a particular row has a value 15.

Then there must be some input table where there is a record where in that field as an attribute value 50. Otherwise, this cannot happen again relational algebra is not turing complete in the sense that, there are algorithms which cannot be coded in relational algebra. We mentioned this earlier too and that is the reason that is a foundational reason of why the SQL language commercial SQL language is based on relational algebra is not turing complete there.

So, we might need to use other programming languages along with the relational algebra coding for solving some of the application problems.

(Refer Slide Time: 26:50)

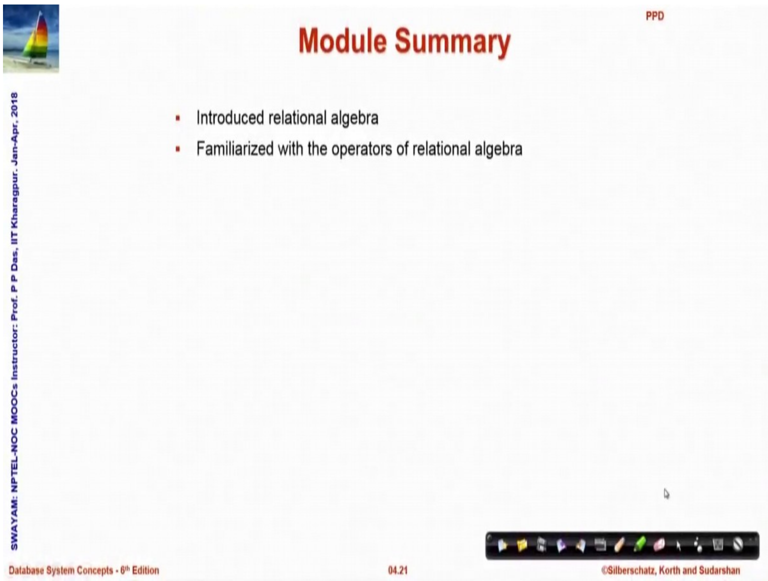


To summarize this is the, so this table is what you not only should remember, but you should become a expert of in terms of the operators of relational algebra which we will start using very heavily as we start doing the query coding and processing. So, we talked about selection, which takes rows selectively we talked about projection which takes out certain columns of a table. We talked about Cartesian product of two relations which make all possible combined relations.

We talked about union of records from two tables having identical set of attributes. We talked about set difference which again is the difference of records of one relation from another, given that they have identical set of attributes. We have shown that set difference can be used to also compute set intersection, it is not in the fundamental operation, but is the derived 1 and we have shown a very interesting operation based on Cartesian product selection and projection called natural join where two tables can be joined based on 1 or more common attributes they have.

Now, I am sure you have already noted that if I am doing a natural join between two tables. Which do not have any common attribute then the result is merely the Cartesian product because the selection around the Cartesian product has no condition to select any of the you know any of the fields any of the rows separately. So, it merely turns out to be a Cartesian product.

(Refer Slide Time: 28:38)



So, in this module, we have introduced the relational algebra and we have familiarized ourselves with the fundamental and derived operators of relational algebra. Going forward in the next module, will take a deeper look into the relational model and start progressing towards the query design and database design.