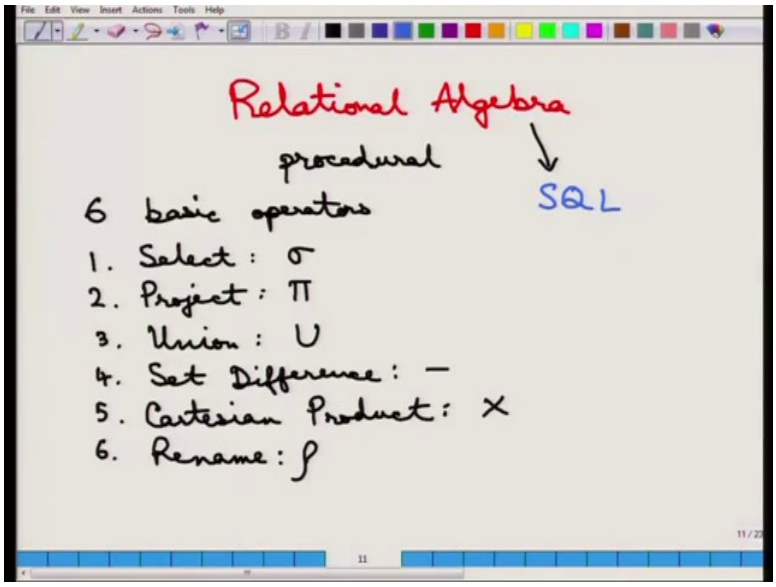
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 03
Relational Algebra- Basic Operators

(Refer Slide Time: 00:15)

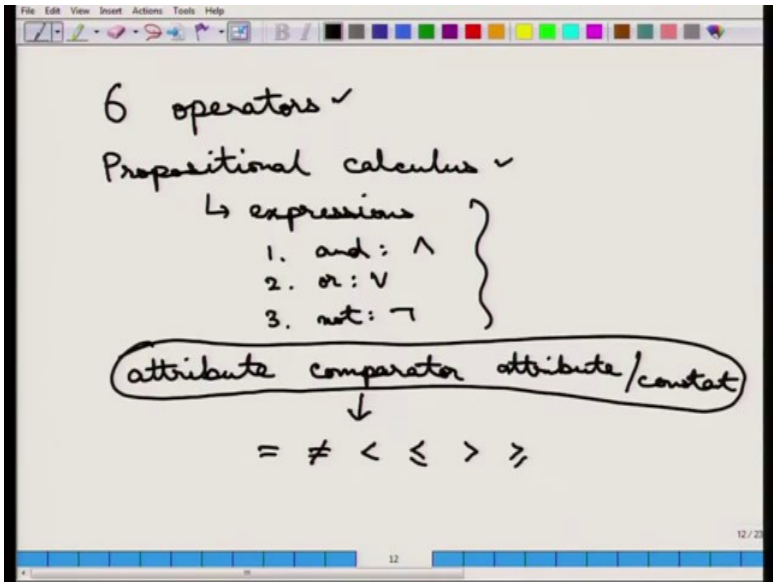


Welcome back, so we will start on our next module, which is on **Relational Algebra**. So, this module is on relational algebra and what do we mean by that, is that, this is a procedural language, to define the database queries. So, they are very important distinction first of all, that this is a procedural language, so this is to define the database queries.

Essentially we will first talk about 6 basic operators; that is the first module that we will be talking about. And the 6 basic operators are, the first one is called **select**, which is denoted by sigma. The second one is **project**, which is denoted by pi, then **union** or the set union essentially, **set difference**, then **Cartesian product** and the last one, the sixth one is called **rename** operator.

Now, before we go forward let us understand that relational algebra is what forms the basis of the SQL or the Structured Query Language that, we will study later. So, relational algebra defines the theory behind SQL, it is not exactly the same as SQL, but it defines the theory. So, let us go over.

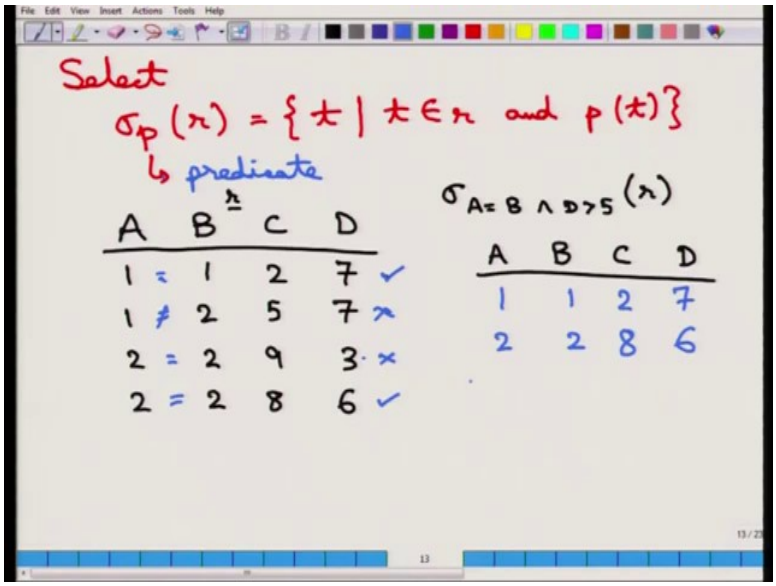
(Refer Slide Time: 02:01)



Other than the 6 operators, so there are these 6 operators that we are going to study, then relational algebra uses propositional calculus. So, this consists of expressions, in the propositional calculus, there are expressions. The expressions are connected by this three basic operators AND, OR and NOT. So, the each term is of the form attribute, there is a comparator with an attribute or a constant.

So, each term is essentially an attribute is compared with another attribute or a constant. And the comparators are the standard 6 comparators, which is equal to, not equal to, less than, less than equal to, greater than and greater than equal to. So, this defines the entire relational algebra, the basic relational algebra. So, we have these 6 operators, then the propositional calculus of these three expressions and the terms are consisting of this form attribute comparator with attribute and constant.

(Refer Slide Time: 03:35)



So, the let us go to one by one over the 6 basic operators, the first one is the **select** operator. Now, just to note that this is different from the select that we understand in SQL, we will go over all of them in much more details, but, the definition of the select is the following. Now, here I am trying to write down a formal definition of select, remember that r stands for a relational instance, so it is a relational. So, sigma is the select operator and p is the predicate on which the selection is done by the way, so this selects all tuples t; such that of course, t is the part of relation. So, it selects all tuples from the relation and p of t. What does this mean? This means that, the predicate, if one applies the predicate on this tuple t that is correct, so this is called the predicate, so this predicate is correct. So, just once more to say, this essentially selects all predicates this selects all tuples from r on which the predicate is satisfied.

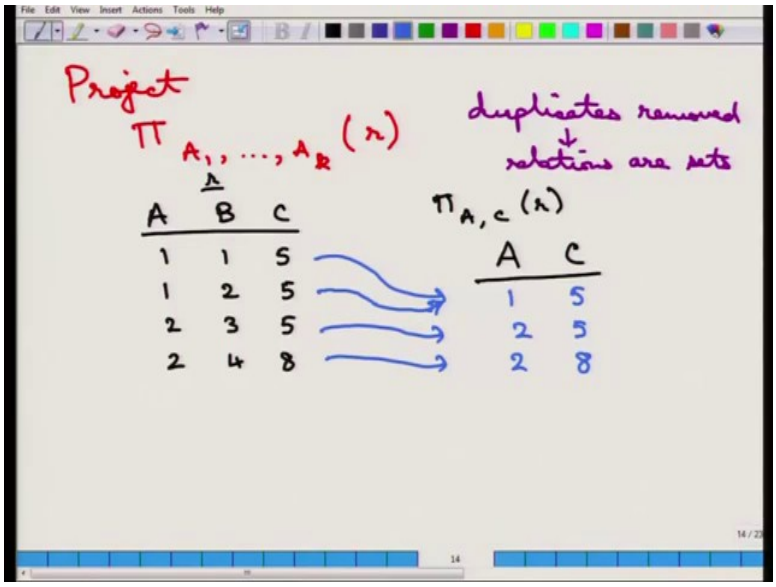
One important thing to notice that, if you do it on a relation r, then it does not change the schema of r. Now, let us take an example; that is the best way of understanding this, so here is the example. So, suppose a relation has got four attributes A, B, C, D. which are these values, let me just write down some values. So, there are just four tuples in this relation r and now, suppose we are applying the following selection on this A equal to B and D is greater than 5 on this r. So, then what is the answer, so first of all, when we say that it does not change the relationship schema.

So, the first thing to note down is the answer will be of the form A B C D as well again,

nothing will be changed. So, what is the way to do it is that, let us test the first tuple. So, we test whether A equal to B, which is A equal to B is correct and B is greater than 7, so this is correct. So, this becomes part of the answer set.

In the next one again we test A equal to B, no A is not equal to B, so this is wrong this is not part of the answer set, this one 2 equal to 2, which is fine A equal to B, but 3 is not greater than this, so this is also not part of the answer set. And the fourth one 2 is equal to 2 and 6 is greater than this, so this is part of the answer set. So, the answer consists of these two tuples of the same form (A, B, C, D) with (1, 1, 2, 7) and (2, 2, 8, 6). So, that is the select operator, let us move ahead with the next operation, which is called the project.

(Refer Slide Time: 06:35)

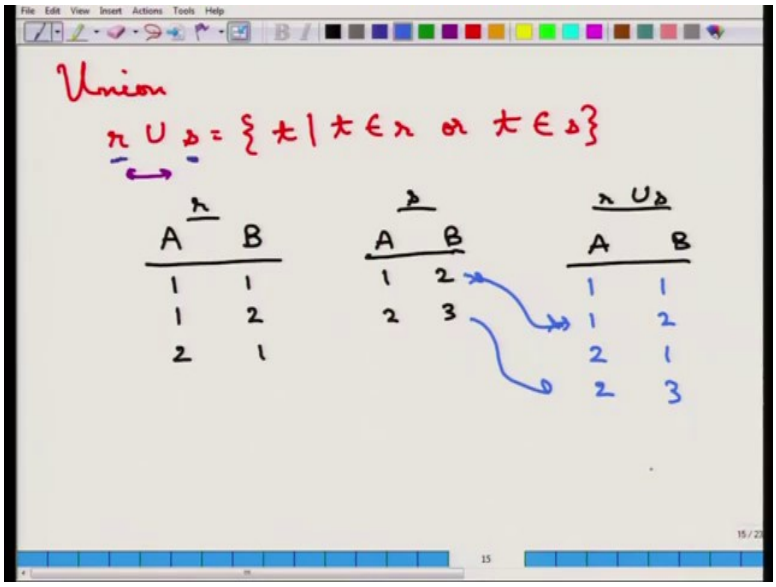


Now, **project** does change the schema of the relationship that even we apply it on. So, project essentially is of the form (Refer Slide Time 06:48). So on a relation r the project operator is applied on certain attributes A_1 to A_k and it essentially just selects out those attributes from the projections. And duplicate rows are removed, so duplicate is removed this is one important thing with the basic projection, duplicates removed. This is done, because relations are sets, so that is why duplicates are removed.

So, let us take an example, so suppose there is this r is your (A, B, C) with the following four tuples (1, 1, 5), (1, 2, 5), (2, 3, 5) and (2, 4, 8). Now, suppose the following projection is done A, C on r, so the projection the schema of this is the same as whatever is projected on, so if A and C are projected on, the schema is A and C. And then, it is simply selected out, so from

here this simply comes as (1, 5), then here this again should have come as (1, 5), but then it merges. So, it does not produce any more tuple. From here it produces (2, 5) and from here, it produces (2, 8); the answer for the project is simply this.

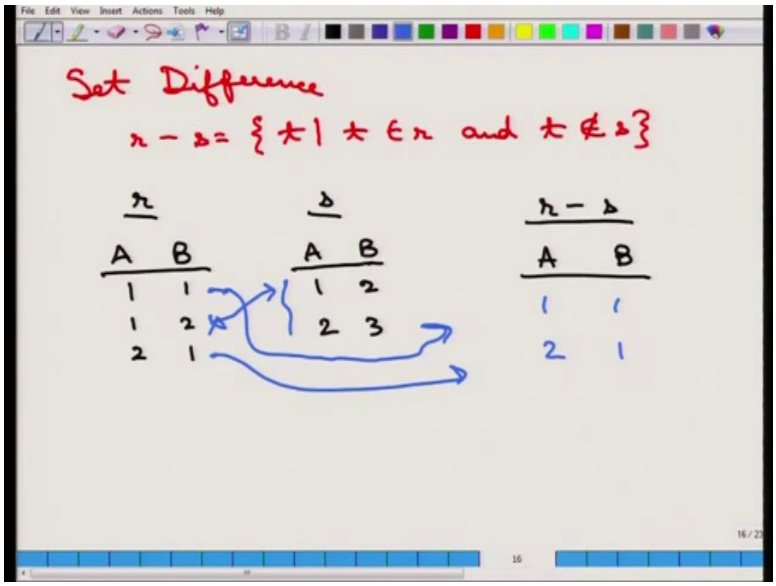
(Refer Slide Time: 08:25)



The next operation is the **union** or the set union. So, set union of the *r* union *s* is all set of tuples; such that $t \in r$ or $t \in s$, this is very easy to understand. This is exactly the same as the set union, once more duplicates are removed, because these are rows. So, one important thing is that *t* and *r* and *s* should have the same attributes, if they do not have then some renaming needs to be done, such that *r* and *s* follows the same thing.

And let us just work out an example, so suppose this is A, this is (A, B), this has got (1, 1), (1, 2), (2, 1) and *s* has got the same schema (A, B), which has got (1, 2), (2, 3). So, the union of r union s produces again A B and this is simply copied, so (1, 1) is copied, (1, 2) is copied from A and (2, 1) is copied from A. Now, then (A, B) from B is not copied any further, it duplicates it, this is duplicated, so (2, 3) is simply copied, so it forms this. So, these are simple things to understand, so union again has the essentially just they set union, nothing more.

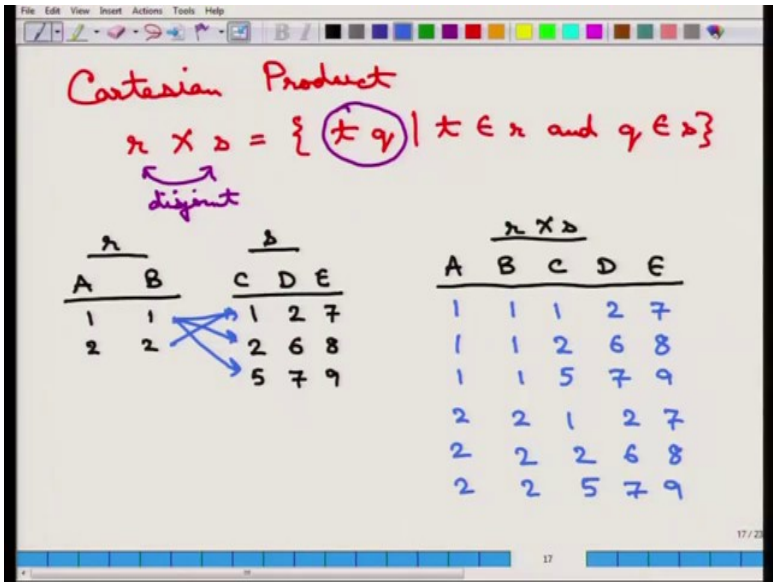
(Refer Slide Time: 09:52)



The next operation is also the simple set operation which is the **set difference** and this is again, what we understand from normal set operation is nothing much difference. So, r minus s is the set of all tuples; such that t is in r and t is not in s. So, once more this does not change the schema and renaming etcetera is done and set, so because these are sets duplicates are removed and an example is the best way.

So, r this has got A B, which is (1, 1), (1, 2), (2, 1). *s* has got (A, B), (1, 2), (2, 3), so $r - s$ everything that is in *r*, but not in *s*. So, (1, 1) is not here, so (1, 1) finds its way through. (1, 2) it is here, so (1, 2) doesn't find its way through. (2, 1) is not there, so (2, 1) finds its way through as well. So, it is just (1, 1) and (2, 1). So this is the set difference.

(Refer Slide Time: 11:08)



So, then the next operator is **Cartesian product**, so Cartesian product is a little bit more interesting. So, it takes out two sets $r \times s$ the first thing that it does is that the schema is changed. So, the tuples that have produced have attributes from both *r* and *s*. So, what it does is that it produces tuples of the form *t q*; such that the *t* part of it comes from *r* and the *q* part of it comes from *s*.

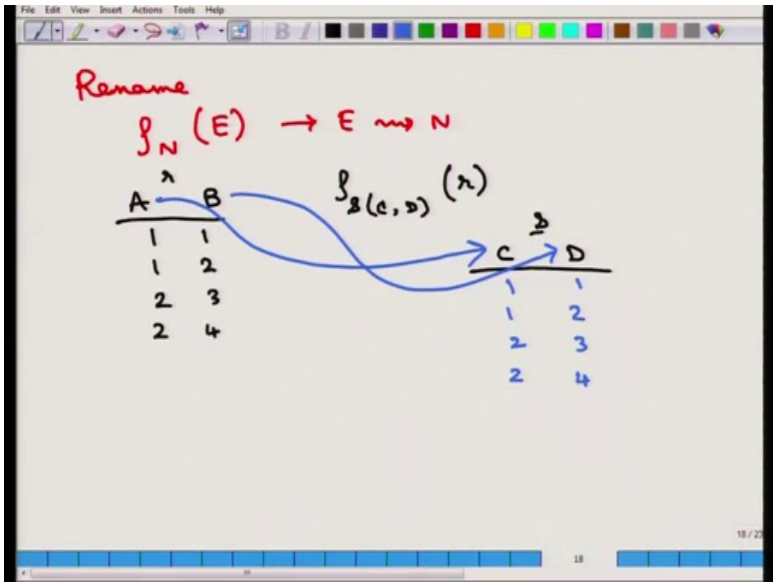
So, very importantly this is of the form (*t*, *q*), which is not the same as either *t* or *q*, so it changes the schema. And, if the attributes are disjoint, then there is no problem it is just an addition of the schema, the addition of the attributes of r and s and if these are not disjoint then some renaming must be done. So, what do we mean by that is that, we will go over all of these things in probably much more detail later, but just to highlight it, what it means essentially is that if both r and s have the same attributes A, then it should be called r**.**A and s**.**A.

Now, here is an example, a complete example on *r* and *s*. The schema of *r* is (A, B) and suppose it contains, simply (1, 1) and (2, 2) and suppose *s* the schema of *s* is (C, D, E) and it computes (1, 2, 7), (2, 6, 8) and (5, 7, 9). Now, $r \times s$, the first thing to notice, what is the schema of going to be, $r \times s$ now since the attributes sets are disjoint. So, the schema is essentially all of them, the union of all of them, it is (A, B, C, D, E).

Now, what are the values of these, the first thing is that essentially, what is done in a Cartesian product is that first tuple from r is taken and it is applied and the attribute union is

applied with all from s. The first tuple that it produces is (1, 1, 1, 2, 7), then it is (1, 1, 2, 6, 8), then it is (1, 1, 5, 7, 9). Now, similarly the second one is taken and all of these things are done, so it is (2, 2, 1, 2, 7), (2, 2, 2, 6, 8), (2, 2, 5, 7, 9), so that is all about the Cartesian product, so it changes the schema.

(Refer Slide Time: 13:56)

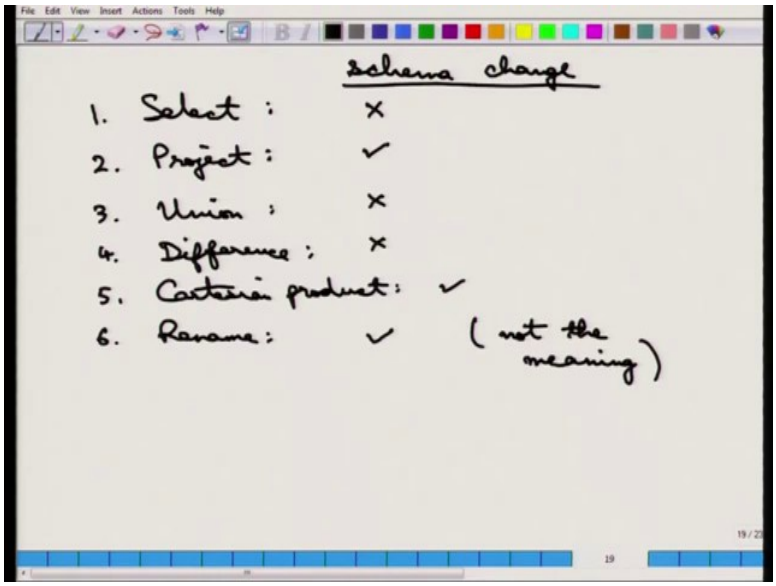


The next basic operator is called **rename**, it is a very simple operation it essentially just renames, so it renames it simply returns E, but now its name has been changed to N that is all. So, it simply renames it and it does not do anything else, so the very simple example of this first thing is that suppose this is (A B). So, this is (1, 1), (1, 2), (2, 3), (2, 4) does not matter, this is *r* and the operation that is applied is on *r*, which is called, let us say you change it to (C, D) on *r*.

So, what it returns is *s* with these values (C, D) this is the same *s*, so it simply copies on (1, 1), (1, 2) there is nothing more, but what it has done is that A has been changed to C and B has been changed to D. Rename of the operation has been done. And, so the schema is actually changed, but not the meaning of it.

Now to summarize all of these things, so let us see.

(Refer Slide Time: 15:16)



So, the first one was select. "Schema Change". Select does not change the schema. The second one was project it does change the schema. The third one was union it does not change the schema. The fourth one was difference, it does not change the schema. The Cartesian product, of course, does change the schema. And rename actually changes the schema, because instead of (A, B) it returns new name (C, D), but not the meaning. The meaning is not changed. So, that is all about these things and then, we can take all of these operators and can compose them, so we can apply them one after another and that is what I mean by composition of operators.