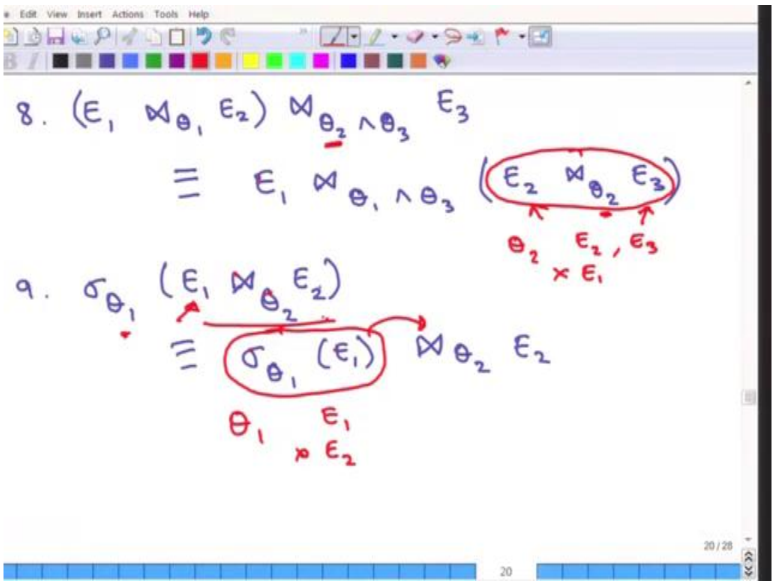
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 25
Query Optimization: Complex Equivalence Rules

(Refer Slide Time: 00:18)


8. $(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3$
$\equiv E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$
θ_2 E_2, E_3
$\times E_1$
9. $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2)$
$\equiv (\sigma_{\theta_1}(E_1)) \bowtie_{\theta_2} E_2$
θ_1 E_1
$\times E_2$
20/28
20

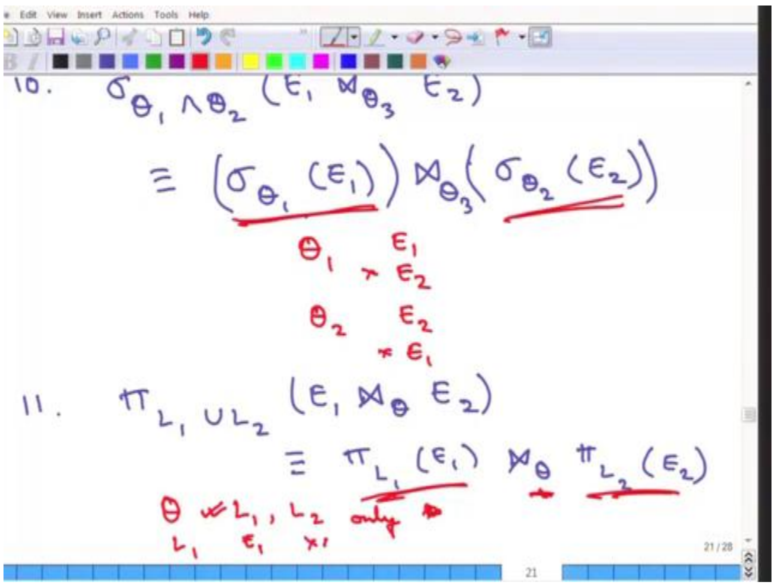
The previous rules were all unambiguous, it does not matter, what the conditions etcetera are and now, I am going to write down a little bit more complicated rules. So, this is 8, E 1 when it is joined on a condition theta 1 with E 2, which is then joined on a condition theta 2 and theta 3 with it each. Note that, these are not natural joins any further, this is equivalent to E 1, this is joined with theta 1 and theta 3, which is then joined with E 2 on theta 2 and E 3.

So, couple of things needs to be there is that, theta 2 must involve attribute from only E 2 and E 3. So, theta 2, it is only between E 2 and E 3, theta 2 is involves attributes only from E 2 and E 3. So, it does not contain any attribute from E 1, it cannot contain any contribute from E 1. This is the condition that has be there, if that is the thing, then theta 2 can be separated and theta 2 can be applied first between E 2 and E 3 and then, it can be joined with E 1, so that is the idea, because theta 2 does not bother itself about E 1.

So, E 1 can be done later, it can be separated out and as very similar rule is the following, E 1. This is some theta and E 2 is a theta 2, this is equivalent to doing the and then, doing the join. Once more, here the condition is the theta 1 involves attribute from only E 1 and it does not involves attribute from E 2. So, in that case what happens is that, this E 1 and E 2 will going to contain attributes from all E 2, but since theta 1 contains attributes from only E 1, the selection is not going to do any selection on E 2.

So, the selection can be done that can be done, a pre selection can be done even and then, that can be joined using the condition. So, this is true only when theta 1 does not contain any attribute from E 2, because it can be as well applied on E 1 earlier, because it is not going to affect, whether E 1 is joined with something else.

(Refer Slide Time: 02:40)



10. $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta_3} E_2)$

$$\equiv \underline{(\sigma_{\theta_1}(E_1))} \bowtie_{\theta_3} (\underline{\sigma_{\theta_2}(E_2)})$$

θ_1 E_1 $\times E_2$

θ_2 E_2 $\times E_1$

11. $\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2)$

$$\equiv \underline{\pi_{L_1}(E_1)} \underline{\bowtie_\theta} \underline{\pi_{L_2}(E_2)}$$

θ L_1, L_2 only

L_1 E_1 $\times$

21/28

21

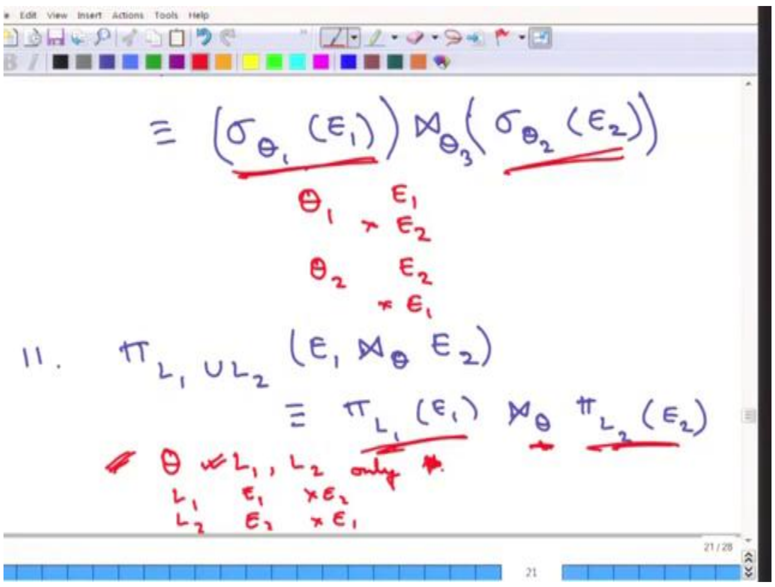
The next rule is again a little similar rule, this is and suppose, there is a condition theta 3 and these can be done in the following manner. And I mean waiting you to think for a while, why, when is this going to happen, as you can probably guessed, when theta 1 contains attribute from only E 1 and not from E 2 and when theta 2 contains attribute only from E 2 and not from E 1, because as we argued earlier, since theta 1 does not concern itself about E 2, so the theta 1 can be applied on E 1 first.

And similarly, theta 2 can be applied in E 2 first and those can be then joined. Now, why are these rules important? Because, you see that these are going to produce much faster join, these are going to be smaller relations and in the expression trees that we solve, this was exactly what was being done. So, branch city was brought into E 1, because the branch city does not bother itself with anything else. So, this is about these thetas, then some projection rules can be written in the following manner.

In this case, what can be done is, one can do the L 1 on E 1 and join with L 2 and E 2, note the similarity of these rule with the previous one. Instead of the selection, there is the projection and again the rule is the same is that, these can be done only when theta involves L 1 and L 2 both. So, there are two couple of things here is the theta must involve attributes from L 1 and L 2 both.

If the theta contains anything else, for example, some other attributes, then that is going to be lost into this. So, theta must contain attribute from only L 1 and L 2, this is only, otherwise it cannot happen, because otherwise those attributes from a lost, when theta will be incomplete, it cannot be applied; that is number 1. Number 2 is that, L 1 attributes L 1 concern itself with only E 1 and not E 2.

(Refer Slide Time: 05:04)



$$\equiv \underline{(\sigma_{\theta_1}(E_1))} \bowtie_{\theta_3} (\underline{\sigma_{\theta_2}(E_2)})$$

$\theta_1 \quad E_1$

$\quad\; \times E_2$

$\theta_2 \quad E_2$

$\quad\; \times E_1$

11. $\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2)$

$$\equiv \underline{\pi_{L_1}(E_1)} \bowtie_{\underline{\theta}} \underline{\pi_{L_2}(E_2)}$$

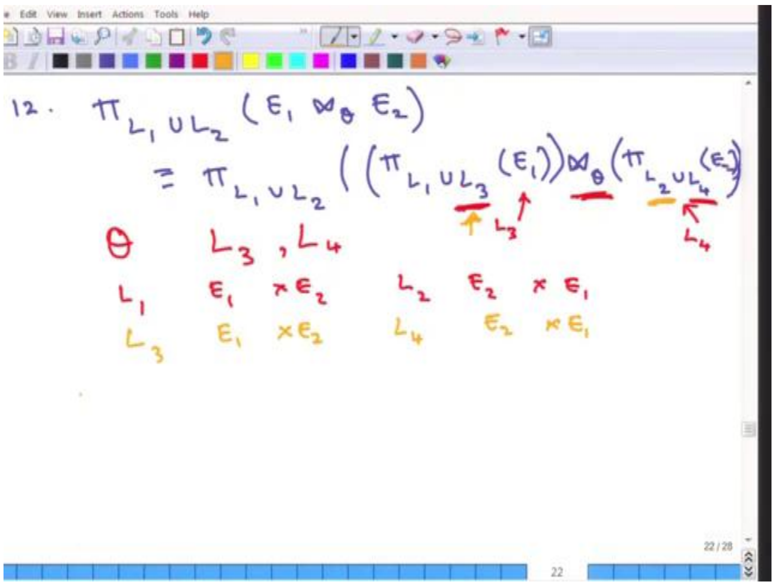
$\theta \; \in L_1, L_2$ only *

$L_1 \quad E_1 \quad \times E_2$

$L_2 \quad E_2 \quad \times E_1$

And similarly, L 2 consents itself with only E 2 and not E 1, this is the same as the earlier, but this is one important condition in addition. Theta must not be anywhere outside L 1 and L 2.

(Refer Slide Time: 05:21)



12. $\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2)$

$= \pi_{L_1 \cup L_2}((\pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\pi_{L_2 \cup L_4}(E_2)))$

L_3 L_4

θ L_3, L_4

L_1 E_1 $\times E_2$ L_2 E_2 $\times E_1$

L_3 E_1 $\times E_2$ L_4 E_2 $\times E_1$

22 / 28

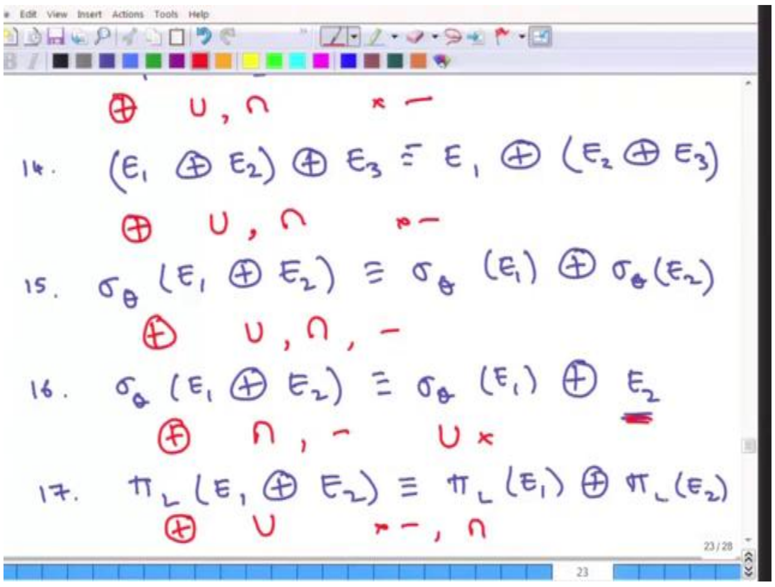
So, a similar equivalence rule can be written, a little bit more complicated equivalence rule can be written in the following manner. We are trying to do the left hand side of this expression with the same as the earlier one and what it does is that, it brings into other attribute. So, there is L 1, this is L 3; note that, there is no L 3 on the left side, but I am going to explain, what L 3 is and then, this can be joined with ((Refer Time: 06:02)).

Once more, there is no L 4 on the other side, but what is being done is the following is that, now theta involves L 3 and L 4, it does not involve only L 1 and L 2, it rather involves L 3 and L 4. So, for these theta to apply E 1 must contain L 3 and if this expression must contain, so this side must contain L 3 and this side must contain L 4. So, it must be selected, it must be projected when this projection has been taken.

So, that is what L 3 must be present on this side and L 4 must be present on this side; that is one thing. Other thing of course, since L 1 is separated out from E 1, it should be the case that L 1 involves only E 1 and not E 2 and L 2 involves only E 2 and not E 1, number that is 2. Also, L 3, now you see L 3 is used in only one side here, so that means, L 3 is concerned only about E 1, it is not concerned about E 2.

So, L 3 cannot be part of the E 2 and similarly, by similar argument, L 4 is only on these side, which is on E 2. So, it is concerned only about E 2 and not about E 1. So, these are the conditions that are required for these two works. Little bit more complicated, but as you can see this is going to be a much faster join, because is a lesser attribute etcetera, than the original one, so that is why these complicated rules sometimes used.

(Refer Slide Time: 07:42)



$\oplus \quad \cup , \cap \qquad \times -$

14. $(E_1 \oplus E_2) \oplus E_3 \equiv E_1 \oplus (E_2 \oplus E_3)$

$\oplus \quad \cup , \cap \qquad \times -$

15. $\sigma_\theta (E_1 \oplus E_2) \equiv \sigma_\theta (E_1) \oplus \sigma_\theta (E_2)$

$\oplus \quad \cup , \cap , -$

16. $\sigma_\theta (E_1 \oplus E_2) \equiv \sigma_\theta (E_1) \oplus E_2$

$\oplus \quad \cap , - \qquad \cup \times$

17. $\pi_L (E_1 \oplus E_2) \equiv \pi_L (E_1) \oplus \pi_L (E_2)$

$\oplus \quad \cup \qquad \times - , \cap$

23 / 28

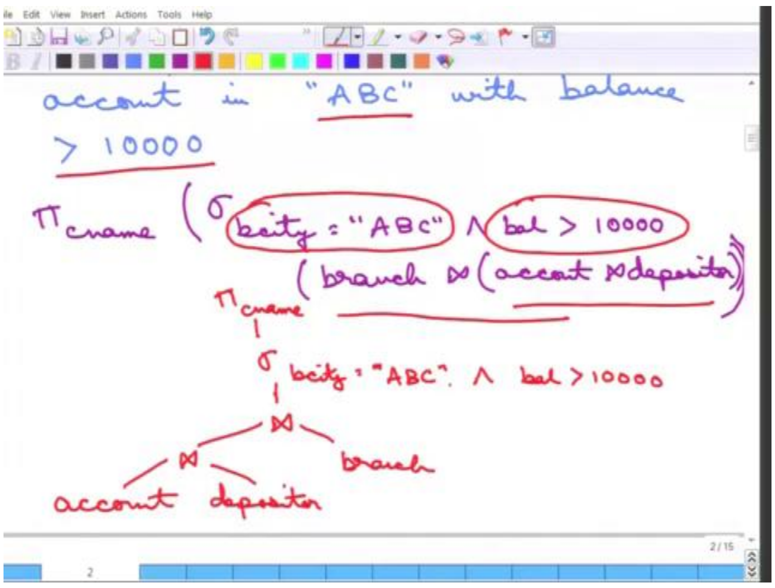
Then, let us go to some of the set operations, the first thing that we are going to say is, suppose E 1, there is some set operator E 2 is going to be equivalent to E 2 and E 1. When is this going to happen not for always, this is when this operated is either union or intersection and of course, not set difference. This is probably easier to understand.

The next one is a more interesting is an order, does the order matter for this operations. So, this is true, when the order is union, this is fine as then the order does not matter is also true, when this is intersection in the order does not matter. But, of course, this is not true, when this is a set difference. The next is on selection using this set of operators. So, if this case, this is equivalent the selection can be separated out and this is true, when the set operated is union or intersection and even set difference, it does not matter.

Because, the selection is going to be on each of them separately; however, an interesting happens, when the following rule is try to be applied in this case, you not doing the selection is just doing the E 2 itself by itself. This is going to work, when the operation is intersection and set difference, but not union and this if you think for a little will, it can understand, why that is the case.

Let us finish off with one last expression is on the projection and if the projection is applied on a set, then it can be separated out only when this operated is union and not set difference are intersection. Again, the reason is the projection works on the union and the intersection, because the projection may lose some of the values. We will next go over an example of how to use these expressions, equivalent expression rules to evaluate the same query using different evaluation plans.

(Refer Slide Time: 10:19)



account in "ABC" with balance > 10000

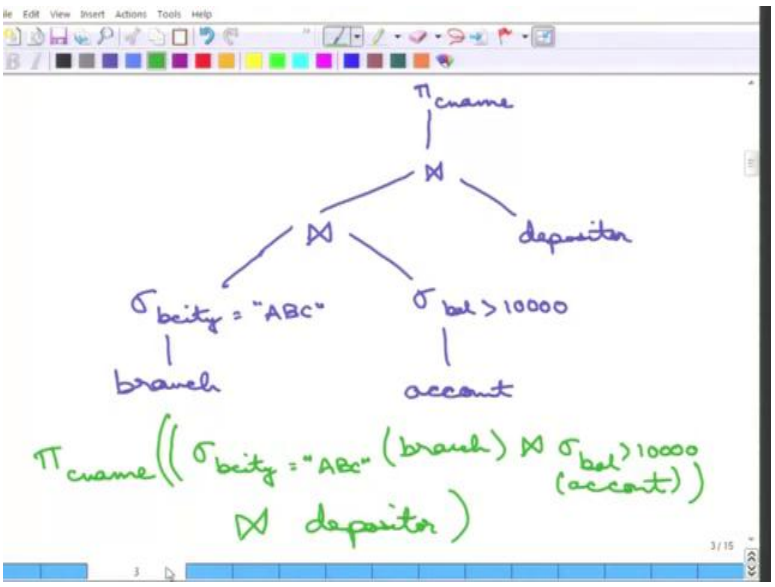
$\Pi_{cname}(\sigma_{bcity = \text{"ABC"} \wedge bal > 10000}(branch \bowtie (account \bowtie depositor)))$

So, this is the banking schema that we have been using so far, the branch customer, account, loan, depositor and borrower. And suppose, this is the query that we need to solve, so find names of all customers with we can account in this city ABC and with balance greater than 10,000. Now, if we want to write it the equivalent relational algebra expression, it can be written in the following manner, I am just trying to solve this.

So, this is one way of solving the query, which is first account and deposited are natural join and then, that is taken a natural join with branch. And finally, the two conditions that the account must be in the city ABC and the balance greater than 1000, these two conditions are applied. Now, if we utilize it, if the relational algebra expression is this way, the corresponding tree can be drawn in the manner that I will show next.

So, the first thing that needs to be done is the account and depositor are joined, the natural join is done; that is the natural join with branch and then, the selection conditions of these two things the branch city is equal to ABC. And balance is greater than 10,000 this is applied and finally, the projection on the c name is taken. This is the equivalent expression tree for that relational algebra equation. Now, as we have argued earlier, this can be made much faster and for that, we can use this equivalent rules that we saw earlier.

(Refer Slide Time: 12:44)


π_{cname}
$\bowtie$
$\bowtie$
depositor
$\sigma_{bcity = \text{"ABC"}}$
$\sigma_{bal > 10000}$
branch
account
$\pi_{cname}((\sigma_{bcity = \text{"ABC"}}(branch) \bowtie \sigma_{bal > 10000}(account)) \bowtie depositor)$

And after application of different equivalence rules, what will happen is that, the new tree can be written in the following manner. This is the branch on which the selection on the branch city is being done, once more this can be done, because the branch city is depends only on the branch. So, the branch city is ABC, then there is an account on which the balance, the selection on the balance is done again this can be done, because the balance comes only from the account table.

This can be then joined, the natural join of balance branch and account can be taken and then, we taken the natural join with depositor. And finally, the projection on c name is taken. Now, what is the equivalent relational algebra expression for this, the equivalent relational algebra can be written in the following manner, this is joined with 1000 of account, which is then through these together is then joined with depositor.

Now, the question is, if we look at this tree, this expression and this tree verses this expression and this tree, why can they be claim to be equal, because we can successively apply the different equivalence rules at we saw last time to get this tree. So, one tree can be transform to the other using this. So, that is why, they can be equal and once more, we can argue that the second tree is more efficient than the first.