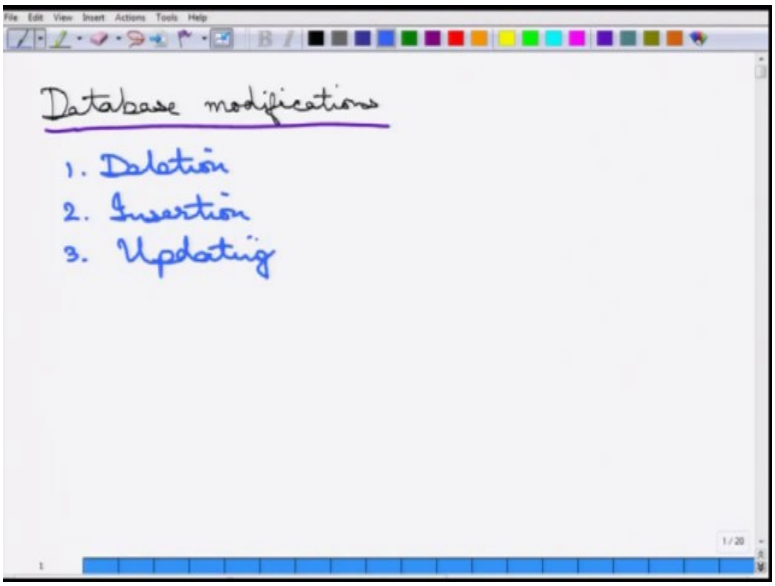
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 07
Relational Algebra: Database Modifications

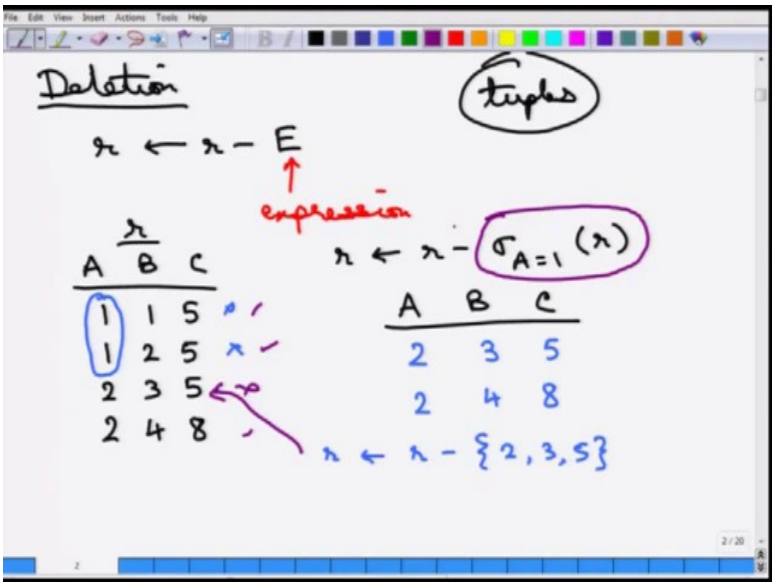
Our next topic will be Database Modifications on the Relational Algebra.

(Refer Slide Time: 00:15)



Database modifications and there are three types of database modifications that we will be talking about Deletion, Insertion, and Updating.

(Refer Slide Time: 00:36)

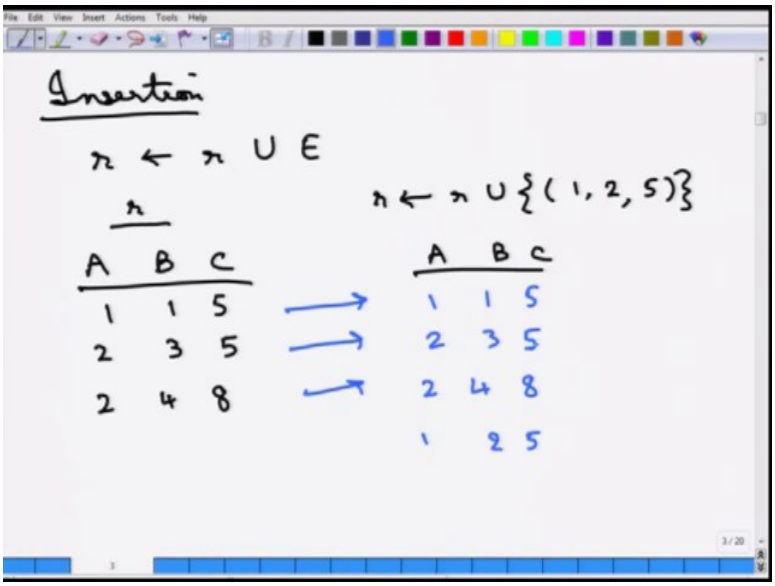


So, let us go over them one by one. So the first one is **deletion**. So deletion essentially means one or more tuples deleted from the relation. So, it is simply the relation r that needs to be deleted, it is assigned in the following manner. So, E is the expression that says which tuples need to be deleted and the rest is very clear. So, the result of the expression is the set of tuples which are deleted from *r*. Now this can be an expression or an actually can it set may be specified and here is one very simple example. Suppose this is your (A, B, C) with *r* with same example that.

So, just one important thing that needs to be remembered is that only tuples can be deleted, the attributes cannot be deleted, only tuples can be deleted from a relation. So, if I now do this r is equal to an expression which is let us say, something like this $r \leftarrow r - \sigma_{A=1}(r)$, then this results in the same schema (A, B, C), but then everything where $A = 1$ is deleted. So, these two are deleted and this results in simply (2, 3, 5) and (2, 4, 8).

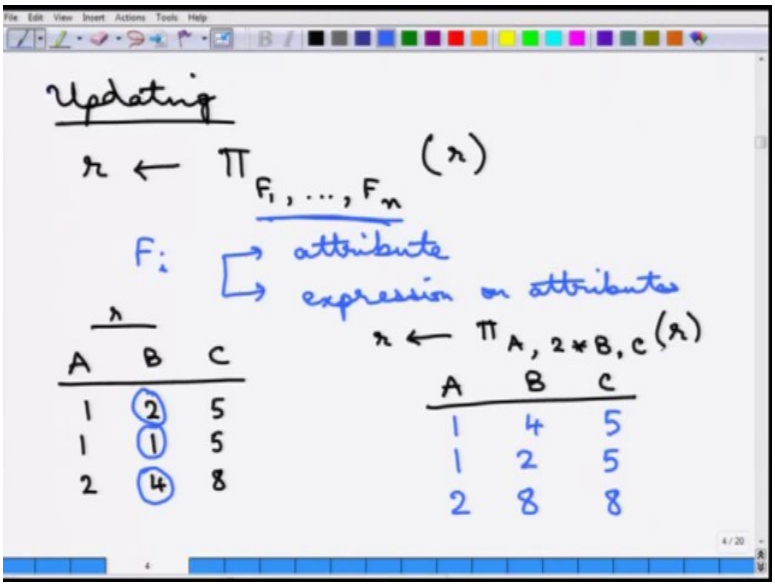
And similarly this can be something like this can be also set, you can say delete (2, 3, 5) which is a set of tuple which is (2, 3, 5). So, then this one is deleted, so this will delete only this and the others will be part of this, but in general it is more useful when expression like this can be specified for deletion.

(Refer Slide Time: 02:25)



The next is **insertion** which is similar in the sense that a tuple is added, so this r gets expression with a union. $r \leftarrow r \cup E$. So, again it is a expression and same kind of things can be given. So here is an example, suppose r is your (A, B, C) which is (1, 1, 5), (2, 3, 5), (2, 4, 8) and if *r* is unioned with, let us say, specify a tuple (1, 2, 5) then of course, the same all these three things are copied and a (1, 2, 5) is added. So, essentially it becomes (1, 1, 5), (2, 3, 5) and (2, 4, 8).

(Refer Slide Time: 03:21)

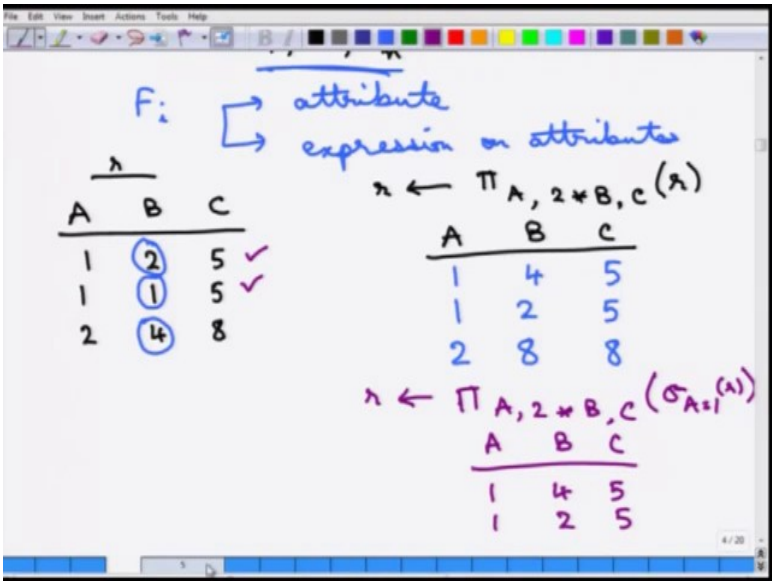


So, the next one is **updating**. Updating is a little more involved in how to write it. Essentially

the updating is given like an expression. $r \leftarrow \Pi_{F_1,\ldots,F_n}(r)$ So, it is the projection of F_1 up to F_n, so all the attributes from *r* are projected on. So, all the attributes of *r* are projected on and each F_i can be either, the attribute itself that was there or some modification of the attributes, so some expression on the attributes.

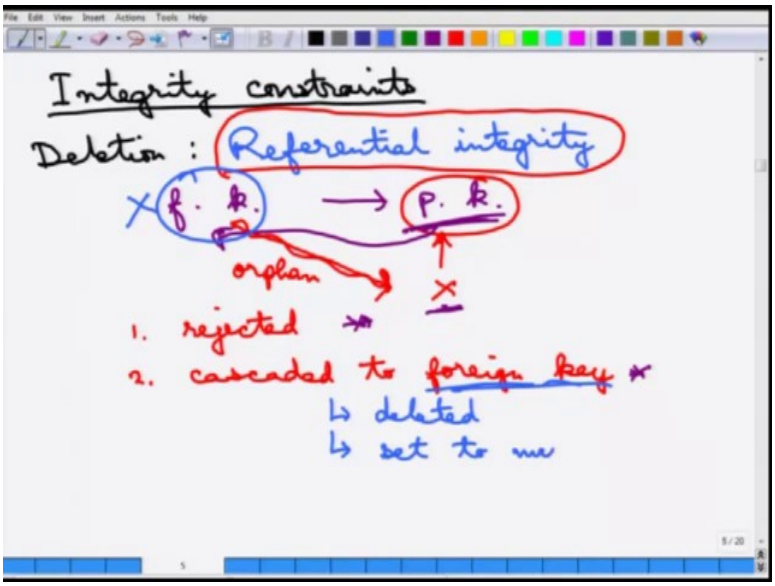
So, an example is the same, let us say *r* is your (A, B, C) which is (1, 2, 5), (1, 1, 5) and (2, 4, 8). And let us say this particular modification is being done, which says A, but then it says 2 star B. So, every B value is essentially doubled up and this is being done, so this results in (A, B, C) with the following 1. So, this is doubled up, this 2 is doubled up, this is 5 the others remain the same. So, this is (1, 2, 5) and this is (2, 8, 8), very simply this is what is being done, Then instead of *r* what can be done is that a little something more can be done.

(Refer Slide Time: 05:08)



So, for example it can be said that $r \leftarrow \Pi_{A,2*B,C}(\sigma_{A=1}(r))$. So that means, first $\sigma_{A=1}$ is done. So, only these two attributes are modified and the resulting then B is doubled up. So, the resulting relation is (1, 4, 5) and (1, 2, 5), so these are the three main database modification techniques, but what is more important for all of this is that all of this can violate some constraints, so these are called **Integrity Constraints**.

(Refer Slide Time: 05:47)

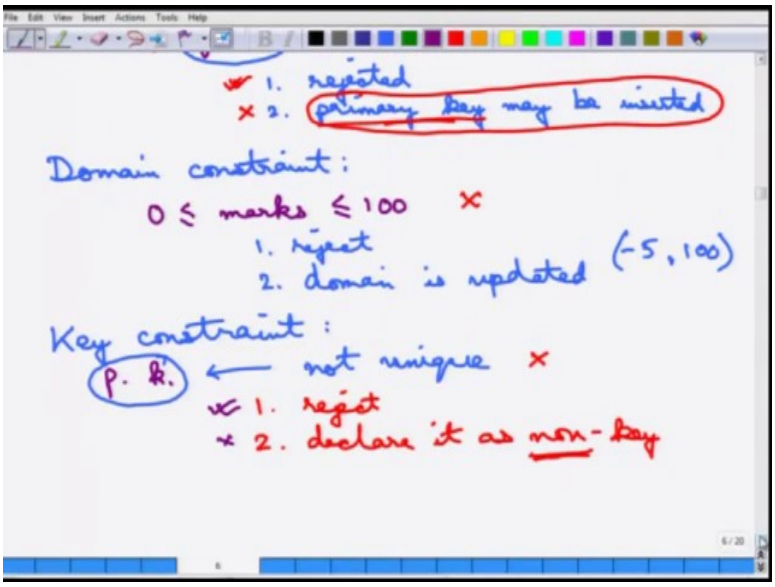


So, all of these three techniques can violate certain integrity constraints. So let us start with deletion. So, what can deletion violate, deletion can violate what is called a referential integrity, this is called a referential integrity. So, what does referential integrity mean? If we recollect what is the foreign key constraint is that every foreign key must be a primary key of some other relation. So, essentially what happens is that if a particular primary key value is deleted from the foreign relation, then a foreign key becomes orphan, because it does not have the corresponding primary key. So, this violates, what is known as, a referential integrity, because this is now referring to something which is not present. So, the referential integrity is violated, so a primary key may not be simply deleted that deletion may not be simply allowed, if there is a foreign key that refers to it. So, what happens if a deletion, or if the database is asked to do such a deletion, there are couples of options first is it is simply disallowed. So, the operation is rejected that is one way of doing it, the other is the deletion is cascaded to the foreign key, there are two ways to handle this. Now, the first one is easy to understand, what do we mean by rejected. So, the deletion does not take place, that simple. Then the second case what happens is that if pk is deleted then the corresponding fk that refers to this primary key also deleted. So, the foreign keys are also deleted from the referencing relations, so these are the two ways of deletion. So, deletion only violates referential integrity and these are the two ways of handling it.

Now of course, as part of this foreign key cascading this can be either deleted totally. So, the entire tuple corresponding to that foreign key is deleted or the foreign key may be set to null,

this is another way of enforcing the deletion.

(Refer Slide Time: 08:08)



So, we next go on what does the **insertion.** What are the constraints that the insertion can violate? An insertion can again violate the referential integrity and this one is probably easier to understand in the context of the previous example. If a foreign key is inserted, then there the corresponding primary key must be present; otherwise, the insertion of this foreign key is not meaningful. So, again there are ways to handle it, so either this can be rejected, so this insertion can be rejected or the corresponding primary key may be inserted.

Now, this primary key may be inserted, but this generally does not make any sense, because if you cannot just insert a primary key, primary key is something very, very important. So, this is not actually not the viable option, so the only viable option is to reject the insertion into a foreign key. So, this is the first kind of integrity that the insertion can violate and insertion can also violate what is called a domain constraint.

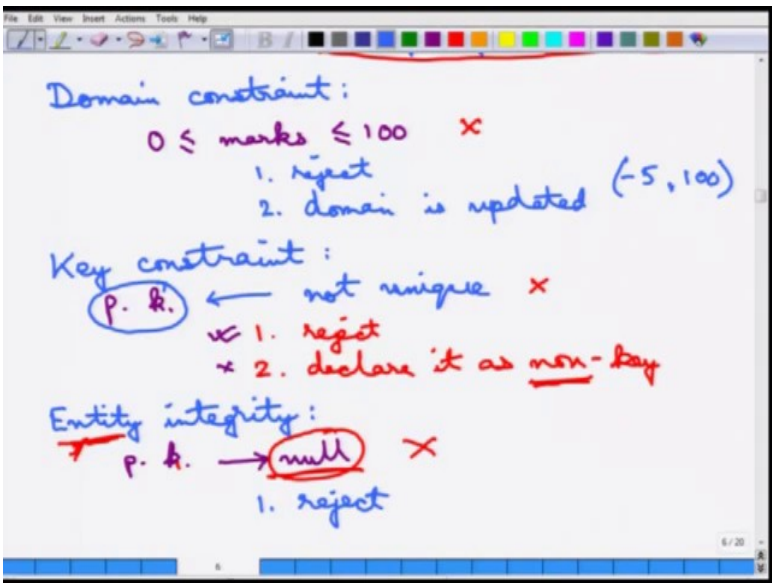
So, domain constraint is probably easier to understand, so there are certain domain values that are specified for example, you can say the marks of a person are between 0 to 100 for a particular exam or whatever. So, if a marks column is being inserted if in tuple is inserted with a marks column outside this range then that is not allowed. So, again there are two ways of handling this, so either you reject the domain is updated.

So, suppose you will later saw that extra marks can be given or marks can be cut for

punishments, etcetera. So, you can the domain can be updated to for example, minus 5 to 100, so there the domain may be updated, so there are two ways of handling this. The next kind of constraint that insertion may violate is called a key constraint. So, suppose there is a primary key or whatever some other kind of key which is there, now if an insertion violates the condition of being a primary key.

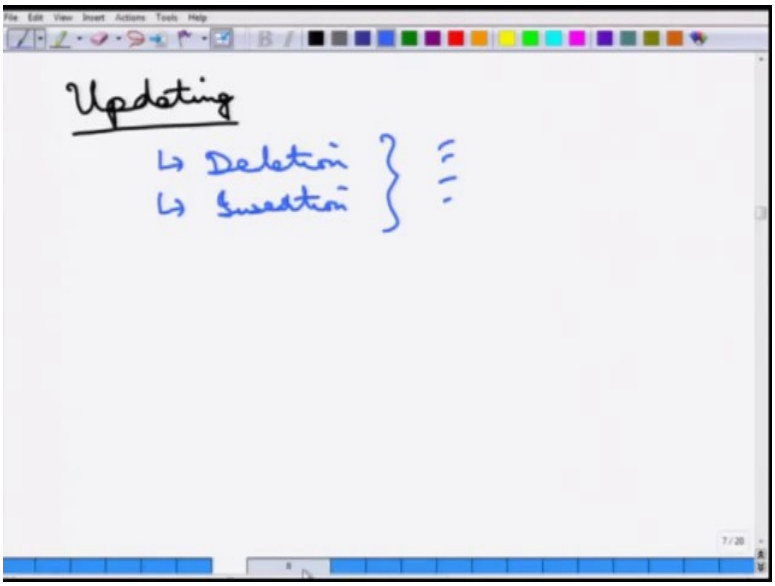
So, how does it violate it becomes not unique, so if another tuple is inserted which has the same primary key or some other tuple then; that means, this is no longer a primary key, no longer a candidate key or super key whatever and this is a problem. So, that must not be allowed, again so this there are two ways of handling this either this is rejected or there is a more extreme option which is declare it as a non key, in which case some other primary key must be done etcetera, etcetera. So, again in general this is not a very viable option and this is the only thing that can be done and that is being done.

(Refer Slide Time: 11:39)



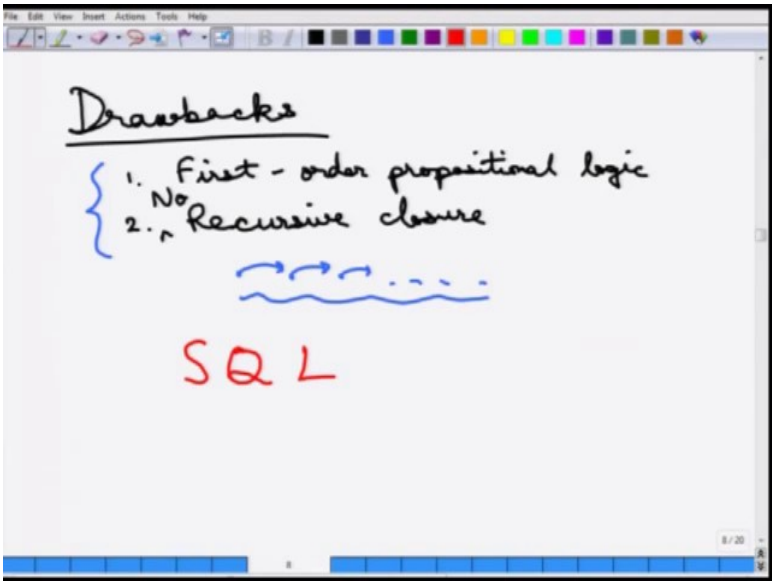
Then there is the fourth type of thing that insertion can violate, which is called an entity integrity. So, entity integrity is that a tuple is inserted, where the primary key is set to null, the primary key is null this again should not happen, because the primary key cannot be null by design. So, because why is this called an entity integrity, what does a primary key mean primary key defines what the entity is about, now if the primary key is null, the entity does not have any meaning. So, again this is not a viable option, so this must be simply rejected.

(Refer Slide Time: 12:27)



So, the insertion violates all these four conditions and if we now go to what the updates updating can violate. Now, essentially updating can be looked upon as a deletion and then an insertion, so any integrity constraints that deletion and insertion violates are also violated by upgrading. So, all those four integrity constraints that we saw are also violated by updating. So, that is what about the relational algebra the all the... So, we looked upon the queries, etcetera and we looked upon the power of relational algebra, the basic relational algebra, the extension of the relational algebra and we looked at all the different, different operations etcetera including the database modifications.

(Refer Slide Time: 13:20)



Now, just to wrap up the relational algebra thing we need to talk about what are the **drawbacks of relational algebra**. So, relational algebra seems to be very powerful, because it can do a lot of queries and insertions and all those things if it can handle, but it has got some drawbacks.

So, the first thing is that this is the *first order propositional logic*, so anything that the first order propositional logic cannot handle, this cannot handle either. Although first order propositional logic is actually very powerful for real life, it may not handle certain other kind of things.

The other very important thing that the database people face almost regularly is that there is no recursion. So, there is *no recursive closure*, so recursive closure that is no recursive closure. So, for example, if you have queries of the form find me supervisors at all levels. So, find me supervisors of this and then supervisor of that person, supervisor of that person up to all the levels it cannot be done, it cannot say at all levels, because there is no recursive closure. So, there is no way to express this in relational algebra, these are the two most important drawbacks of relational algebra, but otherwise relational algebra is very, very powerful. And our next topic which is SQL is based on relational algebra, so this ends the relational algebra module and next we will talk about SQL.