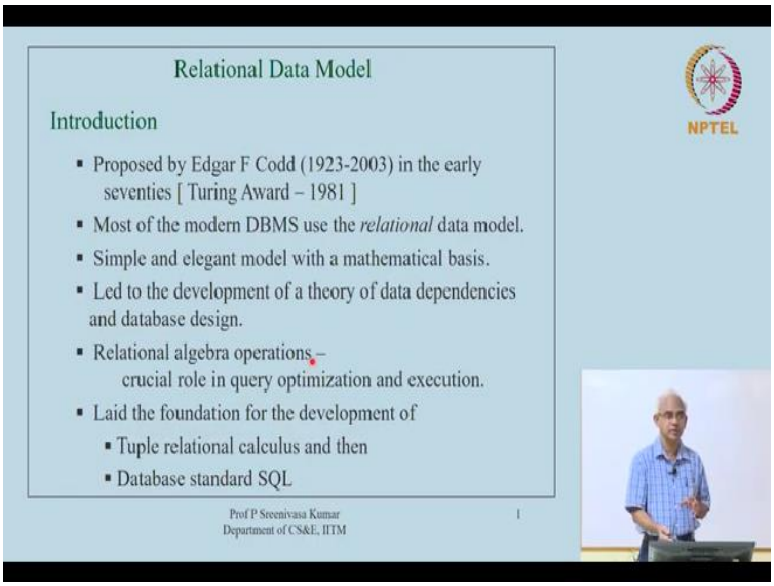
Database Systems
Prof. Sreenivasa Kumar
Computer Science and Engineering Department
Indian Institute of Technology-Madras

Lecture-07
Relational Data Model and Notion of Keys

Discussion on relational data model.

(Refer Slide Time: 00:19)



Relational data model is a very you know important component of the whole course. Though we call it a database system these we generally think of them as relational database systems these days, a database system is considered by default as a relational database system. So, this relational has become such a important thing, before the advent of the relational data model there were other kind of models in us like network model and hierarchical model and all that.

But relational data model, you know, is proposed by Codd in the 70s. And then it became very popular because very simple and elegant model with a mathematical basis. We will see those details as we go along and called Codd Turing award in 1981, much after it was proposed in early 70s. But, about almost a decade later, he got the Turing award, which is one of the highest you know, it is the highest award in computer science.

The introduction of this relational data model led to the development of the theory of data dependencies and we will later on spend a you know lot of time trying to understand what are these data dependencies are, these are essentially to do with evaluating whether one particular relational data model for a particular application is you know better than some other design.

How do we compare database designs. So, it is in this context that we talk about what are called data dependencies and then we define what are called normal forms and then evaluate designs based on these normal forms and then come to a conclusion about what design to adopt. So, the relational database model led to this entire body of knowledge concerned with theory of data dependencies.

And then how to evaluate a database designs, then the as part of this relational model, we will also be discussing what are called relational algebra operations and these algebra operations are the ones actually that get implemented. And finally, when queries given in SQL, I have to be executed by this relational database system, it is in terms of these relational algebra, you know programs that implement relational algebra operators that actually work gets done ultimately.

So, that what it means is that, the SQL is a you know a declarative style of, you know specifying the query and it will get internally translated into these relational algebraic expressions. And as I was mentioning earlier also, we then have an opportunity of you know, looking at this relational algebra operators and then you know, try to make certain transformations on these algebra operations.

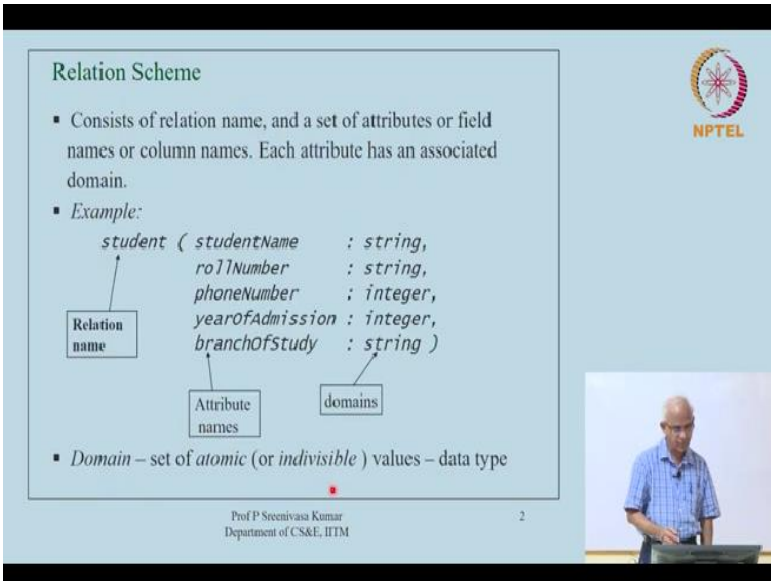
So, that they are the transformations are meaning preserving, but then will help us evaluate the expression in a much better way. So, optimization is possible. So, after the model has been defined setup and theoretical issues have been worked out, it kind of laid the development for this what is called a tuple relational calculus. Relational algebra is one thing, relational calculus is a different thing.

And relational calculus is actually based on mathematical logic is and that later kind of developed into SQL language, so SQL language first and then the of course, that led to the

database standard and so, it is very, very important model to understand and as I was telling you, a data model is nothing but a bunch of concepts that we will use to describe the database.

And the relational data model is used to describe the database at the logical level complete description can be given using relational model okay, so let us now start looking at all the ingredients of this relational data model.

(Refer Slide Time: 05:56)



So the idea is actually pretty simple that there is also very interesting thing that is such a simple idea of a relation as we know of it in discrete mathematics like as a subset of the cross product of sets has an indeed you know, has such a great impact. So, okay in the database context we talk of 2 things here, one is database sorry relation scheme and then the relation instance okay.

Relation scheme will basically consist of the name of the relation, for example here and this set of fields or we also call them attributes in this model also we call them as attributes and but there is an important difference between the attributes that we have seen earlier and the attributes here, I will tell you that. So these attributes are field names or column names. Sometimes we all just call them as column names because it is table are structure okay, and each of these attributes has an associated domain.

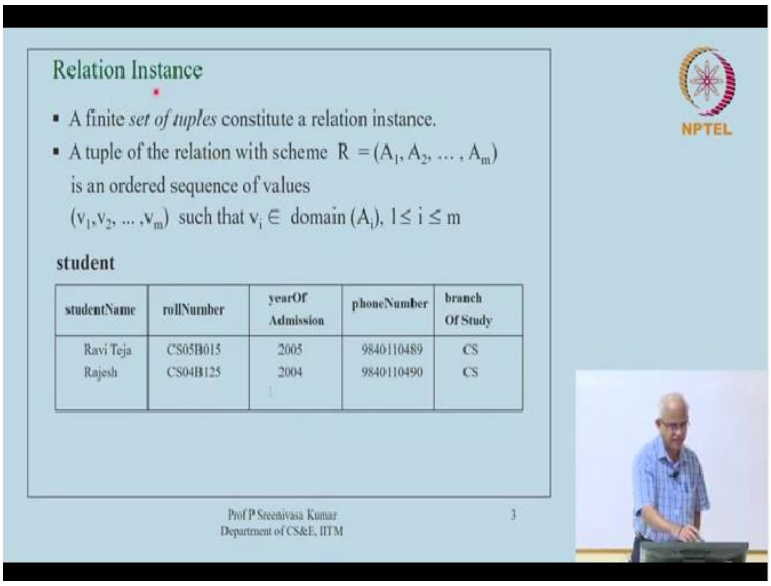
Now, so here is an example. So, we have student name and the domain is strings and roll number we also modeled as a string, phone number is an integer, and year of admission is an integer branch of studies a string. So, these are domains, these are attribute names and then the whole thing is what is called one relation scheme. So the relation name comes and traditionally write it this all the attribute names in parentheses.

And now this what is currently is a domain. A domain is nothing but a set of atomic values. This is most important a set of atomic or indivisible values okay. So, that means this is a single value and we will not you know, try to go into the parts of that and things like that, we will not assume that there it has any components in it will simply deal it as though it is a single value, atomic, indivisible value.

So, as again this in the entity relationship model, when we were talking about attributes, for example, we talked about multivalued attributes, composite attributes and things like that, right. But here now, when we say an attribute, it is always assumed to be an a value that comes from a set of atomic values okay. Now, you can now. So, now this important this assumption is very fundamental to be database model.

And it is so fundamental that it is actually called it is later on in the when we discuss the normal forms, we give a name for that it is called the first normal form okay. Good so that is what relation scheme is about.

(Refer Slide Time: 09:24)



And then we have the relation instance a finite set of tuples consisting, so, they constitute a relation instance and then suppose we have a scheme like this R A 1 through A m which are all attribute names, then we basically have a set of tuples and all this finite number of tuples constitute the relation instance, what exactly is a tuple is easy to see because we are kind of familiar with that.

The tuple is basically an ordered sequence of values, v 1 through v m such that this v i belongs to the domain of the corresponding attribute A i. For all I 1 to m. So, in this case student name Ravi Teja and roll number, some year of admission phone number etc. okay. So you can have dot dot dot so you can it is not an infinite, it is finite number of topics because it is a sect. And it is a finite set emphasis on finite set.

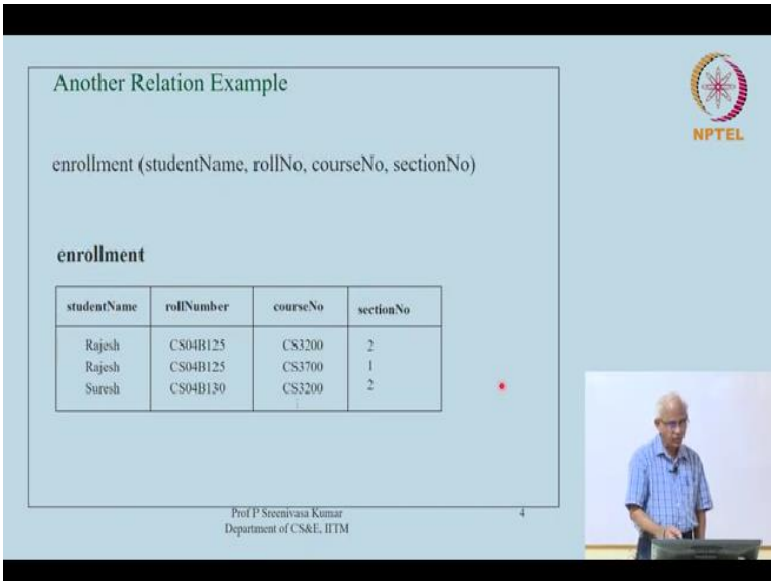
A finite set of tuples constitute a relation instance okay, so at this point of time, we are the only constraint we're placing here is that all these values. So this is a string. So even though it has, it looks like having 2 pieces in that it is a string. So there is a blank character here. So we treated as a string, that is it, okay. So, all these are atomic pieces of data. And each of these things should come from the domain define for that particular attribute.

That is the only constraint we have and there are a finite number of them, this is what is a relation instances. So relation scheme and relation instance. This should be obvious to you that

there are no duplicate rhos in relation instance right. Why is it that there are no duplicate tuples relation instance because it is defined as a set not a multiset, it is a set, so set means we do not have duplicates.

Later on of course, when you deal with SQL tables, SQL tables, we for practical reasons we must allow you know rhos you may allow duplicate rhos inside SQL tables. So, we will discuss that when we come to that. So, but right now from the relational theory, we will assume that this is a set of tuples and hence there is no duplicate tuples at all okay.

(Refer Slide Time: 12:47)

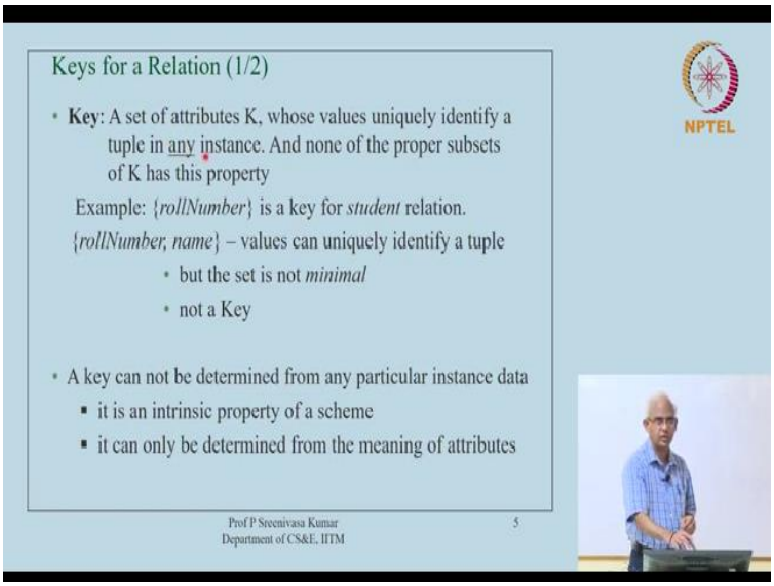


Here is an example. Here is another example. Look at this. I am capturing the enrollment of students into courses. So I am putting student name, roll number okay, it should be same actually, course number and section number. So let us assume that courses have sections and so, course number section number together is what you know, one a student has to enroll for register for. So, in this case now this person Rajesh is registered for these 2 courses.

Whereas Suresh is registered for this course, notice the way information gets replicated, because in order to you know, record the fact that Rajesh has registered or enrolled for this particular course, you must record it like this CS3200 and then since the schema is there like that, we need as the point I am trying to emphasize here is that you cannot put CS3200, CS3CS3700 here, what does it violate.

It violates our principle that the values here should be atomic. It should not be lists, should not be a waste or it should not be anything else it should be atomic basis of data. Because of that, you can see that we need to actually replicate the rest of the values and then change only this value good. So, here is another example. So, we will see a lot more examples of relations.

(Refer Slide Time: 14:37)



Now, let us move on to the notion of keys for a relation. A key is defined like this, a set of attributes K whose values uniquely identify a tuple in any instance**,** that is important any instance of that relation is what is called a key and none of the proper subsets of this K have this property. So, it is in some sense the minimal set of it is the minimal set of attributes that have this interesting property that their values will uniquely identify at tuple in the set of tuples, a set of tuples are there.

So we want to uniquely identify one tuple. What is it that helps us, if you look at this for example, what is the key, what do you think would be key here, can roll number be key here, roll number in this particular relation scheme roll number is not a key because roll number cannot identify a rho uniquely. There are people who are doing multiple courses and so roll number gets repeated in multiple rolls.

And so you cannot do that. So, you can see that actually roll number along with course number would serve as a key in this particular relation okay. So here is another, so roll number for the student relation is in fact of course a key there are no 2 students in the same roll number and so in the student relation, which we have seen a little while it before, roll number is a key okay.

What about roll number and name, you can see that as long as roll number is there, you can throw in any other attribute. And then that a set of attributes will anyway will have the property of identifying that tuple uniquely because it contains roll number, right. So, this set has the property of uniquely identifying the tuple but it is not minimal.

Because you can throw away name. You can you do not need name, you just need only roll number. So, this set though it can uniquely identify a tuple, it is not minimal and hence is not a key because key is a set of attributes whose values uniquely identify at tuple in any instance and none of the profits of sense of that has this property, it has to be minimum okay. Now, one important point about keys is that you cannot look at a database a relation instance.

And then somehow figure out what could be a key, what is the key for that relationship. It is difficult to do that. That is because in that particular instance, it so happens that we have a bunch of tuples and some attribute or a set of attributes. Their values can uniquely identify a tuple in that instance, but then remember that the key is defined that it should in that in not only just in that tense instance.

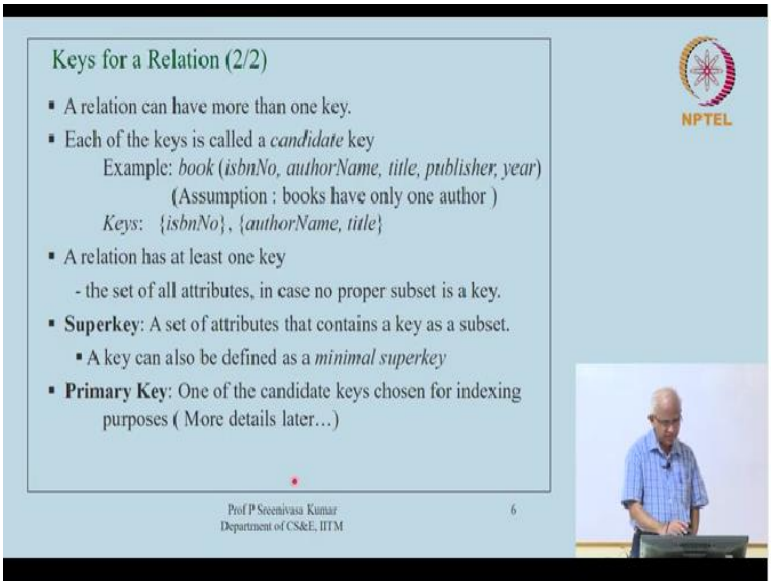
But in any instance, this attribute should have that property. So**,** it is something like an intrinsic property of the scheme. So, only the it can be determined by you know by understanding the meaning of those attributes right it cannot be simply determined by just looking at a particular instance of the relational data okay. So it is an intrinsic property of a scheme.

And so people who understand the domain only can actually identify what should what are the keys for the relationship. Now, an important question here is that does every relation have a key, as a key is it always the case that religion has a key. Having understood what is the definition of key. Now, can we make a statement saying that every relation has a key, you can write okay.

What is the worst case, the worst case scenario is take the entire set of attributes. The entire set of attributes will act like a key because it is a set of attributes. So, set of tuples. So, in any case, the entire if you take the entire set of attributes that is the worst case, but oftentimes you will find some subset of the set of attributes, which will have this property and that is and does a relation have multiple keys is another question.

Can a relation have multiple keys, yes can right nothing wrong. So a relation can have multiple keys and then it always has a key.

(Refer Slide Time: 21:31)



So relation can have more than one key. So here is an example for that. So if it has multiple keys, we will call each of them as candidate key, then new term candidate key, we call all the keys as candidate keys. Later on we will see that one of the candidate keys will get chosen as a primary key when we move on to the implementation stages and things like that. But right now they are all keys, so we call them as candidate keys.

Now let us look at this example of a book. Where isbn number, author, name, title publisher, year are the attributes, I suppose you know, what is isbn number, right. It is a unique number given to all books all you know, its international standard, and standard book number, so the isbn

stands for that. So let us make another additionally, we can make an assumption here that just to you know, illustrate the point that books have only one author.

It is not usually the case, but let us make that assumption but books have only one author. So if that is the case, then there are keys like isbn number is a key because it is, you know, setup like that the it is international standard book number for books and author name and title together serve as a key. No person will write 2 books with the same title, right, there will be some difference in the title.

If you are writing a book and then you are revising it, then you will add the edition number to it. So fundamentals of database systems 4th edition is the title of the book. So when you come up with the 5th edition and fundamentals of database system, fifth editions is the title. So people write books with different titles and so author name, title because we made this assumption that author there is only one author is also can serve as a key for this particular relation.

So we know that so when we say this, it says that whatever be the instance of this particular relation, it is always possible to uniquely identify a book given the author name and title okay, that is what we are okay, so this is the question I was asking a little while before. Does a relation always have a key, yes, it always has a key and the set of all attributes, in case no proper subset is a key will work as a key.

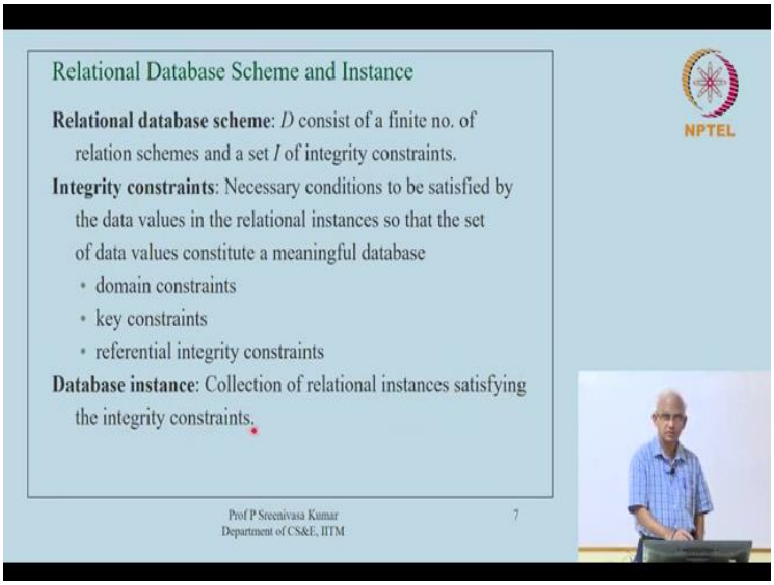
Now, another term, we call something as a super key. what is super key a set of attributes that contains a key is called a subset sorry, the key contains a key as a subset is called a super key okay, so I give you an example of a super key in the last slide, the roll number plus name, roll number and name is a super key because it contains the key which is roll number for the student relation.

For the student relation roll number is the key. So anything you add to that then roll number name, roll number, phone number, etc. they are all will become super keys. So, if you have this notion of the super key then key can be actually defined in terms of that. So key can also be can be defined as a minimal super key. The super key is something which has this property that its

values can be used to uniquely identify the tuple and we are now adding minimality to it, that means no proper subset has this property.

So a minimum super key. So, in some books, they introduced super key first and then define key in terms of super keys as minimal super key. So, these are various terms that you will hear. So, primary key is one of the candidate keys is chosen for indexing purposes and things like that. So, we will see those details later on okay, so, this is a important idea, the key for a relations here. I hope it is clear.

(Refer Slide Time: 26:49)



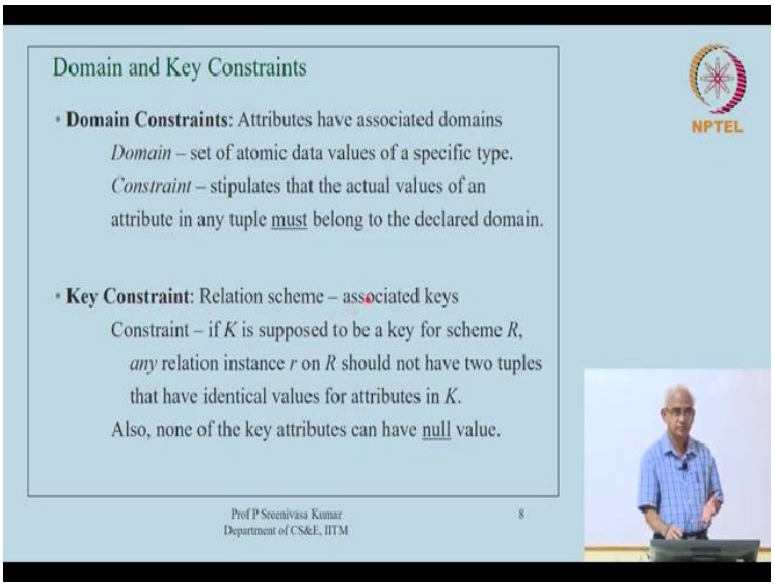
Moving on what is a relational database scheme. A relation database scheme D consists of finite number of relations schemes and a set of integrity constraints okay. Now, what are integrity constraints. We will talk about that, first thing is okay it consists of finite number of relation schemes and internationally it is just not these schemes, but we also need certain constraints and what are these constraints.

Constraints are certain necessary conditions to be satisfied by this data values in these instances in the relations senses. So that that instances constitute a meaningful database okay, now, normally we talk about 3 kinds of constraints, 1 domain constraints, key constraints and referential integrity constraints. So, in the next few slides, I will explain these terms. Domain constraints, key constraints and referential integrity constraints.

And then so we call this relational database scheme. So earlier we just had relation scheme. Now we have database scheme. A database scheme is nothing but a finite number of relations schemes along with this integrated constraints and database instance is a collection of relational instances of all those relations schemes which satisfy the integrity constraints okay. So, we have 2 things here the relation scheme and relation instance, database schema and database instance.

A database instance is nothing but a collection of relational instances but not just any collection, but those that satisfy the integrity constraints. Now let us go into a little bit deeper into what these constraints are. And why do we first call them as constraints and let us get into that okay.

(Refer Slide Time: 29:14)



This basically the idea of domain and key constraints is to capture the domains and domains for attributes and the keys for the relation. So we will see that now. So, attributes have associated domains as I have already mentioned. And the domain is nothing but a set of atomic data values. So the constraint here, what it specifies is that the actual values of an attribute in any tuple in the relation instance must indeed belong to this declared domain okay.

So, that is the constraint would like to turn that information into this constraint mainly because we would later on charge this relational database manager management system responsible for maintaining these satisfying these constraints okay. So that is the reason why we call them as

constraints. So, the corresponding constraint basically stipulates that the actual values of the attribute.

So, the data entry errors are often seen. Suppose you define a cell number as something which has exactly 10 digits, a string with 10 digits, okay and no all numeric 10 digits, that is it, right. And then you see that somebody entered a 12 digit number there, then it is a violation of the constraint right. So, whatever is the declared domain the values must come from belong to that declared domain.

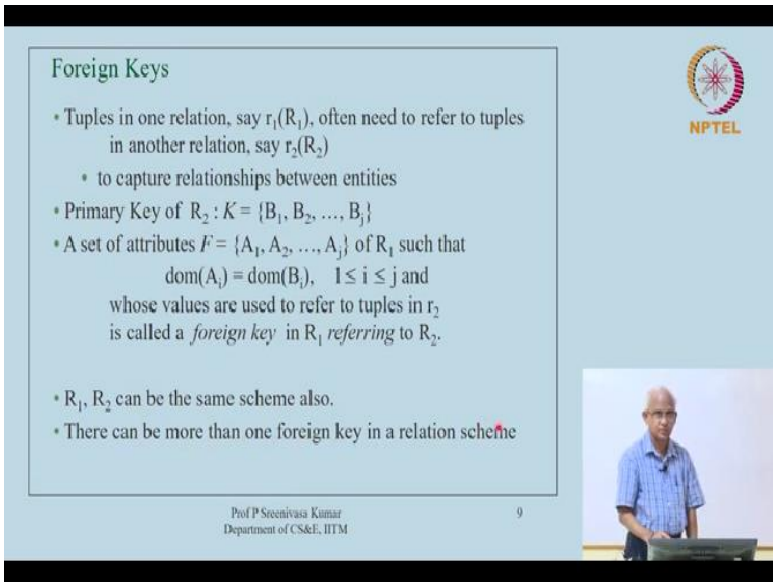
So, we term this as a domain constraint. And in a similar way we have this what key constraints basically, if you in the relation scheme, you say you have kind of declared are asserted that there are certain keys candidate keys. Now the corresponding constraints would be that if K is supposed to be a key for this relation schema, then in any relation instance r little r okay we will use lowercase letters to indicate the instances and uppercase letters to you indicate the schemas in general.

So, in any relation instance r little r on the schema r should not have 2 tuples that have identical values for these attributes of K okay, it is not just enough to say that okay this is key, but the instance should you know satisfied, right. Otherwise it is a useless instance. So, none of the and in addition the key attributes should not have null values could not have is null is a special kind of a value views in order to kind of indicate that we do not know the value or the value is not applicable.

So, we use this null values okay. So, these are reasonably straightforward to understand that that basically we will use this notion of domain and key and then convert this into domain constraints and key constraints. So that you know when you are specifying the database scheme you would not only specify the relation schemas, but you will also specify all these constraints. So, that the when you consider a database instance it should be an instance of all these relations, relations senses and together satisfying all these constraints will later on see.

I mean we are also going to see one more important constraint called referential integrity constraint okay. So in order to introduce this referential integrity constraints.

(Refer Slide Time: 33:56)



I must know introduce another term that uses the name key but it is called foreign Key let us look at this what exactly are foreign keys okay. Now, it often required see, notice one thing that in this model so far we have only one thing called the notion of relation. As a contrast to this, if you consider the ER model we had entities and relationships, so relationships between entities that captured the associations between entities, okay.

But now those relationships also have to be captured as relations in the relational model okay, so we will let us see exactly how that mapping will come out. But in general, you can understand that tuples in one relation may have to refer to tuples in another relation. So say r1 on capital R 1 some trouble in that may need to refer to the tuple in R 2. If that is the case, how does it refer.

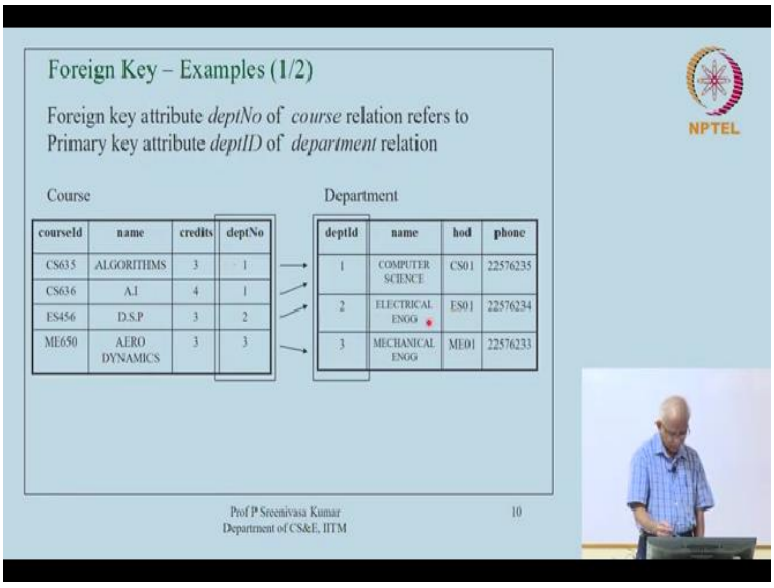
So, **so,** maybe this is to capture the relationships between entities okay. So let me give you an example of that. But before we look at an example, here is a formal definition for that. So let us say primary key of R 2 is B 1 through B j okay, so we are talking about 2 relations R 1 and R 2. So let us stay R 2 has some key called B 1 through B j. Then we will define that set of attributes F of R1 which are A 1 through A j same number of attributes.

But their names may be different, same number of attributes would satisfy this condition such that the domain of A i is same as the domain of B i corresponding domains are same, corresponding domains for these attributes are same. And the values o f these attributes are what we will use in order to refer to tuples in R 2 okay, these are the attributes we would like to use whose values are used to refer to tuples in R 2 is what is called a foreign key in R1.

And specifically will say it is a foreign key R 1 referring to R 2 okay. Now the one little twist here is that R 1 and R 2 can be the same scheme also, sometimes we may use some set of attributes to refer to the tuples in the same relation. I will show you some example of that little later okay, so now let us first understand this and give an example of this and there can be more than one foreign key in a relation scheme.

So this is also possible. So we will call such a collection of attributes as a foreign key because it is basically refers to tuples in a foreign relation okay.

(Refer Slide Time: 37:39)



Here is a simple example. The let us look at these 2 relations schemes course and department. So of course has certain things and then so course has course ID, name, credits and department number to kind of indicate us to what is the department that is responsible for offering that course okay. Now here is department details. So department relation and that has department ID, name, who is the HOD.

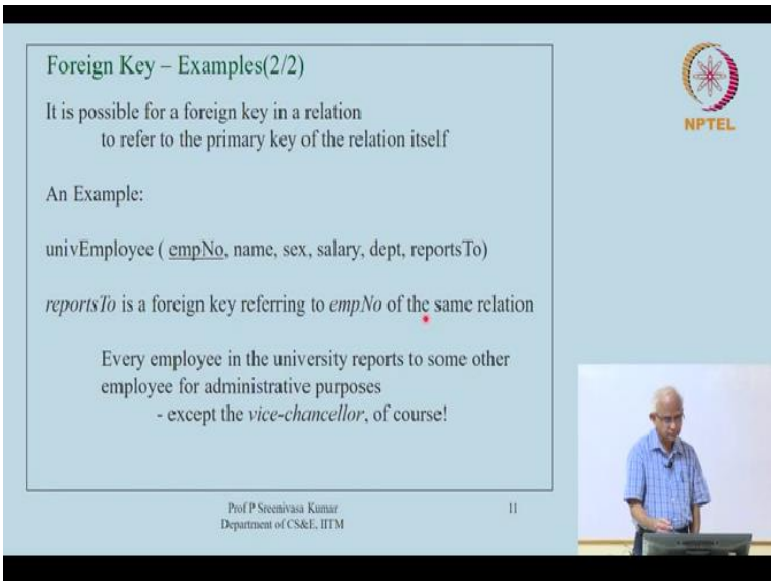
And what is the phone number, probably the office phone number and the office phone number let us say. Now here is the notion of foreign key. So these values we would like to now use in order to kind of refer to the details of what is the department that is offering it. So algorithms is being offered by department number 1, okay, which is actually you can look up here and find out that is in fact, computer science department.

And this DSP digital signal processing is being offered by 2, which is relatively new department. So like that, so basically what we are saying here is that it is possible that we will in one relation, we have certain attributes whose values, we will use in order to refer to tuples in another relationship and obviously the domain of this and the domain of that should be same right. So, that is what we put in the definition.

And if there are possibly 2 attributes or 3 attributes of that kind, then obviously the corresponding domains have to be same. And so such attributes are what we call us the as foreign keys. So in this case, we have taken a department number is a foreign key that refers to department, foreign key of course, relation and refers to the department relations.

So this is how we enable tuples from to kind of link to other tuples. Notice that this is not a physical pointer. This is value based. It is not a physical pointer, right. This is not like, this is a logical thing. It is a value. So if this value kind of represents what is the topic that is talking about okay.

(Refer Slide Time: 40:25)



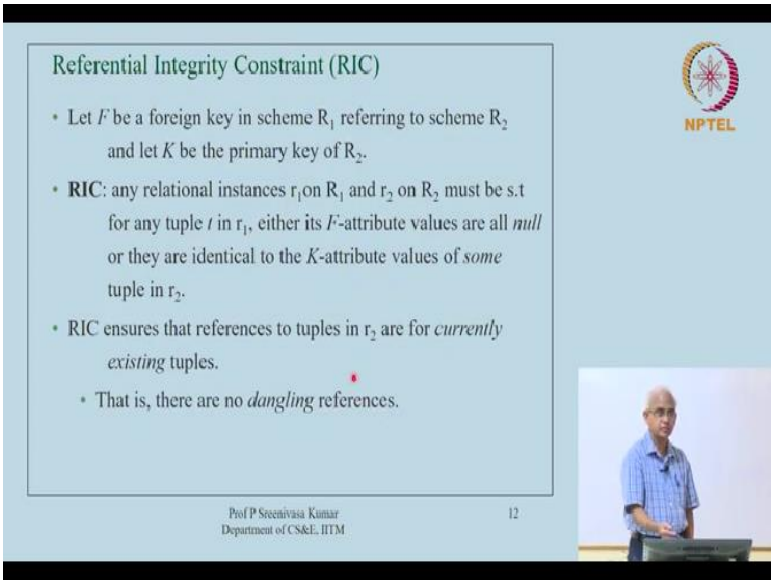
Here is it is possible for a foreign key in relation to refer to the wait, I think, okay. It is possible for a foreign key in relation to refer to the primary key of the relationship itself. Let us look an example. So let us say you have a university employee with employee number, name, sex, salary department and reports to let us assume that you know, just for the example say that all employees are supposed to report to some other employee except possibly the top personally you know stay with vice chancellor.

So, obviously in this case reports to should be having values that come that our whatever is used for employee numbers, let us say employee numbers are some six digit numbers okay. So, this supposed to also should be similar 6 digit numbers so that when you are looking at one tuple of one particular employee from Ramesh we can look at here and then find out what is the value here reports to value.

And that if you look now look up in the set of tuples will tell you who is reporting, who is his supervisor or boss right. So, in this case, we will see that the information about the employee all the employees are in the same relation and so the reports has to basically refer to the same relation instance, it has to refer to the same relation instance. So, the here is a foreign key that refers to the relation itself.

So, here is the normal key and here is a foreign key that that refers to the table the same relation. So, I hope this is clear to now that every employee in okay, so we basically are trying to capture as to who reports to whom. So, a bunch of employees might be reporting to one particular person and that particular person might be reporting to somebody else, and so on there is it you can imagine that there is some kind of a hierarchy in the university except of course, the director, vice chancellor which you will have a null value. Because he does not report to anybody you can imagine that way.

(Refer Slide Time: 43:20)



Now, that we have set up this notion of a foreign key, we can talk about what are called referential integrity constraints, we will call them as RIC referential integrity constraint, which basically you know is some kind of a guard for this foreign keys. So, let us say F is a foreign key in the scheme R1, referring to some scheme R 2. And let us say K is the primary key for that particular R 2.

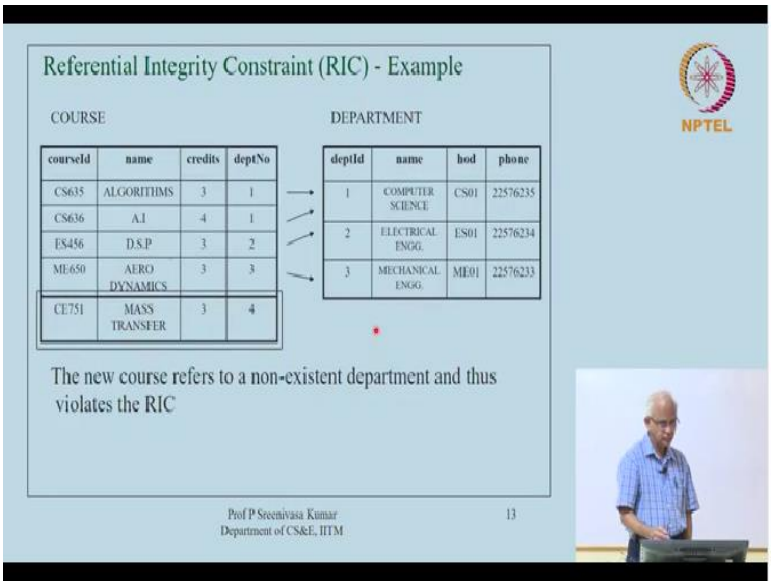
The RIC what it says is that in any religion instance R 1 on this relation, capital R 1 and R 2 it should be such that any tuple t in r1 if you look at its F attributes, F is the set F be a foreign key right. So either it is F attributes values are all null or all null or there are identical to the K attribute values of some tuple in R 2 okay, you got it. What we are saying here is that F is a foreign key in R 1 and K is the primary key for R 2 okay.

So we are the referential integrity constraint request that in any relation instances of both R 1 and R 2 it should be the case that either this F attribute values of this R1 are all null. That means it is not referring to anything or if it has some nominal values, then those values have to be identical to the K attribute values of some tuple in R 2. Basically what we are saying here is that they should not be any dangling kind of tuple.

Dangling references, suppose you referred to something and that something does not exist then you're in serious trouble right. So, we want to make sure that there are no dangling references. So, RIC basically ensures that references to tuples in r 2 from r1 are referring to currently existing tuples they are all currently existing tuples and so, there are no dangling references. So the is a very important thing right.

So that is why we convert that into a constraint and then impose this as a reference as a referential integrity constraint. And then you know, later on we will see that the RDBMS will be given the responsibility of ensuring that the database instance satisfies all the referential integrity constraints, and if it does not satisfy what are the actions to be taken and things like that.

(Refer Slide Time: 46:30)



So, here is a illustration of the violation of RIC. So, supposing we have seen this course and department relations earlier right. So, let us say now we have somebody has come and introduced mass transfer as a new course and said that it is being offered by department 4 and

when now look up in the department there is no department with department ID equal to 4. So this is a reference which is dangling.

There is no department 4 and this is introduced a there is an error year, or it is an error here. We do not know either. In fact, there is a department 4 but its information is not recorded here or actually it is being offered by department 3 itself, but it is wrongly entered as 4 etc. So there is a possibility of course, you might say that okay, I may probably be able to figure out what is the department from the course number.

But we are not allowed to look into the values there. Remember that let us say so, okay. This could be something else right, for example. So in general, we are talking about tuples in one relation referring to tuples in other relation and then whatever is being used as a attribute refer it is value should either be existing as a value here, or it can be null, supposing I do not know what is this particularly course.

Who is offering this course, I am allowed to enter null there, in which case, it is not a problem it can be it has to be updated later. But if you enter 4 and then 4 does not exist, then it is a problem right. So that is why in the definition, if you notice clearly we saying that either the F attribute values are all null or all null or they are identical to the K attribute values of some tuple in r 2 to make sure that there are no references to non existing tuples okay. So that is what referential integrity constraints are about.