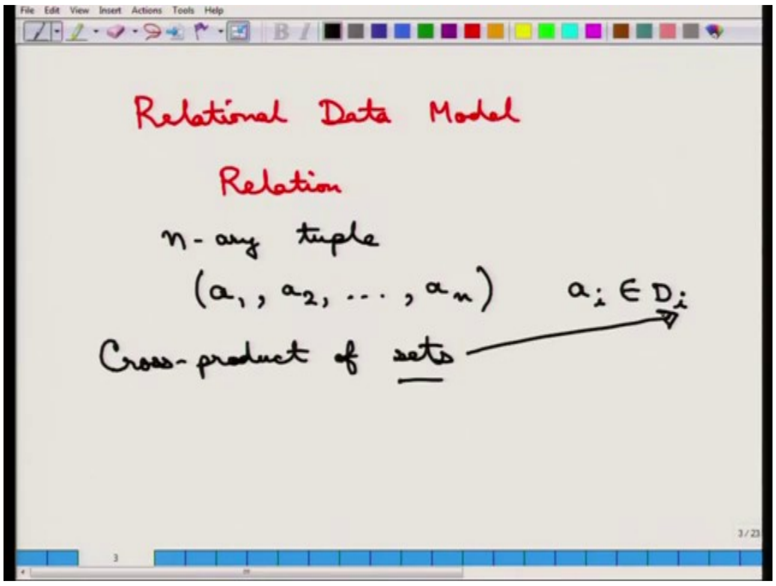
Fundamentals of Database Systems

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

Lecture - 02
Relational Data Model

Let us start on the next topic which is on Relational Data Model.

(Refer Slide Time: 00:13)



Relational Data Model

Relation

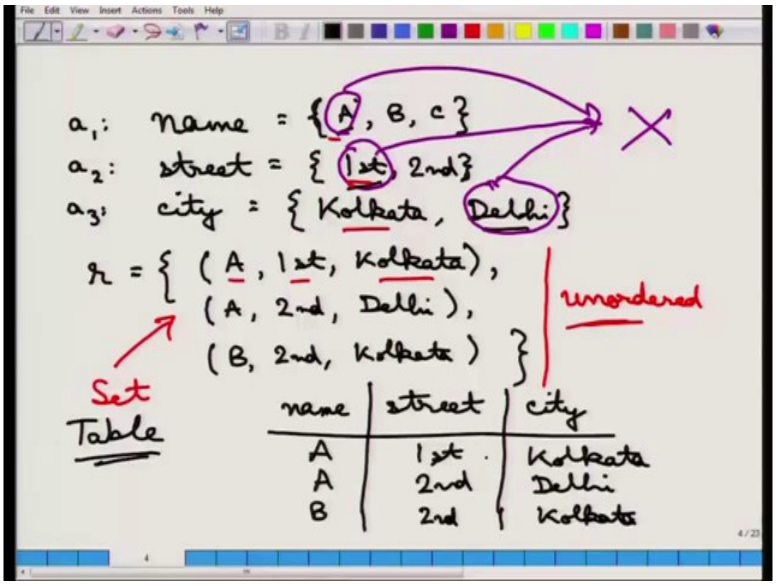
n-ary tuple

$(a_1, a_2, \ldots, a_n)$ $\quad a_i \in D_i$

Cross-product of sets

So let me start off by defining what a relation is. Well, very simply we all understand what a relation is. It encodes the relationship between different types of entities. So, it is defined as an *n-ary* tuple, so what does it mean is that, for example, a particular relation may be a_1, a_2 up to some a_n. So, this is one relation. And each a_i comes from a domain D_i. So, relation is an n-ary tuple, it has got essentially n attributes, each attribute a_i comes from a domain D_i and a particular relation is formed of a_1 to a_n. So, a relation is the cross product of the sets of D_i, These sets are corresponds to these D_i. So, this is the cross product of sets.

(Refer Slide Time: 01:37)



a_1: name = {A, B, C}

a_2: street = {1st, 2nd}

a_3: city = {Kolkata, Delhi}

$\times$

r = { (A, 1st, Kolkata),
(A, 2nd, Delhi),
(B, 2nd, Kolkata) }

unordered

Set

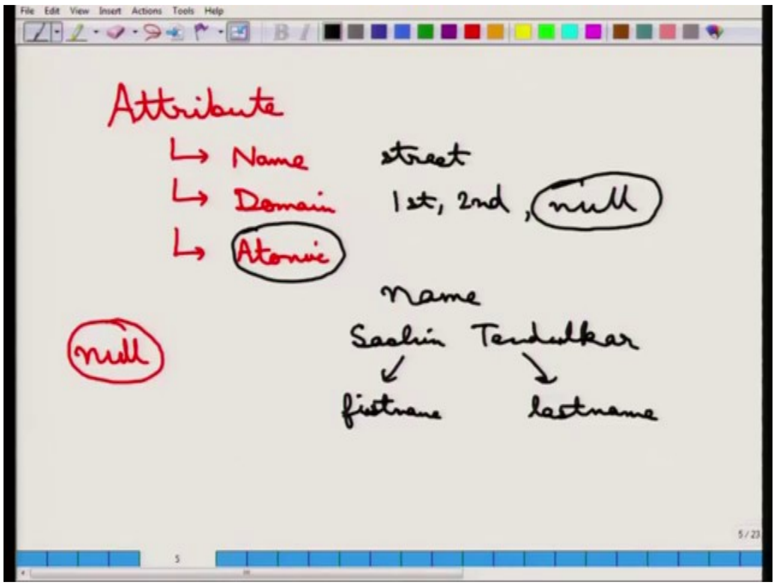
Table

name	street	city
A	1st	Kolkata
A	2nd	Delhi
B	2nd	Kolkata

So, for example, let us take a particular example and say, we have *name* as the first attribute. So, this is one attribute, this is let us say, a_1, this attribute is name. And, Let us say, in the *name*, the particular elements that we are talking about are A, B and C. Let us take another attribute which is called street, which is again very simply say 1st street, 2nd street, 3rd street so on, so forth etcetera. Then let us take third one which is city and let us for example, say this is Kolkata, then Delhi so on so forth. This can be a particular relation. So, it is the subset of the cross product, so any combination can be part of a relation, so for example, I can say that the first n-ary tuple in this relation is *(A, 1st, Kolkata)*. So, what does this mean, that there is an entity, whose name is A, whose street is 1st street and the city is Kolkata, this simply does mean that. And of course, it has it can get many such things it can have *(A, 2nd, Delhi)*, it can have *(B, 2nd, Kolkata)* and so on so forth and this simply defines the relation. Now, note that every member of this relation must be part of the cross product of the domain that we have defined, but it is not necessarily that everything is part of the relation. So, for example, (A, 1st and Delhi), this does not define any relation. So, there is no such tuple like A, 1st and Delhi, This may happen. But all it means is that if you say that there is A, 1st and Kolkata this must come from this, so that is the definition of a relation.

So, that is the first thing, the other thing is relations are unordered, so it does not matter if we write A, 1st, Kolkata then A, 2nd, Delhi then B, 2nd, Kolkata or if we write A, 2nd, Delhi then A, 1st, Kolkata or B, 2nd, Kolkata. So, this is the set of course, and this is unordered, so the ordering has got no meaning and generally what happens is that, generally this is depicted as a table. So, we have all have some idea of what a database looks like and generally what we do is that we depict it as a table. But, this is just a depiction, the relational data model does not depend on how you depict it, this is just a general depiction. So, this is how we will do, so this will go here A, 1st then this is Kolkata then there will be A, 2nd, Delhi then B, 2nd, Kolkata. So, this is just the same relation if we can write it as a table, so that is all about relations.

(Refer Slide Time: 05:16)

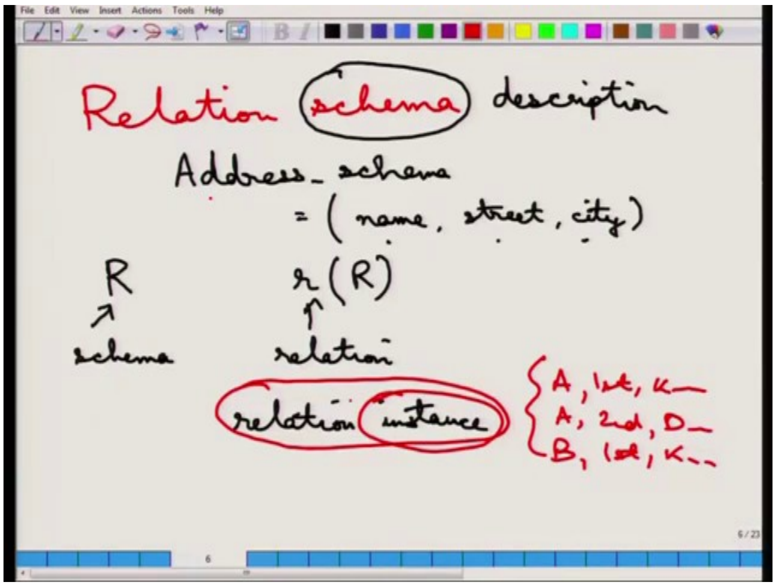


And the next important term that we define is the attribute. So we have been talking about attributes, but we will go a little bit more detail into what an attribute is. So, an attribute has got a name, so it is of course, coming from a domain, so it has a domain and it has got a name. So, for example, street is the name of that attribute and name is the name of that attribute itself and then, there is a domain for each attribute. So, if we go back to the last things, so this is street is an example and the domain here was first and second, then generally these attributes are atomic.

Now, what does atomic mean, this needs to be defined a little bit, atomic means that this cannot be subdivided any further. So, the the first street cannot be subdivided anything further, street may not be doing anything further, but let us take an example of name. So, the name of a person is generally, let us say we can take this name Sachin Tendulkar, this is actually not atomic, because this can be broken into a first name as well as last name.

So, it depends on whether you want to call it atomic or not, but generally attributes are supposed to be atomic. So, that is of one thing, so these are indivisible and these are not set. So, that is the other meaning of atomic and then there is a special value called null, null is part of every domain, so domain of the street can be null. So, for a particular relation, for a particular row, for a particular entity if we do not know what is the value of the street, we can always depict as ***null***. So, ***null*** is a special value that is part of every attribute.

(Refer Slide Time: 07:24)



Relation schema description

Address_schema

= (name, street, city)

R

schema

r(R)

relation

relation instance

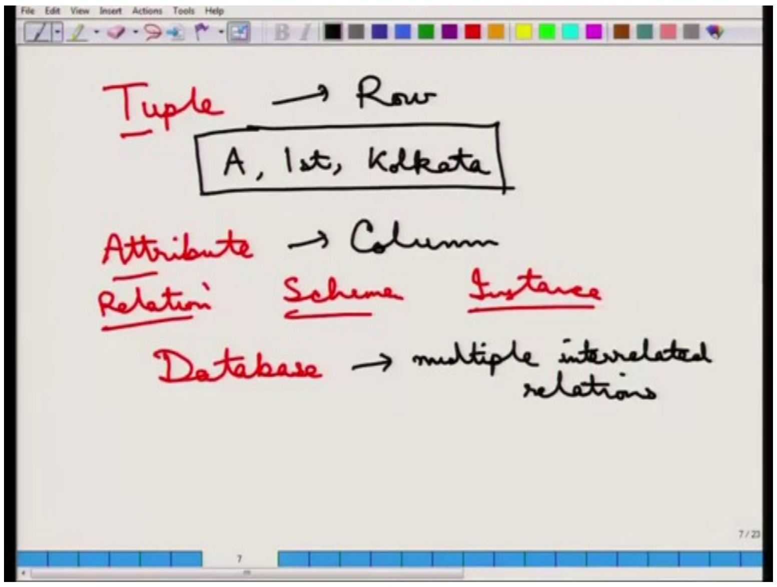
A, 1st, K–
A, 2nd, D–
B, 1st, K..

So, the next concept that we need to define is called a ***relation schema***. so the sets that we defined earlier define the relation schema. So, for example, we can say, going back to our example, address schema. What is an address schema? An address schema says that it must have three fields, the first field is name, the second field is street and the third field is city. So, note that what a schema means, a schema essentially means that this is the description of the relation. So, schema means essentially a description and something more, but very roughly it means the description. And the description of the address is done by this address schema, which is, it contains a name, it contains a street and a city. And the relation is defined over a schema. So, if the schema is for example, if the schema is R, a particular relation r is denoted to be from this schema R.

If the R is the schema and a particular relation is r which within brackets, we have to say that schema of it. So, then it is very clear that the small r, the relation r is part of the R which is the relation schema and a relation instance, is what we saw last time. So, a relation instance, so let us highlight it, this is a relation instance, this is what we saw the table last time some we saw this example A, 1st, Kolkata, etcetera, A, 2nd, Delhi, etcetera and B, 1st, Kolkata, this is a particular instance.

Because, in another relation it may happen in that there is A, 2nd, Delhi and B, 1st Kolkata and so on, so forth. This is a particular instance. So, this is called a relation instance. So once more a relation schema is a description of how the relation should be defined. For example, this address schema and the relation must say which schema it comes from and then there is a relation instance.

(Refer Slide Time: 09:51)



Tuple → Row

$\boxed{\text{A, 1st, Kolkata}}$

Attribute → Column

Relation Schema Instance

Database → multiple interrelated relations

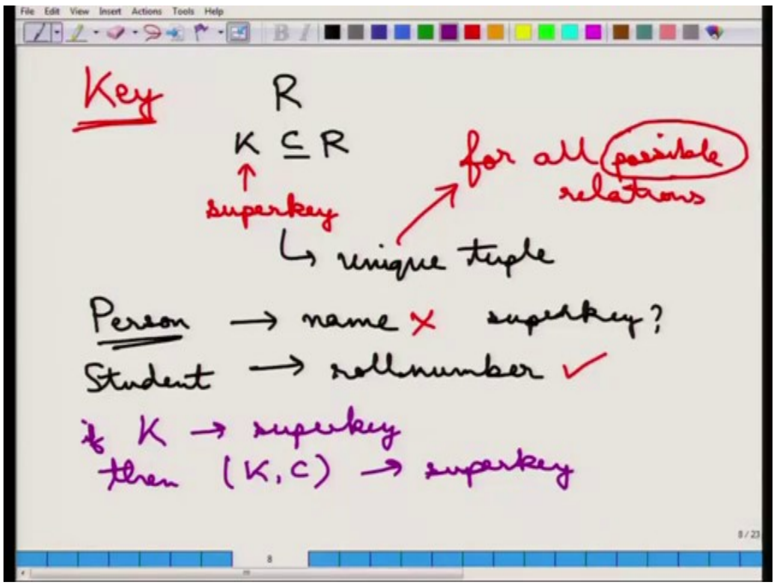
If we then go forward, then there is something called a tuple, this essentially corresponds to a row in a table, this is the one tuple. For example, is that (A, 1st, Kolkata) together consists of one tuple, this is essentially one row, so this is one particular value of the relation that is the tuple and we all understand probably what a tuple means, And just to complete this thing, a tuple corresponds to a row while what we are talking about an attribute that corresponds to a column.

So, these defines about a tuple, attribute, so far what we have got is we have got a tuple, we have got an attribute, we have got a we have defined a relation its schema, its instance, etcetera and now we will formally define what a relational database is. So, a relational database consists of multiple interrelated relations, so a database is a collection of multiple interrelated relations.

So, each relation of course stores about a particular relationship and between two or more relationships there is a connection and there is a relationship, which is also part of the database description. So, the database is not just the collection of the relations, but also the connections between the relations, so that is very important. So, this entirely defines what a database means and it must handle these problems as we went over in the last module, the problems of data isolation, data repetition, etcetera. So, a database tries to handle all of those things.

So, the next very important concept that we will do is called the key.

(Refer Slide Time: 12:04)



So, let us consider a particular relational schema R, now a subset K is called a super key of R if and only if the values of K are enough to determine all the values of the attributes of R. So, to repeat, suppose there is a tuple in R and you do not know all the attributes of tuple in R, but you know, what the super key attributes of the tuple R then it is enough to determine the other values of the tuple, so this is the unique identifying subset of attributes.

So, if you know this then you can complete the entire set of attributes, so a super key essentially identifies a unique tuple. So, to turn it around, we can also say that each tuple has an unique value of the super key.

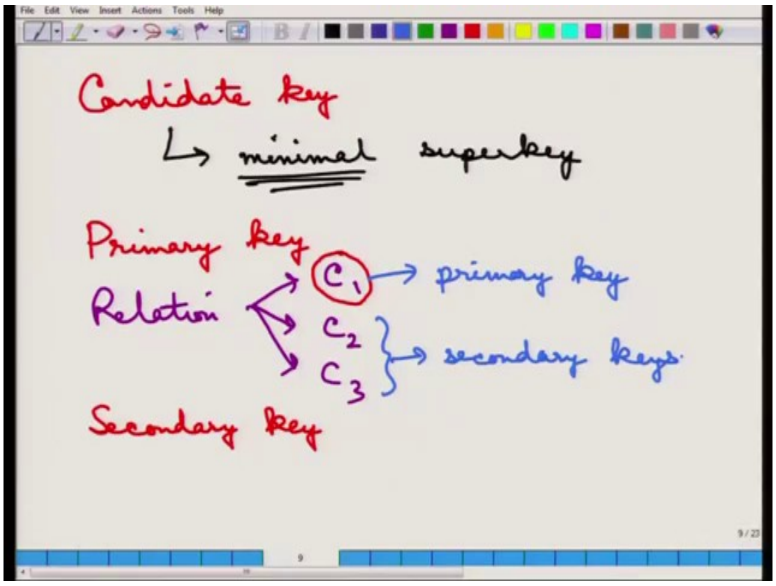
So, some examples. Suppose we consider a person as a relationship and let us consider some attributes, let us consider name as the first attribute, now is name a super key? Well, it is not.

Why? because two persons can have the same name. Now, very importantly and this is a very important point that you must understand, while we are talking about a database and what is a super key etcetera in the context of relation, it must be that the attribute, for example the name here should be able to distinguish the persons for *all possible relations*. So, this super key this is a unique tuple for all possible relations, so what does that mean is that, for a particular set of people for example, it may happen that none of them has a repeated name. So, the name is actually unique for them, but it is still not correct to say that the name is the super key. Because, it may happen that for another collection of persons with the same schema with the same set of descriptions it may happen that the two persons have the same name. So, this is very important. This is for *all possible relations*, so that is why name is not a super key. On the other hand suppose there is a student database and you take roll number. This is a super key, because the roll numbers are designated in that particular manner that every student has a roll number. So, no matter what set of students one takes which university or whatever you go to which institute etcetera, generally the roll number is unique. So, roll number is a super key for the student.

Now the other interesting thing about super keys is that, if K is a super key then anything mixed with K anything after K is also a super key. So, any super set of a super key is a super key as well, so this is much easier to understand.

So, if roll number uniquely identifies the student, roll number plus name will also uniquely identify a student, roll number plus street will also identify a student, roll number plus street plus city will also uniquely identify a student. Because, there is a roll number that uniquely identify, so a super set of a super key is also a super key.

(Refer Slide Time: 16:12)



So, that is the concept of a super key.

Then comes the concept of a *candidate key*, so as you can understand, the definition of super key is a little bit problematic, because any super set of a super key is also super key. So, what people generally tend to work with is what is called a candidate key. Candidate key is a minimal super key, It is a minimal, it is very important. It is a minimal super key. What does that mean? So, a candidate key is of course, a set of attributes. Do remember that every key when we talk about is a set of attributes.

So, a candidate key is again a set of attributes. Minimal meaning if you take any subset of that set it is not a super key any further. So, all the attributes of the candidate key are required to make it a super key. So, the candidate key, in some sense, is the tightest possible set of attribute that one can identify to make it unique and a super set of a candidate key is not a candidate key, because the super set of a candidate key means that it has got something more and; that means, it is not minimal.

So, this is the candidate key essentially gets rid of the problem of the superset of super key. So, that is the called a candidate key, so this is what is called a candidate key and the next important thing that comes is the definition of what is called a *primary key*. But, before we go to the primary key let me highlight one thing that a particular relation may have multiple candidate keys. So, it may have a candidate key 1, it may have a candidate key 2, it may have a candidate key 3, etcetera, etcetera, etcetera it may have multiple candidate keys. Now, note

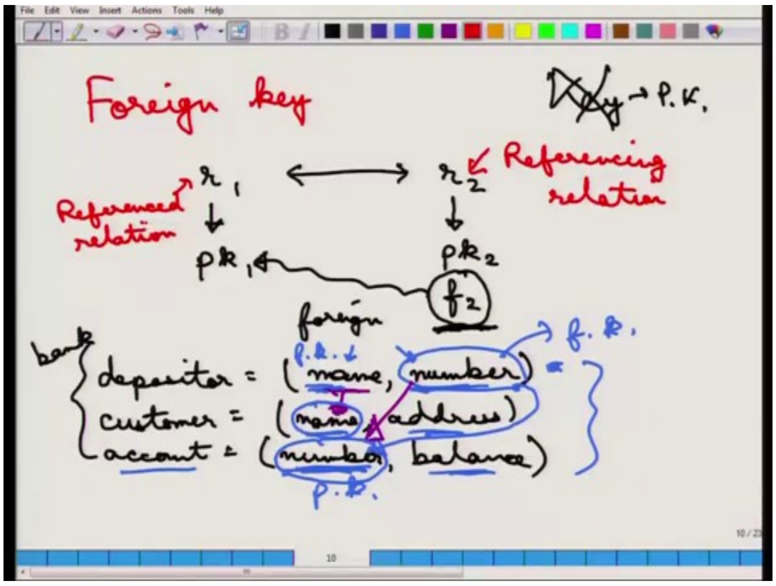
I am still talking about candidate keys of course, it can have multiple super keys, but it can also have multiple candidate keys.

Now, what is a primary key? Versus there is something called a primary key and then there is something called a secondary key. So, both the primary key and the secondary key are candidate keys, and one of the candidate keys is designated by the database designer as a primary key. So, now there is no theory behind this, this is what the database designer, designates one of the candidate keys as the primary key, the rest then becomes secondary keys. So, to repeat, a relation may have a multiple candidate keys one of the candidate keys is designated as a primary key and the rest are secondary keys. Now, if it happens that the relation has a single candidate key that candidate key is of course, a primary key with no secondary keys.

A question will now be coming is that is it guaranteed that every relation has a candidate key, the answer is yes, not in a very strict sense, but for most cases the answer is yes. So, in the worst case or what will happen is the most general cases if one takes all the attributes of a relation that is an uniquely identifying value of the relation, because generally relations are considered to be not repeating. So, the tuple values are not repeating, so there is this problem, there is this concept of the non-redundant information.

So, generally all tuples are unique, so which means that if all the attributes are taken together then that uniquely determines a tuple. So, in the worst case all the attributes of a relation together constitutes its candidate key and that is it I am so a primary key if there is only one candidate key that is a primary key.

(Refer Slide Time: 20:11)



So, next we move on to the last bit on the key thing which is called a foreign key, before we try to understand what a foreign key is we highlighted that relationships have different connections between them. So, suppose there are two relationship r1 and r2 with some connections between them and let us say the primary key of r1 is called pk1. And this has got some attribute pk2 and then this has got some other attribute f2.

Now, what may happen is that this f2 comes from the pk1 of the other relation. So, what I am trying to say is this attribute the f2 of r2 is the primary key of another relation. So, then this is called a foreign key, because this is a key of a foreign table, this is essentially the key in colloquial language means like a primary key, although there is nothing called a key, this is an important point just to note down there is nothing called a key, this key there is nothing called a key it, but generally people mean it is a primary key.

So, this f2 is a primary key of something else, so that is why it is called a foreign key. So, let us consider an example of a bank accounting system, where there is a depositor relationships. So, a depositor relationship has got all of these. Let us say, the depositor has a name and a number and let us say there is a customer relation. So, this is all part of the bank database, there is a customer which has got a name and some address and then there is an account which has got a number and the balance in that account, this is an example of a bank schema.

So, let us try to decode what does this mean and the account is about the different deposit accounts that we got. So, it has got an account number just like our savings bank account number and then the balance and the customer is just like you and me, everybody, we have got a name. And let us assume for the time being that the name is the primary key and we have got an address. Now, the depositor, this, defines a relationship between which customers have which accounts.

So, these numbers essentially mean that customer one whose name is whatever something has this account number. So, this is refers to this account, so this is essentially a foreign key, because this is a number here, so this is a primary key and this number essentially refers to this number. So, that is why it is a foreign key and similarly this name essentially refers to this name, the customer name essentially refers to this name, so that is why these two are foreign keys.

So, this is also a foreign key and this is also another foreign key. This may happen. So, this is why the concept of foreign key is important and we will see later much more about these foreign keys, essentially just a very quick digest is that if a particular name does not happen in customer, it cannot happen in the depositor and if a particular number does not occur in this account database it cannot happen in the number. Because, I mean there cannot be information about an account for which there is no account balance.

So, that is the foreign key and just to complete the story about foreign keys. So, this is called the *referencing relation*, the one which has got the foreign key and this is called the *referenced relation*.

So, that completes this module on the relational data model, so just to recap we studied about what relations are, what tuples are, what attributes are, how is the database defined in terms of those things and then very importantly, about the different keys. So, we started off with super keys, then we defined candidate keys and then we defined primary keys and secondary keys and finally, foreign keys and how is the foreign key helpful or useful.

Thank you, so in the next module we will talk about relational algebra.