Artificial Intelligence:

Representation: Resource Description Framework (RDF)

Prof. Deepak Khemani

Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Module – 04

Lecture - 05

Okay so we are continuing with our study of representation. And we have said that what logicians and mathematicians tend to do is to choose adhoc predicates. Now this is fine provided you are a mathematician or logician who is interested in algorithms for proving things and showing whether your proof procedure is good or bad that kind of stuff. But its not very convenient from the knowledge representation perspective. If you are a knowledge engineer trying to represent knowledge in some domain then the more the number of predicates you have, the more the number of rules that you need to have. So for example we have said all men are mortals. So man is a choice of predicate name and mortal is a choice of predicate name and then we have to establish some association between man and being mortal. Supposing I said all women are mortal or all humans are mortal, all babies are mortal, all cats are mortal. For every such choice I would have had to introduce such rules because if you have to make any useful inferences I must have predicate which link a given category to certain property essentially.

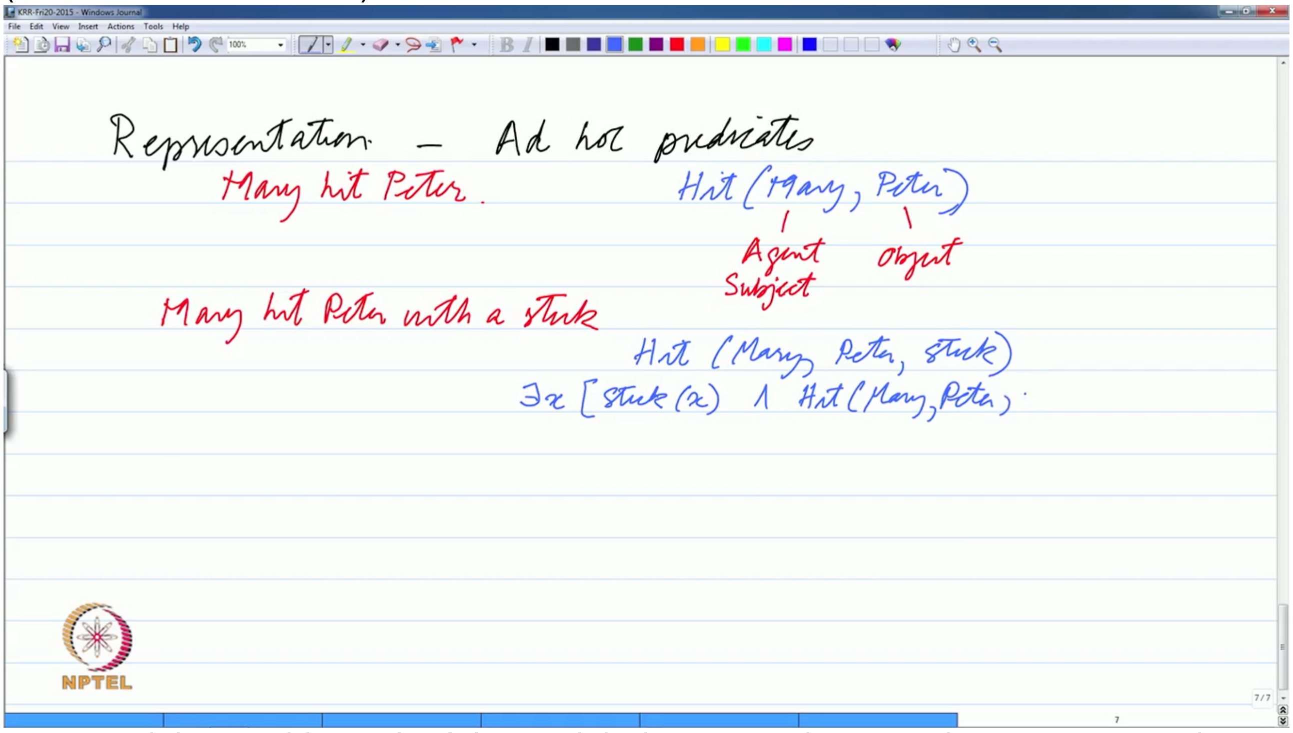
So the more the predicate we choose the more the number of properties we will have. And the more the number of rules we will require. Further we run into problems which we can illustrate by looking into some examples. Let us say that we want to say that May hit Peter. Let us say we just want to represent this. And we will see gradually that talking about actions is also introduce a degree of complexity into the whole process. We will try to see how to make that a neat process. So what’s the simplest way of representing this? Is that as I said if you were a mathematician or a logician, you will say okay I will represent this using a predicate called Hit and I will say that the first argument is the person who is hitting and the second argument is the person who is being hit.

So observe that our understanding of what this predicate, the binary predicate that we have introduced means, as an implicit understanding of the fact that Mary is the Agent or you might say Subject if you are talking in terms of natural language processing and Peter is an object. It is implicit in our representation that when I say Hit Mary Peter, Mary is the person who is doing the hitting and Peter is the one who is getting it. Its not obvious at all, it is just a convention between us that we understand that. And you can see that in natural language processing there are ways to get around this problem. So you can identify who is the doer and who is the object and so on and so forth. For example in English language in active sentences you always start with the agent. So Mary hit Peter you say. You don’t say Peter was hit by Mary, thats a passive sentence. So the grammar tells us who is the agent and who is the object things like that. We need to make such things clear essentially.

So thats one problem. There is an implicit understanding that we require. And we have already said that the semantics of such statement says that there is something called a binary relation called Hit in the domain which corresponds to two elements which we understand by saying that one of them has hit the other person. But what if I wanted to say this Mary hit Peter with a stick. Now suddenly things have changed a little bit. So again if you were not particularly concerned about how efficient is your representation and you know things like that you might say something like this, Hit Mary Peter stick. I mean if you are just being very adhoc you might say something like this.

If you are a little bit more careful you might say something like this. There exists a x stick x and Hit.

(Refer Slide Time: 05:16)



Representation – Ad hoc predicates

Mary hit Peter.

$\underset{\text{Agent / Subject}}{Hit(Mary}, \underset{\text{Object}}{Peter)}$

Mary hit Peter with a stick

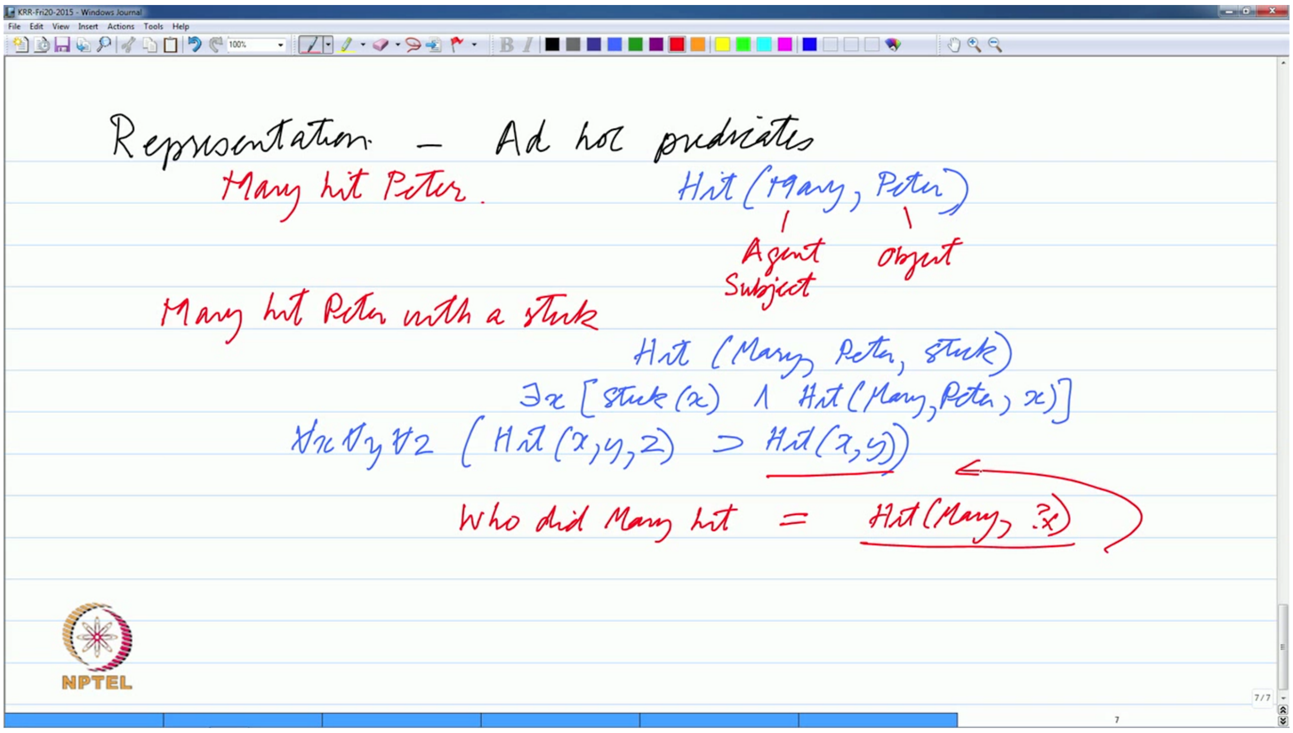
$Hit(Mary, Peter, stick)$

$\exists x\, [Stick(x) \wedge Hit(Mary, Peter,$

so you might say this. Both of them might be acceptable to various people. See it really depends on what you want to do with the representation. We is the kind of inference that you want to make. How do you want to use the knowledge that you are representing. But already you see that you have now two predicates, one is a predicate of arity 2 which says Hit Mary Peter and other is a predicate of arity 3 which says Hit Mary Peter with some object essentially. Now you have to first of all remember what the meaning of first argument is. And secondly if you want to create a knowledge base what will you store essentially.

So for example you might want to store facts like this essentially. For all x for all y for all z, Hit x y z implies Hit x y. You might end up wanting to create rules like this. Depending on what is the purpose. You may want to ask the question who did Mary hit? I may first have to produce the predicate which is a binary predicate Hit x y and then see that okay I can match this predicate with this one and through the process of chaining I can arrive at the answer.

(Refer Slide Time: 07:44)


Representation – Ad hoc predicates

Mary hit Peter.

$Hit(Mary, Peter)$

Agent Subject — object

Mary hit Peter with a stick

$Hit(Mary, Peter, stick)$

$\exists x\, [Stick(x) \wedge Hit(Mary, Peter, x)]$

$\forall x \forall y \forall z\, (Hit(x,y,z) \supset \underline{Hit(x,y)})$

Who did Mary hit $=$ $\underline{Hit(Mary, ?x)}$

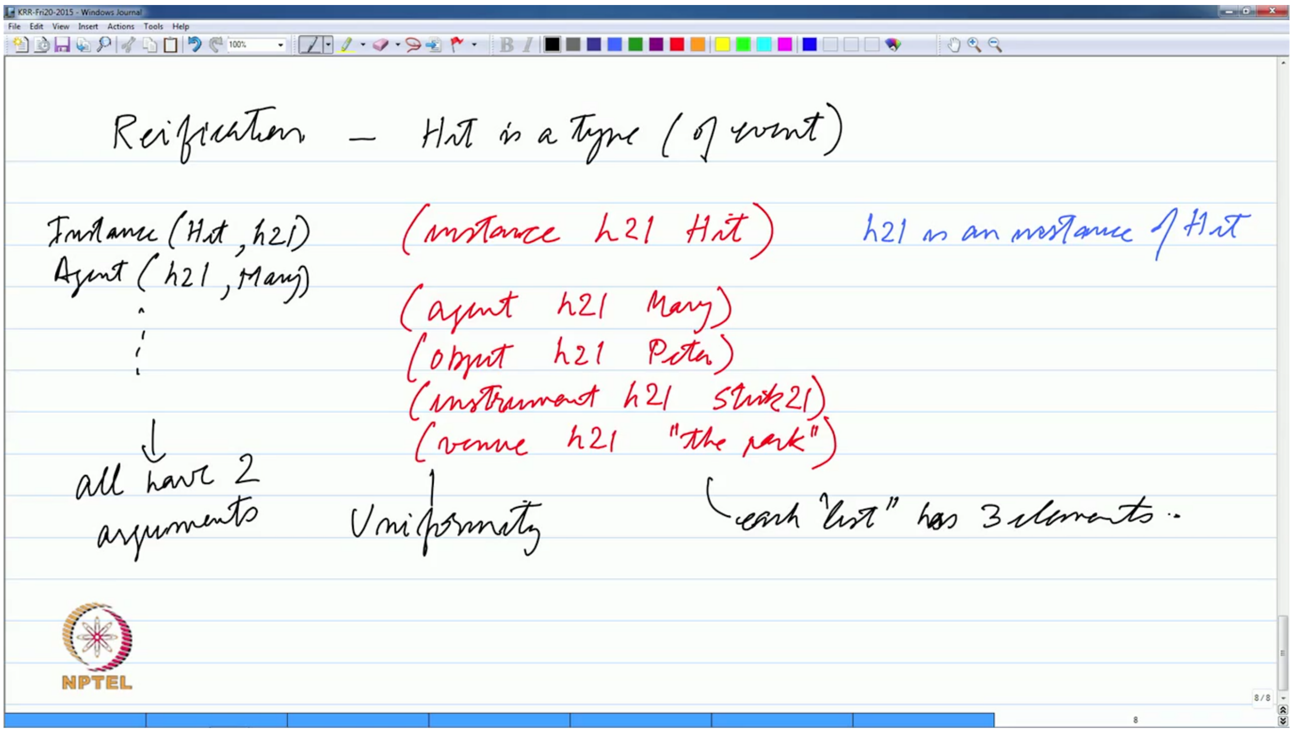
Now there could be more complications. I could say Mary hit Peter with a stick in the park. Now will you introduce predicate with arity 4 now? And what if I say in the park yesterday. Then will you introduce a predicate with arity 5? so you can see that adhocism will not take you very far if you want to represent knowledge in a general sense which is useful in many places. So enter our process of Reification. We say that we can tackle this as something like follows: we can say that Hit is a type of lets call it event, we will talk about event may be next class or something like that. And we can identify that we are talking about a particular instance of Hit essentially. We can say things like this instance h21, thats just a name I am using of type Hit.

So this statement you can read as h21 is an instance of Hit. Which I already defined as a type of event and I am saying h21 is an instance of that. But now you can see we can add more information in a modular fashion. We can say agent of h21 is Mary. We can say object h21 is Peter. We can say instrument of h21 is lets say stick21 without worrying too much about how I am talking about the stick. Venue h21 is whatever, “the park”, however we represent that. We will see shortly that you can instead of having the single entity “the park” you can point to something else, which is a structure is something that we are arriving at essentially and so on and so forth essentially.

So what is striking about this kind of a representation? The first thing is that its modular. That you don’t have to introduce a new predicate of different arity for whenever you add some more information to the event. So Mary hit Peter, Mary hit Peter yesterday. Mary hit Peter with a stick you know these two would be two different predicates of arity 3 and so on. So you dont have to do all that kind of stuff. There is some degree of Uniformity which has come into here. So I have written this in a particular notation. You could have written it as follows. Instance, it doesn’t matter. You could have written it in a more logic-like notation but then but that decides the point essentially.

What is interesting about this is that if you look at this then all have two arguments and in this notation you can see that each list has three elements.

(Refer Slide Time: 13:08)



Reification — Hit is a type (of event)

Instance(Hit, h21)
Agent(h21, Mary)
⋮
↓
all have 2 arguments

(instance h21 Hit)   h21 is an instance of Hit
(agent h21 Mary)
(object h21 Peter)
(instrument h21 stick21)
(venue h21 "the park")

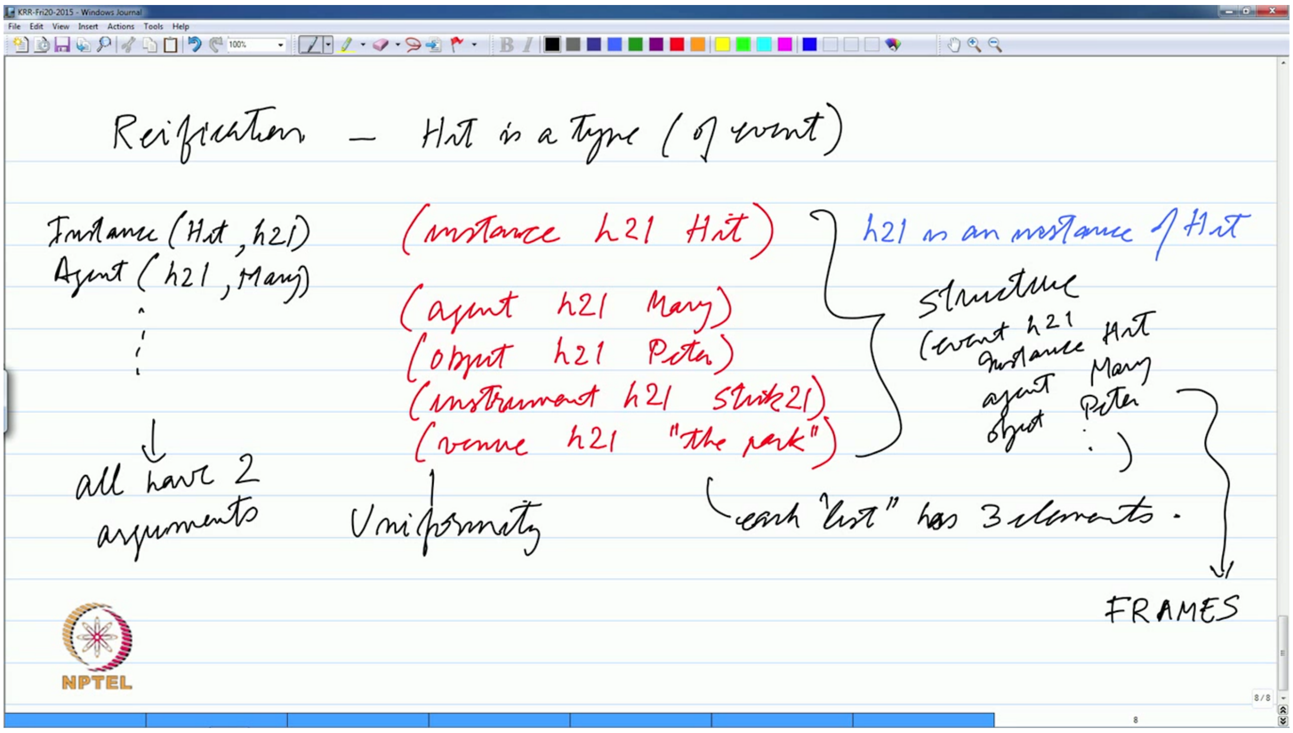
Uniformity

each "list" has 3 elements..

a little bit later, see whenever we talk about knowledge representation and logical reasoning essentially our task is to connect different predicates which are related to each other in some way essentially. So here you can see that all these five predicates that we have written, instance, agent, object, instrument and venue are related to each other in the sense they correspond to the same event. So they are connected by this particular symbol h21 essentially. So we can trace. We can say who got hit in this event? Who was the person who was hit? All these questions we can do through this process of chaining through these kinds of facts essentially.

We have seen forward chaining and we have seen that whenever you want to do chaining, essentially you have a lot of rules and you have a lot of facts and there is the problem of matching. So essentially boils down to searching of the rules and the facts which may mean extra work. Which may add to the complexity of the reasoning process. So it is possible that we may want to put all these things together into a structure. So you are all familiar with object oriented programming and the notion of structure in C. We could do something like that. So we could say event h21, instance of Hit, agent Mary, object Peter and so on. We could have one compound or composite structure to represent all these different facts essentially. This is an idea that we will explore later and its an idea which is called as Frames which you will explore in later class essentially. And the idea of Frames is to keep all these information together so that its easily accessible, you don't have to search for the matching facts and things like that.

(Refer Slide Time: 15:19)


KRR-Fri20-2015 - Windows Journal

Reification — Hit is a type (of event)

Instance(Hit, h21)
Agent(h21, Mary)
⋮

all have 2 arguments

(instance h21 Hit) — h21 is an instance of Hit

(agent h21 Mary)
(object h21 Peter)
(instrument h21 stick21)
(venue h21 "the park")

Uniformity

each "list" has 3 elements.

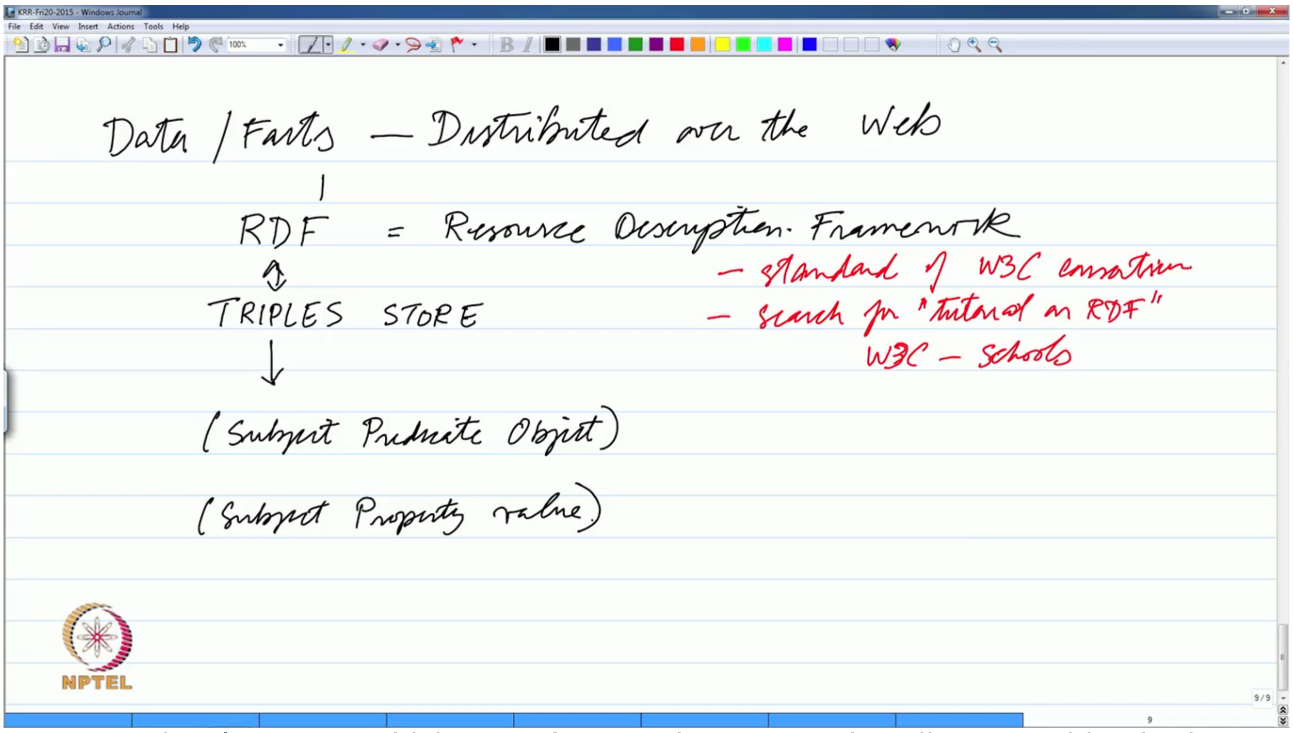
Structure
(event h21
instance Hit
agent Mary
object Peter
⋮)

FRAMES

Right now we want to focus on this part. It has 3 elements and this has become a very attractive thing. And these kind of things are known as triples because they are made up of three elements. And they have become very popular nowadays because of the emergence of the web essentially. Now you all know that you can buy a book from some store or CD from store or something from online store. Your program which enables this transaction has to access someone else's database. So Amazon may have books, flipkart may have a set of books or infibeam may have some books. All these, each seller stores database of books and you want to access those books. So the basic idea is that data has become distributed. When i say data you can say facts as far as logic is concerned. They are getting distributed over the web. And what has come out is a language which is called RDF which stands for Resource Description Framework. And it has become standard for representation on web.

I will strongly urge you to go to this, so it is the standard of the w3c consortium. So you should go to, just search for a tutorial on RDF and you will probably be taken to one of w3c sites. And they have their schools essentially. They have tutorials on everything you need for web based programming. An RDF has emerged as a important tool for web based representation. And RDF we also associate the word with as triple store. Everything you express in RDF is in terms of triple essentially. And they are or all facts are broken down into statements which have free objects and the kind of triples that we are talking about is Subject Predicate Object. Or some people say Subject Property Value

(Refer Slide Time: 19:22)


Data / Facts — Distributed over the Web

RDF = Resource Description Framework

– standard of W3C consortium

– search for "tutorial on RDF"

W3C – Schools

TRIPLES STORE

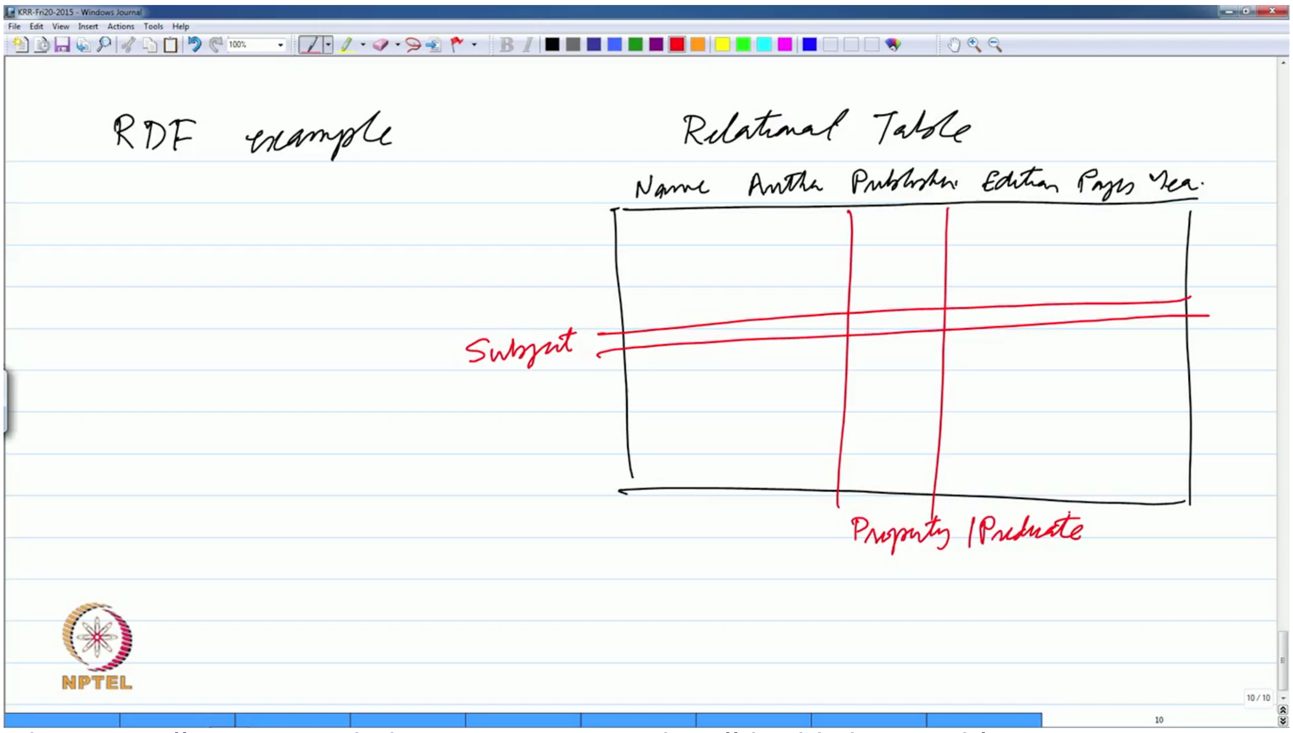
(Subject Predicate Object)

(Subject Property value)

so an RDF is a language which sort of constrains you to describe everything in the world in terms of these three values. So lets take an example. I am not going to the syntax and technical details of RDF you should go and read the tutorial. I will just give you some idea of what's happening here. Lets say you have a relational table, its you were not working on the web, if you had a large lets say physical store, you might have created a table like this for whatever products you have essentially. So this table may have certain, so lets say this table has books essentially, so when you talk about books, you can say Name, Author, Publisher, Edition, Pages, Year and so on. So you will have many columns in your database table. And for every book you have in your store you would have all this information available which is very nice if you can store everything in one place. But now if there are many sellers and if they all want to talk to each other and they want to talk to buyers and so on and so forth, we have to have a mechanism for representing this information in some uniform fashion which is accessible to everybody.

And one question that arises is that what is the name of the property that, we will assume that somehow we will arrive at consensus on the names of property. So if you are talking about a particular row in the table, you are talking about a particular book and in some sense you are talking about a Subject. And if you are looking at a particular value, lets say Publisher then you are talking about property or predicate.

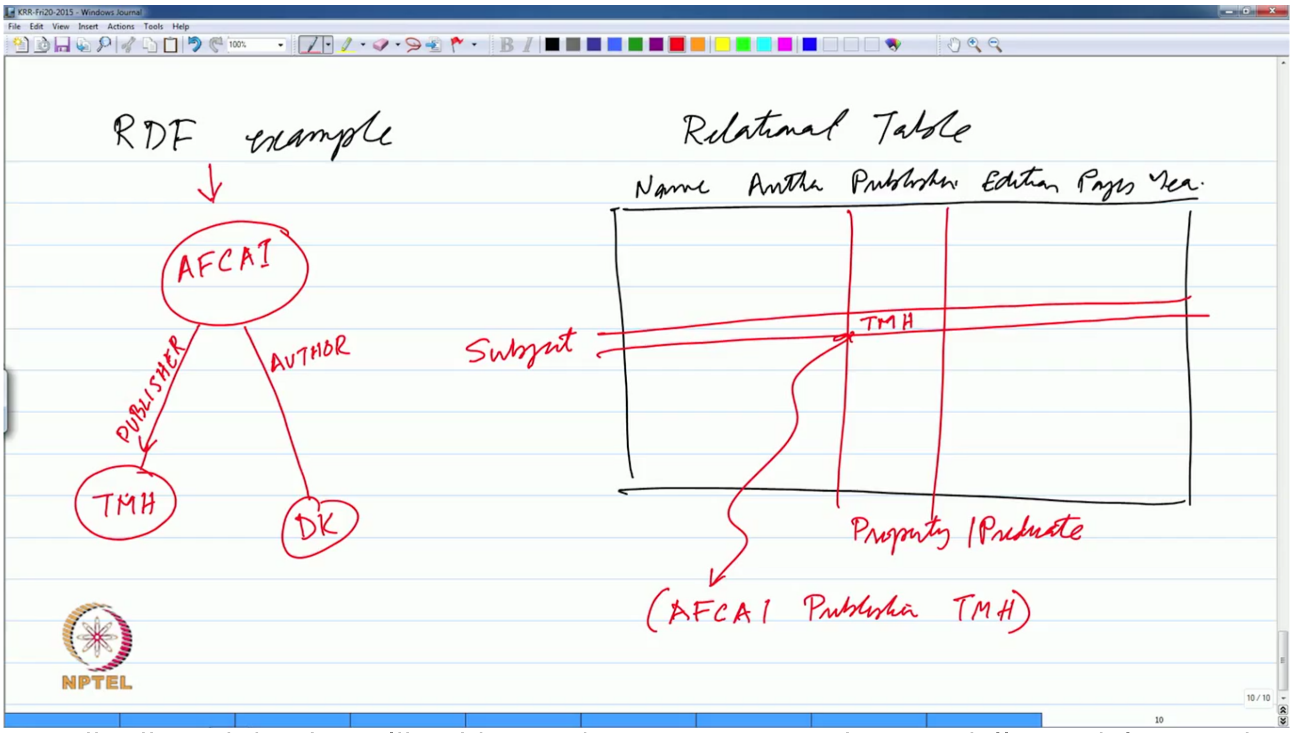
(Refer Slide Time: 22:10)



what RDF allow you to do is to represent each cell in this huge table as a separate entity. But ofcourse we saw the idea of frames and we may want to keep this thing together and so on and so forth but thats a basic idea essentially. So you remember this notion of, so lets say this publisher for a particular book is Tata McGraw Hill which is the old name for McGraw Hill India, because its easier for me to write, you can see that it is Tata McGraw Hill. Then I could, that cell corresponds to a triple, lets say its my book on AI, I will abbreviate with AFCAI, and then thats the Subject. And the predicate is Publisher and the value is TMH or the object is TMH. So every cell in this entire table will be represented separately as a triple.

So we can view this RDF example as saying that I have a book, lets call it A first course on AI. You can think of it a node. And then there is a predicate, you can think of it as an edge. And then there is a value. So this is Publisher. I may have another one for author for example.

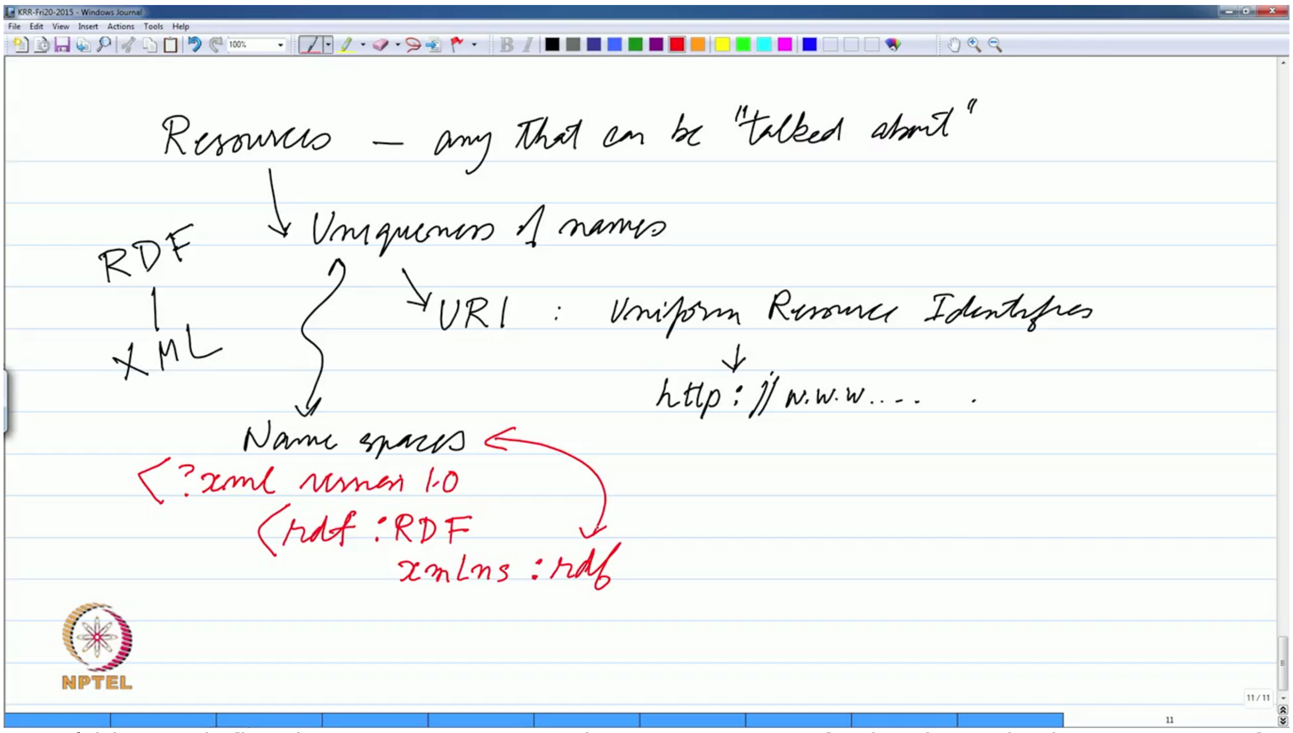
(Refer Slide Time: 24:38)



so a distributed database like this can be seen as a graph essentially, a triple store is a graph. And now a days people call them as graphical databases. And then they have their own query languages. For example there is a language called SPARQL so you should look up the extension of SPARQL. Its an interesting self referential acronym. So what are there resources? They are anything that can be talked about. So one thing that we are interested in is Uniqueness of names. And to do that we always identify resources by what is known as a URI. So earlier we said that instead of saying Mary we will say mary21, instead of John we will say john33 and so on, just to make sure we are talking about something specific. But in RDF you are required to use this Uniform Resource Identifier. URLs are like URLs infact URLs are a type of URIs and typically they will look like http:// something www. Something. Now RDF is build upon XML which is an eXtensible Markup Language which has become kind of defacto standard for representing RDF so you could talk of representing RDF in different ways. And we talk about, for the perspective of uniqueness we talk about namespaces.

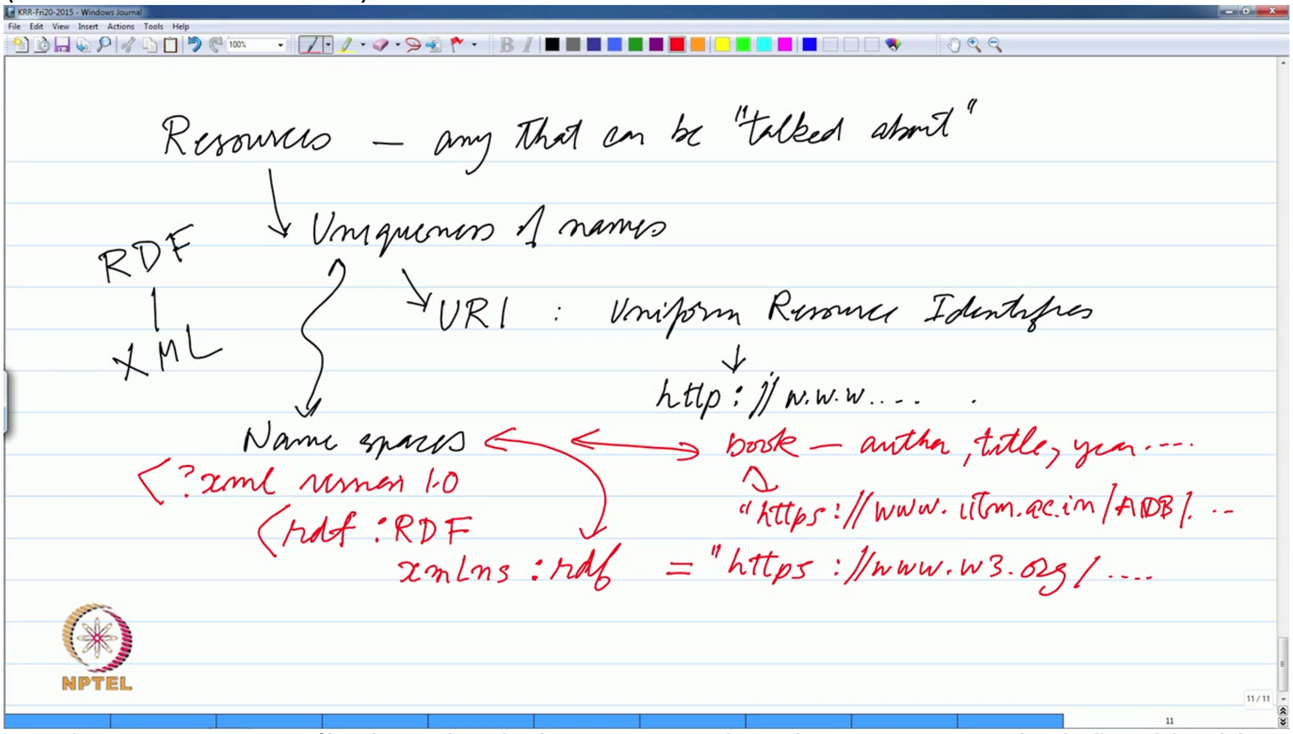
So the idea of namespaces is that it is a resource which defines what is the vocabulary which is used inside. So for example I may have something like this, the RDF document may look something like this, I may say that xml version 1.0 or something and then inside this I will say rdf RDF and I will define a namespace which is defined in xml, so xml namespace rdf. So this is a namespace, rdf is a namespace.

(Refer Slide Time: 28:34)


Resources — any that can be "talked about"
Uniqueness of names
RDF
XML
URI : Uniform Resource Identifier
http://w.w.w.... .
Name spaces
<?xml version 1.0
<rdf:RDF
xmlns:rdf

I could have defined a namespace another namespace for book. So its just a name of the namespace and I will say that in my namespace book I have things like author, title, year and so on. So not only do you need to specify that author refers to author of the book but you have to say that I am taking this word from a particular namespace which is called book. So you have to specify where that namespace is. So this may be for example in some place, somewhere in this website there is this namespace essentially. And I might say that my namespace is in some place , lets say its in my lab, AIDB lab. I have defined this namespace.

(Refer Slide Time: 29:49)


Resources — any that can be "talked about"
Uniqueness of names
RDF
XML
URI : Uniform Resource Identifier
http://w.w.w.... .
Name spaces
book — author, title, year....
"https://www.iitm.ac.in/AIDB/. ..
<?xml version 1.0
<rdf:RDF
xmlns:rdf = "https://www.w3.org/....

so whenever I now talk about book then I mean that the namespace is defined in this particular website. So thats why URIs are looking like web, infact web addresses are a particular case of URIs. Because in principle they can be dereferenced essentially that once you know what the namespace is you know which site is www.iitm.ac.in.AIDB lab and so on and so forth, it can in principle be dereferenced even if they cannot be dereferenced. For example if we dont really have such a website it will become a unique namespace. So thats the advantage that we get out of this. And then so I will define this, I will define that I have two namespaces here, one is rdf and one is book. And then lets say that we are talking of books available in our department library or something like this. Or if we were settling up a shop or something then typically we would say this .

We are now describing an entity, Description and the first thing you will say is that when I say rdf:Description it means I am using the word Description which is described in the rdf namespace. And then I can say that it is rdf:about, so I specify as to what is it that I am describing. So I can give namespace an address, lets say iitm ..and then this book. So what follows after this is everything that I want to express about this rule. So I may say for example book, so book namespace, author . So I can either give a string or I can point to another resource. So I am describing a resource called this book which I am using rdf to describe. So thats the subject, the subject is a book, predicate is this author, and the value is either a string, you can just give a string or you can point to another resource essentially. So you can see that values or objects, so if you say subject predicate object, the object can be a resource so you can have you know big linked graph structures over which you may have to search essentially.

So I dont really want to go into the details of rdf here except the fact that we came to rdf because we want to have a uniform representation. So instead of having achoc predicates with arity 3 and 4 and 5 and so on we are now come down to predicates of arity 2. so we will always talk about subject predicate object or subject property value. And everything on the web is described using such triples as they are called. And such stores are also called triple stores. We will come back to logic and in the next class we will talk a little bit about, when we talk about events like hitting how can we represent them meaningfully. So we will stop here.