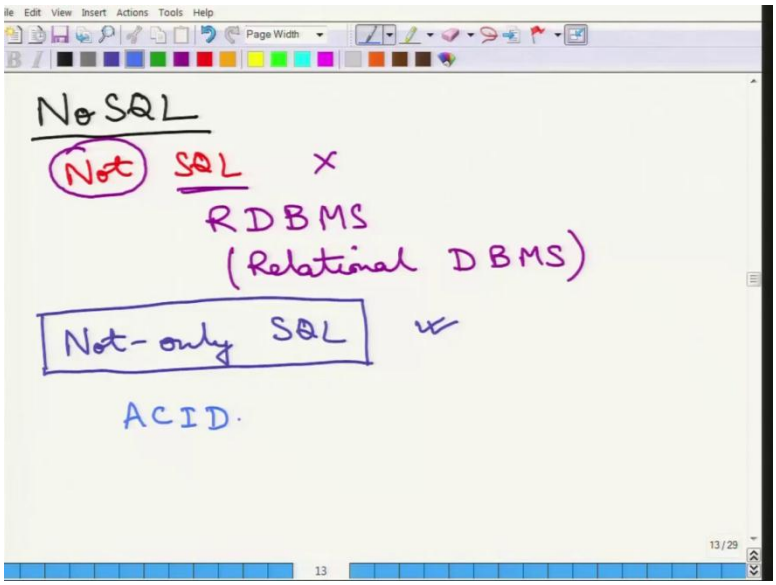
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 45
NoSQL: Introduction and Properties

Welcome, we will start on a new module today which is on NoSQL. So, NoSQL and big data at the last two modules that we will cover as part of this course. Note that these are new additions to the undergraduate database courses and these are actually, because these are new developments in the database paradigm.

(Refer Slide Time: 00:32)



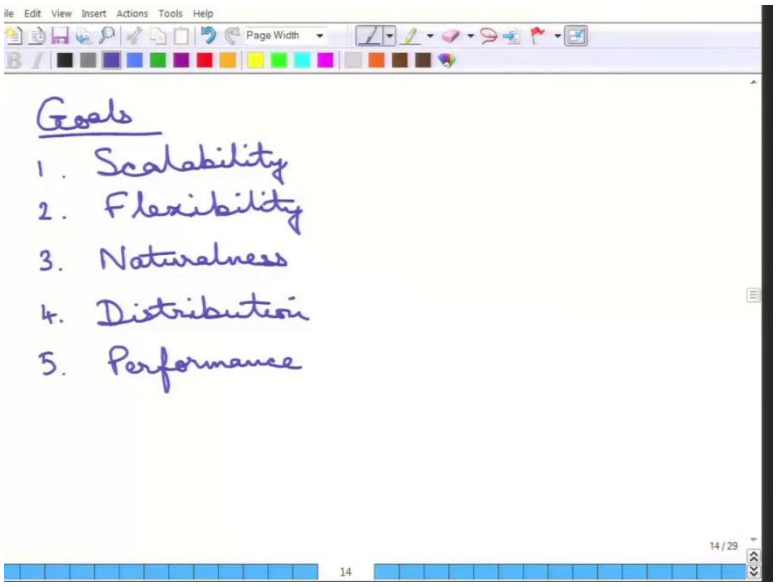
So, we will start off with NoSQL, this is NoSQL. So, as you can probably guess the name NoSQL is that, why is it called NoSQL and what does NoSQL mean actually. So, SQL stands for the Relational Database Management System, so the RDBMS. So, SQL is a quintessential programming tool to handle relational databases and the NoSQL started off by saying that this is we will not use SQL. So, it says it is not SQL, it starts off saying not SQL, because SQL essentially stands for the Relational Database Management System.

So, this is the RDBMS that is the term. So, it is essentially relational, this is the relational model that we have been studying so far in the beginning of the modules, we have been studying the relational algebra, the relational model, etcetera. So, it started off as a not

SQL; however, it is no more a not SQL any further, because the... So, the originally the NoSQL model started the NoSQL databases started by saying that we will not use relational database model, we will not use SQL, we will not use the properties of this ACID properties, the Atomicity, Consistency, the Isolation and Durability this we will not use, we will use something different.

Then, the database researchers and the database community all over the world realize that the RDBMS is just too powerful to ignore. So, then it switched on to something which is now essentially called not only SQL. So, note that NoSQL stands now for not only SQL, it is not SQL, so this is the correct term for this thing. So, it does not restrict itself to using only SQL or rather SQL here stands for the relational database management system, it does not use only SQL or the RDBMS it uses something more on top of that. So, essentially the idea is to get out of acid properties, so not to follow acid properties on that, but to follow something that is not just acid property.

(Refer Slide Time: 02:53)



So, the goals of this NoSQL system whatever it, the goals of NoSQL systems is can be generically summarized as the follows. So, the first one is scalability, so scalability, now one important thing about scalability is that, the database says the relational databases are very scalable, they can go up to large amounts of data, etcetera, etcetera. But, the still there is a problem with scalability when you think of a very large project.

For example, the entire Google web pages or the entire face book graph or things like that when it is really, really large, the relational database models may or may not scale. So, all the searching, etcetera they may or may not scale, all the transactions, the schedules, etcetera may or may not run well. So, the scalability essentially the volume of data that it tries to handle. So, that is one of the goals is to make the system as much scalable as possible, so you keep on adding data without much trouble.

The next is flexibility, so flexibility... So, what does the flexibility mean is that the relational database models part into a particular schema and that schema must be very rigidly followed. So, if there is a student table which requires an id, which is a non null value and a roll number and name, etcetera it must have all of those things, so that is not flexile. So, this NoSQL paradigm, the NoSQL databases wanted to make it flexible. So, it can have different kinds of students for example or different kinds of attributes for a student.

And now if you think of these social networks of the face book, etcetera, not every people will have lots of friends, not every people will upload the photos, not every people will have blogs, etcetera, but it is flexible. So, every people, every person is an identity, is an entity, but it can be flexible. So, that is the other goal or flexibility, then it is naturalness, so again naturalness what it means is that sometimes just to fit it into the relational model, things have to be broken up into this the relational schema in a slightly unnatural way, I mean this is not too much of a concern for RDBMS, in generally they are quite natural.

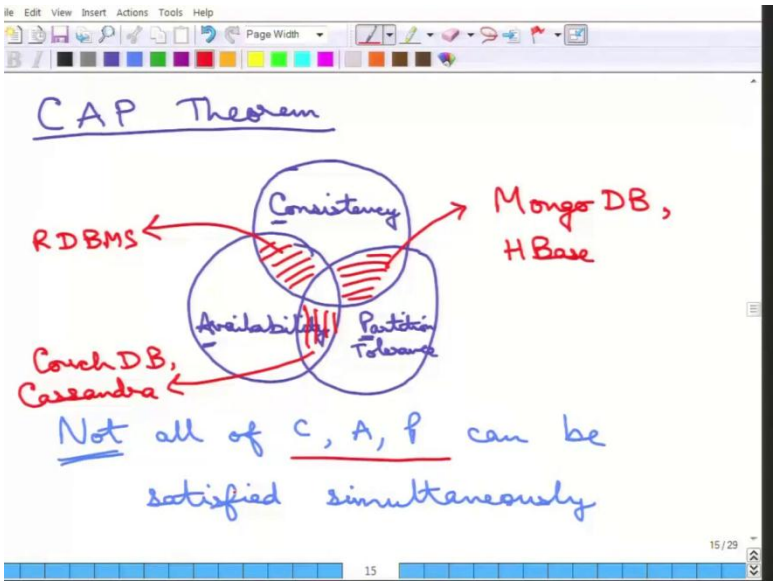
But, sometimes if you saw that bank account thing, the depositor, etcetera then they may not be natural. So, they can have this account and loan number and all of these things together, so that is the naturalness, so this... So, the goals of NoSQL system is to make it much more natural, so that is the thing. Then, the fourth point is extremely important, this is called distribution. So, distribution essentially means is that no matter how larger my SQL system or oracle or whatever, the sum the relational database systems can handle, they are essentially routed to one machine.

So, they the essentially the entire data lies in one machine, one big disk, whatever you can get it, but there are mostly not handling the distributed systems. So, distributed systems are not generally handled very well, very elegantly by this SQL systems. So, the

distribution essentially is to say that the data can be distributed across different machines, it is a normal distributed system paradigm and that has to come naturally within the database system, it is not has to be imposed on top of it, it should be part of the database design itself, so that is the distribution.

And the last thing of course, is that it has to still give it a good performance. So, the system has to still perform very well in a sense that if I want to update something in my face book account or something, it should be quick, it should be almost immediate or my other friends should be able to see it and so on and so forth. And somebody writes a comment I should be getting it and so on and so forth. So, these are the goals of the NoSQL systems.

(Refer Slide Time: 06:56)



And; however, very, very interestingly there is something in this space which is called a cap theorem. So, a cap theorem essentially says that there are three things, so first is consistency, there are three properties of a system that is very desirable, consistency. The second is availability and the third one is partition tolerance, so partition. So, how much it can tolerate partition, so that is the distributed net essentially. So, this is why this is called a cap theorem. Because, this is C, A and P, so this is why this is the cap theorem.

So, essentially we want the ideal system should have all these three properties, but what the cap theorem essentially says is that not all of… So, not all of C, A, P can be satisfied simultaneously. So, not all the three properties can be satisfied simultaneously, so that is

the cap theorem. Now, very, very interestingly we have all learnt that what is the theorem, the theorem must be rigorously proved etcetera. However, the cap theorem although it is called a cap theorem, there is no proof for it, it is just a conjecture that there cannot be any system that has got all of these things.

So, note very importantly that although this is called a theorem, there is no proof it is just a conjecture that is widely believed to be true and accepted to be true, but there is no formal proof that there cannot be a system, where all these three properties can be satisfied. Anyway, so there are systems on the other hand that can satisfy two of it at one time. So, for example, if you take this space here which is that something which satisfies consistency and availability, but not partition tolerance, this is our traditional relational database management systems.

So, before that what does consistency mean is that… So, consistency we have been studying consistency for these transactions etcetera is that things must be consistent. So, this is one of the acid properties that for example, this account balance or sum of account balance should be, the total should be the same before and after the transaction and so on and so forth this is consistency.

Availability is that the database is always assumed to be available. Because, even if it crashes it comes up, it recovers there is this whole recovery protocols and then as if that nothing has happened it to behave as if has nothing has happened and then new transactions can behave that it is again available that is from this thing. So, RDBMS's shared this properties of consistency or rather exhibit the properties of consistency and availability, but not partition tolerance. It is not very good for distribution that is the thing that is the biggest criticism of RDBMS. So, that is why it is consistence and available, but not partition tolerance.

On the other end, we can have systems here, which is let us say consistent and partition tolerance. So, it can maintain consistency and it can be quite nice to distributer, this kind of systems. Some examples are mongo DB, you may have heard the names of this, this is mongo DB and H base. So, these are certain things which are consistent and partition tolerance, but they are not guaranteed to be available all the time. So, some data items in mongo DB or H base may not be available. So, what essentially happens is that?

So, age based let us take an example of age based, there are different tables that go to different machines very simply proved. Now, what happens is that some data item has been inserted into a table which is in machine a. Now, if machine b wants to query it is not immediately available it may or may not return it, because it is in some other machine, so it will take some time to update and so on so forth. So, it may not be available in that sense and machine a may be down.

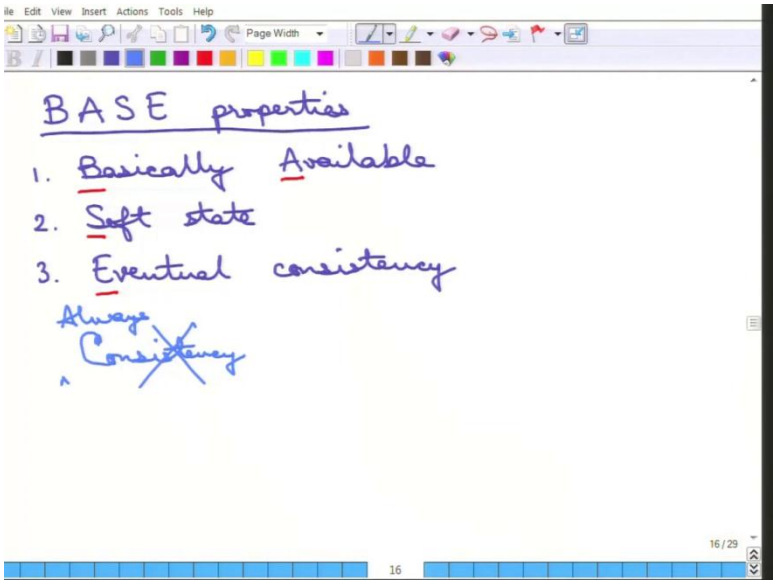
So, machine a maybe may have this is because this is a distributed system machine a may have some faults etcetera. So, it is not available till it comes up and so on so forth so, that is a mongo DB and H base. On the other hand there is this other kinds of systems which is this availability and partition tolerance and these kind of systems, the examples are your couch DB and the probably a bigger example is Cassandra.

So, these systems are available they will always make the data available, because they will essentially what they will do couch DB and Cassandra they will replicate the data. So, they will ensure that if machine a gets a copy, then machine b also gets the copy and so on so forth. Of course, they are partition tolerant they nice scale very nicely with distributed system etcetera, but they may not be consistent. Why are they not consistent? Because, suppose the copy in machine has been updated.

Now, the copy at machine b does not still know about the update in the machine a. So, it is not consistent, because the states now vary between machine a and machine b and somebody which who queries the data may get it from machine b in a inconsistent state. Because, the consistency is maintained or the updates is maintained in machine a, but machine b has not yet got it, so that is the problem with all of these things.

So, every of this systems has suffered from one lack of one of the properties, so that is the famous cap theorem. So, essentially this is what the cap theorem says that not all of this just to repeat that not all of CAP can be satisfied simultaneously. So, then let us move on to the next part of NoSQL is that. So, NoSQL started off by saying that they do not care about acid properties.

(Refer Slide Time: 12:49)



So, what they do care about is something called the base properties. Now, you may see that the name of the properties base is essentially to counter acid and we will see what the properties are, there are three base properties, the first one is basically available, so it is saying that the this is basically available. Now, what does it basically available is that the system does guarantee availability, but not right then I mean may be it is available after some time etcetera, etcetera, but it is not right then, but it is basically available, it is mostly available that is the first property, so that is basically available.

The second property is called soft state. Now, what is soft state? The soft state is the following, so the state of the system is supposed said to be in a soft state I mean it is said to be in a soft state what it means is that the state must change without any input. So, one important thing about RDBMS is that suppose there is an RDBMS that is around and without any new query coming, without any new insertion, deletion etcetera coming the state of the database does not change.

However, for this NoSQL system the state may change, the state may change because it needs to make it available, it needs to make it consistent so on so forth. So, it just it is synchronizes with other machines, other distributed setups etcetera, etcetera. So, that is why the state may be changed, so what it means is essentially is that it is just in a soft state. So, the state may change without apparently any input coming, without apparently any querying done that is why it is called in a soft state.

And the third and the last property is eventual consistency. So, eventual consistency these are the... So, eventual consistency is that data will be eventually consistent without any interim ((Refer Time: 14:50)), so what does that mean is data will be... So, at some point in time data may not be consistent, because as I said that machine a updates and machine b does not get it. But, eventually after some time machine a will translate or will transfer this update to machine b will let the machine b know inform it about the updates, so then machine b will also be updated.

So, it will be eventually consistent without I mean, so without any perturbation in the sense that you do not need to give any more inputs to machine b to say that get go and get yourself updated etcetera machine a will finally, update it. So, it is eventually consistent, but not so one important thing to remember is that this NoSQL systems are not always consistent, they are eventually consistent. So, at some point in time if you take a snapshot, it may not be consistent.

So, why is this called base properties? So, these are the things it is basically available soft state and E that is why it is called base. Now, this you see is just to counter the acid properties that is the thing. So, very, very importantly what it does is that the consistency as we understand in a database system is that it is consistency always, so it is not always consistent. So, always consistency is sacrificed, so it is only eventually consistent. So, this is just to counter the acid properties.