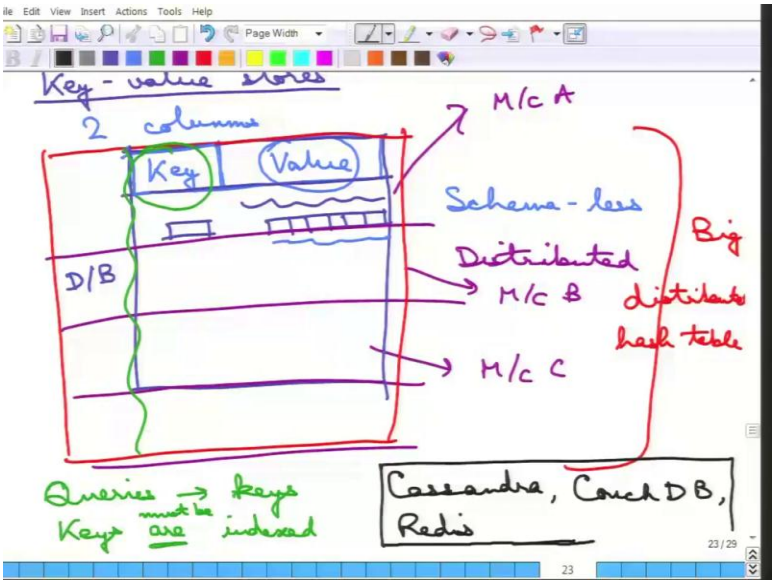
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 47
NoSQL: Different NoSQL Systems

(Refer Slide Time: 00:09)



Let us now move on to key value stores. So, the key value stores, the basic idea is that, there are only two columns. Even, if you want to think of it, there are only two things, the first is a key and then, there is a value; that is it. So, that is the entire database schema you may say or whatever, there is a key; that is a value. It is almost like hashing.

So, what does it mean is that, the key can be anything, but it is generally only the text. So, the key can be generally anything, but it is a idea either assigned by the database system itself or one can assign it is own key, but it is more like a hashing. And the value can be anything; in fact, it is so, I mean it can be even said that the value can be internally broken up into multiple attributes, multiple values and so on and so forth, multiple dimensions and anything it can be done.

But, the key is a single thing, which is generally assigned by the database itself. So, how does one do? Searching it, still it just like one wants to find the particular Tuple. So, one has to specify it is key, once a key is done, it finds it and then, the value is all returned as a single entity, single may be text or may be something else to the system. Now, the

system has to parse the value, the system has to understand that, for that particular database, the value consisted of multiple attributes or it is a text or it is an image or it is something else, but that is up to the particular application.

The database by itself does not say anything about, what the value is, it just save it is value; that is it, it just want to block to it and there is a key and then, there is a value; that is all and it is simply essentially an object for the key value systems. So, then the whole database is essentially just one big table with just the keys and values, this is just one big table and that is your whole database, that is the entire database, this is one big table.

So, you can again consider it as a table thing; that is all. So, the keys are stored and then corresponding to each key. Why is it called a columnar sometimes? It is the keys are stored and then, corresponding to each key, there is a pointer, there is a values are stored to where the object that which the value, the object of that value is stored. It is essentially just like hashing, so the hash keys are stored together and from each hash key bucket; there is a pointer to the actual value or the object; that is it.

So, this in some sense then becomes there is a term called schema less, because it does not matter, what the schema is, the database parse does not care about, what the schema in this value, it just say, there is a key and there is a value. So, for all such databases, the schema is just key and value. So, it does not care, what the schema inside it, whether it is one big object and image or it is actually stored in a relational database manner, etcetera, it does not care.

So, this is highly scalable and this is nicely distributed. Why it can be very easily distributed is that, you cut this database into multiple portions and give these things to machine A and this to machine B, this to machine C and so on and so forth. And if you have more data, you just cut it out more and you give it to machine D, etcetera. So, it can be very easily distributed, because all it needs to do is to just cut it out like this and give the keys essentially; that is all.

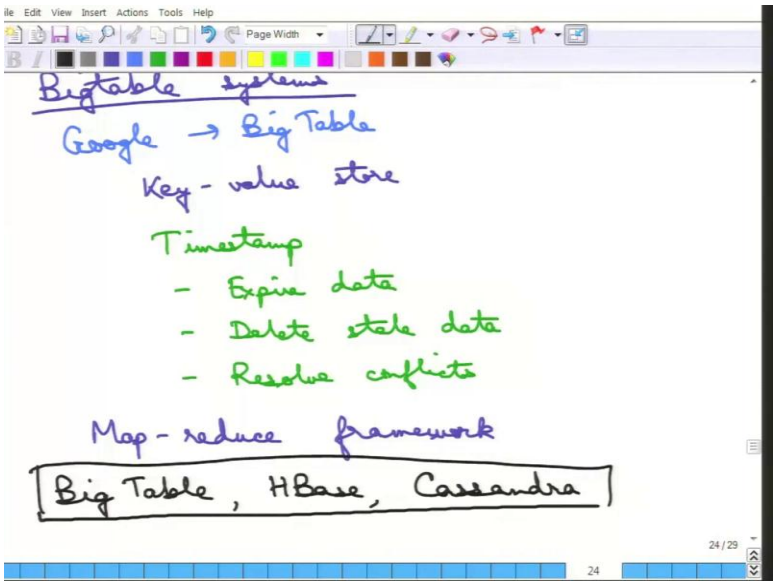
So, this is essentially, very much essentially, let me state it once more, this entire thing is one big distributed hash table; that is all. So, because this is like one big hash table, but this is distributed into different machines. So, that is all that is there, so it is just a big distributed hash table. Now, all the queries are on the keys only, so queries are only on

keys, one cannot query the value itself, just like in the hashing, one cannot say give me the key for which the value is this, it cannot be done.

So, all the queries are on the keys only. So, that is one disadvantage of this thing, because the values cannot be looked inside. So, the keys are necessarily indexed. So, the keys, there must be some kind of keys or keys are generally indexed, because otherwise one cannot keep on searching all the keys. So, keys are indexed, these are must be indexed actually, keys I should say are must be indexed, because otherwise there is no gain in using such a thing.

So, this is all big table and it can use this memory as a cash. So, certain machines, so wherever the queries land up in, the memory of those machines can be used as a cash. So, certain keys which has been queried, it can be again queried just from the cash; that is the thing. So, what are examples of this kind of system? This is Cassandra of course, Cassandra is one big example. Then, there is a Couch DB; that is one example, then there is Redis and then, there are many such examples of this key value stores. So, that is a, these are the examples of this key value stores.

(Refer Slide Time: 05:40)



So, let us move on to the Big table systems. So, you may be knowing that in some time back about the decoder so back, Google had this big table, Google had a schema as a relational, I mean Google had a database not relational, Google had a database, which is called the big table. So, this started the big table system started from that big table; that is

why it is called big table, Google called it actually big table. It is essentially a key value store only.

So, this is just a key value store and it is nothing more than that and but, data can be replicated, it provides better availability and so on and so forth. So, the big table system uses a Timestamp. So, a Timestamp is used to store whatever is happening. So, whenever a data is inserted into the table or whenever it is modified, etcetera different Timestamps are given.

So, Timestamps essentially help to finally achieve consistency in the sense that, if an update happened at particular Timestamp, it can be later on, if the query comes at some other Timestamp, it can be later on synchronized and so on and so forth. So, there are different types of Timestamps and the Timestamps can be used to do something else. So, it can expire the data, so one can say that this data will be valid till a particular time point.

So, you can then use that to delete stale data, an old data can be deleted and of course, to resolve conflicts. So, that is the most important thing about Timestamp and we have already studied a whole lot of how the Timestamps can resolve conflicts and this is the read, write conflict set, so that can be resolved. So, that this is all used and then, there is another important term that is here. So, there is a Map-reduce the frame work, that goes hand in hand with big tables to compute certain things to do certain operations on this tables, the Map-reduce frame work can be used.

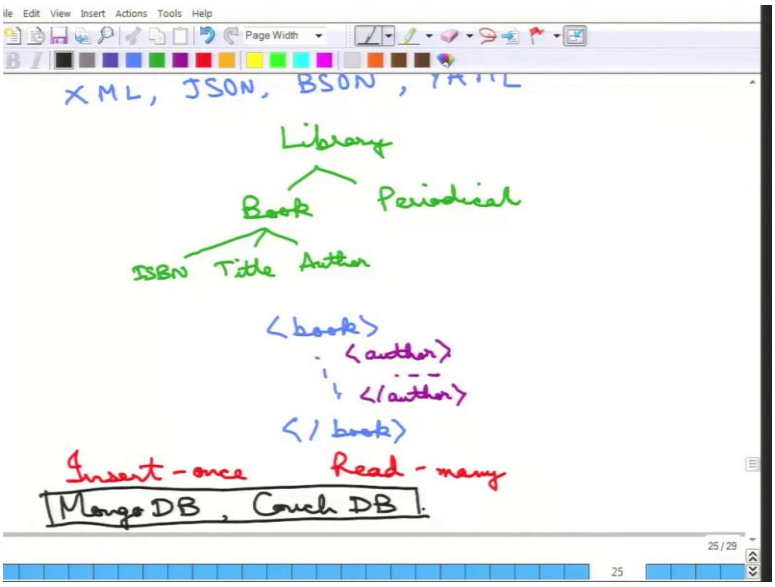
So, the Map-reduce frame work essentially, if you simply understand the Map-reduce is the following manner is that, suppose you want to compute a function on different things, you can map portions of that input to different machines, each machines does it is job and then, you can bringing those, this I mean summaries and can then reduce it to one numbers.

So, an example is that, suppose you want to add 100 numbers and you have four machines in your hand. So, what you will do is, you will map the first let us say, this is a very simple scheme, we will map the first 25 numbers to the first position, the next 25 numbers to the second machine and so on and so forth. Now, each machine will compute the sum of 25 numbers. Why it is done? This is being done in parallel and this is a much lesser work and can be done much faster, etcetera as well.

So, then it reduces a map; that is a map, the first part is a map, then it reduces to these four numbers that you bring it in and submit up. And now of course, this gives you the sum of all the 100 numbers. So, you have to ensure the correctness of I mean of course, sum is a very easy example to see, why it is correct, but there are other things, where the map and the reduce may not be so easy, but that is the basic idea of the Map-reduce frame work and that is being used for this big table systems.

What are the examples of big table systems? Of course, big table itself that I have already said, this is big table, then H base, H base is one very important thing. And again Cassandra, because Cassandra, you can use this Map-reduce frame work and you see these are all key value stores at the end anyway. So, these are all examples of this big table system.

(Refer Slide Time: 09:28)



So, then let us move on to document databases. So, document databases, what is the basic idea of the document is that, it uses documents as the main storage of data. So, there are different document formats, one is XML, one is JSON, you may have heard XML and JSON. You may not have heard BSON, this is a binary JSON and then, there is something called YAML.

So, essentially XML, JSON, etcetera, these are just storages in the document form and these are certain formats of data. So, the data can be stored in certain formats, for example, the XML format, data can be stored in an XML format. An XML is essentially

a tree kind of thing and the data can be said that, it can be stored in a nice tree kind of format and that is the XML.

The flexibility of XML is that, one can define the own schema and that can be part of the XML document. So, one can define that suppose I will have a library, the library is one big chunk. So, you can define in XML like this, so suppose there is a big library, a library will have different things may be a book, may be a periodical, etcetera. So, you can define this entire structure, the tree structure. Book will have a title, will have authors and it can have a, I mean ISBN number, so on and so forth.

And it can have, so you can, then this is your schema, this part becomes part of the XML and you can define your entire library collection using this thing. So, what you say is you first define a library scope, then within that there are book scopes, then within that, there are title values and so on and so forth; that is the XML thing that can be done. So, the document that the XML, the whole thing itself is the database, so the particular book, so within library, there will be particular book, so the XML tags are given like book. And then, within book you will mention certain things and then, there is a closing thing for books, this is just like the HTML tag.

So, this itself becomes a key, because this is all inside one big XML document, so that is the thing. So, document this can be put to a particular location; that is the URI, etcetera and can be these things. But, very, very importantly just like the other systems, what is inside a document, so to find out whether there is a author field etcetera that needs to be parsed.

The database parse by itself does not provide any mechanism to parse it is up to the application to say, I want to find out the authors. So, it will look for that particular author thing etcetera. Note that, this is one of the very, very important advantages of the relational database format, once the schema is fixed, once the schema is given, then the database does all the parsing etcetera.

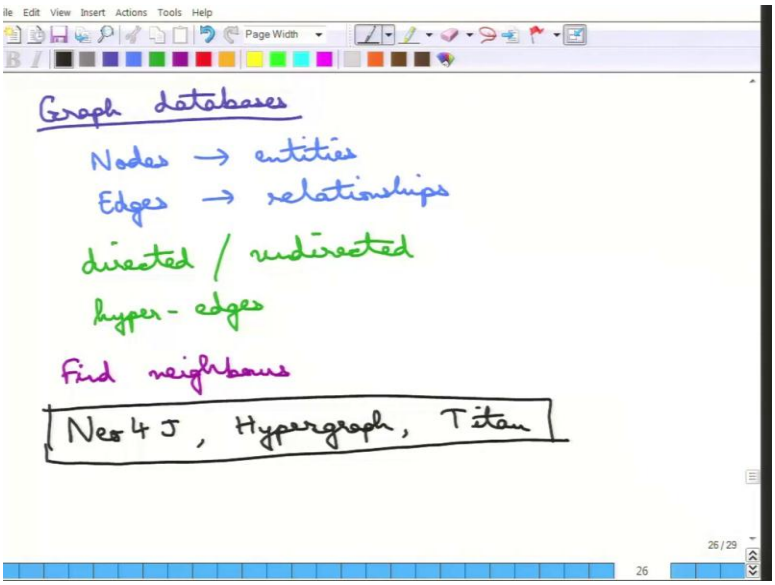
So, once you say I have the schema, I have the following attributes that roll number, name and year and department, etcetera, then you just say I want to find a query on department and the database does it. You do not have to then get that entire student tuple and find out the where the department is, the database does this for you. But, not for

these kind of document databases or in general for this key value stores, it is not big even you, you have to parse the database has to be…

So, again it can use the Map-reduce frame work and this can be very useful. So, the document databases are actually very, very useful in the following sense, where you insert so the data is inserted once, so this is only insert once and read many. So, for example, in typical library etcetera, the library will acquire a book particular once and many, many people will search that book or do something with that book read it. So, that is very these thing and these can scale up much better manner.

So, that is the good thing about document databases and the examples of the document databases are, this is your Mongo DB, it is a Document Database and so is Couch DB couch DB is another Document Database. So, these are the two big examples of this thing.

(Refer Slide Time: 13:58)



So, finally, we have the graph databases. So, the graph database the entire all the Tuples etcetera is represented as a graph. So, what are the nodes, the nodes are the, so the nodes represent the entities in the graph. So, this is the face book graph that we have been talking about nodes. So, what are the nodes in these thing, the every person every account holder is a may be a node and then, the edges encode the relationship between this. So, the frame relationship is essentially an edge.

So, this entire thing is stored like a graph database. So, it is not easy to store this in a relational manner, but it is much more natural and easier to store it in this graph format for a social network. For example, it is a very natural node edge kind of a relationship, it can be directed of course, it can be directed or undirected depending on what the application is or what the kind of database it is saying, it can have hyper edges as well.

So, what are hyper edges is that, it is not a single edge between two nodes, it can a particular edge may be over multiple nodes, for example, a group in a face book can have multiple nodes. So, that one group may be represented as a hyper edge of all the members in that, all the nodes in that. So, why are graph databases useful? Graph databases are useful for certain types of queries; these queries are let us say you want to find neighbors of everyone.

So, I want to find friends of friend and these kinds of things that, if it is in a relational format or if it is in some key value format, etcetera it is much harder to find it. You have to get all the values of myself and parse them and then, get the values of those etcetera. But, in a graph thing, it is much more natural to find friends of friends or find neighbors etcetera, this is much more natural and it is actually being shown by face book and other people, other companies that this is much faster as well, much more efficient, much more scalable.

And what are the some examples of this is, there is this NEO 4 J, then there is Hyper graph, then there is Titan so on and so forth, there are many other examples that one can find out, so these are the examples of these thing. So, what are the things about NoSQL is that, so NoSQL, although it started as an NTSQL movement, actually it said that no we do not want SQL any further, it is not so more, it is just not only SQL, it cannot live without the relational database.

The relational database as I said earlier is just too powerful and so it is essentially a compromising saying that in some cases, the relational database does not scale or the consistency requirements, the transaction requirements are just too strict for certain applications and it may not be required. So, acidity is sacrificed and you rather go for this distributed setup, which is much more scalable, so that is the thing.

But, NoSQL as is not good for every scenario banking scenarios must use this consistency and transaction, protection, durability, etcetera in NoSQL system just fail.

And even nowadays, it is people are realizing more and more that consistency matters a lot, so eventual consistency may not be desirable in all kinds of applications. So, there are now people are although NoSQL is very much hyped and it is taking over, it is nowhere near relational databases.

And actually database researchers are again going back to this traditional DBMS things, I am trying to fix certain problems in NoSQL using this traditional RDBMS methods. So, most legal sys systems still use this RDBMS, etcetera and NoSQL is currently well it is as I said it is currently too hyped etcetera and NoSQL. But, however, NoSQL there one important thing about NoSQL is that, with the advent of big data and cloud computing etcetera most of those try to use No SQL system. So, they try to use the properties of NoSQL or do rely on NoSQL as the backend databases as the support systems for all of that. So, that ends the module on NoSQL and later, we will touch on a little bit on big data.