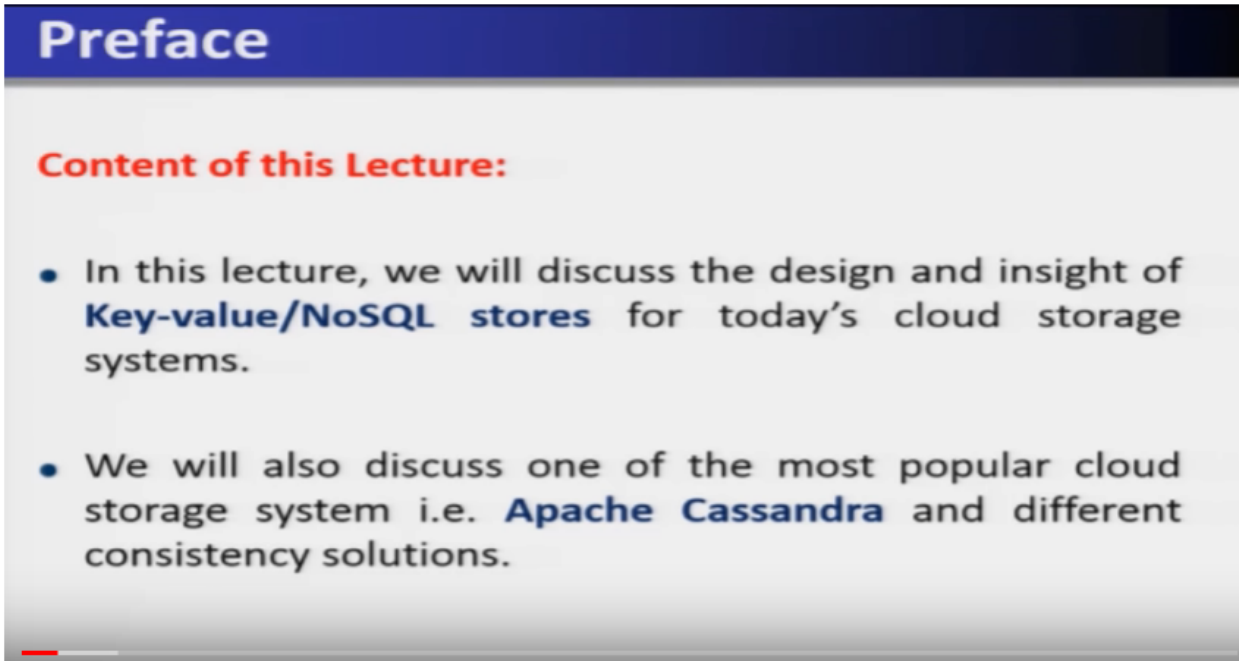
Lecture 12
Design of Key Value Stores

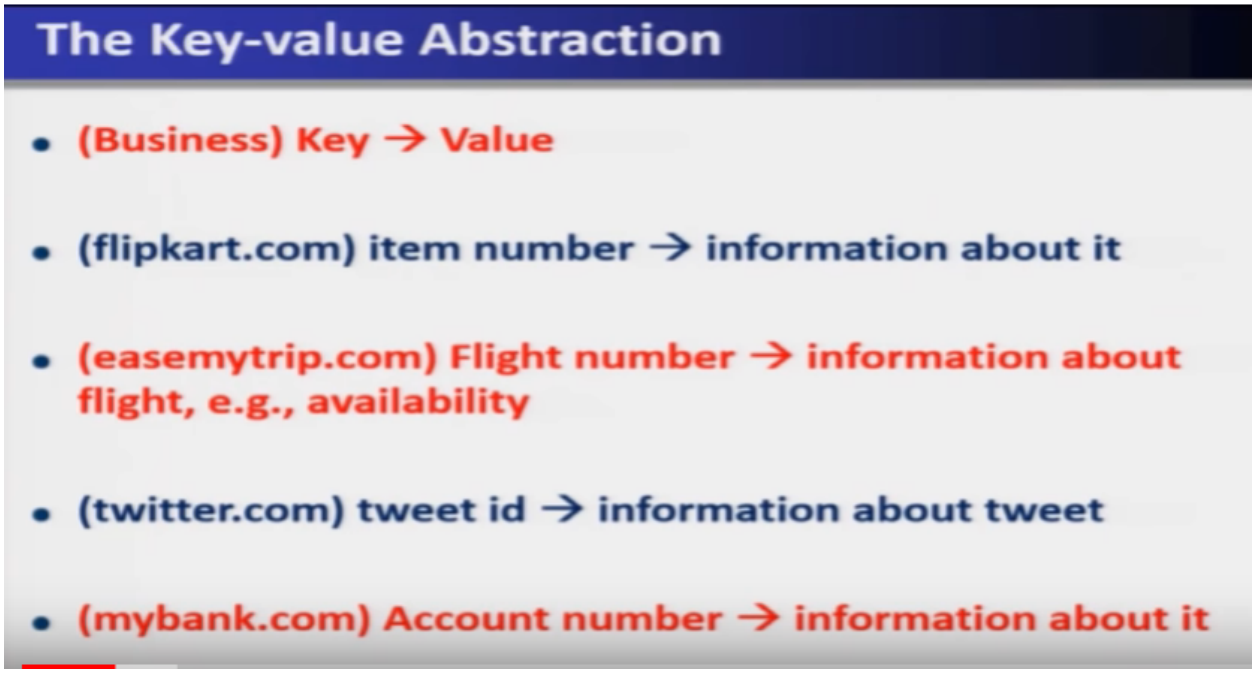
Refer slide time :(0:23)



Preface content of this lecture in this lecture we will discuss, the design and insight of, key value store, which is also known as no sequel stores, which is used for storage of the big data system. These key value

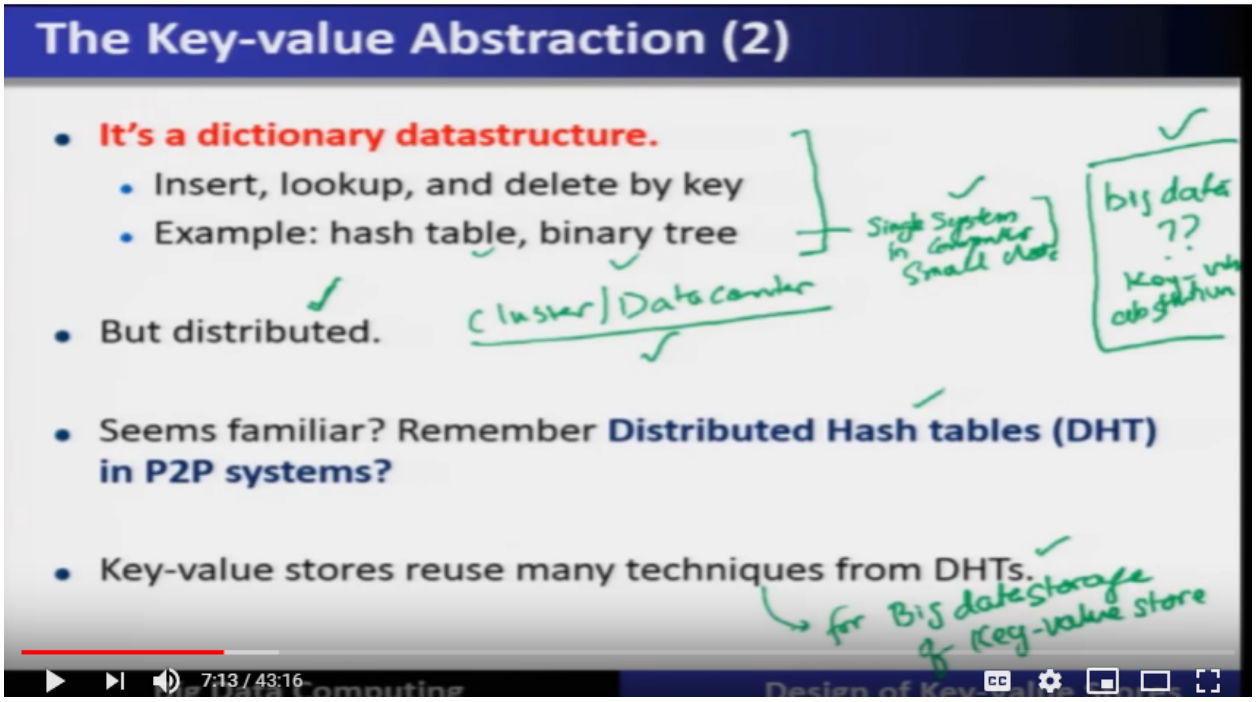
stores are also provided as, the storage system for the big data and also supported, as the cloud storage systems. We will also discuss, one popular, key value store which is called,' Apache Cassandra', which is one of the most important and widely used, no sequel storage, used for the big data storage and also I will able add the cloud storage systems, will also see, the different forms of consistency solutions which are provided by Apache Cassandra in this lecture.

Refer slide time :(01:20)



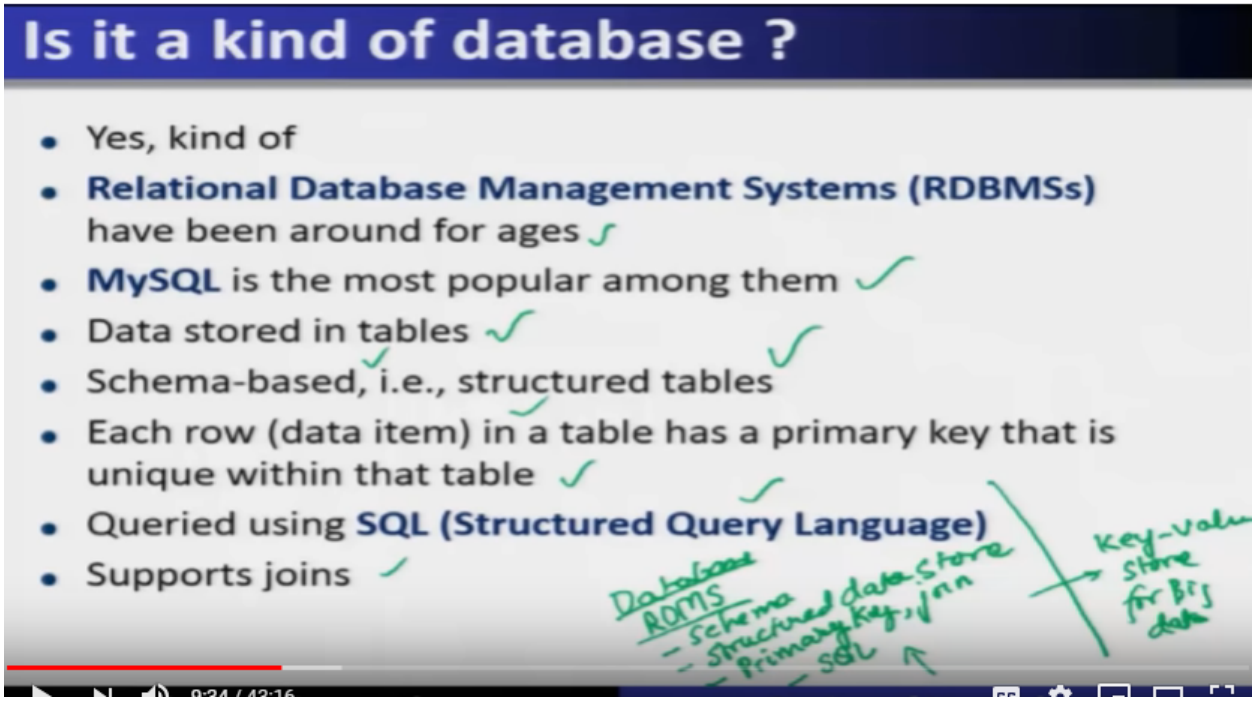
Let us understand the key value abstraction, key value for, any business and important, item or entity is called a,' Key', and all the attributes related to that becomes, the value for example in a flip. com company, they are the item which they, deals as the sales is called the,' Key', I and all the information about that item becomes, the value for example how much the quantity of that item what are the reviews of that item? And so, on and so forth similarly for another company is my trip. Com which deals with the flight reservation, there the key will become, the flight number and the information about the flight for example the seat availability and so on, will become the value. So, key value store is, defined in this manner, similarly in a Twitter.com the tweet ID becomes, the key and the information about the tweet that is the time, day and the text of the tweet becomes the value of that system. In bank .com in various bank operations, the key becomes the account number and the information pertaining to that account becomes the value. so, this way the key value abstraction, will provide the storage, of the data in different businesses and different companies, which is required to be stored, in a large amount, that we are going to see how we are going to handle this key value abstraction, through the databases which are available, in this domain of technology.

Refer slide time :(03:20)



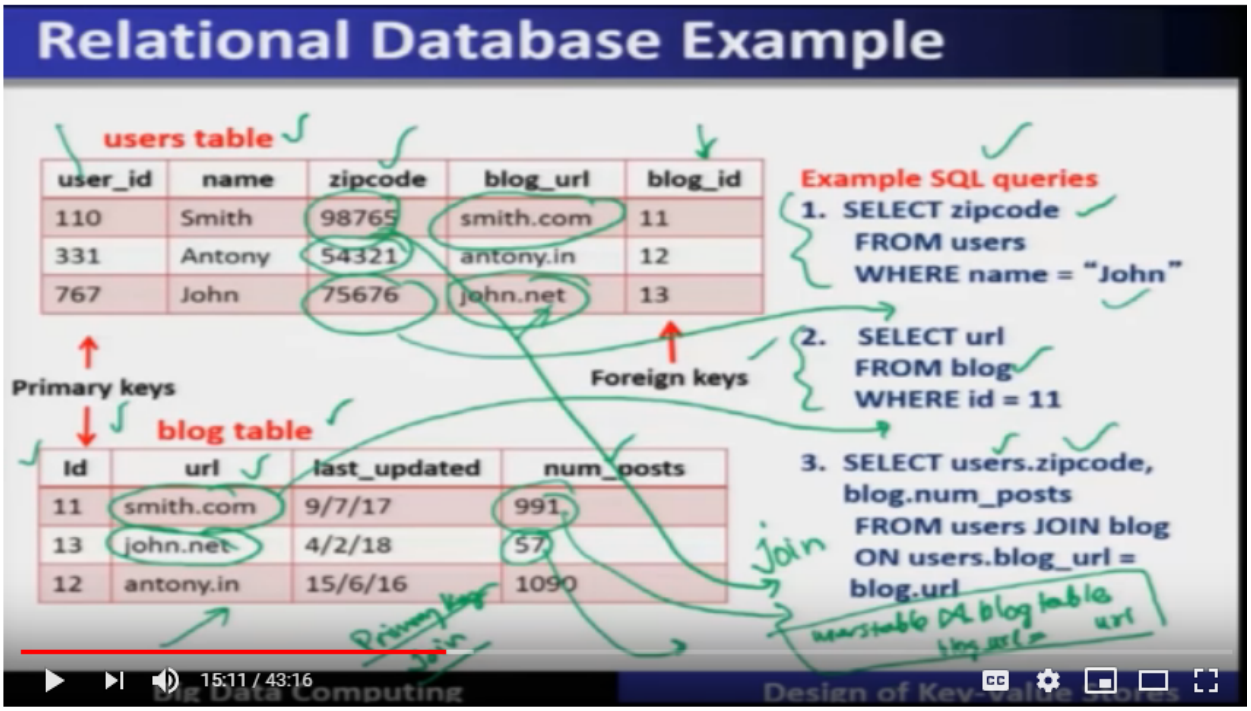
So, key value store, which we have just discussed, can be visualized as the dictionary data structure, for example, to have this abstraction key value ,abstraction for example the operations which require to be supported, in the Destiny data structure such as insert lookup and delete by key. And this kind of key abstraction can easily be provided by hash table binary tree and so, on these data structures. So, in the dictionary area data structures what we have seen that in the form of hash table and binary tree can implement these key value abstraction but the issue, there it is that these dictionary data structure, can be supported by a single system, in a computer, now the size of the data if it becomes very big, then this becomes a problem that means a single system cannot store the entire data hence, this kind of scenario which is available for providing the key value abstraction using dictionary data structure like, high stable and binary tree is good enough for a small amount of data. So, when it becomes a big data, this key value store how we are going to handle, the this key value store, abstraction, becomes the point of discussion in this lecture. The way to store the large amount of data, is not is to be handled in a distributed manner that is not one system but a cluster or a data center, will be used to store, the big data that is in the form of a key value store, hence this particular, concept of storing the key value or providing the key value abstraction for a big data is not done through the single system but it has to be achieved through a distributed computation system or we can understand it by providing this entire storage, in the form of the cluster or in the form of a data center. Now this particular concept seems familiar if we recall the use of distributed hash tables in a peer-to-peer system, now we are going to see that this key value store may reuse many techniques which we have known in the peer-to-peer systems in form of distributed hash table we will see how this entire technique, can be utilized to design this key value abstraction for the Big Data system.

Refer slide time :(07:17)



Is it a kind of database, yes it is a kind of relational database management system, that have been around for ages, the most important among them is my sequel, database system which stores the data in the form of tables, which are the schema based structure tables where in the each row that is the data item in the table hi as a primary key and that is unique within that particular table. And these queries are done using a language which is called, 'Structured Query Language' or a sequel which supports the join, therefore we see that the database which are also called,' Relational Database Management System', has the schema and the data has to be organized according to this schema hence it is called ,'Structured Data Store', and this particular data has this particular system of a database management system in our DBMS highest from primary key, I mean the concept of a joint operates over there and the queries or the data can be queried with the help of a language which is called,' SQL'. Now let us see, how we are going to use this concept for our key value store, for the Big Data. So, the question is can we use this concept of, the existing structured, data that is structure tables which are managed in the form of well-known relational database management system. So, can we use it or we cannot use it both these questions are to be answered in the further slides.

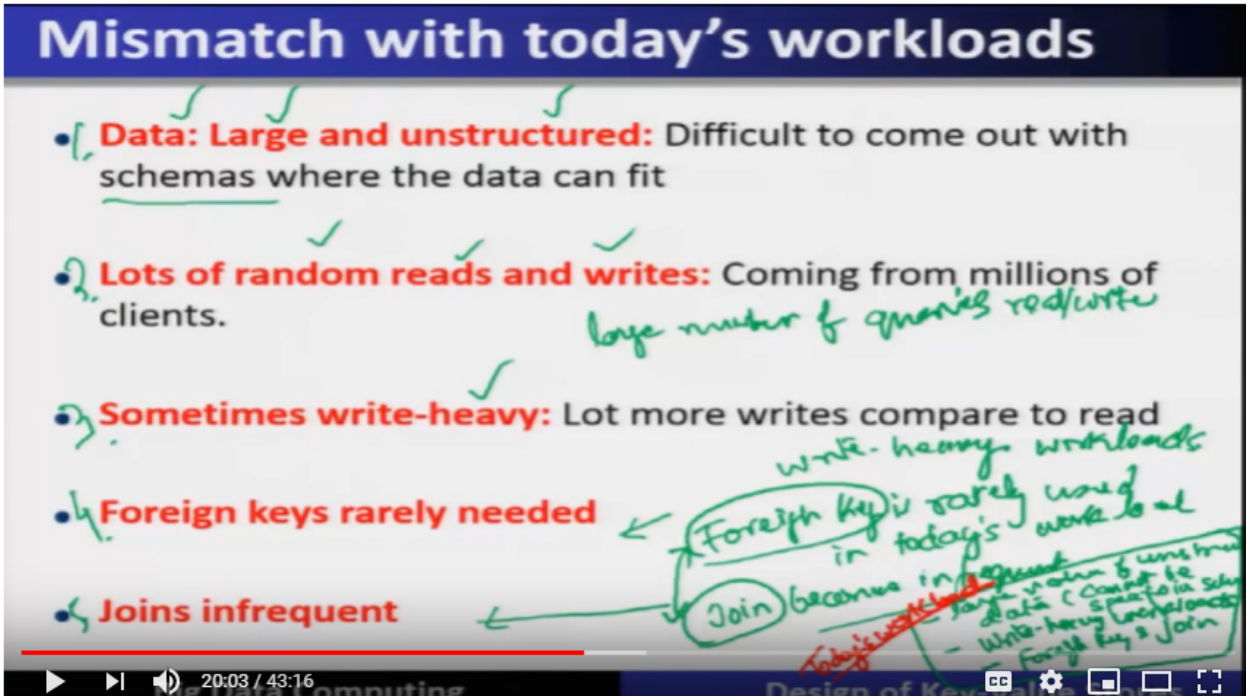
Refer slide time :(09:38)



Before that let us understand our DBMS the concept of storing the key value store or the, the data, in a structured format and performing the queries and the concept of primary key and a joint operation just we will review it and then we will go ahead. So, in a relational database management system data is organized, in the form of a schema and they are called as the,' Tables'. So, here in this picture, we have in this example we have shown the users table and a block table, where in the users table this is the primary key, users ID will become a primary key and all other attributes, name zip code block URL and block ID becomes other attributes in the schema, similarly the block table has ID as the primary key and all other, all other attributes like URL last updated and number post are also a part of the schema of the block table. Now as far as in users table if we go inside the detail of the attribute block ID, this we call it as foreign key, this block ID is referring to the IDs which are the primary key, of a block table hence this becomes, a prime foreign key and now this particular database once it is defined then we can perform the SQL queries, on top of it the SQL queries are such that if you want to find out, the zip code from the users, where the name, of the user is the John if you want to find out that zip code this particular query will fetch this particular value over here, similarly if you want to run another query on this particular database let us say select URL from this particular table block we're ID becomes 11. So, this particular URL smith.com will become the result of this query, now furthermore if we require to, to select the users zip code. So, users zip code and, and block number posts together from the users and the block file together and we're in the users block URL, is equal to the block URL on the other table that means this particular query can be satisfied when we join two different tables and that is the users, table when we join with the block table, on the key as the users on the on the key users block, block URL ,is equal to the to the block, to the URL of the other table. So, this requires the join operation, this particular requires the join operation to

perform this particular key. So, the outcome will be the when the users block URL and the block you and, and the URL in the block table if they are same for example the Smith both are same and let us see the when Smith, is the same then we want to find out the users zip code the user zip code is this value an and blocks number post nine, nine one. So, these values will be given as the output similarly in another case, when this John net and they are same then in that case the number of post is 57 and the, the zip code is sorry, zip code is five seven four seven five six seven. So, this way we have shown that once the relational database, is defined and data is stored in the form of the tables that is in the structure schema, the and every table defines a primary key, then we can perform the different queries that is SQL pissed on the primary key are based on the, the operation which are called, 'Join Operation'. So, therefore in our DBMS the primary key and the join operation plays a very important role, in that retrieval storage and retrieval of the data.

Refer slide time :(15:10)

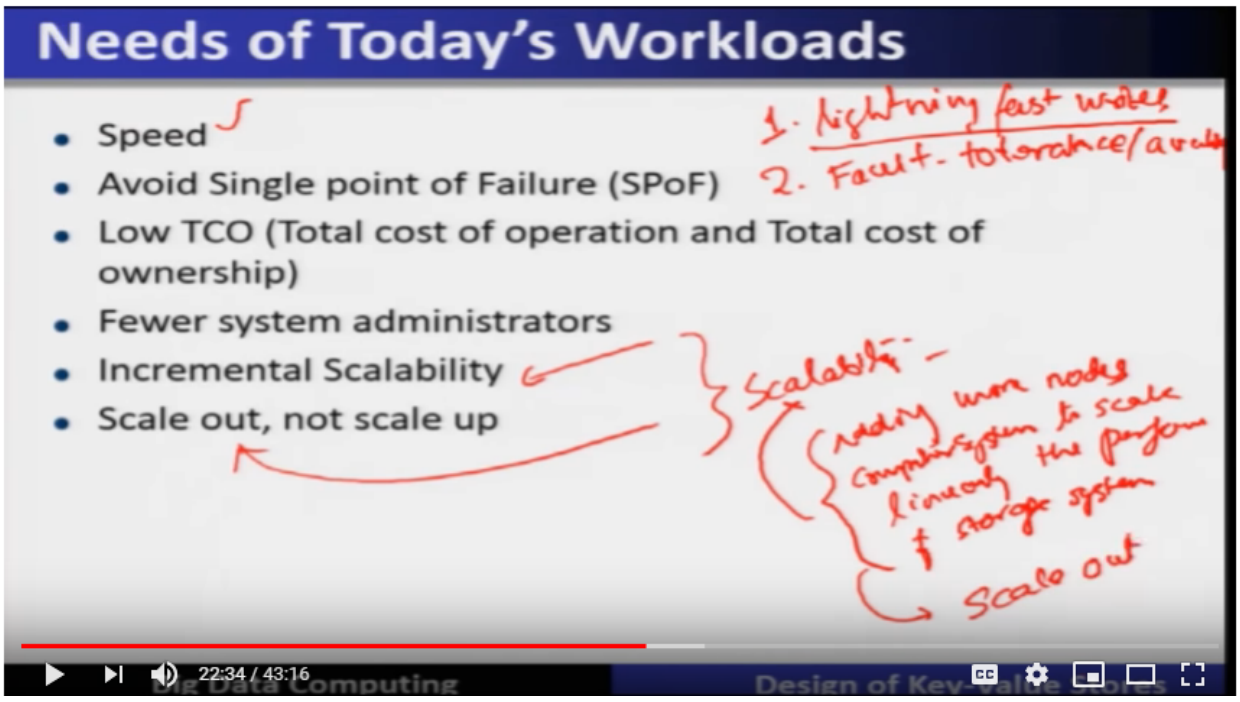


Now as far as in today's workload, when we see that, whether this kind of our DBMS ,that is the structure data can be further utilized or not now to answer this question let us understand the today's workload. So, there is a mismatch in today's workload with the existing our DBMS or a structural data system in the following manner, first is that the first mismatch is about the data. So, in today's workload the data is characterized by a very large, volume and also the data is unstructured, meaning to say that when it becomes unstructured, then it cannot fit in any of the schema, which is a predefined. And second thing is the data volume is too large it cannot fit in to several even tables which can be stored on a one computer system. Now another mismatch in today's workload, we can think of in the terms of read and writes, which are in large number of random, read and write operations coming from millions of clients. So, how

are you going to handle this large, number of queries that is the read and write operations, third thing is about today's workload, is that which differs from the previous our DBMS systems which were handling the workload, is that these workloads have the right heavy operations. So, most of the time, these workloads requires the right operations, with compared to the, the lesser number of read operation but read and write put together are much larger in the volume compared to the previous our DBMS s hence, this is known as the write heavy , workloads.

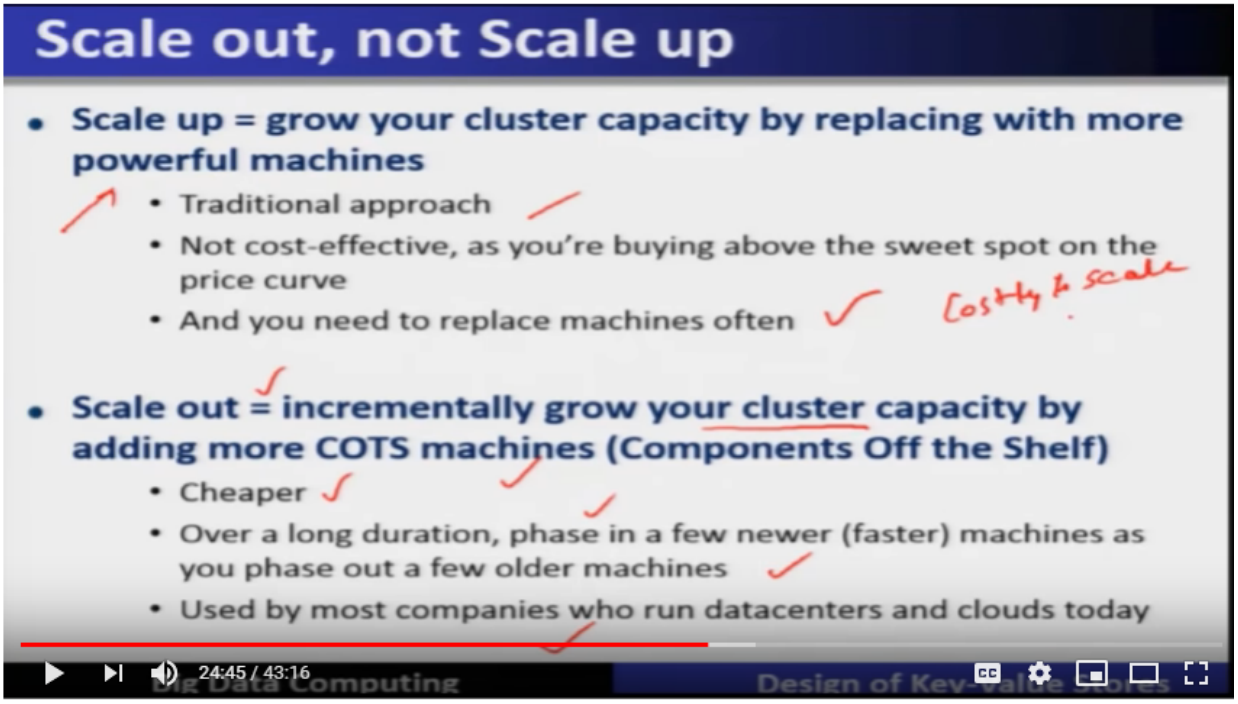
Four thing is that, in this particular workloads we are not going to handle we are not going to use this foreign key and this foreign key is rarely used. So, foreign key, is rarely used in today's, workload, also we see that this joint operation, also is becoming infrequent in today's workload, therefore the foreign key and the joint operation, is not very much used in today's workloads. So, therefore we are going to design or we are going to design a database management system, which is going to handle today's workload which has these following characteristic or the requirements to handle the large volume, of unstructured, data which cannot be, specified in schema. Second thing is that it is the right heavy, workloads light of right operations are too many numbers and finally the foreign key and joint operations are not very required in the today's workload. So, let us see with this particular, requirements how we are going to design a new database system, which is going to cater to these today's workloads and how our why and we are going to provide how we are going to provide, the key value store for the Big Data system.

Refer slide time :(20:07)



So, therefore in today's workload, what is needed is the speed? In which these write heavy workloads are to be catered. So, that means a lightning-fast, writes has to be supported, in today's workload. Second point is that we have to avoid, in today's workload the single point of failure, that means the data that means we are not going to be affected, by the failure of a node, in this kind of system, that is called,' Fault Tolerance'. And availability third important criteria is about low cost of operation and total cost of ownership and fewer system administrators are required to manage, this entire big data system and also it will be handling able to handle the incremental scalability, that means to support the scale out. So, let us understand, the scalability aspect scalability by scalability we mean that, as the data volume grows, we keep on adding more system and therefore the capacity, of that handling of storage of a big data system automatically increases, without that is called scalability, scalability means we can keep on adding, more nodes or a more computer systems, to scale linearly, the performance, of a storage system, this is called as the,' Scalability', this technique of adding the computer, without replacing with the new one is called a,' Scale Out'.

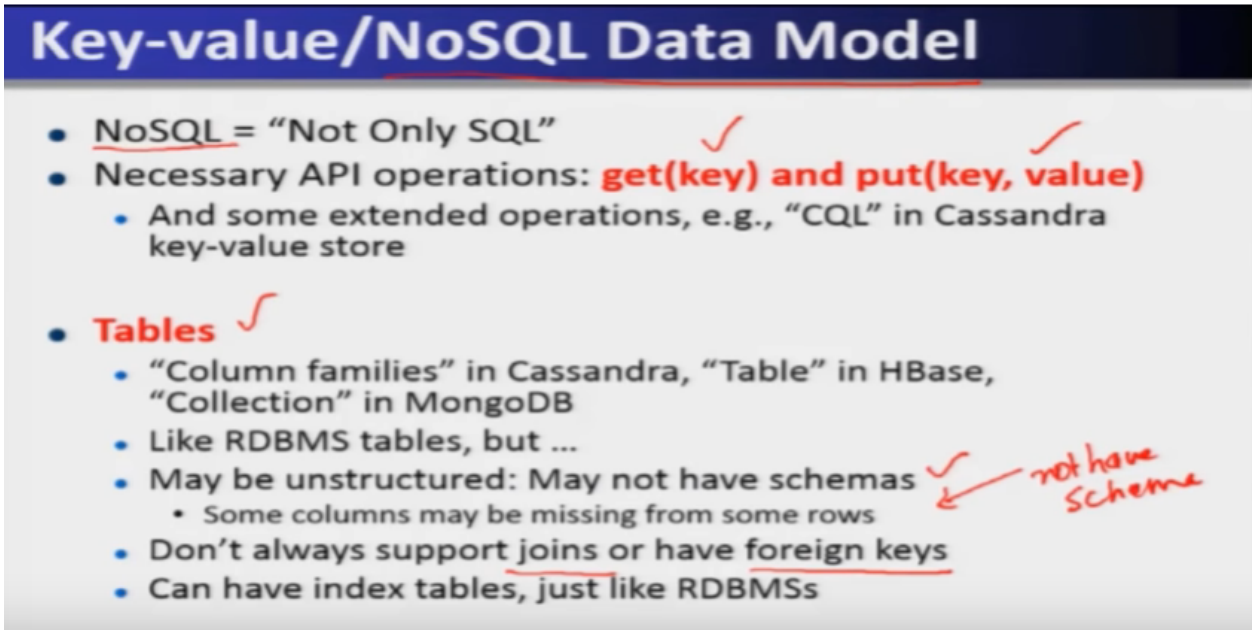
Refer slide time :(22:33)



Let us understand about, what you mean by a scale out? And which is supported, in the new today's workload system, to handle the big data system. Now is scale out means, that it will support to increment, incremental e grow, your cluster capacity by adding more, component of shelf systems, that means this

way of, of achieving the scalability becomes, cheaper why because we are not going to replace with the costlier system, we keep on adding more number of systems, that is called,' Scaling Out', and also over the long duration, we have to face in a new phase, in a new phase a few newer faster machines as you phase out a few older machines. So, that is called a,' Scale Out'. So, hence this becomes a cheaper way of scaling up the, the entire system. And that is how we are going to build the cluster systems, which is being supported by a scale out technology, this particular is killing out is supported by oil is being used by many companies, which runs the data centers and the cloud today, in contrast to scale out there is a scaling up scaling up, means the traditional computer system, we are going to replace with a high capacity powerful machines, by to increase the, the capacity of the system in terms of memory and the processing capabilities and so on. So, this is a very costly affair, by providing the scalability which is called,' Scaling', up by replacing with a very powerful machines, that means the old machines, need to be replaced with the newer machines this becomes a costly affair, in compared to the two the scale out.

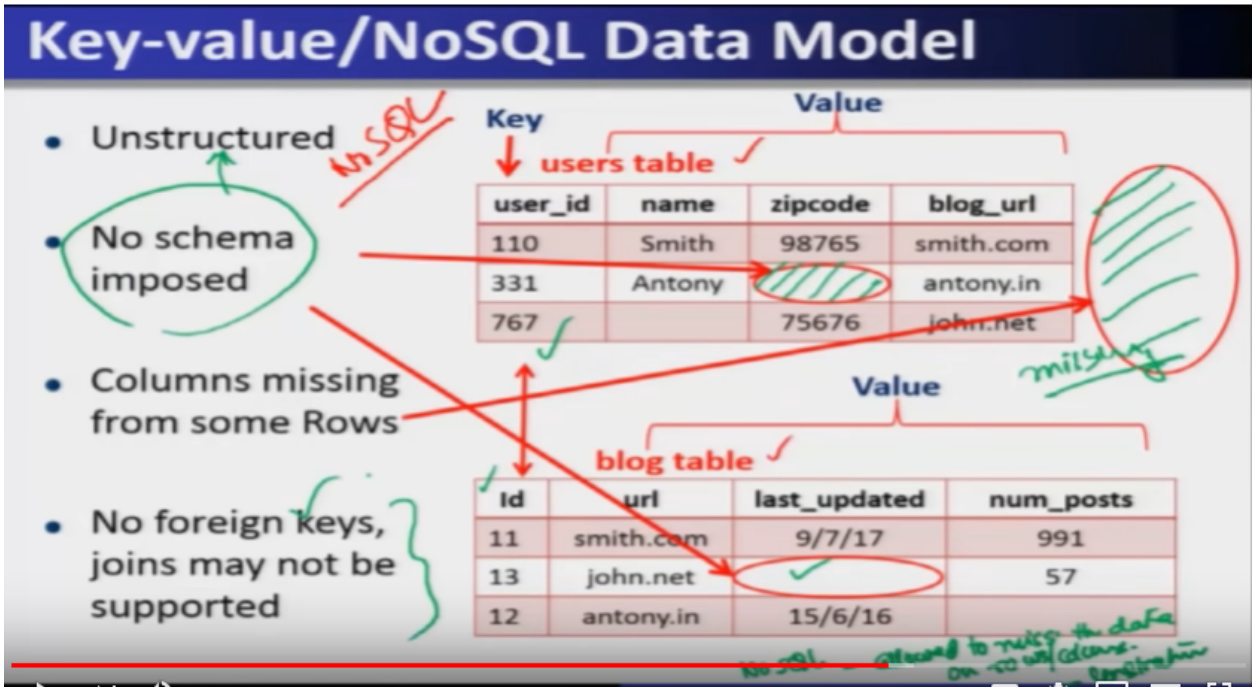
Refer slide time :(24:49)



Mechanism of scalability, now we are going to understand this concept of a key value store, of the know sequel data model. So, key value store normally is provided, in today's to handle, the today's workload in the form of no sequel data model, now let us see understand what do you mean by no sequel data model, no sequel stands for not only SQL. So, not only SQL means beyond SQL whatever is limitations, of SQL it will go beyond and it will provide the support, to the today's workload, that is why it is called,' no Sequel Data Model'. And which is used to store the large amount of key value store, abstractions, now this new sequel is to data model provides the necessary API operations and the two most important API, is

which are provided by no sequel system are get by key and put by key. So, put by key value, is basically, writing the key value pair and get means we want to read the key value pair. So, given a key we want to read that key value pair. So, these operations are provided as in, in the different no sequel database systems which are available as of today. Now and some extended operations are being provided, in the form of for example are also provided for example the CQL that is called, 'Cassandra Coil Language', is being provided by the Cassandra database system, Cassandra provides a no sequel data model and this is we are going to understand here in this lecture, about this particular data model that is called,'Cassandra', which supports no sequel for providing the key value store for big data systems. So, in the no sequel system, the tables are often called the,' Column Families', for example in case and right is the tables are called,' Column Families', and in HBase, it is called ,'Table,' and in MongoDB it is called collection. Like our DBMS tables and here, it is called, 'Column Families'. These particular tables may be unstructured, that means they do not have any fixed is EEMA and some column may be missing and some rows may also be missing hence, they are not called a, 'Structured Data'. So, there is no schema which can fit this data hence it is called, 'Unstructured Data'. So and also it doesn't support, the join and the foreign keys and also can have, the index tables like RDBMS. So, these are the some of the features of the key value store which is provided in the form of no sequel.

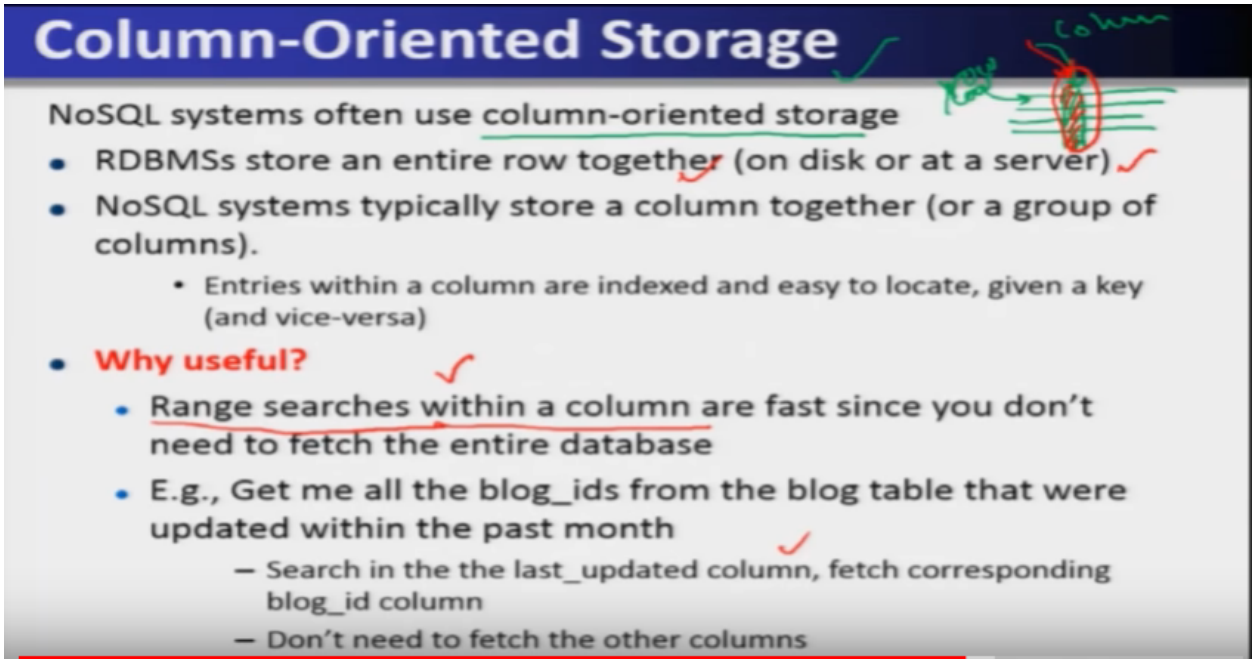
Refer slide time :(28: 13)



So, let us understand these, concept which is called, 'Unstructured'. Where no schema is there and what do you mean by this? And how no foreign keys and joint operations are supported. Let us take the same example of the two tables, the user table and the block table which captures the data in, in this today's

workload system or in a no sequel system. In today's no sequel system, let us understand by this example and here, we can see that this particular table users table which is called a, Column Family'. And sometimes table in HBase, here we can see that in some of the attributes and even some of the columns are missing, if they are missing that means sometimes it is called a, 'Null Value' or there is no schema which is imposed if the entire column is missing. So, therefore this particular model, data model is called, 'Unstructured Data'. Similarly an entire column can be missing and from some rows and also, you can see the same thing here, these particular entities are missing, some of the entries are missing from the table which is not possible to be there in RDBMS but, in the no sequel this is allowed and this is not a problem, hence it is called, 'Unstructured Data'. Why because there is no ski, bar which can fit this, this type of data. Which is coming in today's workloads another thing is they're. So, primary keys are there, but there is no concept of foreign key, foreign keys are not required and also there is no joints which are supported in no sequel system.

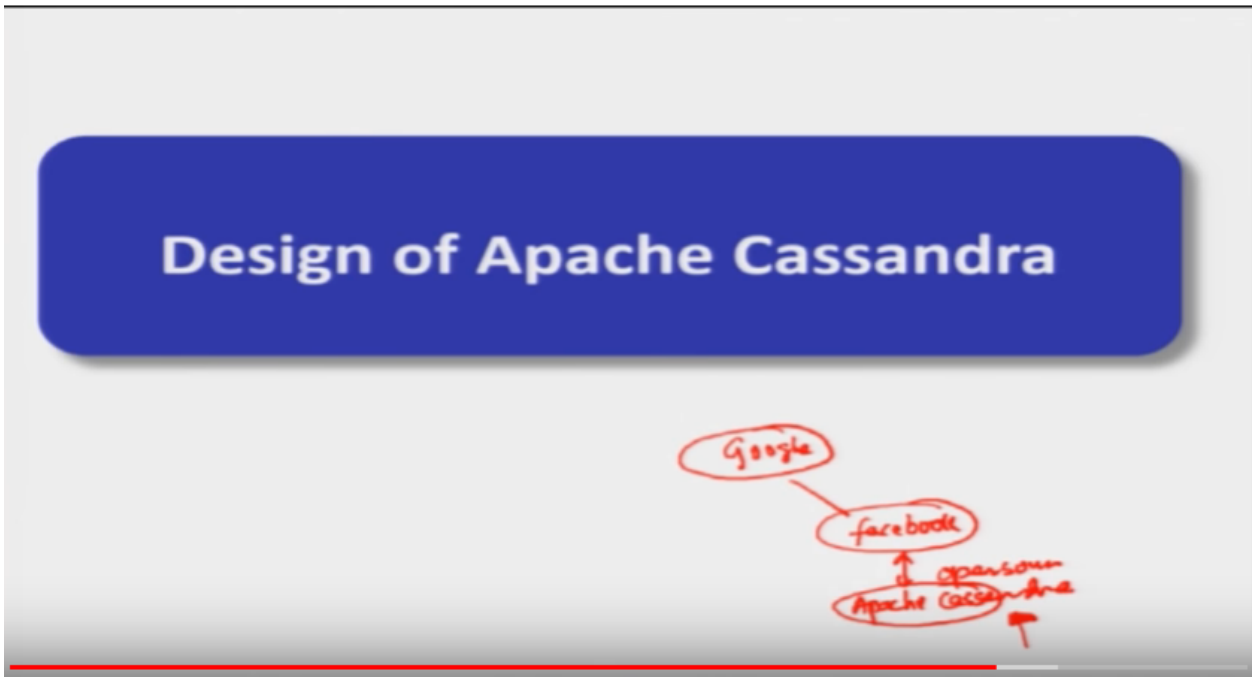
Refer slide time :(30: 48)



Now, this storage system is called column-oriented storage system. Now, we have to understand this concept in a better in more details. So, no sequel systems often use column-oriented storage. What do you mean by column oriented stories? That means the sequel system uses to store are used to process the row wise and now, in no sequel system if it is, storing the entire columns and column wise operations if it is, being performing then it is called, 'Column-Oriented Storage'. So, no sequel system often uses the column oriented storage. So, RDBMS store, the entire row together, on the disk or at the server whereas the no sequel system typically store a column together or a group of columns. Now, entries within the columns are indexed and easy to locate a given a key and vice versa. So, that means we are here, in the

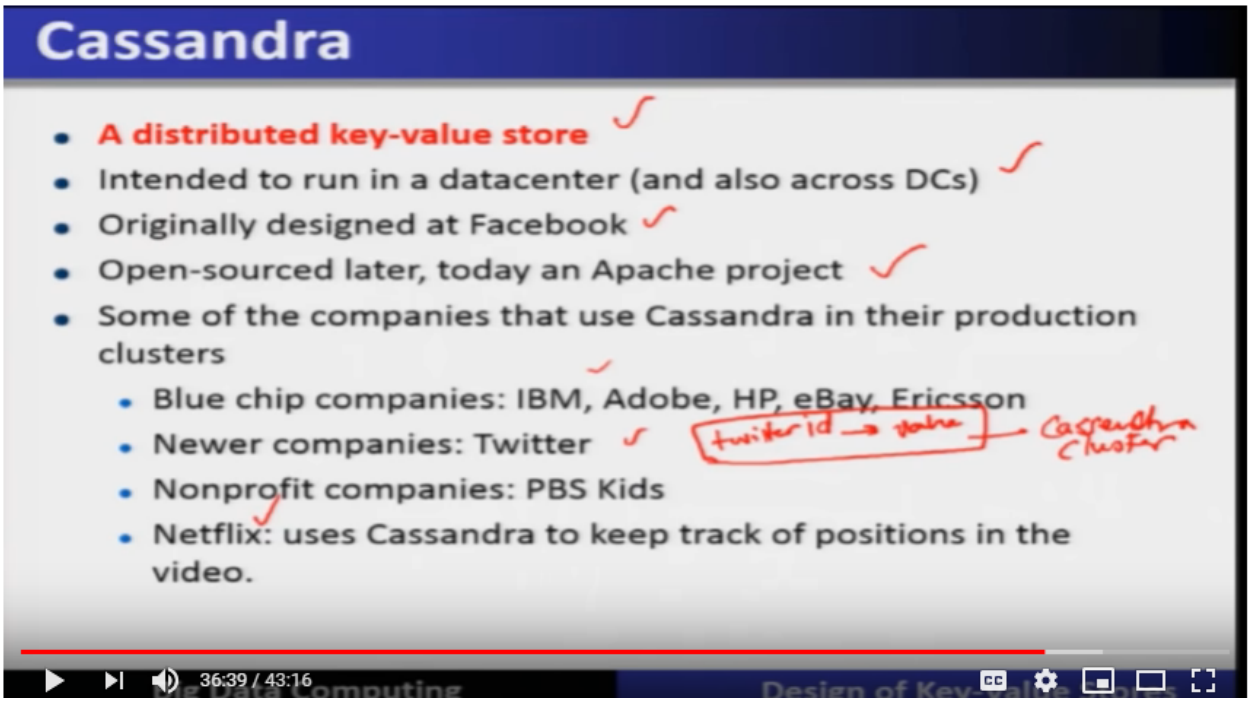
column oriented storage we are handling the columns. That is why it is called a, ‘Column Oriented Storage’. In contrast to the RDBMS which uses, the rows together and they are stored together in these forms. Why this is all useful concept is? Because, the range searches within the column or fast since you don't need fetch the entire database. So, that means if your query is targeted to be answered from that column itself, all the entries of the column then the column is required to be pressed and this and it can without fetching the entire database. So, hence the range queries can be easily supported, within the columns. We will see this kind of operations in further detail how, that is all implemented in no sequel systems. For example get me all the blog IDs, from the block table that were updated within the last month if that is, the query then we have to search in the last updated column fetch the corresponding block ID column and we don't have to fetch all other columns in this case. So, this kind of query which is very common in today's workloads and then that is why the today's no sequel databases are column oriented storage.

Refer slide time :(33: 37)



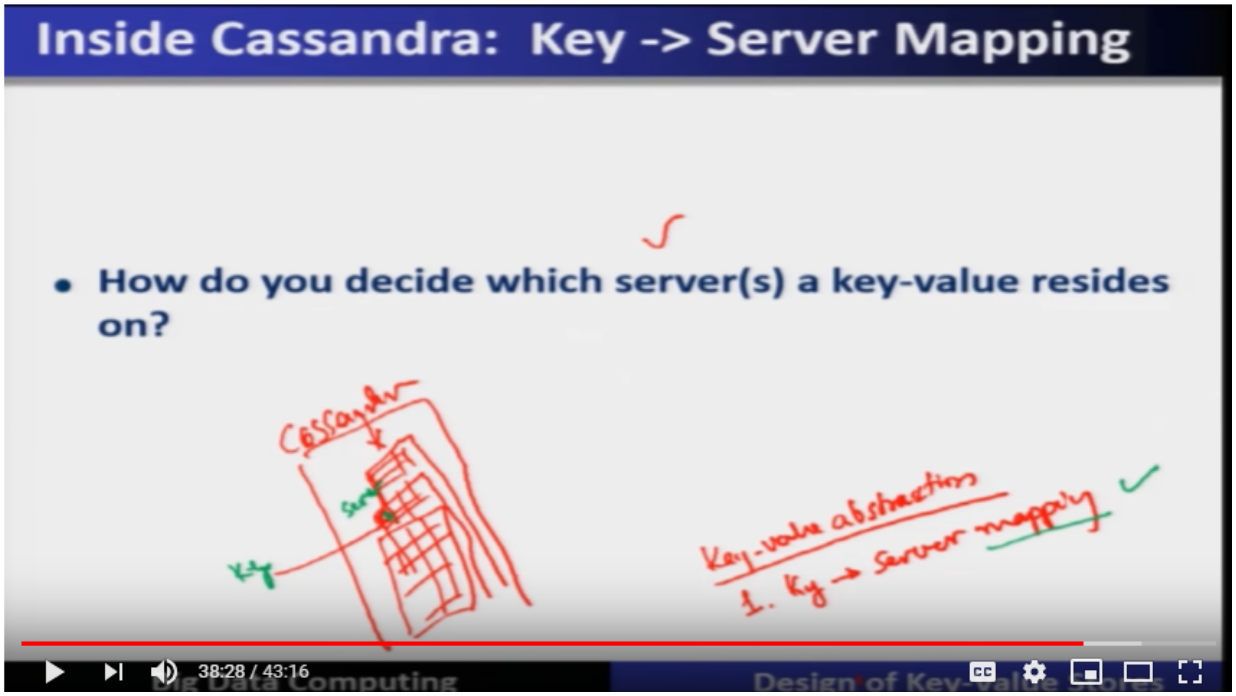
Now, let us go and discuss the design of Apache Cassandra, Apache Cassandra before we go ahead let us see there are two, there are two companies, one is called, ‘Google’. And this Google has inspired this no sequel system, which is now taken by the Face book and then Face book, has developed this Cassandra and made the open source and hence this is called,’ Apache Cassandra’. So, we will see the development and we will see why it is so, important in today’s scenario and what are the important things about Cassandra, we are going to see in the design, of Apache Cassandra.

Refer slide time :(34: 34)



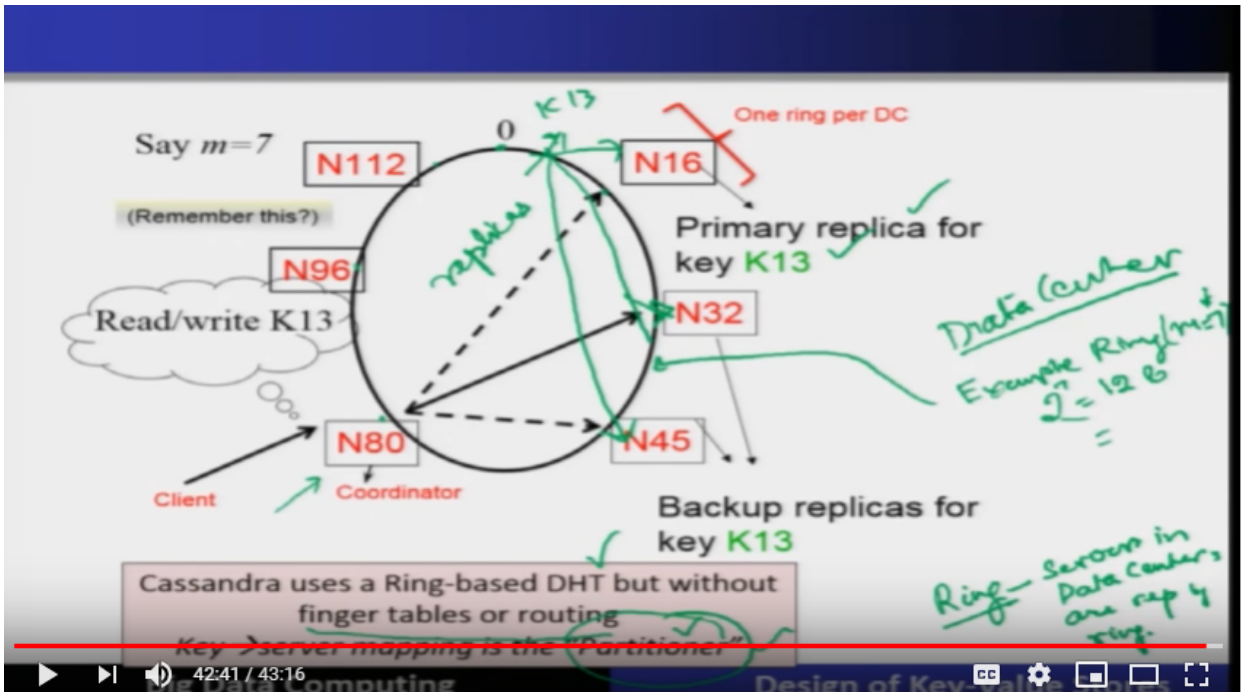
So, Apache Cassandra is a distributed, database management system or it is also called as a,' Distributed Key-Value Store'. Now this particular database management system, it is intended to run in the data center or across the data centers, it is not meant for to run on a single node, but it is meant to run on a data center, originally it was designed at the Face book and it was open sourced later, by the Apache project, some of the companies that use Cassandra in their production, clusters are blue chip companies like IBM Adobe HP eBay Ericsson newer companies like Twitter also uses, the Cassandra for restoring their tweets and nonprofit companies like PBS, KIDS also uses it and Netflix, also uses Cassandra to keep track of the positions, in the video while you watch the movies on the Netflix. So, Cassandra is so important that most of the companies, in the reduction cluster, they use this Cassandra for storage system, of the large key value store, as per their requirements, for example we have already seen that the twitter ID the twitter company, uses the twitter ID, this becomes the key and the value becomes, the information about that tweet, this particular key value is stored in by the twitter in their Cassandra, production cluster. We are going to see now the more detailed design of the Cassandra, how it supports the storage of the large volume of key-value store that is how it is going to support the today's workload, by providing the abstraction key value, to the different companies.

Refer slide time :(36: 44)



Let us see, the inside of Cassandra to see the inside of the Cassandra we will divide, the task which is designed inside the Cassandra, the first thing is the key the mapping between key to server mapping. So, the Cassandra supports, the key value abstraction, to support this key value abstraction, this key, to the server mapping, is the first most important task of the Cassandra design. So, that means now you know that this particular database is supported, on the clusters, this is called,' Clusters', which has several nodes and this runs, the Cassandra. Now here for a particular key, which server really stores, that key with server stores this particular key is going to be very important and how we are going to do this mapping, we are going to see so how do you decide which server or a set of servers, basically stores that key value pair on it. So, to decide this

Refer slide time :(38: 32)



We are going to see the inside of Cassandra. So, Cassandra uses a ring based, distributed hash table, for doing this kind of mapping and key to the server mapping, is done by the concept which is called a, 'Partitioner'. We call it as partitioner so; let us understand what do you mean by the ring? So, the so, the nodes are the servers within the cluster or in a data center they are represented in a form of a ring so, for example here in this, in this case we are making a ring,of size M is equal to seven, when M is equal to 7 then two days power 7 that is 128, different points will be there on the ring and on these points, we can place different servers. So, there will be so, this will be called as a,' Ring', of all servers which are there in a data center. So, data center is represented as a form of a ring and if, if the number of servers are more than 128 then, M value will, will increase and so on. So, the ring depends upon how many servers are there and it will be organized in the form of a ring. So, so whenever there is a key, required to be stored on these particular servers which are organized in a form of way of a logical ring, then one of these particular server, will become a coordinator, to handle this key to the server mapping, with the help of a program which is called the partitioner.

So, that particular client even the help of partitioner it will decide, where this key is going to be stored or mapped, on which servers. So, it will follow for example key 13 it will follow, the servers following this 1k 13 lies over here. So, the successor of K 13, will store this particular case for example the successor of K 13 will become N 16 it will store this key and another copy, of this key is stored on n 32 and another copy will be stored on n 45. So, the copies are called, 'Replicas'. So, it's not called as a,' Primary Replica or a secondary replicas'. But they are called as a replica all the copies are same and that can be served this is called a', Partitioner'. So, this kind of mapping from, key to the server is called a partitioner, in these terms and this uses the concept of the, the ring based technology which is we have seen in distributed hash table in a peer-to-peer system. So, when's this peer-to-peer systems, that means all the nodes are same it's a pure a distributed system without having any client-server architecture in it. This ring which Cassandra uses, it's differs from DST in one form that it is only using the ring, of DST and it is not using

the other concept of, distributed hash table such as finger table it is not using a routing is not using. So, Cassandra uses the ring without any routing within it.