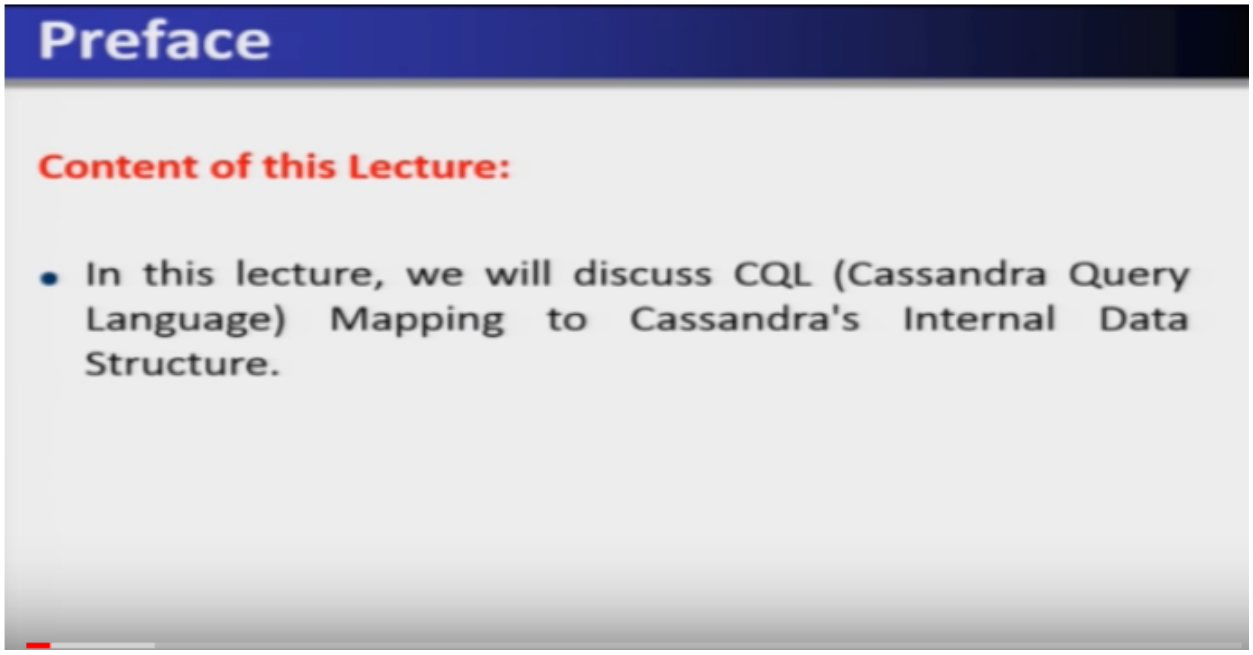
Lecture 17
CQL (Cassandra Query Language)

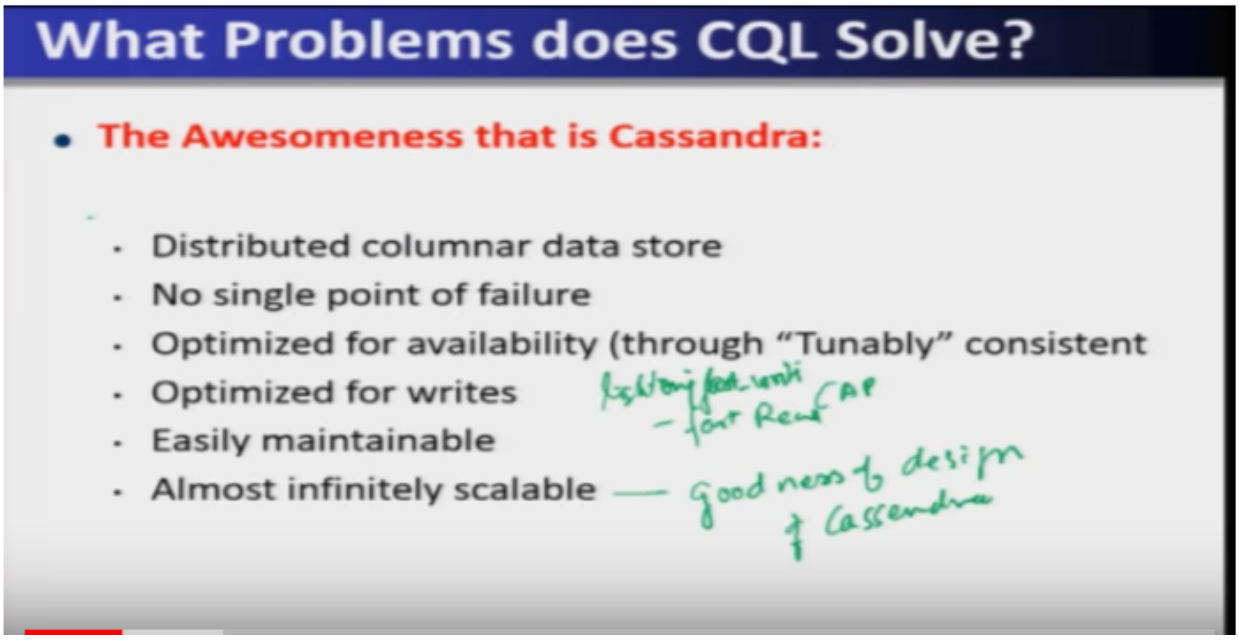
CQL Cassandra query language.

Refer slide time :(0:16)



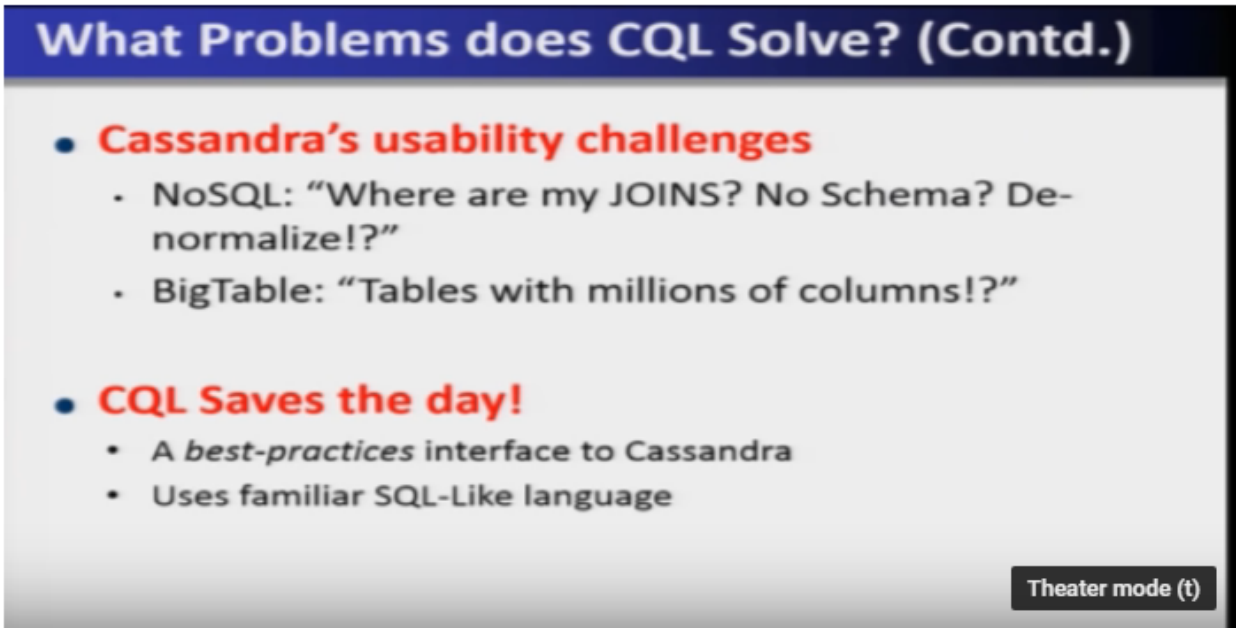
Preface content of this lecture; this lecture we will discuss, CQL that is Cassandra query language, its mapping to Cassandra's internal data structure, all these internal details, we are going to see in this part of the lecture.

Refer slide time :(0:34)



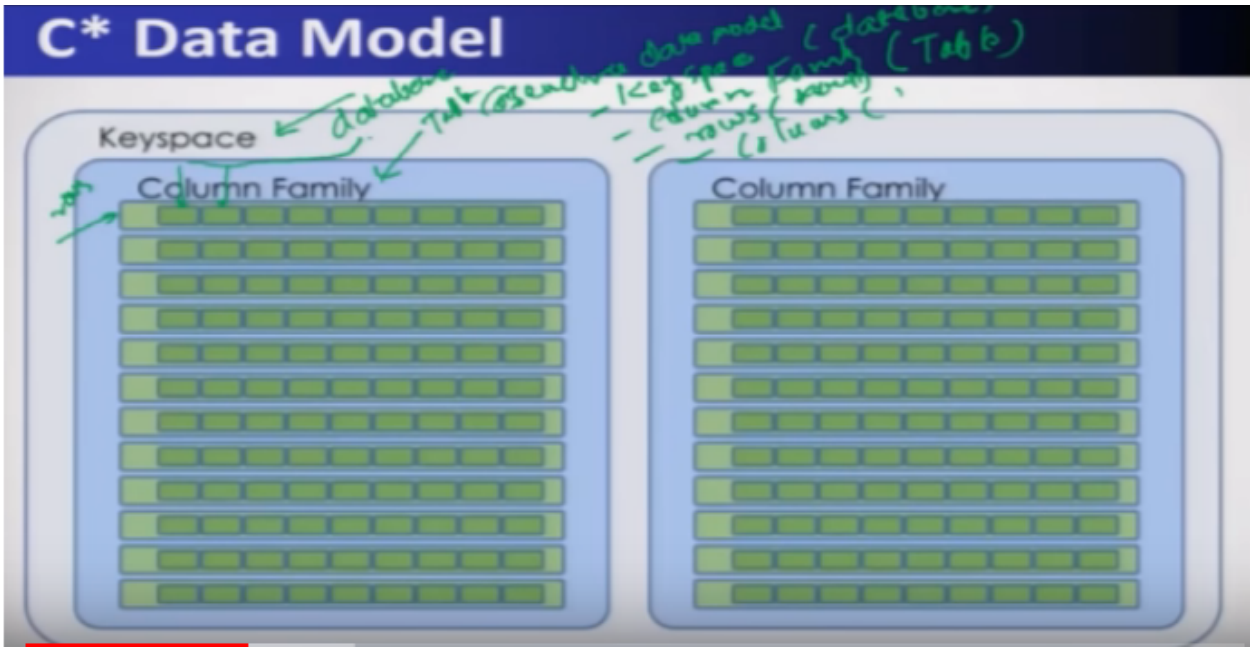
What problem does, the CQL is going to solve, to understand this let us first see, the awesomeness that is the Cassandra, Cassandra is a distributed columnar data store, that we have seen in the previous discussion, here in Cassandra there is no single point of failure, that is it is handled, it is handling the fault tolerance effectively, Cassandra is optimized for availability, although it is tunable, it is having a tunable consistency also, as far as the cap theorem is concerned, Cassandra is optimized, for the writes, here the writes are lightning, fast writes in the Cassandra, all the reads are also very fast. And this Cassandra is easy to maintain and also this Cassandra, is infinitely scalable, that is this is the good part, of the design of Cassandra. So, all these are adding the awesomeness, to the customer.

Refer slide time :(02:07)



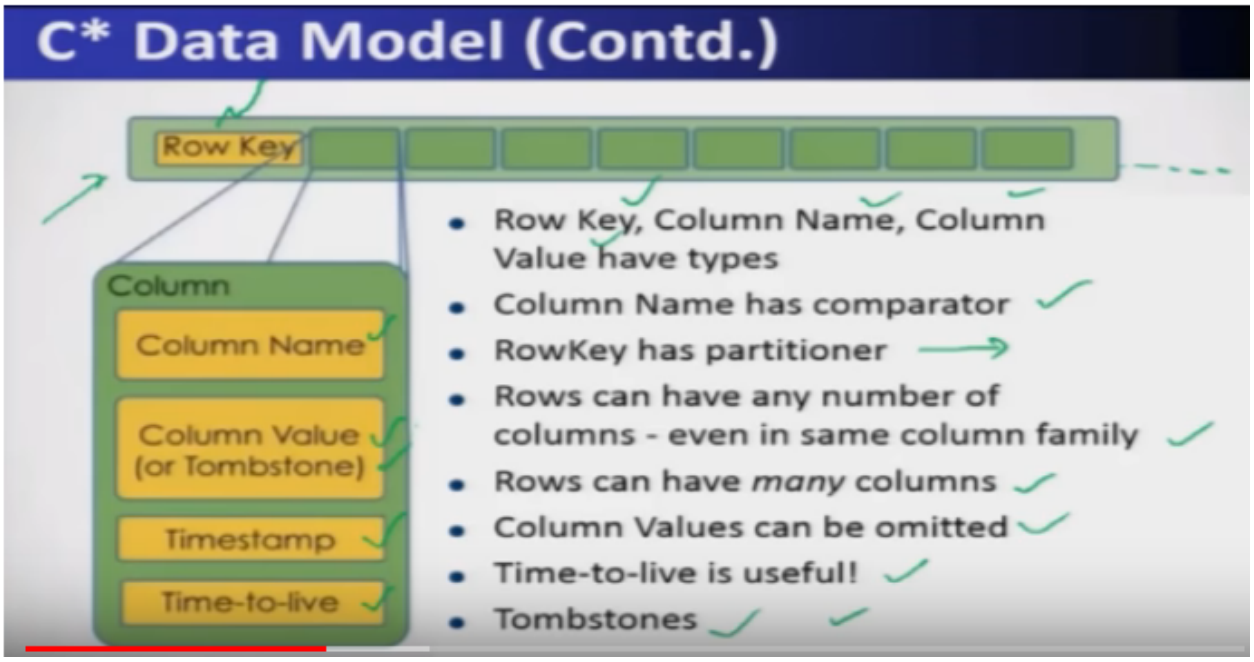
Awesomeness to the Cassandra, what problem does CQL solve? We are going to see, further more detail that Cassandra's, usability challenges is that again, is of no CQL that is there are no joints and also no schema and there is a big table, which contains all the, columns hence there is no demon de normalization. So, big table and here, the tables with the millions of column, sexist in this Cassandra's, now the CQL saves the day that is the back best practices, which are interfaced to the Cassandra. And also Cassandra uses SQ Like languages; given all these let us see.

Refer slide time :(02:52)



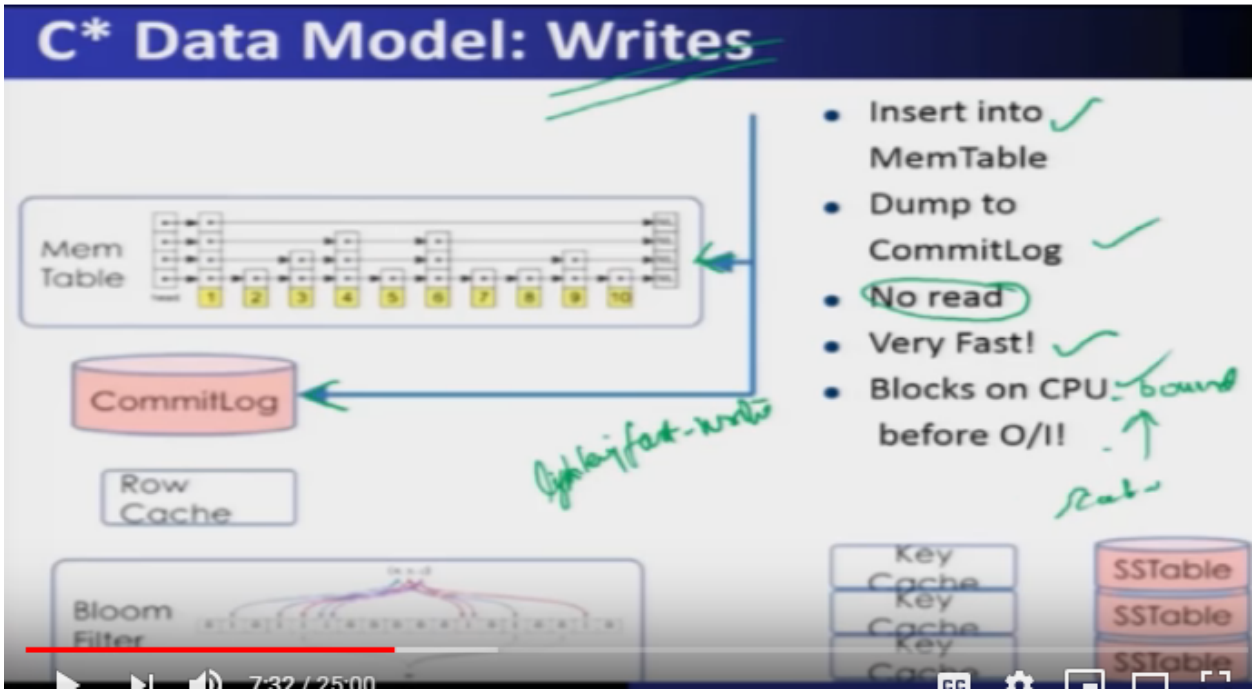
The more detail of Cassandra data model. So, Cassandra data model consists of a key space key spaces the entire database, of R DBMS system and the column family is nothing, but the tables of database management system. So, so the Cassandra data model, consists of consists of key space and then it is divided, into the column families. So, column family is nothing but a table, in a database and this is the database, then column families will have, different rows. So, this is the rows of a table and you will have different columns. So, that different the rows have multiple columns. So, this particular aspect which we have now seen, this key space refers to the, database of R DBMS, with respect to R DBMS and column family is a reference to the table, in R DBMS and these are all called,' Rows'. And every row is having multiple columns.

Refer slide time :(04:41)



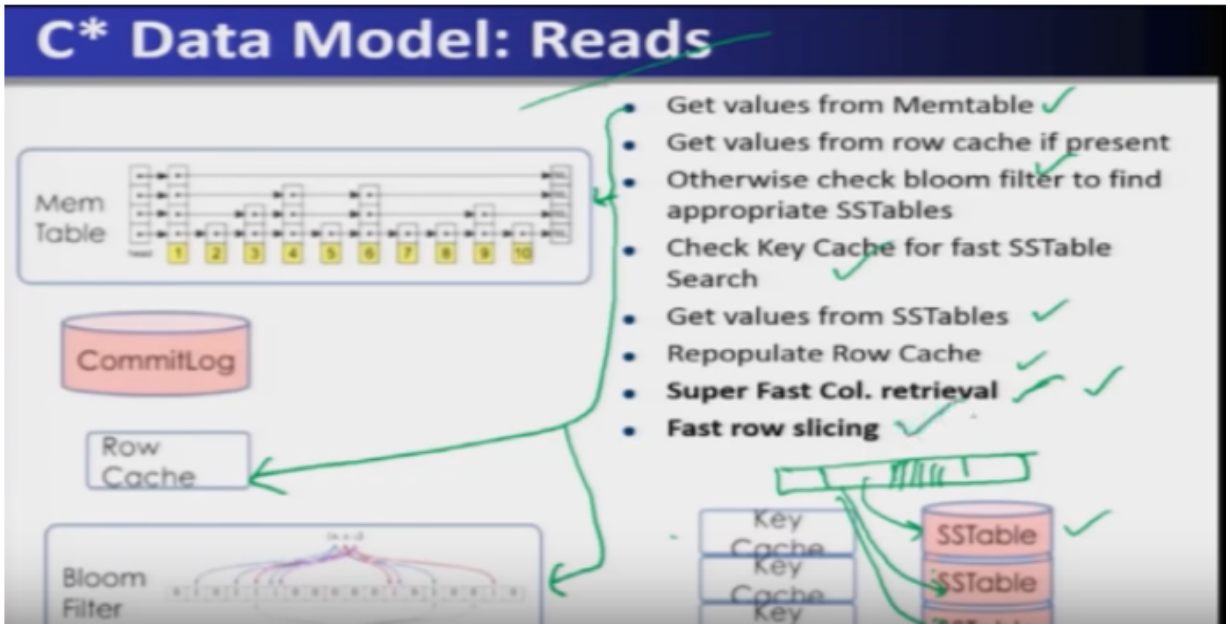
So, this comprises of the data model, let us consider the row in word row of, the column family. So, here the row contains the row key and further the every column contains, the name of the column, then column values or a tombstone, then times, time will tend to live. So, here the row key, column name, column values, how different types? And column name has a comparator and this particular comparator is, defined in Cassandra row key has, the partitioner. So, that means these row key will be, used to store, the rows on to the cluster or on different nodes and rows can have any number of columns, even in the same column families. So, so that means, this row can have any number of columns, it can be a millions of columns, also can be there in a row and rows can have many columns, column values can be, omitted here. And there will be the two more functions, one is called time to live, that is automatically after, this time to leave the data will be deleted. And there will be a tombstone that is if the, the data is deleted that particular flag will be set, it will not be immediately deleted by the system.

Refer slide time :(06:14)



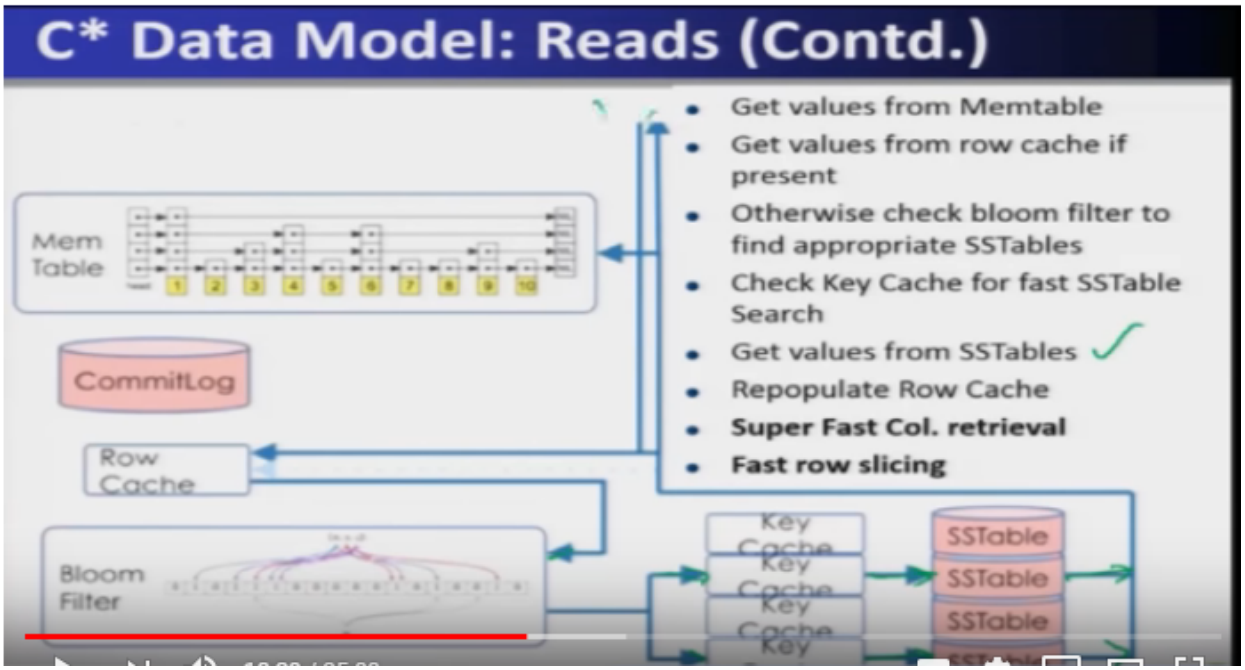
Now let us see the data model, of Cassandra and we will see how the right operation is being performed. So, right operation virtually has to do nothing, in the Cassandra data model. So, here we can see that if there is a request to, write. So, it will be first go and write into the Memtable and then it will write into the commit log. So, commit log, will ensure the, reliability against the failures. So, that is why using this but there is no lead is required and hence this particular right operation, is very fast this is called,' Lightining', first read/write operation is supported in the Cassandra, here we can see, also that these operations does not require any IO and all the operation, has to be done is an IO in CPU, bound operations is performed in the right operation, hence this requires, hence this is quite scalable. So, as we add more number of CPUs this will automatically, get scaled in linearly.

Refer slide time :(07:37)



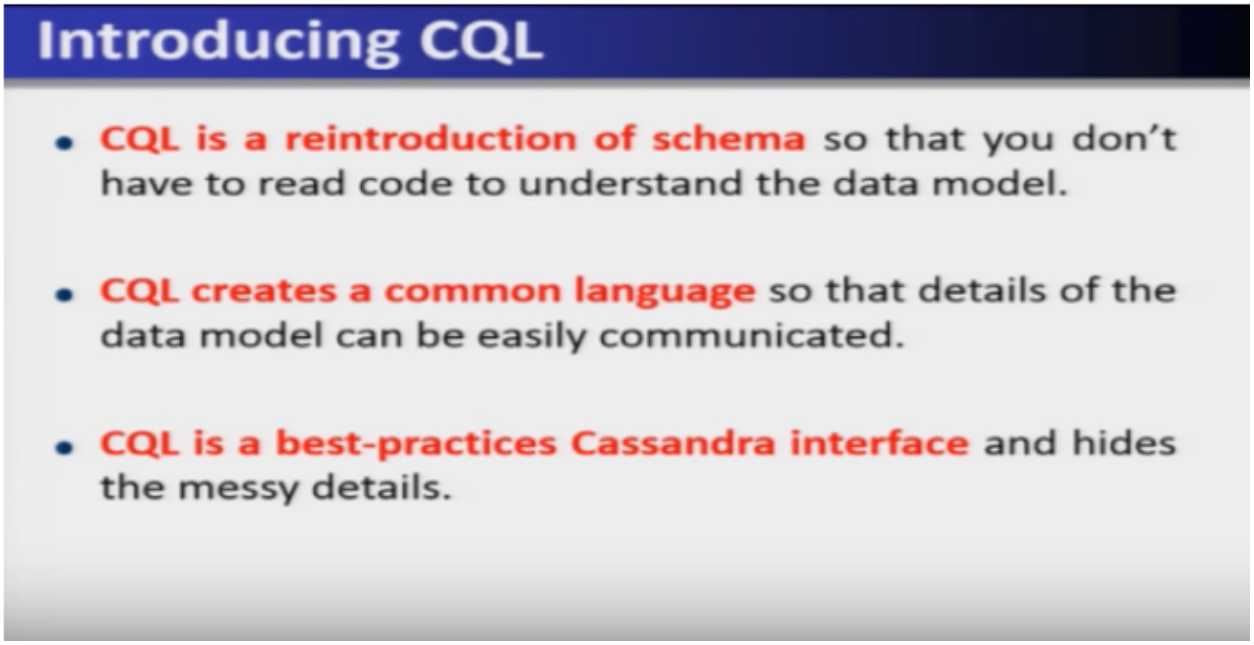
As well as the, the read is concerned, reads are supported, with several set of steps, in Cassandra. So, let us see how the read is supported, reads are not as fast as, the writes but still it is faster. So, it when the read operation, is invoked first it will try to get the values from the Memtable, it will first look up the Memtable and if the values are present, then it will be then operation, read operation, is done, if it is not there then get the values from the row cache. And if it is present, then it will be done otherwise, if it is not in this memory or in a row cache, then it has to go and search the disk, for these values, for that it goes and checks, the bloom filter, bloom filter is also a very fast, way of indexing or basically searching whether the data is, resides in which SS table. So, checking the bloom filter is to find the appropriate as a SS table. And so, it will, now check the using bloom filter after that it will go and check the key cache, for the first Ss table search and get the values from, after getting get, get the values from SS table. And then repopulate the row cache, this is super fast, column retrieval and why because, you know that a row, maybe split into different, SS tables because the slices, of the rows of slices of the columns are is stored, indifferent SS table. So, multiple SS tables need to be consulted, therefore sometimes if, if a particular row contains, that column value which is required by the read operations and it will be extremely fast read, otherwise it has to access more than one as a SS table and all the row entries, have to be now assessed. So, this way it will be possible, for the first row.

Refer slide time :(09:52)



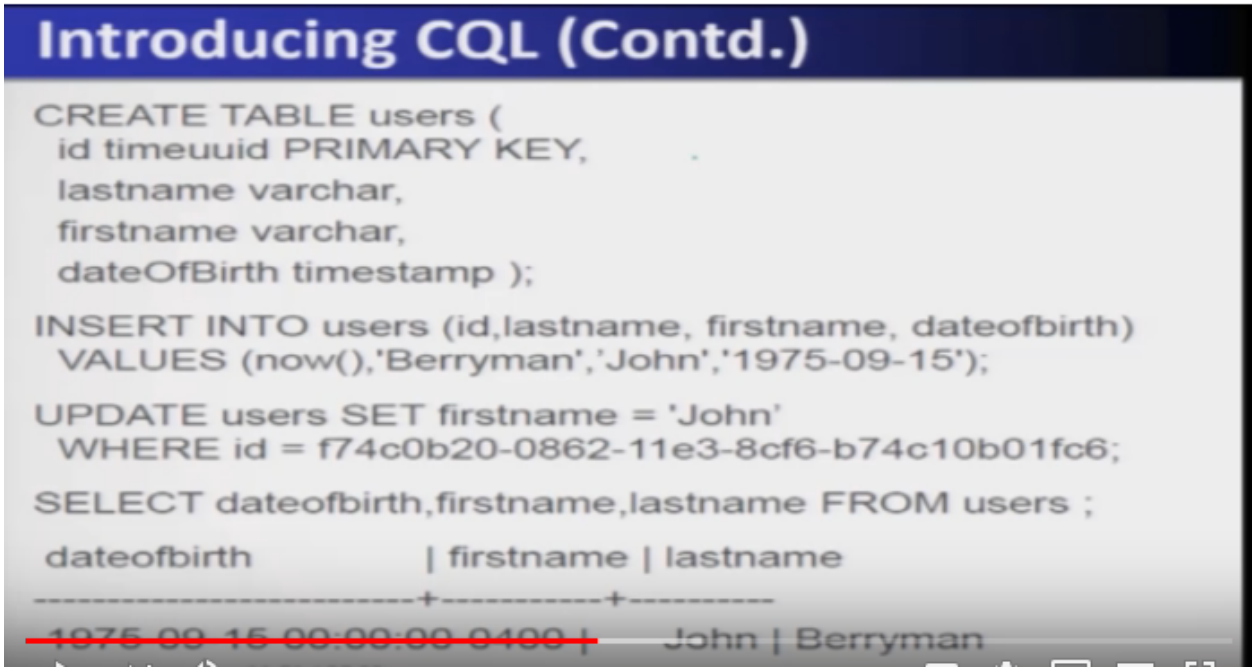
So, all these scenarios interactions, which we have shown here, for the read operation so, first it will go and check, the Memtable and row keys if it is not there, in the row key then it will be checking in into the bloom filter and based on that it will go and check, the in the key cache. So, key cash in turn will index into the, SS table and SS table in turn will fetch these values which are required, use the read operation, get the values from the SS table and give it back to the read operation.

Refer slide time :(10:33)



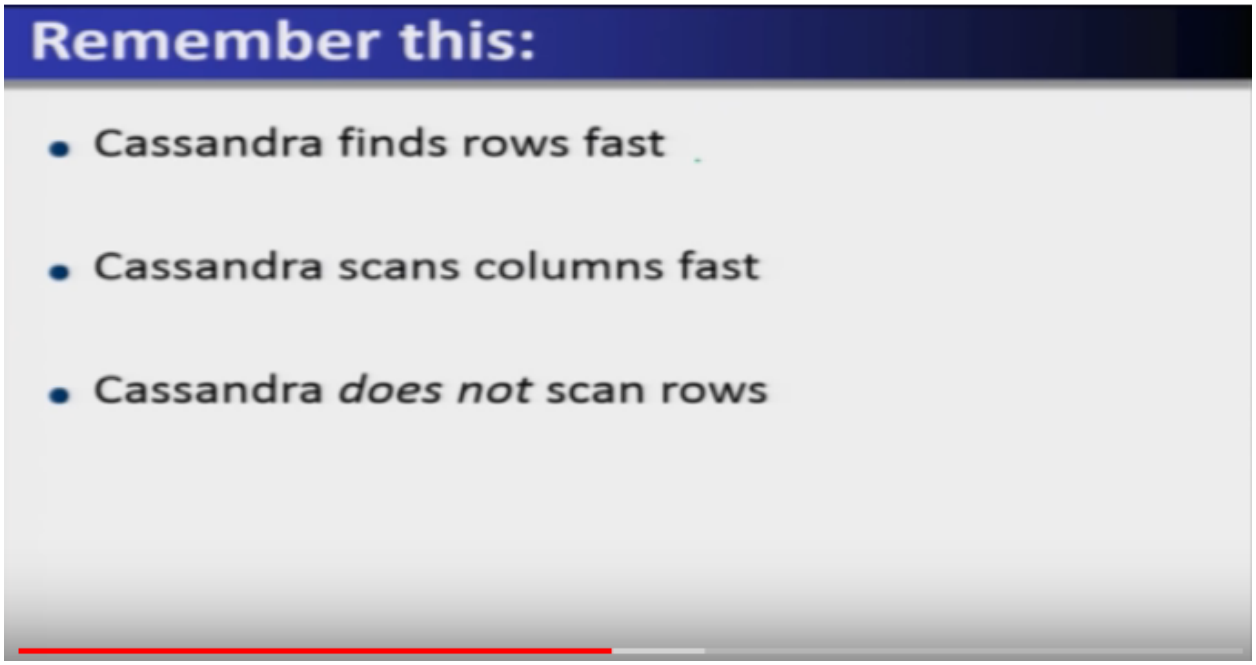
Now let us see, about Cassandra query language to access, this particular key value pair, this database Cassandra provides, easy interface, that is looking that is quite similar to SQL that is called,' CQ'. Let us go and discuss more detail of it so; CQL is reintroduction of a schema. So, that you don't have to read the code to understand the data model. We need to say that the interface, by which the users are going to access, is quite familiar and this is nothing but similar to the SQL, but internally their implementations are quite different. So, how this CQL is into the internals of Cassandra that we will discuss in more details. So, CQL creates a common language. So, that the details of the data model can easily be communicated. So, CQL is the best practices, the Cassandra interface and hides many messy details.

Refer slide time :(11:35)



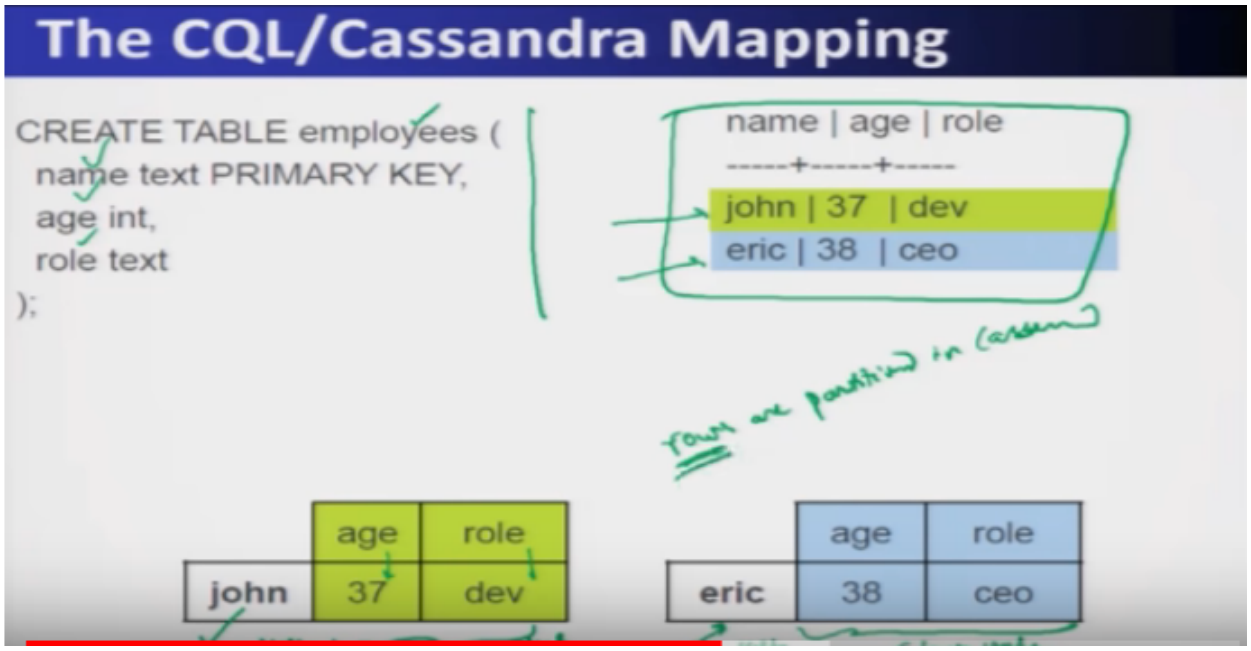
So, to give a very, quick overview of what kind of construct CQL provides. So, for just like we have seen in SQL all these commands similar commands are available, in CQL that we will see in the further slides.

Refer slide time :(11:56)



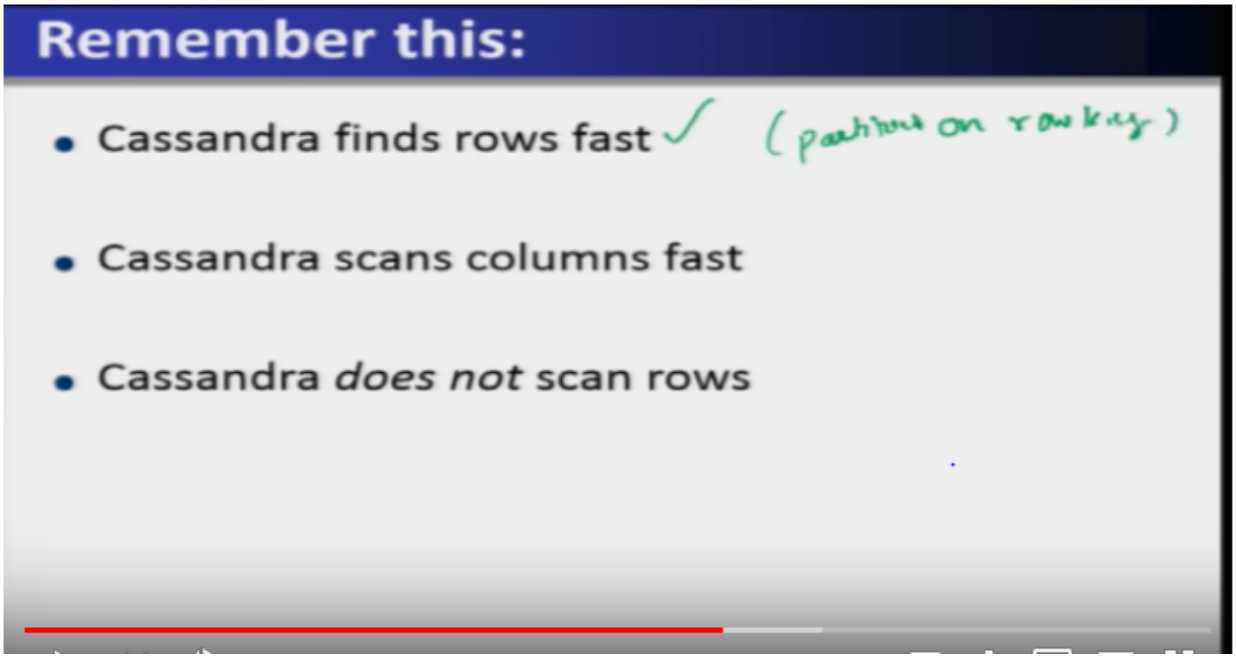
So, remember that Cassandra finds, the Rows when you fast Cassandra, scans the columns very fast Cassandra does not scan through, the rows these are some of the, the features, of the Cassandra, how it performs how it does. So, we have to see all these particular and three different steps, in our operations.

Refer slide time :(12:14)



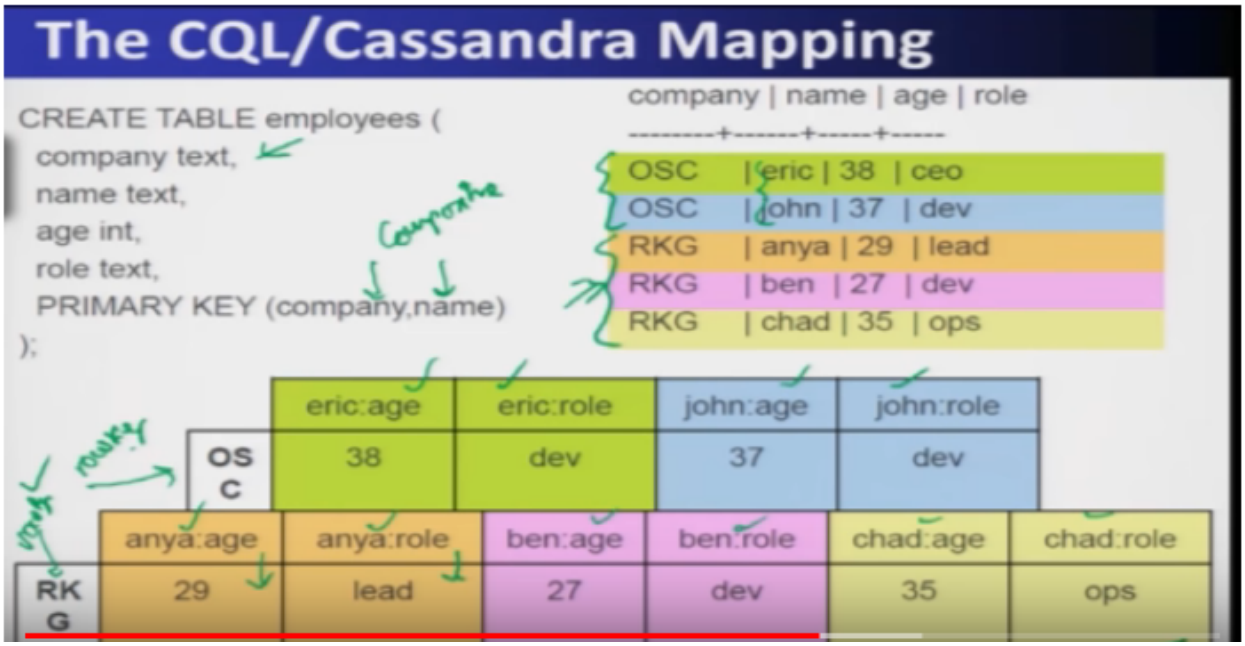
So, let us see that, what are the CQL commands CQL commands, of Cassandra and how it is internally being mapped into the Cassandra. So, if there is a create table, CQL command that is the create table with the employee name. And name is the primary key and age and role are the two other attributes. Now let us populate, this particular database, having two tuples one is called,' John', that is called,' Eric'. Now as far as internal mapping of this CQL command, is concerned the entire operation. So, this particular name, is a is a primary key. So, the further couple for the row a key, that is John will become the row key, for the for this particular tuples. And corresponding age and the roll number and corresponding age and role, they will be having their different details, noted. So, this is the, the row key and this entire row is identified using row key and others are the column values, similarly Eric row key will be used as the row key and these are all column values. Now these rows are partitioned in Cassandra. So, these rows are now replicated and stored, in the cluster, having the row keys and they are being accessed very fast.

Refer slide time :(14:33)



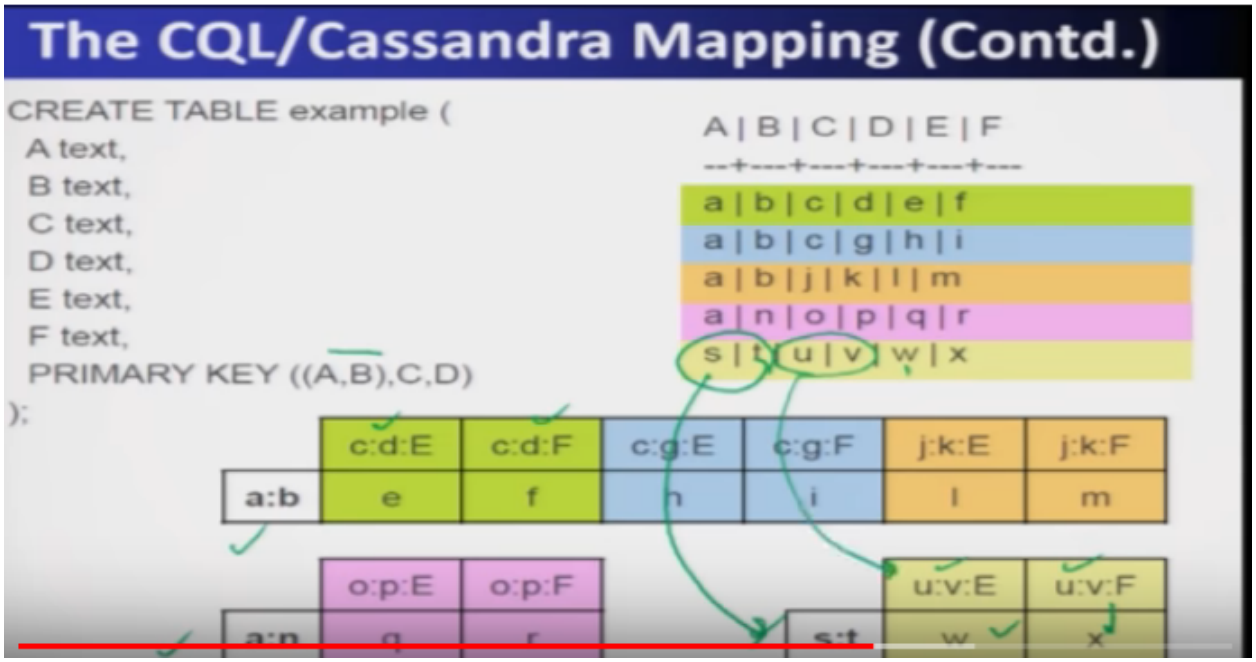
So, as far as this particular, important point, which was there that Cassandra finds that was very fast why because the, the row keys, they are partitioned across the, they are partitioned, on the row keys, hence they are very fast to access.

Refer slide time :(14:47)



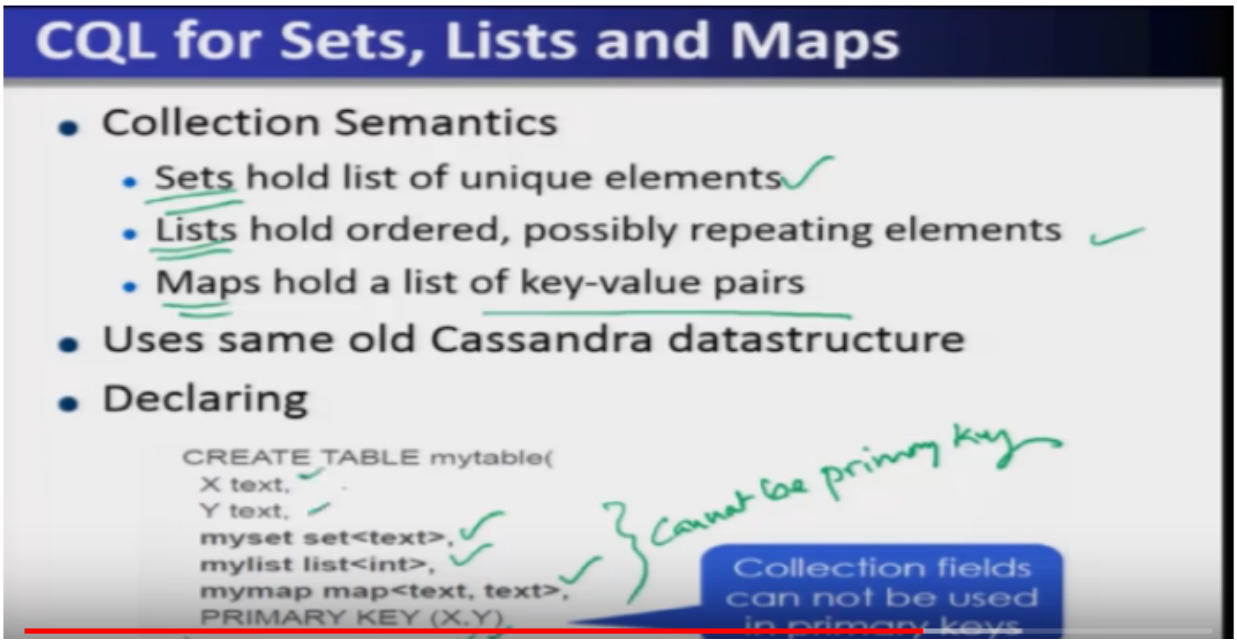
Now the next, example which we will see about CQL is that we are going we have added one more attribute, that is called,' Company'. And now the primary key, is now the composite key, now let us see the, the table or a column family, which will look like in this manner and it has two of the, there are two tuples with the OSC company name and there are three tuples with RKG company name. Let us see that five different tuples will be converted into two different row keys. So, the first row key is OSC and the name is the Erich. And so, OSE has two different Erik name so, both are were here, in that same row key, which is having the value OSC similarly it has the, the age values, column values, those values are also added. So, this becomes a row key similarly RKG will be the another row key and this will have this name Anya and Ben and chair there each and roll their values or column values are now, put into the different column values. So, so this will be partitioned across using the row key and will be stored into the Cassandra. So, this is the CQL to the Cassandra, internal mapping.

Refer slide time :(16:42)



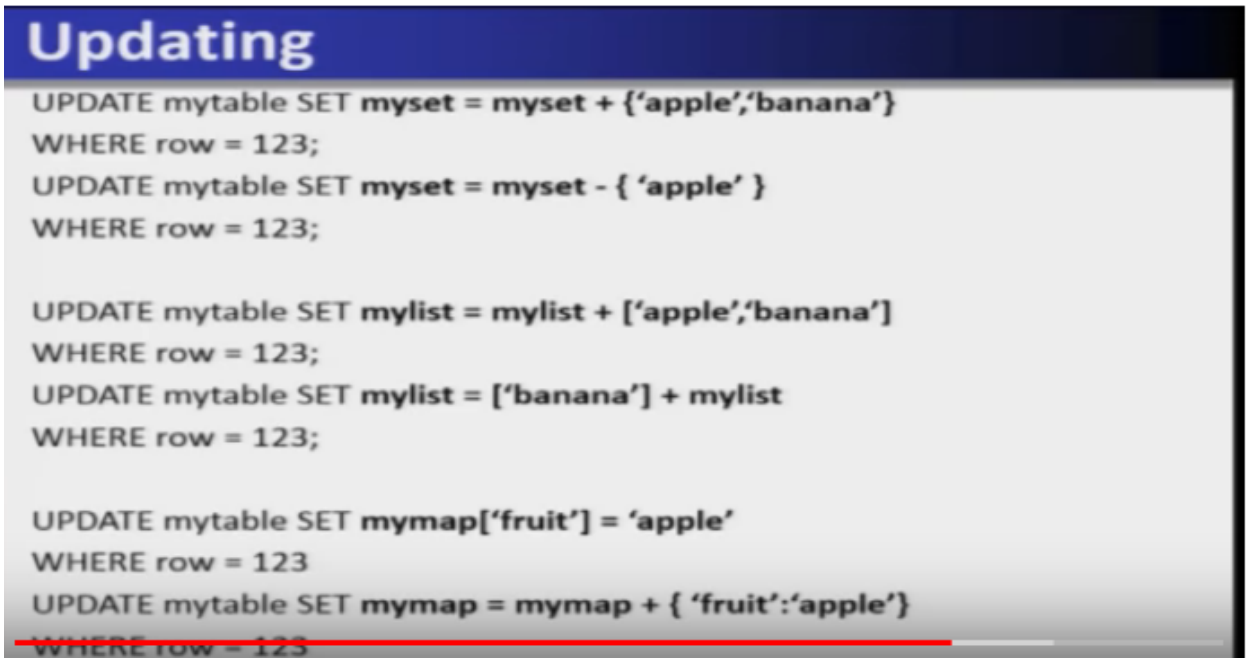
Similarly we can extend it into the more complicated, example where in the primary key, is now another a composite of four different keys, A B and then C D. So, A B is one of the primary key and then C D and their values are corresponding, values are now put in this column similarly for a 2 n. Now let us consider about s 2 T so, this is the a B they will become the primary key and the CD UV and their CD values are now noted down, as for evaluate will be now W and for F value it will be X and U and B these values, will come as the column attributes. So, this way the

Refer slide time :(17:41)



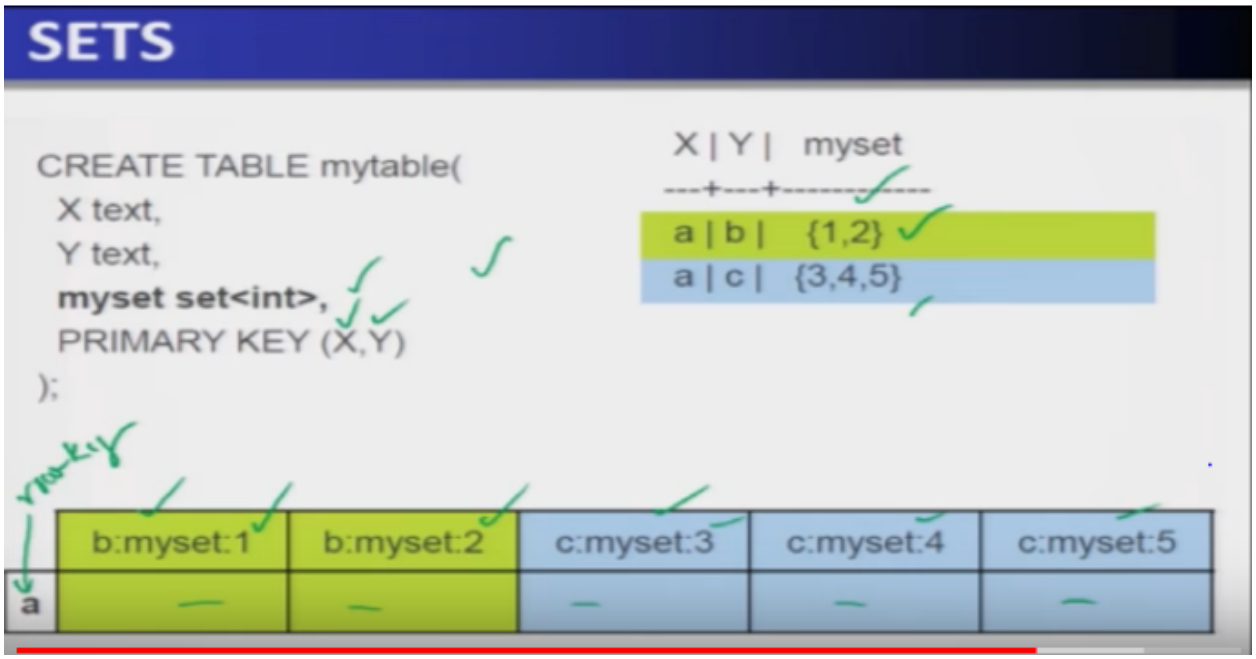
entire row key will be mapped, now CQL also supports the sets list and map. So, they are the collect they are the collection semantics. So, set holds the list of unique elements, set means the list of unique elements. And the list is nothing but an ordered possibly repeating the elements that is called the,' Set'. And map is the list of key value pairs, now uses same old Cassandra data structure. Let us see how we can declare, these things, that my set is equal to set and a text and my list is a list of integers and my map will be the map of key value pairs and note that these fields that is set list and map cannot, cannot be the primary key. So, the primary key will be, from the normal attributes, that x and y can be the primary key.

Refer slide time :(18:59)



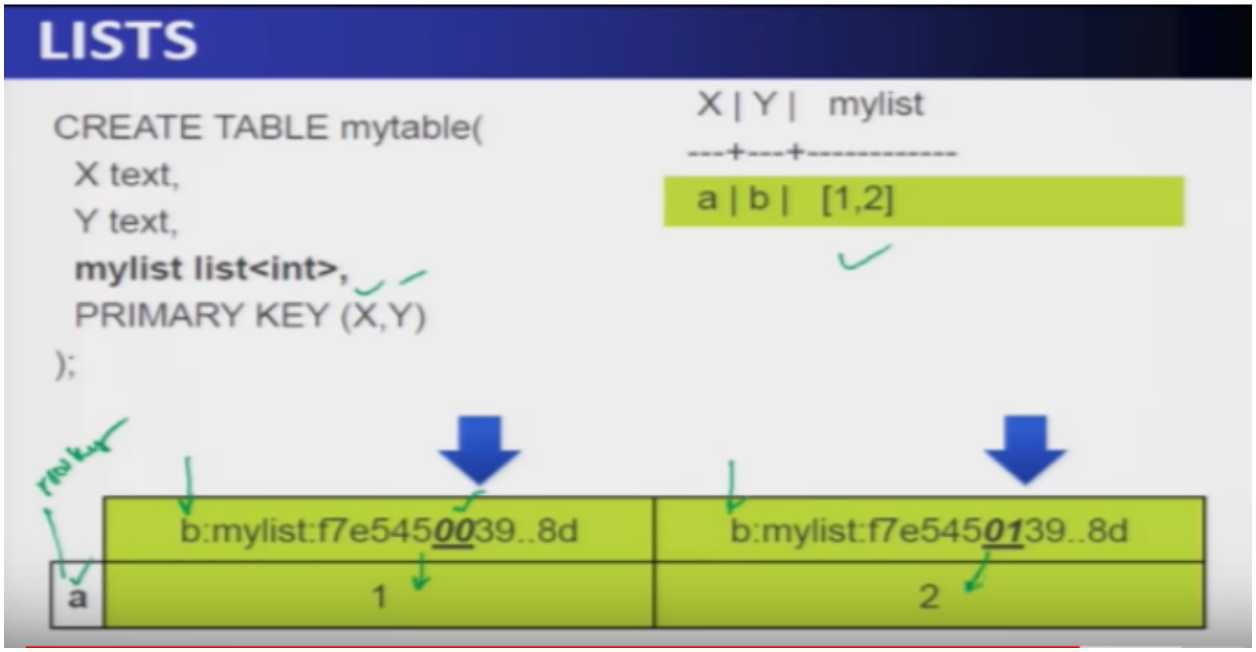
Now there are some operations, all these operations are supported, to insert into the into, the table, these values, whether it is lists or it is set or it is the, the map or it is the normal, very values these values can be inserted into the using insert into the command, of off CQL similarly we can update, it we can add and we can delete the values.

Refer slide time :(19:22)



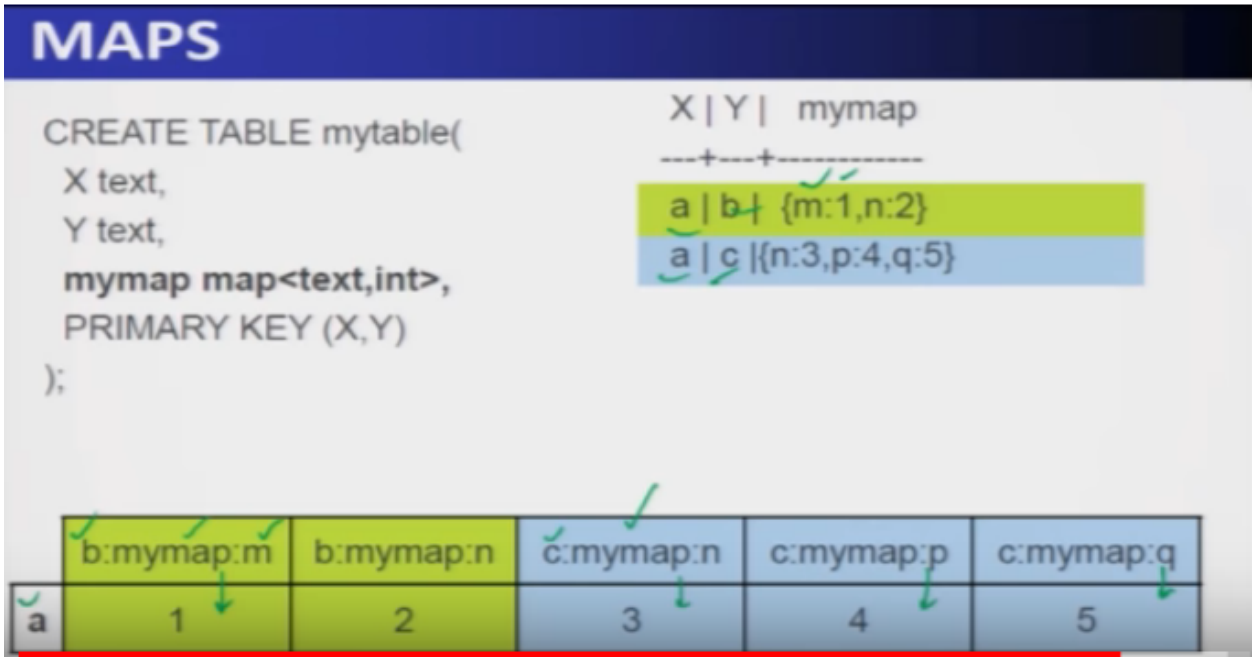
Now let us see about the set how it is being mapped into the Cassandra, data model. So, that means if we are given, the, the CQL command for CQL query for create table, by table and here we are using the set, my set and the primary key will be XY. So, the set will be the set of elements, here two sets will be given. Now this entry is shown like this that a will be the, the row key. And as far as the columns are concerned so, primary key are x and y. So, Y will be the on the primary key, so, so Y will be 4b and 4c, now for B there are two different members, in, in the set, two different elements in the set. So, my set with the member one, my set with a member tow, with a member three, four and five. Now you see this there will be no, values in this column values and everything will be there in the column attributes. So, this will be the, the internal mapping, of the Cassandra sets into the,

Refer slide time :(20:53)



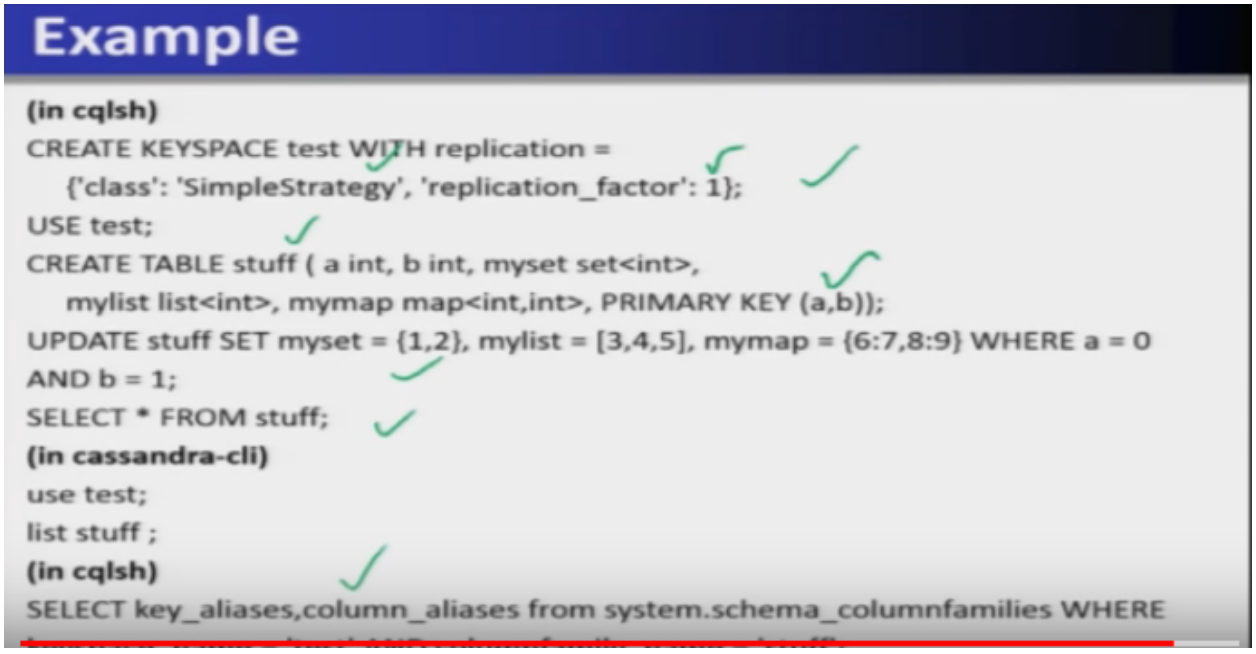
into the Cassandra details so, another construct which is there in Cassandra is called a,'List'. Now let us see the example of a list here, the elements of the list may be repeating also. So, the primary key is x and y so, here a will be there as a row key and the other row key will be the B which will be there on the column, together they will identify, this part in our my list values. So, my list is some arbitrary value. So, in that there will be two values so, 0, 0 and 0 1 will the first value will be 1, the other value will be2, in this way all these list values are being used here in this case.

Refer slide time :(21:41)



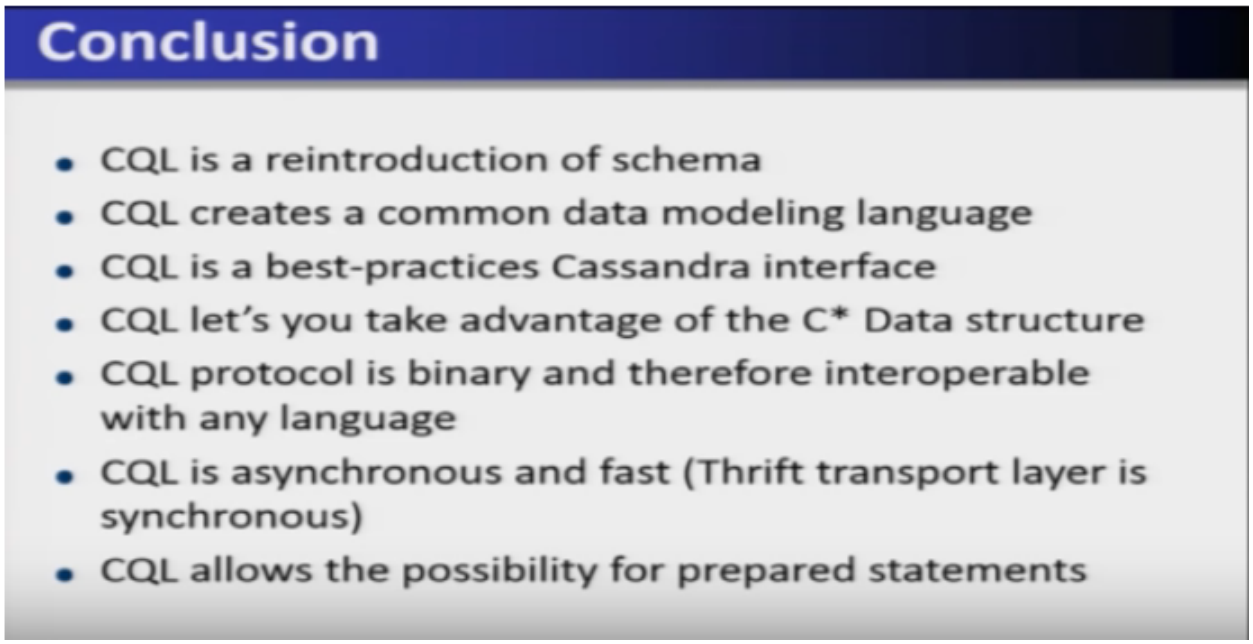
So, Maps means key value store so, key and value will be stored, here in this case let us see this example also. So, A and B they are the primary keys, A and B they are the primary keys they are they are restored and from the key value pair from the map. So, key will be M and the value will be stored over here. So, therefore in the second row A and C will be the primary key and they are key value pair that is from my map. And 4nthe value will be 3 and 4 P with the value will be 4 and it will 5 in this way, this particular map, of Cassandra CQL will be stored internally.

Refer slide time :(22:27)



So, let us see some examples, of CQL so, we can create first the key space, which is named as a,' Test', with the replication factor is called 2:1 all these things we have already, studied and the petitioner is simple strategy, which is being used, for replication or storing the data and then we have created a column table, column family that is called,' Create Table'. And the name is stuff and all these are the elements and the primary key will be a Band we can perform an update, into it and then we can perform the select, operations. And from select all the values from the stuff which we have just, defined and furthermore we can create, the soul select, using key alias column alias, from the system is schema column families, where key space name is equal to test and column family stuff. So, we have seen that, the CQL is very familial and it looked like SQL commands, but internally it is being handled in a different manner that we have already discussed and covers.

Refer slide time :(23:57)



Conclusion CQL is a reintroduction, of the schema CQL creates a common data Modern Language, CQL is the best practices, the Cassandra' interface and CQL lets you take advantage of the data structure, of the Cassandra CQL protocol, is the Binary and therefore interoperable with the languages, CQL is asynchronous and fast, CQL allows possibility for prepared statements. Thank you.