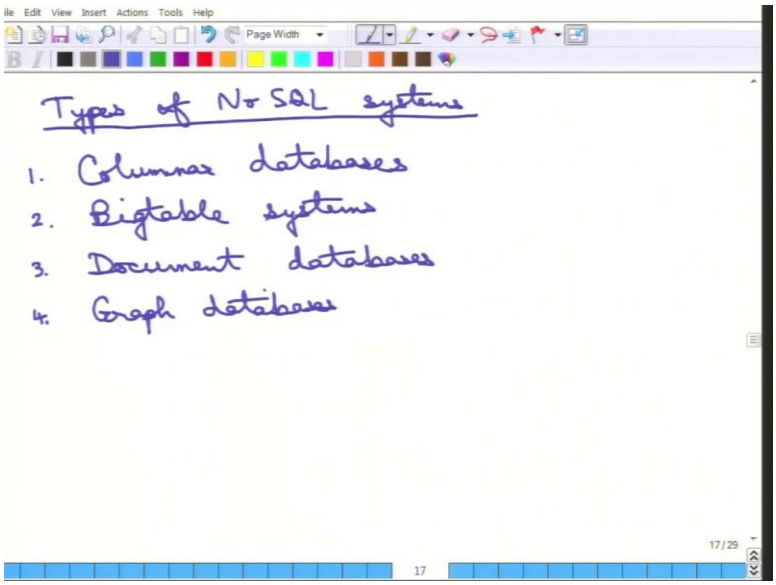
**Fundamentals of Database Systems**
**Prof. Arnab Bhattacharya**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kanpur**

**Lecture -46**
**NoSQL: Columnar Families**

What are the types? There are different types of NoSQL system.

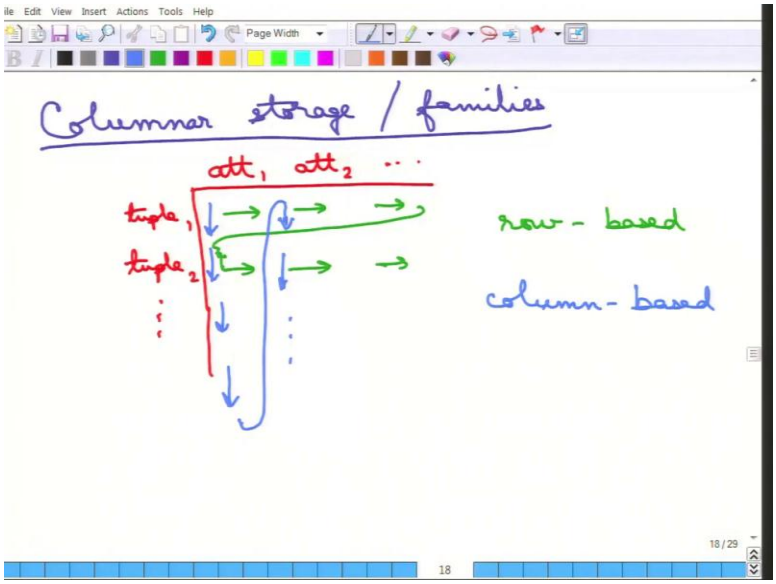
(Refer Slide Time: 00:15)




So, there are four main types of NoSQL systems. So, we will just briefly go over them, note that the whole issue of NoSQL systems may be another half course or whatever I mean, it can be very large, but… There are four main types of NoSQL systems. The first one is the columnar database family, so columnar databases, the second is the big table systems, the third one is the document databases. You may or may not have heard the names of this.

So, these are nowadays becoming very, very popular and so, may have heard the names of these things anyway and the fourth one is graph databases. So, let us go over each one of them one by one.

(Refer Slide Time: 01:17)



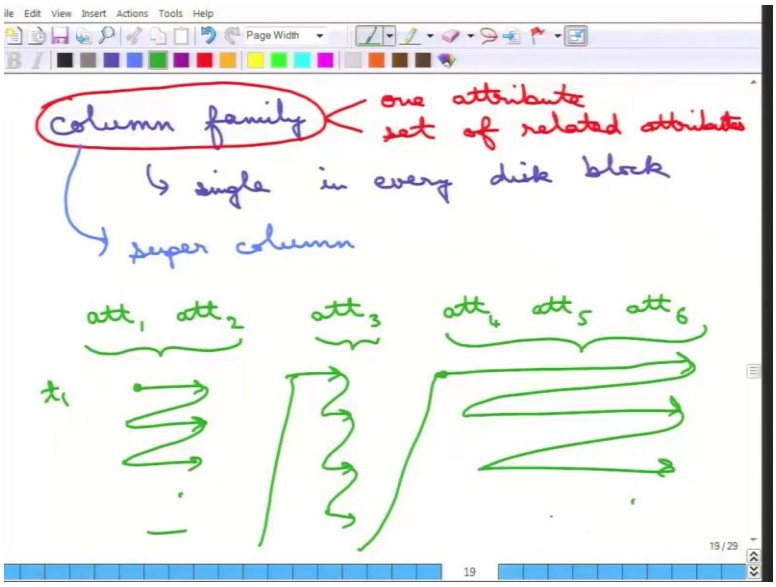

So, the first one is columnar storage, so columnar storage or columnar family let us just say columnar storage or columnar families or columnar databases, etcetera. So, one important thing is that, how is data stored in DBMS is they are stored in a row manner. So, what does it say to store in a row manner is that, think of a typical relational table. So, there is tuple 1, then tuple 1 has attribute 1, then attribute 2, attribute 3 and so on and so forth and after that it is tuple to it.

So, how is it actually stored in the disk? Disk is essentially, I mean the one dimensional line, because logically it is stored in tracks, sectors, one sector after another and so on and so forth. So, what it done is that, each tuple is taken, all it is attributes are stored and then, the next tuple's attributes are stored and then, the third tuple and so on and so forth, so that is called the row wise storage. So, to highlight, what I am trying to say is, this is a normal database table, this is all the tuples, so this is tuple 1, tuple 2 and so on and so forth and while, this is attribute 1, attribute 2 and so on and so forth.

So, in a row based storage, what is being done is that, this is stored, then this is stored then this is stored, then this is stored, then this is stored, then this is stored, so that is, so this is called a row based storage. On the other hand, what is being done for the columnar families is that, this is stored, then this is stored, then this is stored and so on and so forth. So, all everything up to this is done, then attributes 2 is stored. So, what is this is being done is that, this is the column based storage.

So, in column based storage, what is essentially done is that, the first attribute of all the tuples are stored. So, the attribute 1 is taken, it is value for tuple 1 that is stored, then for tuple 2 it is stored, then tuple 3 and so on and so forth and up to all the tuples. Then, the attribute 2's values are stored, tuple 1, tuple 2, so on and so forth. So, it is stored in an attribute wise manner; that is the column storage; that is why it is called a columnar family or column based storage or columnar family, columnar databases, etcetera.

(Refer Slide Time: 03:42)



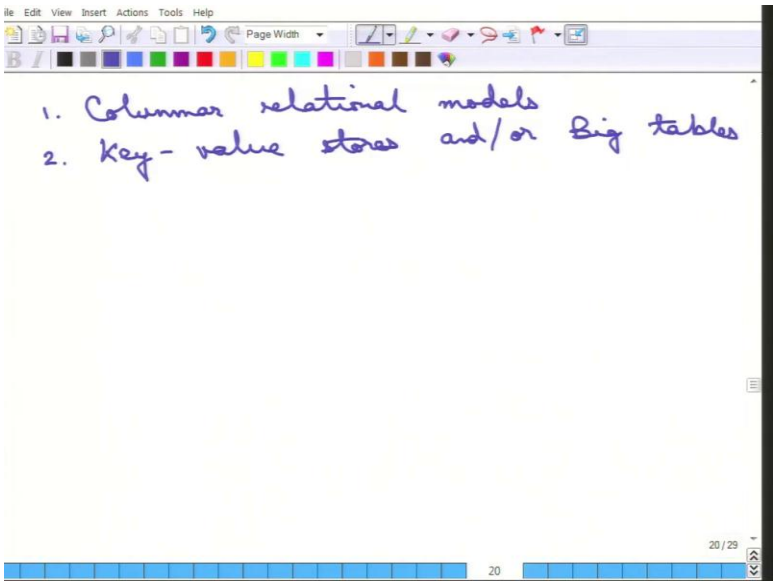

So, essentially this is the idea of storage and there are couple of things is that, so a single disk block or this is a single, what is called a column family, there is a concept of a column family. So, a column family is that instead of storing it one column at a time, you can group certain columns together and that can be stored. So, a column family this is stored in a single I mean, so every disk block contains only a single column family, so single column family in every disk block.

So, every disk block stores only a single column family. So, if it is a new column family it is a next block, it is stored and so on and so forth; that is the, this is the concept of the column family. So, either a column or a set of related columns together, so this can be either one attribute or set of related attributes, so that is called a column family. So, then this set of columns is sometimes also called a super column, so the column family storage this is set of things is also called a super column, so that is the super column.

So, essentially suppose the example is that attribute 1 and attribute 2 is pertaining to one super family, then attribute 3 by itself is one super family, then attribute 4, attribute 5, attribute 6 is another super family. Then, what happens is that this is stored, so this is tuple 1 this is stored, then this is stored, then this is stored and so on and so forth. And after that is finished a new disk block starts, how many disk blocks it takes and then, this is stored, then this is stored, then this is stored and so on and so forth.

Then that is finished, then this is stored, because this is all part of the same column families, so then all the attributes are stored one after another, then this is stored and then, this is stored and so on and so forth. So, that is the idea of a column family or a super column.

(Refer Slide Time: 05:54)

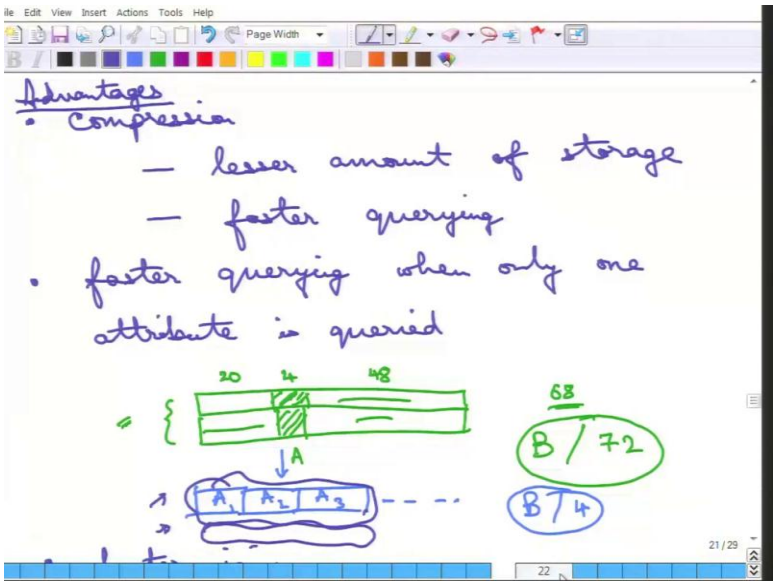



Now, out of these columnar families there are two main types of things, so first is the, so this is the columnar relational models. So, there are essentially two main types of things, the first is the columnar relational model and the second is the key value stores, also sometimes these are also or the big tables, what are called the big tables. So, very, very importantly, whenever we say column storage, so even the column storage is considered to be part of NoSQL, note that column storages can be relational models.

So, that is why it is called not only SQL, because these are part of these relational models or also NoSQL. So, columnar storages just by itself does not mean, just the fact; that is NoSQL does not mean it is not RDBMS, columnar storages can be RDBMS, so these

can be relational models as well. So, this is essentially, what we have been just studying about, so this is RDBMS and it is stored one after another.

(Refer Slide Time: 07:05)




Now, couple of things about columnar relational models is the following is that, see when attributes are stored, when these are stored in an attribute wise manner, it allows compression of attributes. Now, what does it say means to say it allows compression of attributes? Suppose, there are many tuples which share the same attribute, suppose there are 100 such tuples, which share the same attribute.

So, what can be done is that, only one couple of that attribute value can be stored and it can be said that this is shared by tuples, whose ids are from whatever 900 or some 1 to 100 or 213 to 312 and so on and so forth, so that will allow compression. Now, compression saves two things, first of all it takes lesser amount of storage, because it is here of course, compressing it that is the thing. Also very, very importantly, this can allow faster querying. Why will it allow faster querying? There are two reasons, why it will allow faster querying.

So, first of all when… So, suppose you may need to find all tuples whose value is greater than something, etcetera. So, now, you go one tuple and instead of going over all the tuples you go to that value and you skip over that 100 tuples all together that is one way of doing it, that is the one good thing about compression. Even, if this is not compressed, so these are the gains of this thing, even if it is not compressed, faster querying can be

allowed, faster querying is also true when only, I mean when only one attribute is queried.

So, what does this mean, so this needs to be understood in a little bit manner. So, suppose there is a selection query, which says select everything from the relation, where A greater than 4. What is the basic manner of doing it is that, it goes over the tuples, then it goes to the attribute of A, checks it and if it is correct it says it is correct, it is output otherwise it is not, then it goes to the next tuple, checks the attribute of A and so on and so forth.

Now, how much, so it is essentially going over the disk is essentially going over all the other attributes, which are not A and skipping it is not checking it, but it is actually going over. Now, in the disk, what does it mean to go over it is actually taking more blocks to study more blocks to query, so this is what I am trying to say. So, suppose this one row this is attribute A that I am interested in this takes let us say 4 bytes and suppose there is another 20 bytes here and another some 48 bytes here.

So, to go from attribute A to here you are essentially wasting a going over 68 bytes this 48 plus this 20 only, then you are reaching the next A. So, in a disk block, how many such tuples can be stored or how many such tuples, where only A search can be stored this is essentially the disk block size B divided by 20 plus 4 plus 48, so this is 72, so that many number of tuples can be stored. So, that many number of attributes A can be searched by accessing one disk block.

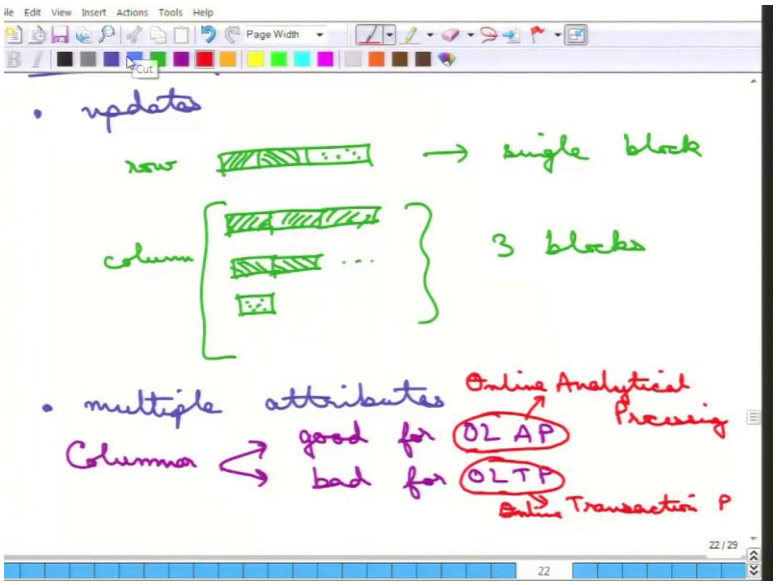
However, if this is stored in a columnar manner, then what is being done, is essentially attribute A of tuple 1 is stored, then attribute 2 of tuple 1 is stored and so on, so forth, this is all four blocks and so on, so forth these are all stored together. So, then how many such attribute AS can be stored in 1 block this is B by 4. So, you see that is a larger number much larger number, so it is a larger number and, so it is quite beneficial if it is stored by attributes, because just by one disk block access more tuples and more attributes or more, more attributes and more tuples can be searched.

So, the single attribute in more tuples can be searched, so that is the, so more tuples can be decisions about more tuples can be taken by searching them. So, that is the one good thing about this thing and just to continue this allows also faster joins. So, why what are

joins, what joins are generally on two columns, one column from table A and another column from table B.

Now, essentially it just go over that table A and other attribute in the table B the attribute A in table the first table and attribute B in the second table and that again by accessing lesser number of disk blocks more such joins the join condition between more tuples can be checked. And only if that attribute A values and attribute B values are matching, then the actual tuple needs to be bothered about the actual tuple needs to be output. So, the even the joins are faster, having said all of these are essentially the advantages of columnar storages.

(Refer Slide Time: 12:11)




Now, what are disadvantages there are certain disadvantages as well. So, the disadvantages are the following is that updates, so updates are not good updates are not good at all, because, now what does updating mean updating generally means that particular tuple is updated of course. I mean the particular attribute either, even if it is one attribute is updated or multiple attributes are updated. So, multiple attributes suppose there are multiple attributes are updated for a single tuple.

Now, how many blocks does it require in a row based storage to get the complete tuple it is just one disk, because tuples are stored particular tuples are always stored in a single disk block. Now, here whatever is the number of column families or super columns, that many blocks need to be accessed. So, here is the idea is that, if it is row based storage

this entire tuples of suppose this is attribute A this is attribute B and so on, so forth and this is attribute C and everything is stored in a single block.

So, if there are three updates that needs to be done, the tuple is brought from the memory and this is one single block tuple is brought to the memory from the disk one single block that is it. However, if it is row based storage this is in one block this again is in another block, because in this other block there are attributes only of the same type this is not other attribute. So, that is there not there this is again attributes of this type only.

And again, so the third attribute is again in another block, so this will require three blocks, so that is the difference between, so this is in row and this is in column. So, updates are much slower if the update involves more than one attribute it is much slower for columnar families columnar databases. Also if the query or the join, if the query generally, if the query touches multiple attributes, then it is also not faster probably it may not be faster it is not clear it will depend on lots of other parameters it may not be faster.

Because, it again goes to it accesses the tuple when if it checks if that value passes it needs to go to the other disk block to access the attribute of the same tuple and then, it needs to see that and so on, so forth, so it may or may not be faster. So, when multiple attributes are done it may not be any faster. So, that is anyway the algorithms need to take care of this handling different things and keeping the idea or the tuple id etcetera and so on, so forth can be done.

So, then essentially there are two terms in the database that use in the commercial databases, that is being done is that. So, columnar families or columnar things are good for OLAP and bad for OLTP, so the OLAP and OLTP these are two new terms, so OLAP this stands for online analytical processing or analytic processing and this stands for online transaction processing. So, what does this mean analytical and P stands for this is online analytical processing and the other is of course, online transaction processing.

So, the same thing, so what is OLAP essentially versus OLTP, OLTP is essentially transaction. So, just single updates are being done or some small part of anyway small part of the database is updated. So, a single tuple or a couple of tuples like that account A account B it is only account A tuple is updated and account B tuple is updated. These

transactions touch only a very small part of the things, but they touch probably multiple attributes of these things.

So, for that columnar families are not good, because the relational databases can do it much faster if they are doing it. However, for this analytical, so the online analytical processing, what happens is that suppose you are trying to find out the summary of certain things the average over all balance all accounts, then the relational databases the traditional row based relational databases will take a longer time as opposed to columnar databases, because they access that column much faster and can do this summaries and aggregation operations much faster.

So, these are the two standard terms OLAP and OLTP and one big example of this relational database is you must remember the name, which is Monet DB. Monet DB is the example of a columnar relational databases Monet DB.