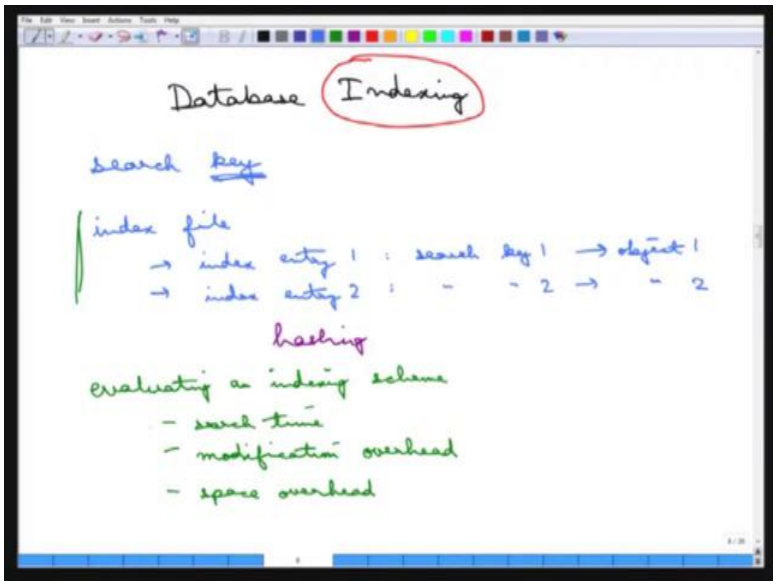
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 18
Database Indexing: Hashing

We will start on the next topic which is on Database Indexing.

(Refer Slide Time: 00:15)



So, we will first try to understand what does the word indexing mean and you all have probably seen examples of indexing is in a book towards the end of the book, there is a word is written on the page number 2, where the book is found is written and that is called an index. So, what does indexing help one do is to make the search faster.

So, imagine that a book has no index and you want to find a particular word. What is the way to do it? You have to go through the book probably page by page, word by word to find that particular word. If the index is there, you essentially just go to the index, find the word that you are looking for and just simply go to the page number. Now, also important in the index is that the words are stored in an alphabetical manner. So, that finding the word inside the index is also faster, so that is the whole concept of indexing that is the whole idea of indexing and databases use index very heavily, so that the queries can be fast.

So, what are the kinds of queries that we are talking about? Think of this select query, select all branches or select all loans whose amount is greater than 300, etcetera. Now, how do you find out all such loans? The naive way of doing it, that the simplest way of doing it is to go through all the loans and select whichever is larger than 300. But, suppose an index is built and we will see exactly how the indexes are built, suppose the index is built which tells you all loans that are greater than 200 and all loans that are less than 400 very easily, then all one needs to do is to find it inside this bucket of the loans and not the other one, so that makes it much faster, so that is called the indexing.

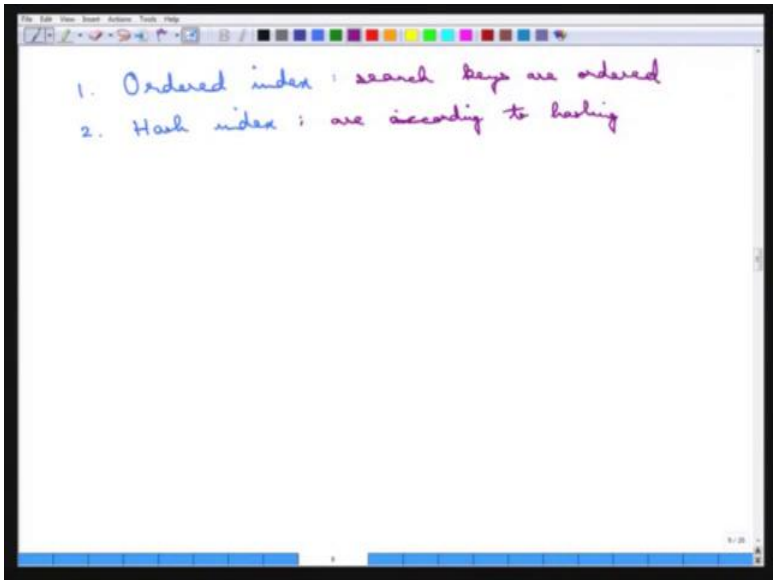
So, what is being done in an indexing is a search key is used. So, we are talking about searching, so there is a search key that is being used. So, we want to find this particular key and how is this done. So, an index file is maintained, there is an index file which contains multiple index entries, so each index entry 1, index entry 2, etcetera, so the multiple index entries are maintained. So, what is a look index entry? It contains a particular search key and a corresponding object to it.

So, as soon as I make this kind of design, the first thing that comes to one's mind is we have seen this and this is exactly what is being done in hashing, hashing does something similar. There is a search key and then there is a actual object corresponding to the search key, corresponding to the hash key. Before we go into the types of indexing, let us just see that how does one evaluate an index. So, evaluating...

So, there can be different types of indexing schemes, evaluating an indexing scheme requires the following thing is that on what are the ground that we will index, it is the search time of course,. So, the whole point of indexing is to reduce the search time, so it must be able to do so, then the modification overhead. So, what is meant by the modification overhead is that, suppose the data in the original database has changed. Now, how much time does it require to change the corresponding index or the entire index file that is the modification overhead and the space overhead.

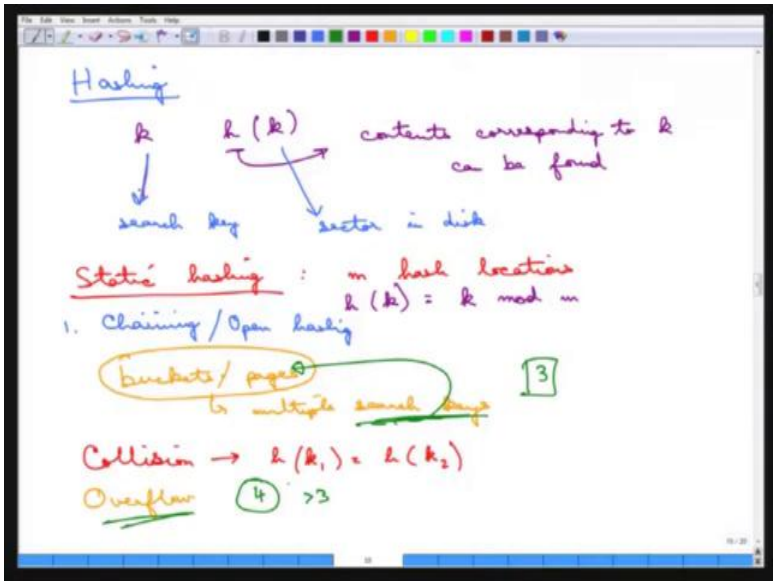
So, suppose the original data is one gigabyte, so how much extra space does the index needs, because you see the index is different from the original data. So, there will be some extra space, so what is that space overhead that we are talking about. So, these are the three criteria for evaluating an index.

(Refer Slide Time: 04:22)



Then, there are two basic types of index schemes, the first is called an ordered indexing scheme and the second one is a hash index. So, in the ordered indexing schemes, the search keys are ordered, the search keys are stored in an ordered manner and this is what we see in the index of a book. The search keys are stored in an ordered manner in the index and in the hash key, the search keys are in some hash order or according to hashing. So, there is essentially no order or you can, one can say that there is a hashing order, so it depends on the hash function. So, let us go to the indexing that we all know about which is the hashing.

(Refer Slide Time: 05:10)



So, we will talk about hashing, hashing there are two main types of hashing. So, before that what is a hash function does, the hash function does the following is that there is a key k, then a hash function is applied on that, hash function h is applied on that. So, that gives a location, where the object is stored or where the contents corresponding to the key, contents corresponding to k. So, that can be in the object, that can be the actual some other else, whatever if corresponding to key can be found, so that is the whole point of hashing.

Now, what does it mean in the context of a database? The key is the search key and the hash key, there is a sector or in the disk sector in disk, where the contents corresponding to the search key are stored. So, that is the context in the database, so that is the hashing and then there are different kinds of hashing, the first one is called the static hashing. So, what are the ways of static hashing? The first one is chaining.

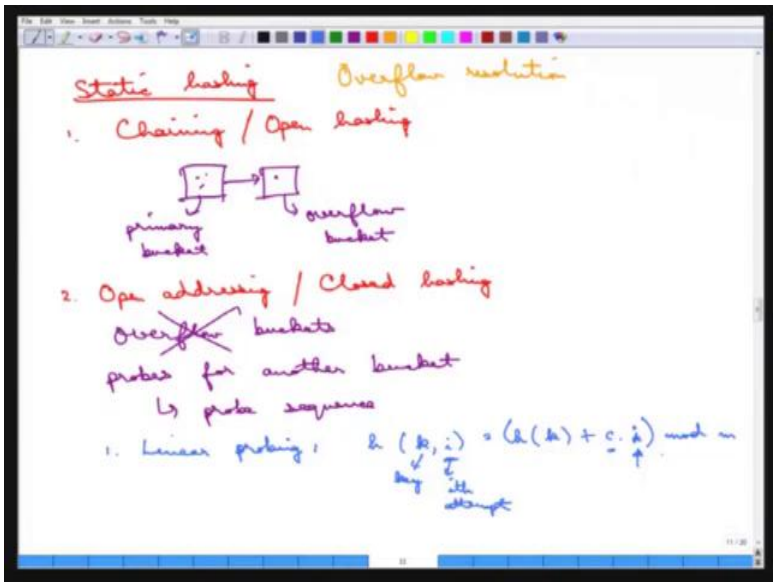
So, static hashing by the way static hashing essentially says that there are m hash locations. So, everything that needs to be hashed is one out of this m, so very common example of hashing is that if there are h locations, there is a k mod m. So, any key gives you a number, once you take a module as according to m, gives you a number between 0 to m minus 1 which is what, where the key is stored or the contents corresponding to the key are stored.

Now, what is in the context of chaining? Chaining is also sometimes called open hashing. What does chaining mean is that, before we go into the chaining, let us go back to the database context and in the context of hashing, what is being done is the following is that in the database, the objects are stored according to buckets or pages in the disk. So, a single page can contain multiple search keys, so this can contain multiple search keys.

So, once a search key is given the point of the hashing is to find the bucket, where it is stored and a bucket or a page can contain multiple search keys. So, in normal hashing in the whole sense what we termed as collision, what does a collision mean is that when two hash keys have the same location. So, h k 1 is equal to h k 2 then they collide, because they are both trying to store in the particular position. In the context of database, there is no concept of collision, because multiple search keys can be stored in a bucket.

So, the collision does not make sense what makes sense is overflow, the concept of overflow. The overflow is the following is that when there are so many search keys, so suppose in a particular bucket three search keys can be stored. Now, overflow may happen when more than three, suppose four hash keys are trying to go to the same bucket, then the bucket is said to overflow. Because, the bucket or the page or the sector or whatever does not have enough space to contain these four things, it can only contain three, so anything greater than three is a problem and that is what this the overflow is about.

(Refer Slide Time: 08:56)



Let us talk about what the static hashing means, so there are static hashing and the first scheme there is called the chaining. So, chaining is that when... So, all these static hashing, etcetera what we are trying to do is something called an overflow resolution mechanism. So, as opposed to a collision resolution mechanism, this is called an overflow resolution mechanism. So, chaining, what does chaining or open hashing does is that when there are multiple keys that are going to the same bucket, so this suppose this is the bucket and then there are multiple keys going to it, another bucket is lynched to it, chained to it.

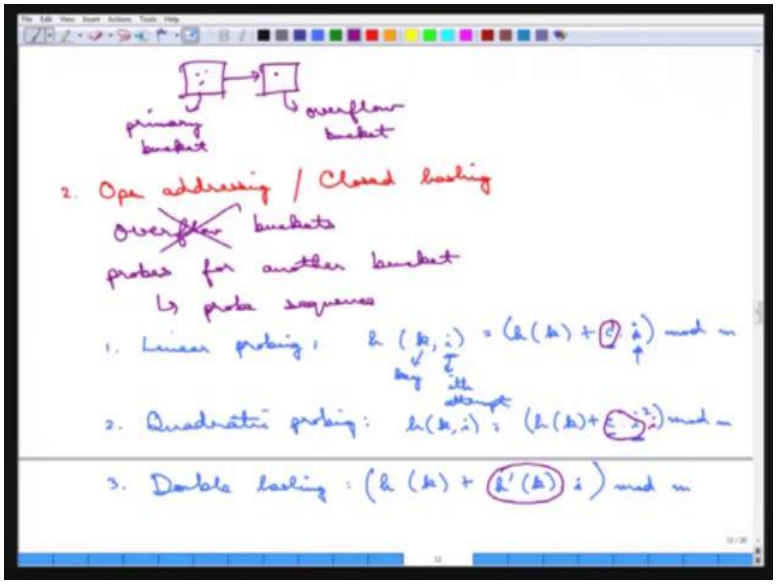
So, the next key that falls to the same bucket is sent to the bucket that is chained to it. So, that is called a chaining. So, these are the overflow buckets, this is the primary bucket, where a particular hash function finds it is way and if it is full, it goes to an overflow

bucket, so that is chaining. As compared to chaining, the next one is called open addressing or closed hashing, so it does not employ overflow buckets.

So, overflow buckets are not employed, what it does is that it probes, it uses a something called a probe, it probes for another location, another bucket and it probes according to a probe sequence and then there are multiple ways of doing the probe sequence and let us just list them, the first one is called a linear probing. So, the intervals of this probe sequence remain fixed. So, suppose this is h k i, so what does i is the i'th attempt to...

So, this is the key and this is the i'th attempt, so whenever there is some problem with the bucket. So, whenever the bucket is full, it takes another attempt at hashing it. So, this is the i'th attempt to find another hash position, so this h k i is simply h k plus some constant c times i and of course, this is all module i assuming this is anything, so this is a constant and this is a linear. Why it is called a linear thing? Because, the position corresponding to the original position is changed linearly that is why this is called a linear probing.

(Refer Slide Time: 11:44)

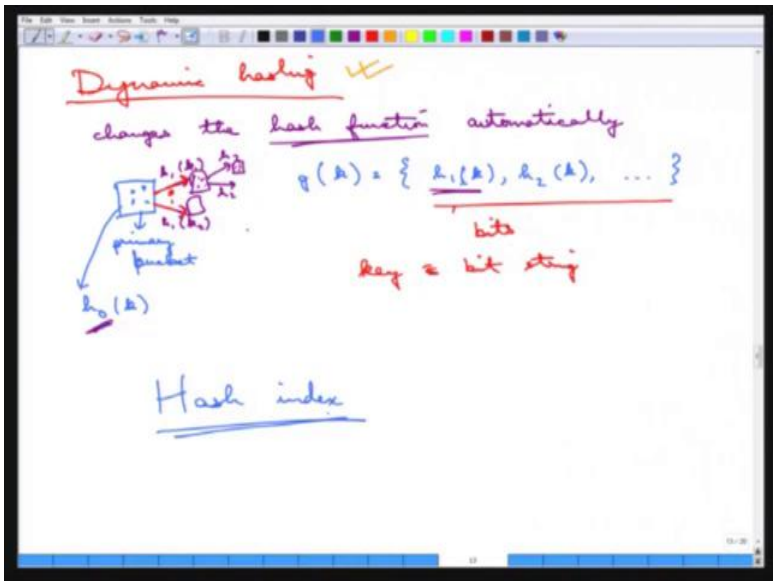


Then one can understand there is a quadratic probing, where this function changes to i square. So, it depends on the this is why it is called the quadratic probing and then the third one is called a double hashing, where instead of using the i'th attempt a completely new function is evaluated. So, it is h k plus some completely new function h dash k that is being done, so instead of a constant this is h k i which is of mod m.

So, one thing to notice that everywhere there is this i function, but for linear probing this is a constant for quadratic probing this is c times i, this is essentially you can write this as c times i times i and for double hashing this is the completely new function double that is the double hashing. So, this is the, these are the three different ways of static hashing.

Static hashing; however, has the following problem is that suppose one has allocated m buckets to start off with and the data really does not fit into any of this m buckets. So, the data for is much more than m buckets, then what can happen is this static hashing performance degrades. So, the problem of static hashing is that it starts off with the idea of how much data is going to be hashed, but it cannot adopt automatically if the data is much more than what it has guessed or much less than what it has guessed.

(Refer Slide Time: 13:37)



So, that is the problem with static hashing and to handle that there is something called a dynamic hashing. So, the dynamic hashing tries to attempt to rectify the situation by dynamically adopting the hashing function itself. So, it changes the hash function itself, the hash function is not static anymore automatically to adjust to the volume of data that is coming. We will not talk about many methods of dynamic hashing that we will talk about one very simple way of dynamic hashing that is called, it was originally just called dynamic hashing.

Suppose, this is the primary page that one goes to primary page or primary bucket, whatever one can call, so this goes to and suppose it has overflow, then what happens is

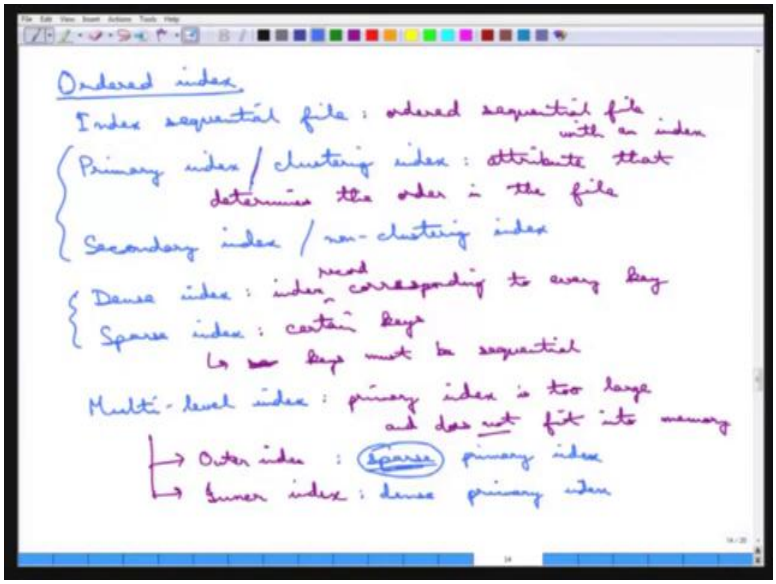
that this is produced by some function. Let us say h 0 of k. So, this is the 0'th function, then there is a series of function, which is g k there is series of function that is h 1 k, h 2 k, etcetera, etcetera, there is a series of function that can be done whenever the data overflow, so all data that overflows from a primary bucket. The next function in the series is applied, so this goes to h 1 k 1 and this is h 1 k 2.

So, the even though h 0 k 1 and h 0 k 2 is the same that goes to the same thing, it goes to different bucket after that and that is the dynamic and even if one of them fails then it utilizes h 2 and so on, so forth. So, that is how it keeps on growing and that is how it keeps on automatically adjusting, because you see that between k 1 and k 2 it is not just h 0 that is the function it is h 1, because h 0 has over flown whereas, for other k 3 it is simply h 0, because that bucket has not over flown.

Now, one very useful way of what getting this h 1, h 2 is just the bits. So, if a key is represented by it is bit stream, then one can simply apply the bits in order to get the different branch branches in this hash tree, one can call this a hash three. So, if this is the bit is 0, it can go here if the bit is 1 it can go here and so on, so forth that is a very simple example. But, the whole point is that this is does what is called a dynamic hashing, because this adopts dynamically to the situation and if the data is reduced it can collapse back in the same manner.

So, this is all about dynamic hashing and the next important thing that we will cover is called a... So, this is called about this is a hash index, because the data is arranged in a hashing manner, the next in that we will cover about is the ordered index.

(Refer Slide Time: 16:46)



So, little bit of the ordered index that we will do, so the ordered index file, there is an index sequential file corresponding to this and then there is a primary index, this by the way is the ordered sequential file with the index. Then, there is a primary index, this is also called the clustering index, this attribute determines the attribute that determines the order in the file and then there can be secondary index or the non clustering index which is any other attribute.

So, the file is ordered according to the primary index and any other attribute on which the index is done is called a secondary index, this can be one way of doing it then there is other way called a dense index, this is a different terms that is one needs to know dense index is that there is an index record, index corresponding to every key. So, that is a dense index as opposed to a sparse index, where there are index records corresponding to only certain keys.

Now, how does this sparse index works? If there are index records corresponding to only certain keys, keys must be sequential here; otherwise, it cannot happen one cannot find a particular key, if there is no record corresponding to it and the keys are not sequential then it cannot be happened. So, that is a thing about sparse index, so this is another way of classifying the indexes, then one more important way is called the multi level index.

So, multi level index means there are two indexes, so this may happen if the primary index is too large and the entire index does not fit into memory. So, that can happen

when a multi level index and then this can be broken into an outer index and an inner index. So, the outer index is the sparse, because it does not fit into any main memory, so not everything can be indexed. So, that is why it uses sparse and this must be a primary index. While the inner index can be a dense primary index, this is the way a multi level index was this is very important that this has to be sparse and this is called the outer index. So, that is about the ordered index. Next we will cover one very important topic in this indexing which is on the tree based indexing.