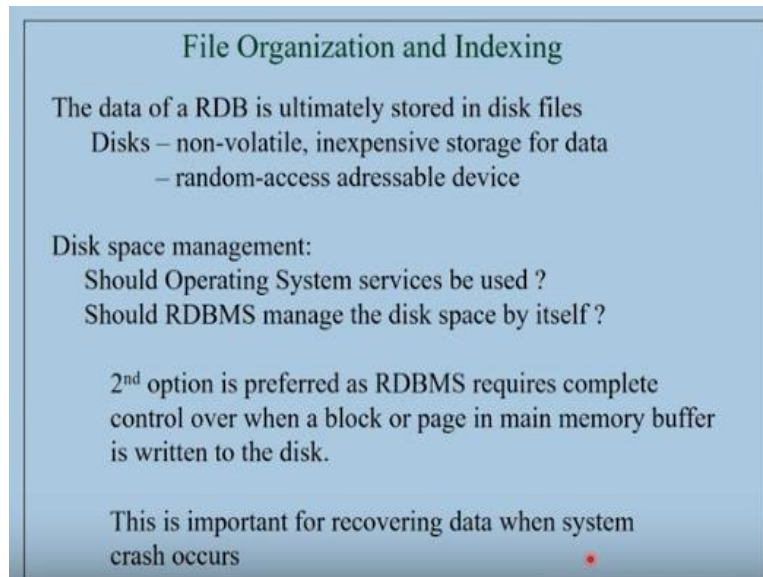


Introduction to Database Systems
Prof. Sreenivasa Kumar
CS & E Department
Indian Institute of Technology-Madras

Lecture - 25
Introduction to File Organization

Start off a new section of the course which is to do with how data is stored on computing systems.

(Refer Slide Time: 00:29)



File Organization and Indexing

The data of a RDB is ultimately stored in disk files
Disks – non-volatile, inexpensive storage for data
– random-access addressable device

Disk space management:
Should Operating System services be used ?
Should RDBMS manage the disk space by itself ?

2nd option is preferred as RDBMS requires complete control over when a block or page in main memory buffer is written to the disk.

This is important for recovering data when system crash occurs

The data ultimately needs to be stored in some stable storage, right. So we need storage that is not volatile. So we cannot be in memory of the computing systems because computer systems may lose power and then you know lose data if it is there any main memory. And the amount of data that typical databases hold with not anyway fit into main memory.

So and at this at any point of time you may not need the whole of the data that is there in a database. You need to store some part of the database in the, in the main memory. We will see how exactly the transfer of data from the non-volatile storage to the main memory will take place. So disks, these hard disks have been the mainstay of this non-volatile storage for a number of years.

So these are relatively inexpensive, you know storage medium. And they are in fact also random access addressable devices. So we will see how exactly they are random

access. Just like your memory is random access these are also random access devices. Random access devices to be contrasted with serial kind of devices, which are you know which are actually not any longer being used like tapes.

Tapes are not animax. They are to be you have to rotate the tape to a appropriate place before you can actually start using data from some segment of the tape. Now there are of course, advances that are taking place in storage technologies. So flash drives are already you know are available and we can see that this the flash drive based external drive disk storage is also available, right.

So but these things largely are we are currently using them for personal backups and things like that. It is also possible to use them for database data storage. But as of today, though they do offer performance benefits, cost of these devices compared to the disks is actually pretty high. So for I think for a few more definitely a few more number of years, we will continue to use this magnetic medium based disk, hard disks as the main storage for databases.

Now actually one more thing that you need to know is that in large enterprises, a single disks are not any more use actually. It is a collection of disks that are used and the collection is organized in such a way that it will give you faster access to the data and also the failures of this disks can be tolerated. The data of a database has to be available to us at any point of time.

Once we accept that we are going to manage this data, the data has to be available permanently for us. We cannot afford to lose data at any point of time. So but disks are electromagnetic devices. They do fail and so we have to take adequate precautions to ensure that the data is available in spite of the failure of a single disk sometimes ((4:32)). So we will see how exactly we can address that issue a little later in the module.

Right now we will take up and understand a single disk how it is and then in general, we will abstract out the lot of details about the disks and then think of them as a medium that offers you a bunch of blocks, and then we will see how exactly we can organize information on this blocks.

Now another important question that comes up in the context of management, storage management is that should the operating system on which the system is working cannot we make use of the operating system services as and then you know get the operating systems to kind of manage the disk space that the RDBMS actually needs. Now you would have studied how paging systems have you know are to be implemented in the operating systems course, right.

So how virtual memory organization is handled and when a processor demands a particular data, how it is brought from the disk and is made available on any memory buffers and how when a block or a page it is also called page right, a page is written by the process that is under consideration, how is it again transferred back to the memory all that, right. Now and it has its own policies right.

So when to replace the so if there are you have seen a buffer of pages available for in the main memory and if you want to bring in a new page from the disk and then you want to vacate some frame in the memory where you can put the new block which of the frames to kind of empty. So that is an issue. So there are some specific policies where you will use and then transfer certain pages.

Certain pages will be transferred automatically by the operating system back to the disk right, least recently used and so on. You have studied some policies there in the operating systems course. So the main point about this, from our perspective, from the database management system perspective is that if you go by the root of the operating system, then the operating system has the control about when to transfer a page that is there on the main memory back to the disk, okay.

Now the crucial question is, is this okay for us? And the answer for in from the perspective of a database management system is that, that state of affairs is actually not acceptable for us. DBMS, the database management system needs to really have very tight control on when a page that it is being modified by the RDBMS system is actually transferred back to the disk.

See once it reaches the disk, then it is in permanent storage, it is in permanent storage. So you recall that in the discussion for the database management systems, we made statements about when you know the when the transactions run when the transactions or logical, you know pieces of work that the server accepts to do on behalf of a end user, like transferring money from one account to another account.

And in reality, these operations consists of several sub operations, okay which need to be performed on the database which is residing on the disk etc. So under these circumstances, you want to give a guarantee saying that this entire operation will be done in an atomic manner. That means it will be done entirely or it will be denied completely.

You cannot you know have thousand rupees reduced from A's account and not credited to B's account if you cannot stop the transaction somewhere in the middle, right. So it has to be done in the entirety. So this kind of guarantees are expected out of the RDBMS system when it does this transaction management.

So under these circumstances, if you update something in the main memory and the page that you have updated slips out of your control and goes to the disk and you actually do not want it to go to the disk, you are in trouble. So you really want to know when exactly it is going and if it is going then you know then you should know that it is going and you should take appropriate measures for you know cancelling that in case it is necessary.

What if the transaction has to be aborted in between. For some reason, if the transaction has to be aborted, then whatever partial work that it has done should not be visible for the end users at all. It should not affect the database on the disk. Ultimately the database is on the disk or on the stable storage. So these are so we will appreciate these concerns when we actually talk about transaction management and error recovery in the next module that we are going to study.

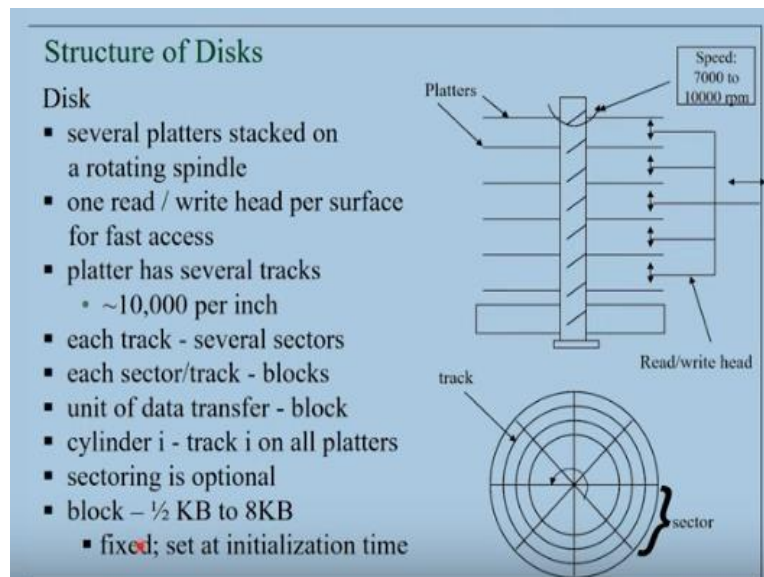
But as of now we can assume that there are these issues related to transaction management which will which are which has to be taken into consideration. And those things dictate that we actually need to have a very tight control on when pages

move from the main memory to the disk from buffers to the disk. That is why most of the RDBMS actually takes control of a certain amount of disk space.

It completely takes off the whole amount of disk space from the operating system and manages the disk space all by itself, okay. So this is the second option that I am talking about, should RDBMS manage the disk space by itself? Yes, that is what is preferred. Because we really need to have very tight control on when a buffer is written to the disk. We will appreciate this when we actually study the error recovery mechanisms in transaction management module.

Okay, so now let us move on to looking at a single disk. And as I was telling you that we may actually not store you know data only in a single disk. We will see later on how to handle multiple disks and what are the issues that are there in multiple disks and things like that. We will start studying a single disk.

(Refer Slide Time: 11:59)



These are electromechanical devices. I am sure you would have used the external hard drive for backing up and things like that. And in addition to the USB flash drives, we also sometimes if you want a largest array, you will actually buy an external hard drive, right. So if you place a hand on your external hard drive, you can see the vibrations as it is operating. So it is an electromechanical device.

So it has, the structure inside is that there is a fixed you know some kind of a stand kind of thing and then you have a rotating spindle. You have a rotating spindle. To the

rotating spindle, we have lots of platters that have attached, circular plates that are that are permanently fixed or attached. So that when the spindle rotates the plates rotate.

So these plates are also called platters. So there are multiple platters on the spindle and the spindle rotates at a speed of 7 to 10 to 12,000 rotations per minute. You can see how fast it rotates. It keeps spinning. So they are called spinning disk. That is why we are calling it spinning disk. They spin at very fast rate. The current spin rates are something like 12 to 15 RPM rotations per minute.

Now while they okay, so let us first look at the physical structure. So we have this rotating spinning spindle to which lots of platters are attached. And then we have a head assembly which has several read/write heads, several read/write heads. And these read/write heads can actually be node mode so that they can read, they can read some part of the disk, okay.

And the disk material is magnetically coated so that you know you can electrically transform that to store update. And then you can read it that read that you know and then figure out that there is a bit there, a 0 or 1 something. So basically read of course is 0 or 1. Now so the important thing to remember here is that these read/write head, has to be physically moved inside and outside so that it can read a circle like this.

So if it is on this particular circle, it can read off bits from this particular circuit. So we have this concentric circles. So each of these things are actually called a track. This is called a track. So when the read/write head is positioned on a particular track, then it can read off bits on the track. It has a sensor. Now how many tracks are there you will be very surprised that there are actually very densely packed track cell.

Something like 10,000 per inch. So an inch is 2.54 centimeter size. So within 2.54 centimeters we have 10,000 of these tracks. You can see how closely they are positioned. And so each of these this read/write head has to position itself on exactly one of those tracks. So it has to be very precisely controlled and you know stepper motors are used in order to position that on exactly one track.

Now the tracks are optionally sometimes, you know partitioned into what are called sectors. If it is there sectors are hardwired when the device is manufactured. But it is optional to keep sectors. But the track is of course the most important thing. So and now if it has sectors, either the sectors or the entire track itself is divided into blocks. A block is the basic unit of data transfer from the disk, okay.

And these blocks are the ones that are given certain numbers. So there are a number of blocks within each track and those blocks are numbered and you can actually put, you know ask for a particular block number and then that block will be read. Now another interesting thing here is that these things are all fixed to a particular mechanism. So they all move in unison, they move in unison okay.

So it is not the case that this read/write head is on one track and this other read/write head is on another track that is not possible. These are not independently moving read/write heads. They all move in unison, okay. And so at any point of time they are on the i th track say 100th track on all the platters. On all the platters they are on the 100th track, okay. So now you can imagine that this 100th track on all the platters is in some sense forms a virtual cylinder.

It is a cylinder, okay. So within this cylinder, we have so many surfaces. So each so we are on 100th track, let us say. So 100th track on surface one. And then each surface actually has, each platter has two surfaces. The top surface and the bottom surface. So both surfaces are actually used for storing data, okay. So that is why this this has two arrowheads. You can see that it can either read that or read this, okay?

But the most interesting thing is you should remember that at any one point of time, we are only reading one track, one track of one surface. We are not simultaneously reading several tracks. So that means what so these, one of these, one of these read/write heads, there is a read/write head for a per surface, right. So one of these surfaces one of these read/write heads will be electronically chosen.

It will be electronically activated so that information from that particular read/write head starts coming off, okay. You can switch between this track to this track in a jiffy because it is electronic switching. It is a electronic switching. You just switch off this,

this sensor and then switch on this sensor. Then you will start reading from that surface and then stop reading there and then start reading some other surface.

So it is possible to do that. So that is why we will think of the 100th track that is there on all surfaces as some kind of a virtual cylinder. And then we can actually read off several blocks from this virtual cylinder one after the other, okay. Once you are on surface one, so how does the reading actually happen? This only moves this way in a horizontal manner. So suppose it is on 100th track it will stay fixed there.

Now your read/write head is fixed and the platter is spinning. So you basically, if you are looking for the 100th block under particular track, then you basically wait for the 100th block to pass over you in this process of rotation. And once it starts processing over you, you will start picking up bits. That is how data transfer happens. So and the unit of data transfer is one block okay.

And the block, the size of the block ranges from half a kilobyte to something like 8 kilobytes, kilobytes. A kilobyte is 1024 bytes. So something like 512 bytes to 8 kilobytes. That is the kind of typical size. So this is fixed. And this is set at the initialization time for the other formatting of the other disk. And you can dynamically change it. Once a block is fixed, it will be permanent as long as using this other disk.

(Refer Slide Time: 21:22)

Data Transfer from Disk

Address of a block: Surface No, Cylinder No, Block No

Data transfer:

- Move the r/w head to the appropriate track
 - time needed - seek time – ~ 12 to 14 ms
- Wait for the appropriate block to come under r/w head
 - time needed - rotational delay - ~3 to 4ms (avg)

Access time: Seek time + rotational delay

Blocks on the same cylinder - roughly close to each other

- access time-wise
- cylinder i , cylinder $(i + 1)$, cylinder $(i + 2)$ etc.

Now let us look at what is an address of a block. So I told you it is a random accessible device. So it will, the disk controller actually accepts a block address. A

block address consists of the surface number, the cylinder number, and the block number. Surface number can also be thought of as a track number.

You can also call it like that, but it is the cylinder that actually decides cylinder number is the one that decides what I see in the track. So there are the 10,000 tracks here. So there are 10,000 virtual cylinders. So a cylinder number will be given and then within the cylinder which track you are reading because it consists of as many tracks as there are surfaces here. So that is why you need a surface number.

Surface number, cylinder number and within the track what is the block number that you are interested in. So then the data transfer will actually take place. So in order for the data transfer to take place, what you need to do is to move the read/write head to that particular cylinder. So the read/write head might be somewhere here, somewhere here and you are asking for the 100th thing.

So the entire assembly has to be moved to that place, okay. This is what is called the seek time. You have to seek the appropriate cylinder. This is the one that actually takes, this is a physical movement based on stepper motor. So this is the one that actually takes a longer time. So it is about 10 to 12 14 milliseconds for you to move to a appropriate track, right. And also of course, depends on where you are starting.

If you starting at one place and then you know you have to pass over 500 tracks, then it obviously takes longer time to go. And then you basically the switching the surface number is actually used to switch the read/write head appropriately, okay. So you have to, depending on the surface number, you set the read/write head from where you are reading.

Then you basically then after moving to that particular cylinder, you wait for the appropriate block number to pass over the read/write head. And how long does it take to pass over? It will take one rotation, maximum one rotation. So the one rotation is about 3 to 4 milliseconds because it is making 10 to 12,000 rotations per minute per second sorry RPM. RPM it is RP sorry rotations per minute.

So from this RPM that is given, we can calculate how much is the time it takes to do one rotation. So one rotation time is what it can maximum take for you to get the block under your read/write head sensor. So this is what is called rotational delay. So there is a seek time where you know you need to go to the appropriate track and there is a rotational delay where you will wait for the appropriate block to come under a read/write head sensor.

And once it comes there is a transfer time. But the transfer time is so little so you will actually ignore that. The transfer time is when you are transferring this 4000 kilobytes. The block size let us say is 4000 kilobytes then you have to transfer this 4096 bytes from the surface. That happens in a jiffy, so you really do not bother too much. This is compared to this, it will be very small.

So the access time is modelled as seek time plus rotational delay. Now you will notice one thing that now the blocks on a particular cylinder, there is a cylinder, right virtual cylinder, the blocks that are there on that cylinder in some sense, they are all closer to each other from an access point of view, access point, from access time point of view.

From an access time point of view, all these blobs that are sitting on a virtual cylinder or closer to each other compared to another set of blocks that are sitting on the other cylinders. The moment you switch from the 10th cylinder to the 50th cylinder the read/write head has to be moved, okay.

So but as long as you are in the same on the same cylinder, and you are asking for different blocks on the same cylinder, all that you are doing is if necessary, switch the surface numbers, and then wait for the block to come under the read/write head. You do not have to move the read/write head. There is no seek time involved as long as you are reading blocks from the same cylinder, you get that.

So that is why we will say that the blocks on a particular cylinder are in some sense closer to each other from a access time pointer. And so if you want if a file wants, say 500 blocks, then you try to allocate all those 500 blocks on a particular cylinder. Do not put them on one surface. But if you put in a surface then the read/write head has to

keep moving like this on the surface. Instead what we will do is to put them on the cylinder.

Because if you place them on the cylinder, then read/write head will go to that cylinder first and then probably switch between the platters and then pick up. So there are so many say let us say there are 20 some 10 platters, let us say. So you have 10 tracks available. On those 10 tracks, there are so many blocks. So within those blocks you try to put your file, okay. So from an abstract point of view, the blocks on the same cylinder are kind of closer to each other.

So and if you start ordering them on that particular you know access time point of view then the cylinder i blocks are all close to each other. And after that this it is the closest bunch of blocks are on cylinder $i + 1$ then in next cylinder and then and so on the next cylinder. So when you have to pick up or you know ask for a consecutive in some sense a consecutive set of blocks then you will pick blocks from cylinder one, then cylinder two and cylinder three etc., okay.

So from a abstracting the way all these it is actually such a amazing thing that on a small about a one and a half, you know centimeters height we have so many platters and then the platters are rotating at such high speeds and then certain read/write head goes inside these platters and then the probability of this read/write head touching the platter is you know very high actually.

But then it actually is a fantastic piece of engineering that it actually works. And then you know and as it works, as it works, it actually generates small amount of dust particles as it ages it starts generating like this purpose. The whole assembly is actually is hermetically sealed. The whole thing is hermetically sealed. So anything from outside will not enter.

But inside there can be certain particles that are getting generated and so slowly, the air quality changes so to say, okay. So there are I met a colleague in chemical engineering, who spent a huge number of years studying the ecosystem inside the disk. He was working in IBM. So

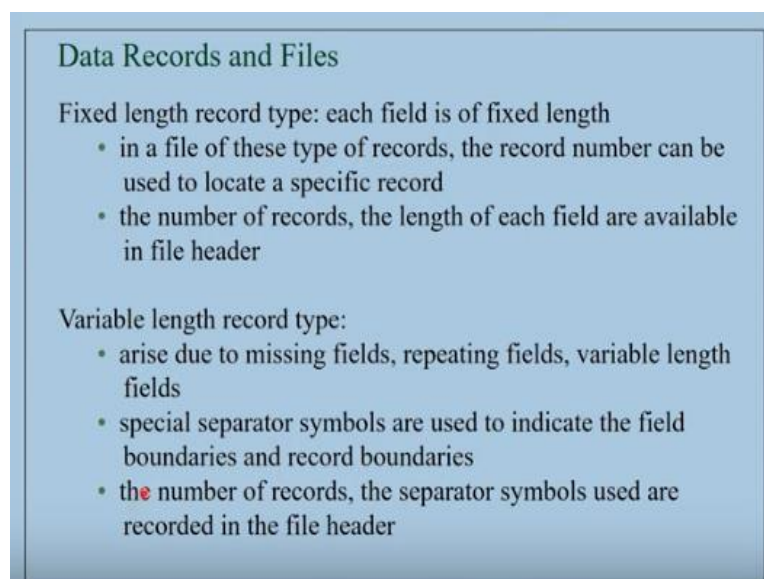
in these disks, how exactly the, you know as the disk gets used, how does the whole inside things you know changes and things like that.

So based on that they will be able to you know tell predictions about what is the mean time to failure, when is it likely to fail. So the failures in hard disks typically happen when the read/write head crashes onto the surface. Once the read/write head crashes onto the surface, then surface crackers will happen and the whole assembly will not work properly. So we will look at the failure probabilities and other things little later.

But right now let us understand that we can abstract out all these things and then assume that this secondary storage is controlled by a disk controller. And what the disk controller will do for you is to take a disk address, and then a buffer address in the memory. And then you block transfer that block of data, which is about say 4 kilobytes onto this buffer in about depending on where the read/write head right now is, things like that.

So it may take on the average 10 to 12 milliseconds for you to get back that 4 kilobytes of data, okay. That is a lot of time ahead for us when the processors are running at 2.5 gigahertz or something like that. So the memory operations are happening so fast milliseconds look like centuries. So that is the main one of the main issues with the forward block.

(Refer Slide Time: 32:06)



Data Records and Files

Fixed length record type: each field is of fixed length

- in a file of these type of records, the record number can be used to locate a specific record
- the number of records, the length of each field are available in file header

Variable length record type:

- arise due to missing fields, repeating fields, variable length fields
- special separator symbols are used to indicate the field boundaries and record boundaries
- the number of records, the separator symbols used are recorded in the file header

Now let us look at how the, so we will all be looking at file organizations. So before we do that, what do we place on files? It is the data records that we place on the files. So and what are the kinds of data records that are possible? They can be either fixed length record types, where each field is a fixed length. Now you know the length of the field. It takes so many 25 bytes okay 25 bytes.

This field takes 25 bytes, next field takes 30 bytes, next field takes 40 bytes, etc. You know the length of the fields. So such kind of thing if you have 10 fields in the record, then you know you can add up the field size as and then you will get the record length. So you know how much exactly for the record or you know takes so many number of bytes of data.

So the advantage with this kind of records is that within the a file is nothing but a sequence of bytes for us. A file is nothing but a sequence of these bytes basically, okay. So within that sequence of bytes you can actually precisely locate as to where your record starts, because you can calculate how many you know bytes to move before you can pick up your record. Because all records are of fixed length.

So if each record is taking 100 bytes, then you know that the fifth record starts on (0) (33:43) right. So you can use the record number, suppose you are given the record number you can precisely calculate which byte it starts from and then start reading from that byte onwards. Now these number of records that are there and the length of each of these fields all this information has to be stored in the file header.

A lot of information gets stored in file header. So when you are doing a file management, you will first read the file header and then figure out the meta information about how information is organized in the file and then start operating that file, okay. So as against this fixed length records, we have variable length record files also. We are looking at basically just data records, we have fixed length records and variable length records.

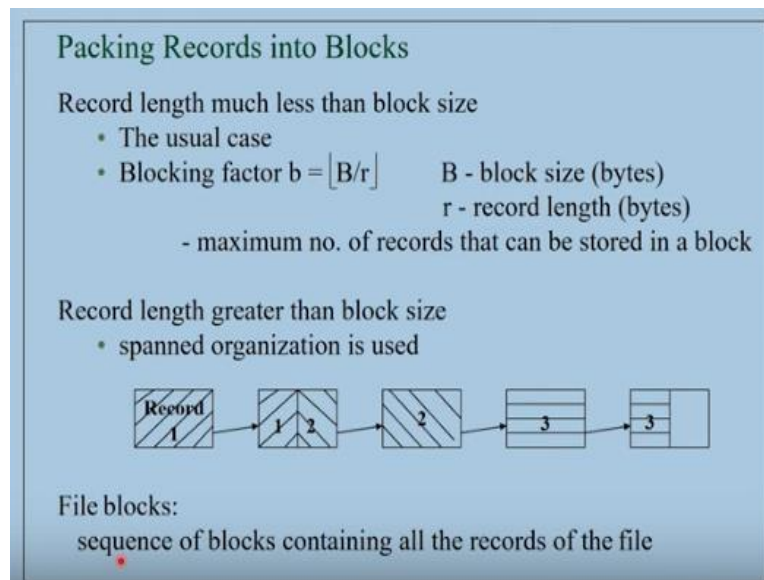
The variable length records they do arise because there can be some missing fields and there can be some repeating fields in a particular file. And there can be variable length, you know a field is naturally modeled as a variable length value. So if this is

the case then we need to put some special separator symbols to indicate as to what are the field boundaries, where one field has ended and where another field is starting.

So if you imagine a record as a sequence of bytes, then if you do not have a fixed length records, obviously fixed length fields obviously then when a field ends, you should know when it ended. So you need some end markers for the fields. So you need separator symbols to indicate when the field and then you also now require separate separator symbols to figure out when the record has ended, okay.

So record and what are these separator symbols that are used to indicate the field you know field separation and the record separation, that also has to be recorded in the file header. And how many number of records are there? What are the separator symbols used etc. all this information thing has to be there in the file header, okay.

(Refer Slide Time: 36:08)



Now, so now records are sequence of this sequences of these fields. Fields are nothing but sequence of bytes and so you have records. A sequence of these records is what is a file is, right and so now you have to organize this information on to on the disk. So since the unit of transfer from the non-volatile storage for the main memory is at this so we will be packing these records into these into blocks into units which are of block size.

So that these units can be actually stored as blocks on the disk, okay. So there are two cases. If the record length is much less than the block size, which is usually the case

in Enterprise Information Systems, where you typically have employee records, student records and you know things like that. So you do not have you know photographs you do not have music, you know things like that.

Then record lengths are likely to be less than the block size. And so in this context, we define what is called a blocking factor. The blocking factor is the maximum number of records of a particular file that can be stored in a block. Since the block size is fixed for the best, so what is the and the record length is available to us so you can divide the block size by the record length, and then we usually take a policy of not to store a fraction of the record into a block.

So you usually store the whole of record into a block. And so you will take the floor of this number, block size divided by the record length and then you get a number. So that number is the maximum number of records that you can store in a block. So you will pack records into block size units. And then you put integral records into this thing. You do not put a fraction of a record into this units.

And now you have your file, which is a sequence of records is divided into these units, each of them or block size. Now you are going to map these store these sequence of blocks, file blocks, they are called file blocks onto the disk, okay. So that is how we will organize the file onto the record, onto the disk. Now the record length could be greater than the block size if you are actually using some media elements and things like that.

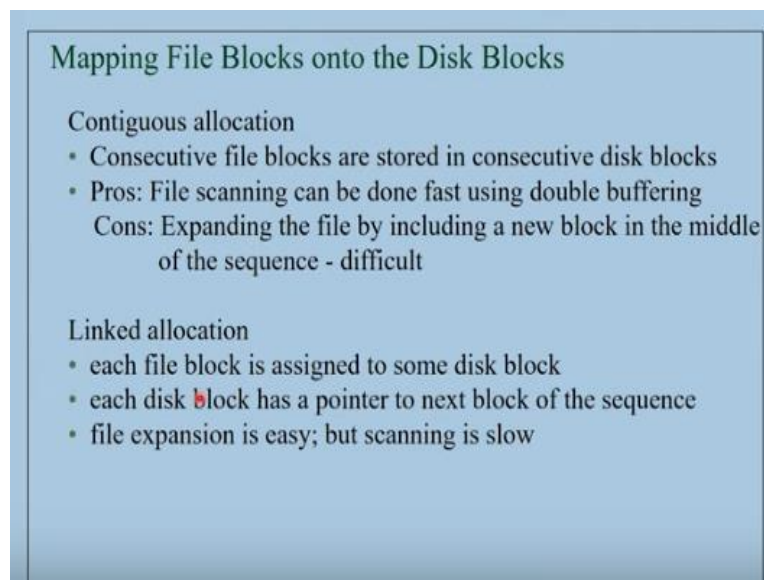
In which case of course, it will be something like this. Yes, what is called a spanned organization has to be used. That means, you know record one starts in one block and then it stops somewhere in the middle of the second block and then the second record starts there and then and it goes on and so on. So you have to use extra information to figure out when you know how many blocks are being used per record, and where a block is ending etc., where a record is ending etc. Okay.

So the usual case is that record length is much less than the block size. So we will, in most of our considerations, we will assume this case where a spanned organization is not needed. We will use the we will assume that an integral number of records are

placed in a block and the blocking factor indicates the maximum number of records. Now once you prepare, once you imagine that a file which is a sequence of records is broken into these units, so you have file blocks.

These file blocks have to be stored on the disk. So you will map these file blocks which is a sequence of blocks containing all the records of the file onto the disk blocks.

(Refer Slide Time: 40:23)



Mapping File Blocks onto the Disk Blocks

Contiguous allocation

- Consecutive file blocks are stored in consecutive disk blocks
- Pros: File scanning can be done fast using double buffering
- Cons: Expanding the file by including a new block in the middle of the sequence - difficult

Linked allocation

- each file block is assigned to some disk block
- each disk block has a pointer to next block of the sequence
- file expansion is easy; but scanning is slow

Now what are the various ways you can map these file blocks under the disk blocks? It is possible for you to keep the first block of the file on some disk block, place the second one on some other disk block etc., right. So you can do that. As against that, you can do what is called continuous allocation. That means consecutive file blocks are stored in consecutive disk blocks. What do you mean by consecutive disk blocks?

The consecutive disk blocks are the disk blocks that are lying on the same cylinder, okay. Blocks on the same cylinder are thought of as consecutive blocks because they are in some sense consecutive from an access standpoint of view, okay. So if your file has 5000 blocks, then you will try to place them on cylinder 5 and cylinder 6 cylinder 7 like that, depending on how many blocks are there available in each cylinder.

Now if you do this the scanning of the file is very fast because once you start reading from a cylinder you can read some say a large number of blocks from that particular cylinder itself of the file, okay. So reading will be very fast. And we can also use what

is called double buffering technique in order to you know make use of this fast file scanning.

That means, you can ask the disk controller to start filling one buffer and then you know while it is filling this buffer, you can actually you know use a previous buffer that was given to the disk block earlier, disk control earlier to fill the previous block. So while you are using that block, the next block will be filled by the disk controller. So you can use some buffers and then.

But the disadvantage of this kind of a contiguous allocation is that in case you have to expand your file, expand your file and then insert some new block in the middle of that sequence. There is some kind of a sequence in which you have organized your records of a file. So for some reason, let us say you want to introduce a new block into the file.

Then you know you are in trouble because you have put this file blocks on contiguous disk blocks and if you want to maintain this policy then you have to read off a lot of disk blocks and then again write them back after inserting this new block. So expanding the file will be a little difficult while keeping this thing. So as against this there is what is called a linked allocation.

In a linked allocation, each of these file blocks is stored on some disk block. And then so the file block 1 is stored on some disk block. File block 2 is stored on some other disk block. Then there is a pointer from the first disk block to the second disk block, saying that the next file block is actually stored there. So these pointers are not memory obviously not memory pointers they are disk pointers.

A disk pointer is nothing but a disk address. So you store the file, first block of the file onto some disk block and then store a pointer to tell in that block to tell as to where the next file block is located. What is the disk address of the block where the next file block is, okay.