

**Database Management System**  
**Prof. Partha Pratim Das**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 25**  
**Storage and File Structure : File Structure**

Welcome to module 25 of Database Management Systems. We have been discussing about storage and file structure.

(Refer Slide Time: 00:22)

PPD

## Module Recap

- Overview of Physical Storage Media
- Magnetic Disks
- RAID
- Tertiary Storage

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 6th Edition      25.2      ©Silberschatz, Korth and Sudarshan

In the last module we have talked about different storage options.

(Refer Slide Time: 00:27)

PPD

## Module Objectives

- To familiarize with the organization for database files
- To understand how records and relations are organized in files
- To learn how databases keep their own information in Data-Dictionary Storage – the metadata database of a database
- To understand the mechanisms for fast access of a database store

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P P Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.3 ©Silberschatz, Korth and Sudarshan

And in this one we will talk about the; organization of database files, what should be the typical structure to store the records in the files. And how the overall database which manage itself we will talk about those issues.

(Refer Slide Time: 00:41)

PPD

## Module Outline

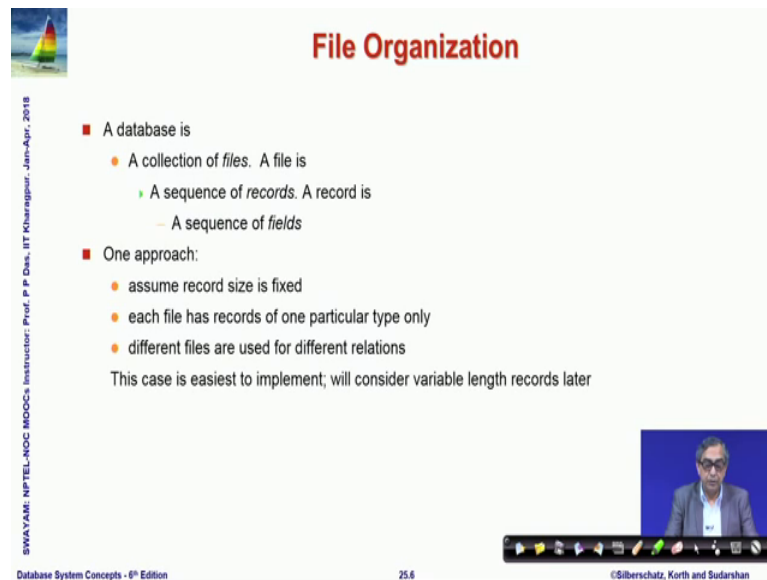
- File Organization
- Organization of Records in Files
- Data-Dictionary Storage
- Storage Access

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P P Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.4 ©Silberschatz, Korth and Sudarshan

So, the file organization; so if you look at a database; what is the database? It is a collection of relations.

(Refer Slide Time: 00:43)



**File Organization**

- A database is
  - A collection of *files*. A file is
    - A sequence of *records*. A record is
      - A sequence of *fields*
- One approach:
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations

This case is easiest to implement; will consider variable length records later

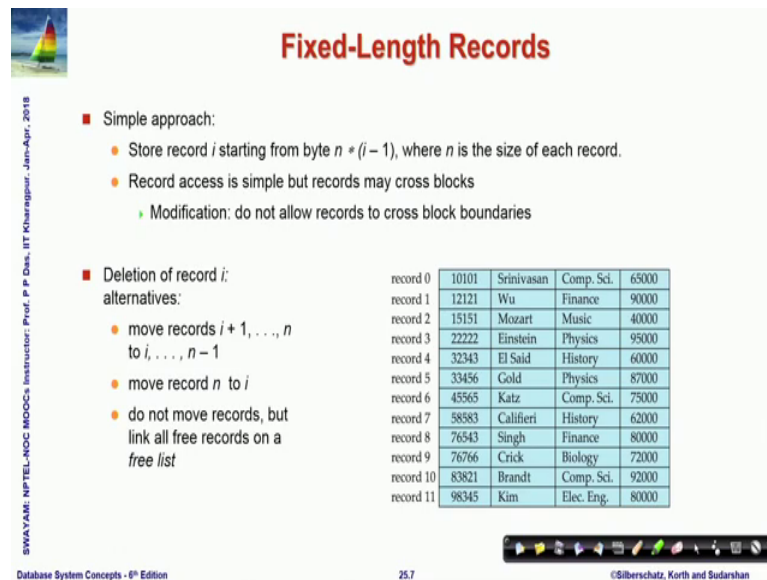
SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 8<sup>th</sup> Edition 25.6 ©Silberschatz, Korth and Sudarshan

So, it is a collection of files every relation is a file. Now, what is a file? A file is a sequence of records, and what is a record? It is a sequence of fields. So, this is the hierarchy that exists and this will have to be kept in mind, when we design the organization of how we keep this data.

Now, one starting approach could be we can assume that all records are of fixed size which makes a life easier and each file has records of only one type again a simplifying assumption and different files are used for different relations. So, this is a easiest case to implement.

(Refer Slide Time: 01:30)



**Fixed-Length Records**

- Simple approach:
  - Store record  $i$  starting from byte  $n * (i - 1)$ , where  $n$  is the size of each record.
  - Record access is simple but records may cross blocks
    - Modification: do not allow records to cross block boundaries
- Deletion of record  $i$ : alternatives:
  - move records  $i + 1, \dots, n$  to  $i, \dots, n - 1$
  - move record  $n$  to  $i$
  - do not move records, but link all free records on a *free list*

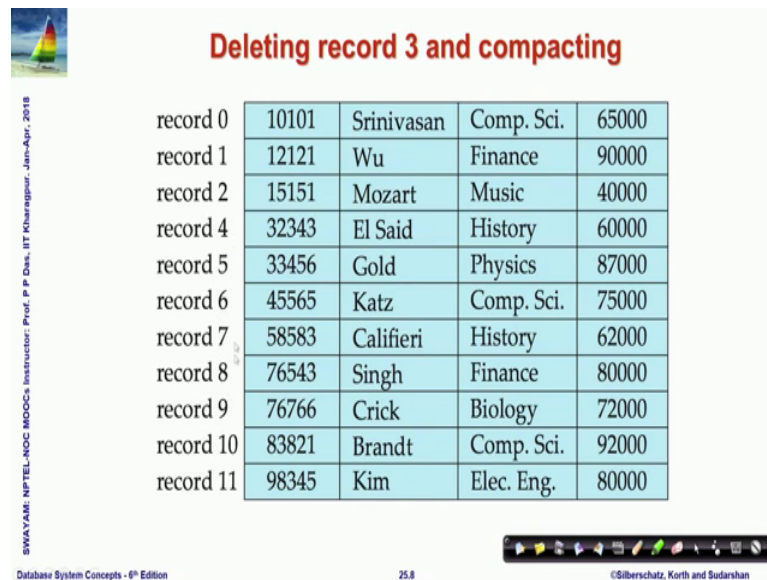
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califeri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.7 ©Silberschatz, Korth and Sudarshan

So, we will start with that; so this is what we have we store these are fixed size records. So, we store them one after the other and based on the fixed size, we can easily know what is the starting address of any record and we can access it accordingly. Now, if a record is deleted, then there are several things that I can do see that this is a different alternatives, that is if I record delete record I then. So, if we delete any record then we can actually move the records. So, that we consume that space or we can take the last record and move it there or we can simply do not do any move, but use an some additional pointers to denote that these records have become free rather give it to a free list.

(Refer Slide Time: 02:22)



### Deleting record 3 and compacting

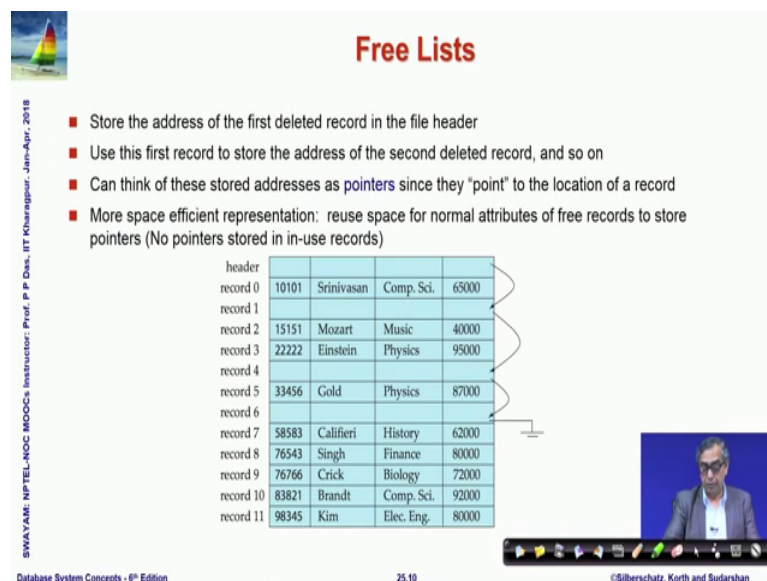
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.8 ©Silberschatz, Korth and Sudarshan

So, these are the three main strategies. So, here we showing the first one the record three has been removed. So, all records have moved up in this it is we have move the last record 11 in the place of record 3. So, record 3 is gone, but still the whole thing reminds compact only the point that must be noted that in the earlier one, where well we moved everything then the ordering that existed here of this key of this key field is maintained, but if we move the last record the naturally that ordering has got destroyed. So, it will have implications in terms of indexed organization that will cover in the next modules.

(Refer Slide Time: 03:08)



### Free Lists

- Store the address of the first deleted record in the file header
- Use this first record to store the address of the second deleted record, and so on
- Can think of these stored addresses as pointers since they "point" to the location of a record
- More space efficient representation: reuse space for normal attributes of free records to store pointers (No pointers stored in in-use records)

header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.10 ©Silberschatz, Korth and Sudarshan

The third option could be use a free list, which is a nice one because you would you do not neither here neither you destroy the order that existed and no one have to really move records which is expensive, but you just start with a pointer and keep on pointing to the empty records.

And once you delete it you use that space itself to point to the next deleted record. So, whenever you have to you know delete a record all that you need to do is adjust this point. So, which is pretty fast and quite efficient way of getting this linked together in terms of; so there is as such no space over it and it is a fastest possible that you can do now in contest to fix length record.

(Refer Slide Time: 03:54)

**Variable-Length Records**

- Variable-length records arise in database systems in several ways:
  - Storage of multiple record types in a file
  - Record types that allow variable lengths for one or more fields such as strings (**varchar**)
  - Record types that allow repeating fields (used in some older data models)
- Attributes are stored in order
- Variable length attributes represented by fixed size (offset, length), with actual data stored after all fixed length attributes
- Null values represented by null-value bitmap

Null bitmap (stored in 1 byte)  
0000

21, 5	26, 10	36, 10	65000	10101	Srinivasan	Comp. Sci.	
Bytes 0	4	8	12	20 21	26	36	45

Database System Concepts - 8th Edition  
25.11  
©Silberschatz, Korth and Sudarshan

If the record becomes variable length, then certainly every record may of very different size and it is very common for example, we have types like varchar a lot of strings are varchar. So, we just we do not know how much it will take. So, the typical way you represent that is a you represent as to; what is a starting pointer of a particular of the actual value and the size of the value the number of bytes it will take. So, when we say twenty one five which we mean that this field will actually start from location 21 and we will have 5 locations 5 bytes, then the next 1 is 26, 10.

So, this will start to 26 and go for 10 such; so what happens is; if you look into this part of the data, then that part is actually for all practical purposes the fix length 1, because here you are just keeping double x for the variable length data or you have some field

which is a fixed length data anyway or you have a null which is stored in one byte, and then you have all the variable stuff at one end. So, you can actually make part of this fixed length by using this kind of encoding. So, this is what is explained here.

(Refer Slide Time: 05:22)

**Variable-Length Records: Slotted Page Structure**

The diagram shows a page layout with a **Block Header** and **Records**. The header contains:
 

- # Entries
- Free Space
- End of Free Space

 Arrows indicate that the header points to the records and the end of free space.

- **Slotted page** header contains:
  - number of record entries
  - end of free space in the block
  - location and size of each record
- Records can be moved around within a page to keep them contiguous with no empty space between them; entry in the header must be updated
- Pointers should not point directly to record — instead they should point to the entry for the record in header

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018  
 Database System Concepts - 9<sup>th</sup> Edition 25.12 ©Silberschatz, Korth and Sudarshan

So, for variable length records a one main issue is if you if you keep it like this, then since you are using actually you are using pointers here we saying that this data actually is on 21. So, what will happen is if you change the position of the record if you relocate the record, then all these references will have to be updated. So, that becomes a slotted thing. So, what the slotted page structure does is it does a [li/little] little bit of adjustment it ports a puts a records here at the at the end.

And it has a header it has a. So, it has a block header as in here and the block header has actually pointers to the records and then you have a an entry which points to the end of the free space where more records can still be stored. So, when you refer to a particular record you do not actually refer here. So, you do not refer here, but you refer here. So, what you maintain is the header is actually not changed, but if there are relocations required adjustments required, then that will be done with respect to this. And so, this value will change, but any references made to this location will remain invariant. So, that is the basic idea of the slotted page structure, which can allow you to have the variable length record with easy re locatability in the design.

Now, let us see the given this what is the organization of the records in the file.

(Refer Slide Time: 07:05)

**Organization of Records in Files**

- **Heap** – a record can be placed anywhere in the file where there is space
- **Sequential** – store records in sequential order, based on the value of the search key of each record
- **Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed
- Records of each relation may be stored in a separate file. In a **multitable clustering file organization** records of several different relations can be stored in the same file
  - Motivation: store related records on the same block to minimize I/O

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-April, 2018

Database System Concepts - 9th Edition

25.14

©Silberschatz, Korth and Sudarshan

So, there are different organizations that have been tried out the simplest is the heap is a record can be placed anywhere in the file where there is space. And you can link to that that is that is one way certainly there is nothing very smart in terms of doing that, but you can you would possibly like to do better than that. So, one is you can store the; records in a sequential manner let us store records in a sequential order in terms of certain search key.

So, based on the value of the search key you put them in the sequential orders. So, what it will mean that it will become easier to search the records in that way, but it has consequences or you can hash you can use a hash function on a some of the attributes of the record and the results specified on which block which disk block the record will be placed. So, these are the different option and a records of which relation may be stored in a separate file that is a basic convention, but in some cases there could be multi table clustering as well.

So, let us quickly take a look at these options.



(Refer Slide Time: 08:16)

**Sequential File Organization**

- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key

10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P P Das, IIT Kharagpur, Jan-Apr, 2018

So, these sequential file organizations. So, these things are kept sequentially here as you can see there all consequentially here and this is the link key of those.

(Refer Slide Time: 08:34)

**Sequential File Organization (Cont.)**

- Deletion – use pointer chains
- Insertion –locate the position where the record is to be inserted
  - if there is free space insert there
  - if no free space, insert the record in an overflow block
  - In either case, pointer chain must be updated
- Need to reorganize the file from time to time to restore sequential order

10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

32222

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P P Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 6<sup>th</sup> Edition 25.16 ©Silberschatz, Korth and Sudarshan

So, this is the issues of deletion and if you delete you use pointer chains. As you have we have discussed earlier, and if you have to insert then you look for a free space, if you find a free space you can put it there you insert it there if there is no free space then you have to use a overflow block, where you can go and place that separately as the dilemma

shows here; in a multi table clustering what you would do is more than one relation could be kept in the same file.

(Refer Slide Time: 09:00)

**Multitable Clustering File Organization**

Store several relations in one file using a **multitable clustering** file organization

*department*

dept_name	building	budget
Comp. Sci.	Taylor	100000
Physics	Watson	70000

*instructor*

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

*multitable clustering of department and instructor*

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

SWAYAM: NPTEL-NOC MOCs- Instructor: Prof. P P Das, IIT Kharagpur, Jan-Apr, 2018

For example, here we are showing two relations department name building and budget these attributes doing department and instructor, id name, department name, and salary is a other instructor in a way keeping them together here naturally, where we keep them together.

For example here in we have one which is here in we have one, which is and entry of record from the department relation. Similarly here is another which is from the department relation whereas, these are entries from the instructor relation; please note that since we are doing it multi table with a department we do not need to keep these information in as a part of the record, but what you mean is if there is a computer science entry here. Then all those records which follow this computer science entry are actually instructors in the computer science till I actually come across another departments entry where which will be followed by instructors for that department. So, that is a basic multi table convention that is to be followed here.

(Refer Slide Time: 10:26)

**Multitable Clustering File Organization (cont.)**

- good for queries involving *department*  $\bowtie$  *instructor*, and for queries involving one single department and its instructors
- bad for queries involving only *department*
- results in variable size records
- Can add pointer chains to link records of a particular relation

Comp. Sci.	Taylor	100000	
45564	Katz	75000	
10101	Srinivasan	65000	
83821	Brandt	92000	
Physics	Watson	70000	
33456	Gold	87000	

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 9<sup>th</sup> Edition

25.18

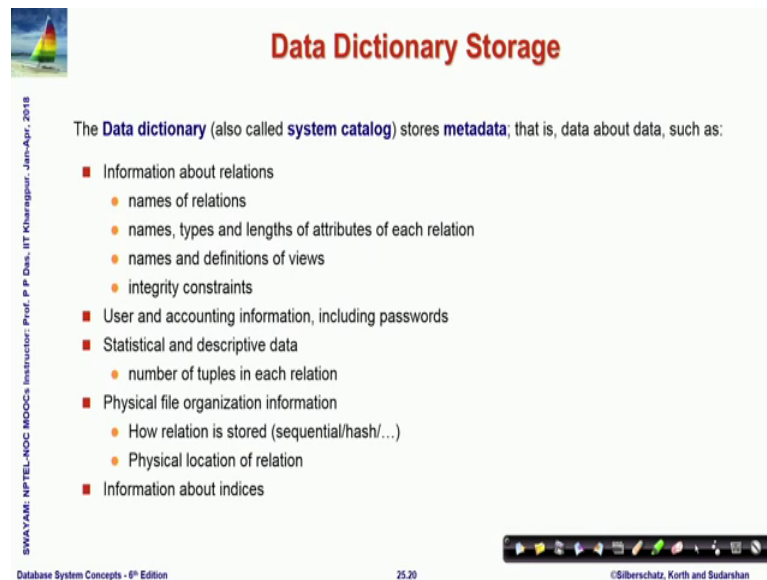
©Silberschatz, Korth and Sudarshan

Now, it is actually good for queries that involved joining department with instructor, because based on the value of the department you have the instructors club together and they could be very easily quickly taken together and it is also good for single queries with departments and it is instructors, because as you can see you can if you want to know for example, who are the faculty for at computer science department; then it be very easy to answer that, because you need to search for computer science and then you know all the list of the faculty will be in consecutive block.

So, you can easily lift that, but certainly this is not true, if you want to involve queries which have department only; because that department information are all now partially distributed. So, if your query has the department based information to be to be accumulated, and then this may not be a good option. So, that will result in then you can have supporting pointer chains to actually link the department information. So, this is a one kind of a design that you have ok.

Now think about; so the whole so, we have; so, far talked about the relations and relations going to either single files or multi table relations; multi table file where multiple relations are on the same file. Now, if you look at the database as a whole. So, what is a data base?

(Refer Slide Time: 12:08)



**Data Dictionary Storage**

The **Data dictionary** (also called **system catalog**) stores **metadata**, that is, data about data, such as:

- Information about relations
  - names of relations
  - names, types and lengths of attributes of each relation
  - names and definitions of views
  - integrity constraints
- User and accounting information, including passwords
- Statistical and descriptive data
  - number of tuples in each relation
- Physical file organization information
  - How relation is stored (sequential/hash/...)
  - Physical location of relation
- Information about indices

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 8<sup>th</sup> Edition 25.20 ©Silberschatz, Korth and Sudarshan

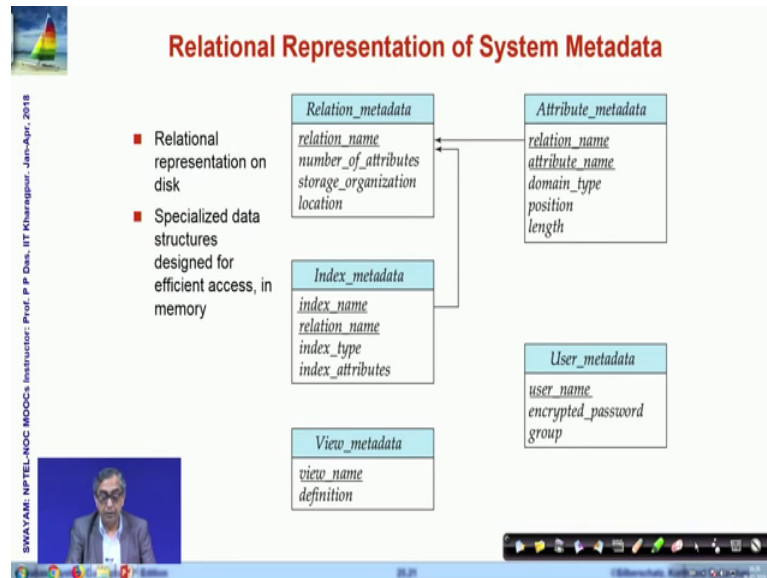
The database as a whole has a whole lot of tables; and so far we have just been focus on the fact that tables we know the tables we know their attributes and the data resides in inside, but if you think in terms of the database, then somebody; somewhere we will need to remember that what are my tables? What are my relations? For a given relation I need to know what are the attributes that the relation has, what is the; you know length type of this attributes I did remember what are the views that I have created on the database the constraints that exist.

So, all of these information which you can say is databases own metadata information needs to be also maintained; and what is done is that; also is maintained as a database within the database system. And such a metadata system is usually known as the name of data dictionary or system catalogues.

So, it has informations like this. So, you put them again, you create the schema design based on the all this metadata information that you need, also you can have you will need to maintain information for users, accounts, passwords and so, on. Then you may have statistical information, where you would like to; we will see the use of statistical information when we talk about indexing in the following week you will see that you need to know, what is the you know how frequently the different queries are fired, what are the number of peoples in each relation and so, on; you may also need to have

information in terms of what has been your choices in terms of the physical locations of file the storage options and so on the index files.

(Refer Slide Time: 14:05)

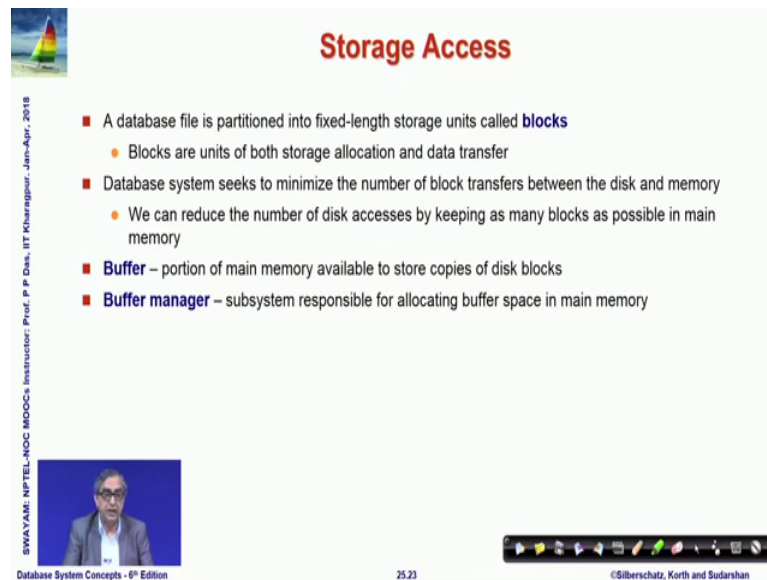


So, here is a sample one. So, what if you look into; so you can again see a number of schema. So, this is saying that the; this is the relational metadata schema which is talking about, what is the different relations? So, every record here is not keeping the data of your application, but its keeping the information that here you have these different relations. For example, couple of modules back we are talking about the library information system.

So, in the library information system we had different we designed different relations the book issue, the book catalogue, the book copies and so, on. So, those relation names and the how many attributes they have? How will you organize the storage, where is that storage will go to this particular table? Then depending on the kind of index that you are creating we have still not discussed about index we will do subsequently; but those index information can be preserved the view information we can have attribute metadata.

So, it is for the relation name what is attribute name and what is the type of that attributes. So, if the relation name is say book catalogue, then the attribute name is title, then what is the domain type. So, we will say this is a varchar; then the position of that attribute, the length if it is given the user metadata all of this are typical things that will go into this system catalogue or the data dictionary that we will require.

(Refer Slide Time: 15:47)



**Storage Access**

- A database file is partitioned into fixed-length storage units called **blocks**
  - Blocks are units of both storage allocation and data transfer
- Database system seeks to minimize the number of block transfers between the disk and memory
  - We can reduce the number of disk accesses by keeping as many blocks as possible in main memory
- **Buffer** – portion of main memory available to store copies of disk blocks
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. Das, IIT Khargapur, Jan-April, 2018

Database System Concepts - 9th Edition

25/23

©Silberschatz, Korth and Sudarshan

So, finally, the access to the storage the database file as I have been repeatedly saying is partition into fix length units called blocks, because blocks are defined; so that they can be easily allocated and transfer and they are the fastest unit of data that can be transferred between the disk and the memory.

So, unlike many of our typical algorithmic considerations; so when we talk about different algorithms, what we try to minimize? We try to minimize certain expensive operations in the CPU; say the comparison operation or the assignment or may be the memory read operation. But in terms of database systems block is a basic unit of data transfer and the data transfer two and from the disk is the most time taking factor much takes much larger time compare to any in memory operation that we do. So, this kind of becomes the primary unit of cost that we want to minimize.

So, normally we will see that as we talk about index saying and other different kinds of mechanisms, our primary target is to minimize the number of block transfers. So, certainly we can do that by; can reduce the number of disk access by keeping as many blocks as possible in the main memory. So, we can how can you minimize that transfer, if we can keep more of the blocks in our main memory and naturally of course, there is a limitation because main memory is much smaller. So, often we make use of different buffers.

So, a portion of the main memory will be kept to store copies of the disk block. So, every time you need a block you may not want to need to bring it from the; disk storage. So, you keep it in the buffer in main memory, and then you have a management strategy to manage this buffer. So, whenever you want to actually access a record which should be in a particular block; you check whether that block is already available in the buffer; if it is available in the buffer it use that if it is not available in the buffer, then you take it from the disk you will need quite a bit of cycles for that and as you get that from the disk then you keep a copy in the buffer so that it can be used in future.

(Refer Slide Time: 18:33)

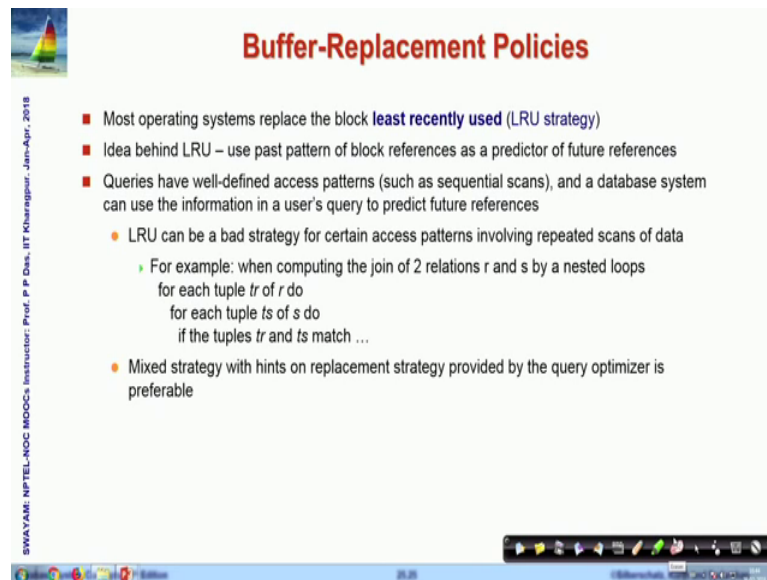
**Buffer Manager**

- Programs call on the buffer manager when they need a block from disk.
  1. If the block is already in the buffer, buffer manager returns the address of the block in main memory
  2. If the block is not in the buffer, the buffer manager
    1. Allocates space in the buffer for the block
      1. Replacing (throwing out) some other block, if required, to make space for the new block
      2. Replaced block written back to disk only if it was modified since the most recent time that it was written to/fetched from the disk
    2. Reads the block from the disk to the buffer, and returns the address of the block in main memory to requester

Now as you keep on doing that, naturally soon you will run out of the buffer memory. So, you will come to a situation, where we need to bring a block from the disk to the memory, but the buffer does not have enough space. So, then we will have to create replace some of the blocks and create space for that. So, here is a basic buffer management strategy.

So, as I said if we if we start if the block is already there in the buffer, then that is given out if the block is not there in the buffer the buffer manager will need to allocate some space how do you allocate space by throwing out some other block which is not required or replace the; then replace the block return back to disk and if it was modified and make space free and then read from the disk and keep a copy in the buffer. So, that is a simple strategy as you can see.

(Refer Slide Time: 19:36)



### Buffer-Replacement Policies

- Most operating systems replace the block **least recently used** (LRU strategy)
- Idea behind LRU – use past pattern of block references as a predictor of future references
- Queries have well-defined access patterns (such as sequential scans), and a database system can use the information in a user's query to predict future references
  - LRU can be a bad strategy for certain access patterns involving repeated scans of data
    - For example: when computing the join of 2 relations  $r$  and  $s$  by a nested loops
      - for each tuple  $tr$  of  $r$  do
      - for each tuple  $ts$  of  $s$  do
      - if the tuples  $tr$  and  $ts$  match ...
  - Mixed strategy with hints on replacement strategy provided by the query optimizer is preferable

Now, certainly when you have to replace the block in the buffer, then the question is which block would you replace. Now if you recall from your from similar situations in the in the operating system in terms of memory management, you have read about different strategies for doing replacement and one of the very common strategy more often used is the LRU strategy of the least recently used strategy. So, the idea of behind LRU is use the pass pattern of block references as to predict the future. So, least recently used is if this is not been used in the recent past. So, it has less likely hood of being used in the future.

Now, to queries; now here we are trying to do the similar thing in terms of queries. So, they have a well defined access pattern and database system can make use of that and as it turns out LRU can be a bad strategy for example, often you are doing computations in terms of such what you say such nested loops.

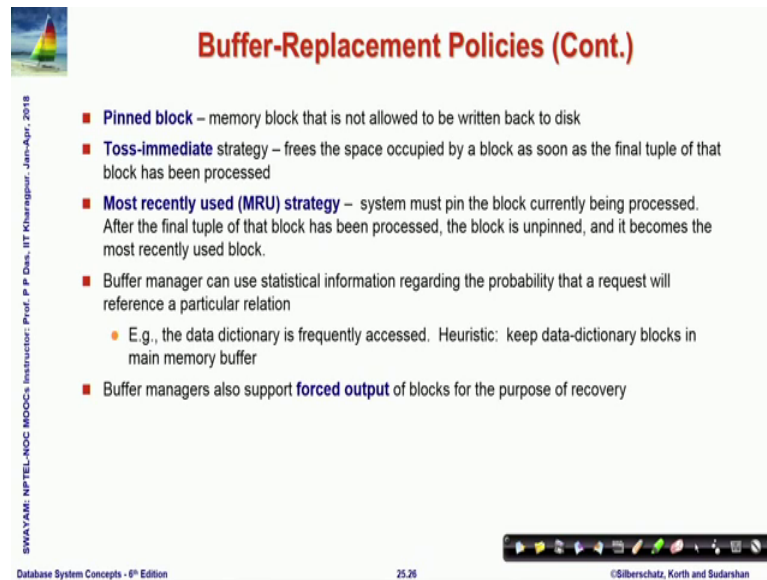
So, you have for each tuple you do this; so you have basically trying to do a join. So, you have two relations and you are trying to do a join. So, when you do this when you are going through the inner loop, there will be lot of transfers that will happen; and the original block where you have been holding this is here and you are not accessing that for quite some time.

So, while you are doing this if you if this block, which was which was holding  $r$  at that time if that turns out to be a LRU; then you will throw it away, but with that when you



complete this loop and come back here, you will again have to read it from the disk. So, this is not LRU for such nested computations may not be a good strategy, so may be some kind of mixed strategy would work better.

(Refer Slide Time: 21:50)



**Buffer-Replacement Policies (Cont.)**

- **Pinned block** – memory block that is not allowed to be written back to disk
- **Toss-immediate** strategy – frees the space occupied by a block as soon as the final tuple of that block has been processed
- **Most recently used (MRU) strategy** – system must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the most recently used block.
- Buffer manager can use statistical information regarding the probability that a request will reference a particular relation
  - E.g., the data dictionary is frequently accessed. Heuristic: keep data-dictionary blocks in main memory buffer
- Buffer managers also support **forced output** of blocks for the purpose of recovery

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 25.26 ©Silberschatz, Korth and Sudarshan

So, there are several that are used in terms of buffer replacement one is called pin block, where you mark a certain memory block which is not allow to be return back to the disk it has to stay in the buffer or a toss immediate strategy is quite often used. So, it frees the space occupied by a block; as soon as the final tuple has been return.

So, it is a toss immediate. So, as soon as you are done you just throw it out you are you are done with it so you write it back. Another which is commonly uses most recently use the; if whenever the block is currently being processed, then the system will kind of keep a marker a pin. So, that it is not removed, but after the final tuple has been processed, the block will be unpinned and then it becomes a most recently used block and you can go with defining the most recently used block and having the strategy.

(Refer Slide Time: 23:04).

**Module Summary**

- Familiarized with the organization for database files
- Understood how records and relations are organized in files
- Learnt how databases keep their own information in Data-Dictionary Storage – the metadata database of a database
- Understood the mechanisms for fast access of a database store

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 9th Edition

25/27

©Silberschatz, Korth and Sudarshan

You can certainly use different kinds of statistical information and in summary; so on this we have talked about the basic organization of database files starting from the fixed record to variable record handling of the different file organization and try to take a look in terms of how records and relations are organized in terms of the in terms of the files and what are the options that we have and we took a look at the data dictionary storage the basic system catalogue, where database keeps its own information.

And then noted that block happens to be the major unit of data transfer between the disk and the main memory and therefore, that is the unit of defining unit of cost that we have to incur, and to minimize that we have a buffering strategy in the main memory, where the disk blocks will be kept a few disk blocks will be kept for quick use whenever required. And there needs to be various different smart replacements strategies for good management of this buffer.