Files and File Structures

- **We looked at the three levels of the database system architecture earlier**
- **At the logical level, a database is seen as a collection of tables, objects, trees, or networks**

  Database application system developers and users are not burdened with the low-level details of data storage.

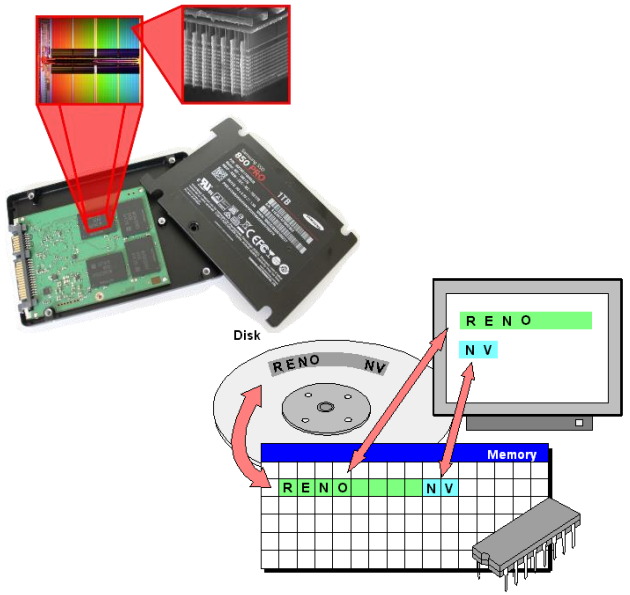
  But how data are stored and retrieved can seriously affect the performance of a database system.
- **The basic problem in physical database representation is storing a file consisting of records**
- **Each record usually has data fields formatted in certain order**
- **Data are transferred between disk and main memory in blocks**

  Main memory – RAM

  Cache memory – high speed memory, managed by the operating system (O/S)

  Disk memory – direct access storage

  Sequential storage – cylinders, drums, etc.



- **A block is a contiguous sequence of bytes that can range from 512 bytes to several kilobytes**

Files and File Structures (Cont.)

- **A file can be several blocks big or smaller than a block**

  But data are transferred between disk and main memory in blocks.

  Therefore, how data are stored and accessed can seriously affect the performance of a data base system.

- **Even though blocks are of a fixed size, record sizes may vary**

- **For the purpose of exploration, we can consider two types of records**

  Fixed length records

  Variable length records

- **Fixed length records – all records are of the same length (that is, the same number of bytes)**

  For example, a record may contain 4 data fields for maintaining deposit records for a bank.

  Branch Name: character 20
  Account Number: integer 8
  Customer Name: character 20
  Balance: real 4 **Total of = 52 bytes**

Files and File Structures (Cont.)

	Branch Name	Account Number	Customer Name	Balance
0	DeKalb	510	Thomas	400.00
1	Sycamore	400	Jones	350.00
2	Rochelle	750	Chen	700.00
3	Rockford	900	Green	500.00
4	DeKalb	511	Abdul	900.00
5	Rochelle	755	Alvarez	750.00
6	Sycamore	406	Huang	600.00

- **Note: Assume the records (i.e. Accounts) are ordered within a branch according to ascending order of account numbers. For example, Account 510 in DeKalb comes before Account 511.**

- **What are the problems with this approach?**

  1. **It is difficult to delete a record from this structure. The space occupied by the record to be deleted must be filled with some other record or there must be a way for marking deleted records so that they can be recognized as deleted.**

  2. **Unless the block size happens to be a multiple of 52 bytes, some records will cross block boundaries. That is, part of a record will be stored in one block and the rest in another block. It would thus require two block accesses to read or write such a record.**

  3. **Maintaining the order of the records is difficult. When a record within a sequence is deleted, other records in the sequence may have to be reordered.**

1. Files and File Structures (Cont.)

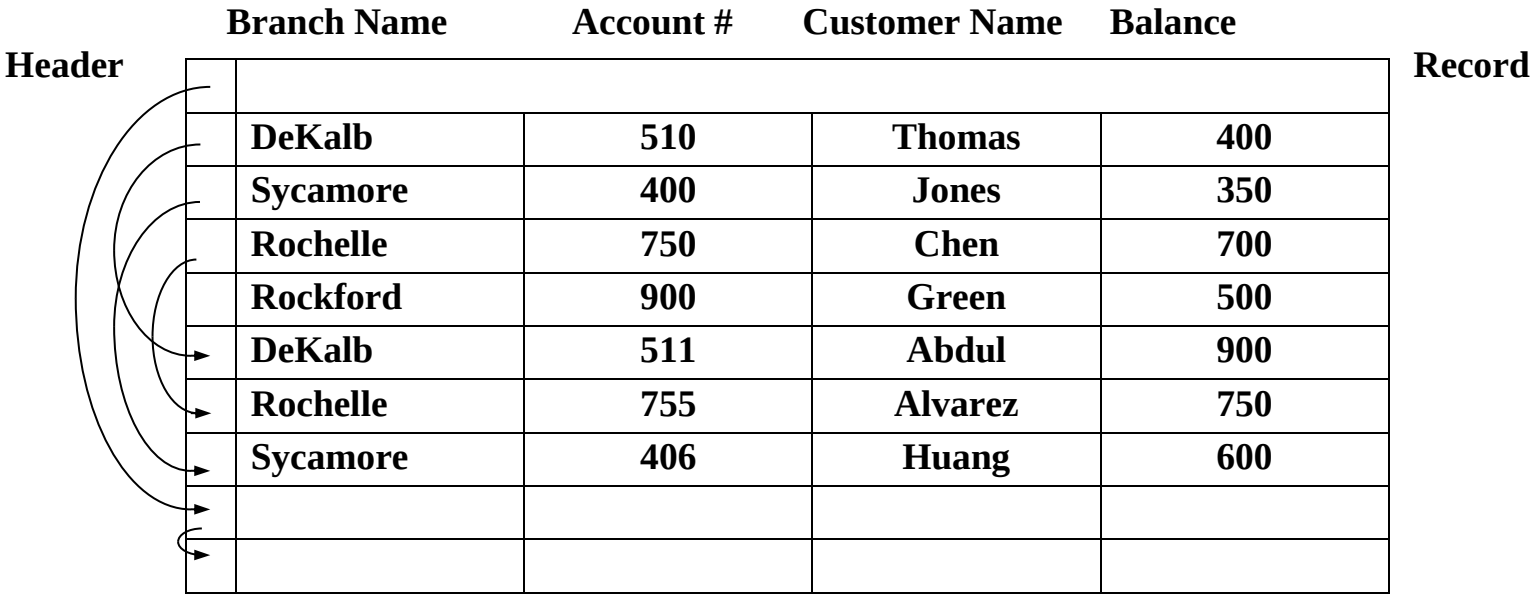
- **To counter some of the problems, we can use a header record, and pointers.**

  The pointer is nothing but a storage memory address to a record in a file (not a physical arrow as shown in the examples below!).

  The header record points to the first empty record in a data file.

  We can also have related records connected through pointers.

  For the previous example, the bank records could be shown as below:



Files and File Structures (Cont.)

- **Insertion and deletion of records in this approach is quite simple**

Suppose we delete the record (Rockford, 900, Green, 500), the previous structure will change as shown below. Note: the deleted record is also linked to the chain of empty records using pointers.

	Branch Name	Account #	Customer Name	Balance
	DeKalb	510	Thomas	400
	Sycamore	400	Jones	350
	Rochelle	750	Chen	700
	DeKalb	511	Abdul	900
	Rochelle	755	Alvarez	750
	Sycamore	406	Huang	600

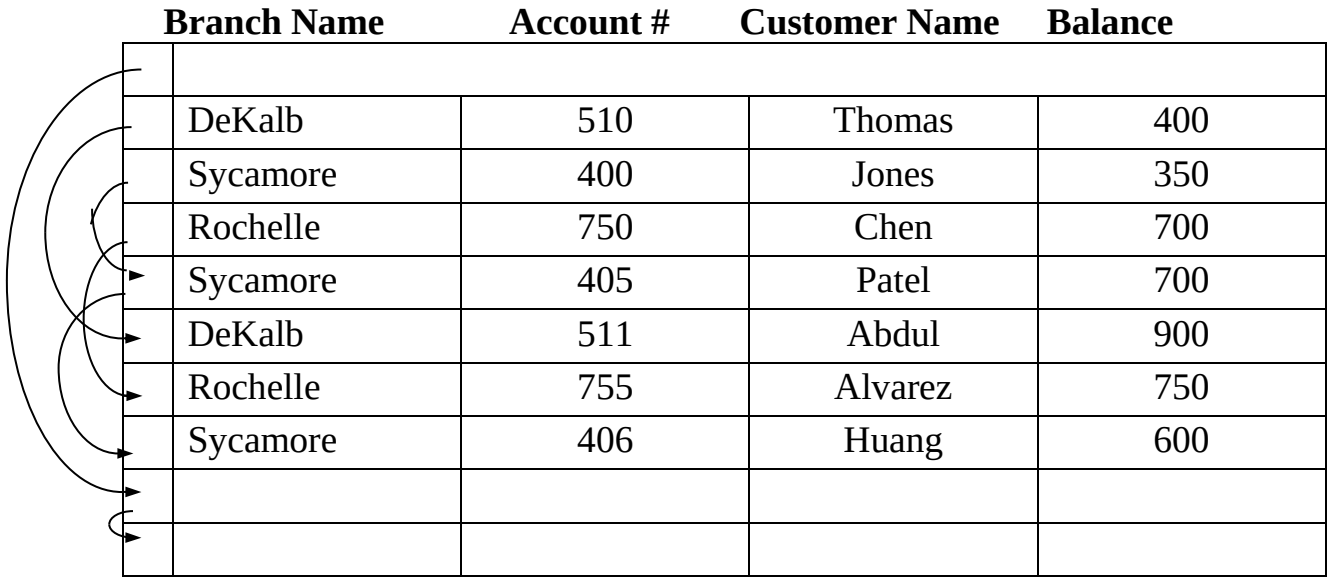
To show the space occupied by the deleted record is now available, we can either link the last empty record to this new empty record or point to this new empty record from the header record.

In reality, the data will not be deleted from the space occupied by the deleted record, but the record will be linked to the list of free records to show it is empty. Data can be deleted from a memory location only if it is written over with some other data or the disk is reformatted or defragmented.

Files and File Structures (Cont.)

If we insert a new record (Sycamore, 405, Patel, 700) to the earlier structure, what would be the resulting structure?

In this example also, assume that the records within a branch are ordered according to ascending order of account numbers.



	Branch Name	Account #	Customer Name	Balance
	DeKalb	510	Thomas	400
	Sycamore	400	Jones	350
	Rochelle	750	Chen	700
	Sycamore	405	Patel	700
	DeKalb	511	Abdul	900
	Rochelle	755	Alvarez	750
	Sycamore	406	Huang	600

- **It is possible to insert the new record at the empty location pointed by the header record or at the last empty record.**

- **Note that the bank records within Sycamore are sequenced according to account numbers.**

Files and File Structures (Cont.)

- **What are some of the problems still with this approach?**

  1. **We have not solved the problem of record length extending beyond a block size.**
  2. **Some information such as Branch Name is repeated.**
  3. **We need procedures to keep track of pointers and updating them.**
  4. **We may jump back and forth to find related records and this may slow down processing.**

- **To counter some of the mentioned problems, we can use an anchor block and an overflow block structure**

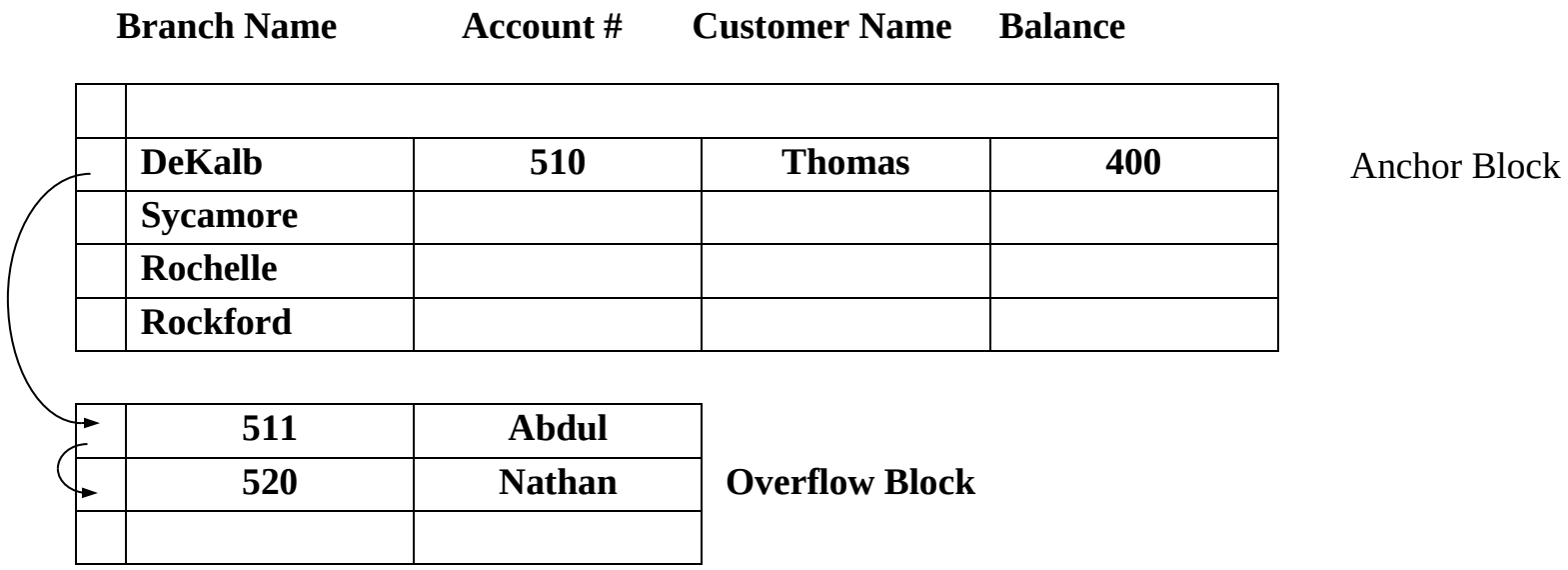
  An anchor block contains the first record in a chain of related records and the overflow block contains the records related to the first record in the anchor block.

- **By using an anchor block and an overflow block we can save some disk space and disk access in accessing related records.**

- **But if we want to access unrelated blocks, then we will have to retrieve several blocks and that can slow down performance.**

Files and File Structures (Cont.)

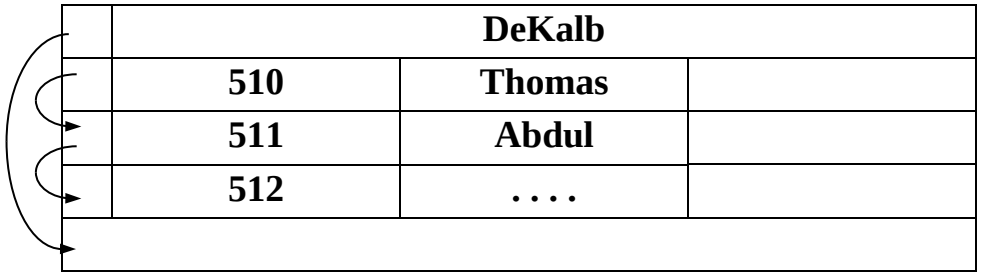
- **The figure below shows the earlier file structure organized using an anchor block and an overflow block**
- **Note in this structure that the Bank Branch Names are not repeated in the overflow block and the related records within the DeKalb branch are linked together with pointers**



- **But it is not efficient to keep related records of different bank branches in the same overflow block.**
- **A better alternative may be to have separate block structures for each branch as shown below.**

Files and File Structures (Cont.)

Block 0



Block 1

	Sycamore		
	400	**. . .**	
	405	**. . .**	
	406	**. . .**	

Block 2

Files and File Structures (Cont.)

- **This approach is not completely efficient for the following reasons:**

  1. **We have to have a way for locating the block for a particular branch.**

  2. **If a block is insufficient for maintaining the records for a branch, then we may have to chain it to another block. This may slow down performance, if we have to jump between blocks to retrieve a particular block.**

- **A better approach may be to maintain related blocks in a bucket**

  A bucket is a collection of related blocks.

  However, maintaining related records in a bucket alone may not simplify the process of searching, retrieving, inserting, or deleting records.

Caution!

- **After performing a series of insertion and deletion operations, students would try to sort the final set of records manually and arrange them in a sequence from top to bottom in a file. This is a mistake!**

- **Sequencing of records in a file should done using pointers, and the final set of records after a series of insertion and deletion operations may not be visually appealing but it is okay**

- **Remember the record operations are done using software and not done manually!**

Files and File Structures (Cont.)

- **Summary**

  Organizing records into a file is not easy.

  It requires marking what was deleted.

  When something has to be inserted, the new record has to be kept in sequence.

  Requires overhead for manipulating pointers.

  Sorting and searching for a particular record within a list of records is not easy.

  But records have to be stored in files.

  Records have to be sequenced using pointers.

  Deleted space has to be kept track throughout the operations.

  All these requirements justify the need for a better way to do the file and record operations in a database.

  The methods described above are not used in database management systems, but they give a better understanding of what is involved in creating, accessing, and maintaining records.

  These methods were described only to give a historical perspective of the problem of managing data.