Intro to DB

CHAPTER 11
STORAGE & FILE STRUCTURE

Chapter 11: Storage and File Structure

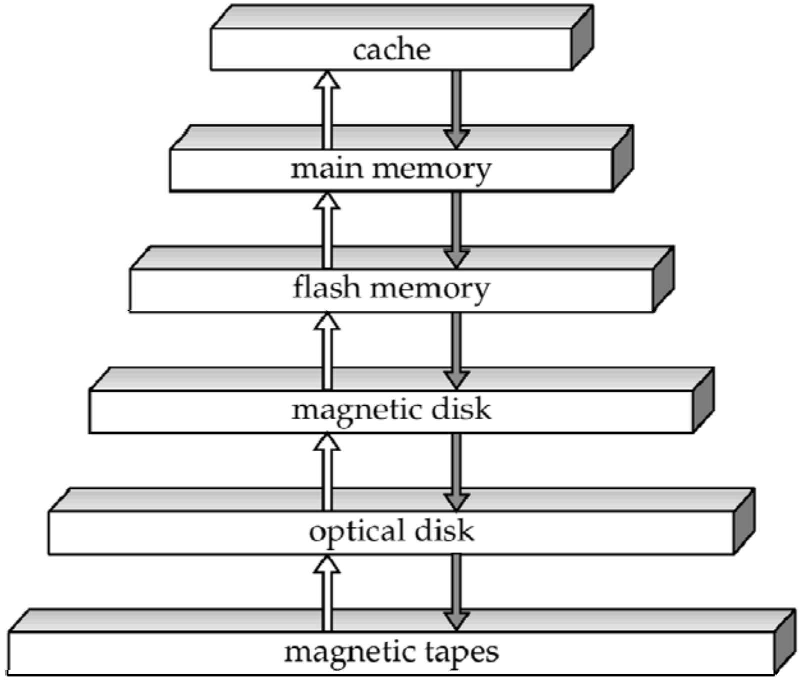
- Overview of Physical Storage Media
- Magnetic Disks
- RAID
- Tertiary Storage
- Storage Access
- File Organization
- Organization of Records in Files
- Data-Dictionary Storage
- Storage Structures for Object-Oriented Databases

Physical Storage Media

- Data access speed
- Cost per unit of data
- Reliability
  - data loss on power failure or system crash
  - physical failure of the storage device
- Persistence
  - **volatile storage:** loses contents when power is switched off
  - **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter-backed up main-memory.

Storage Hierarchy - Operational

- **primary storage**
  - Fastest media but volatile
  - cache, main memory
- **secondary storage**
  - next level in hierarchy, non-volatile, moderately fast access time
  - also called **on-line storage**
  - E.g. flash memory, magnetic disks
- **tertiary storage**
  - lowest level in hierarchy, non-volatile, slow access time
  - also called **off-line storage**
  - E.g. magnetic tape, optical storage



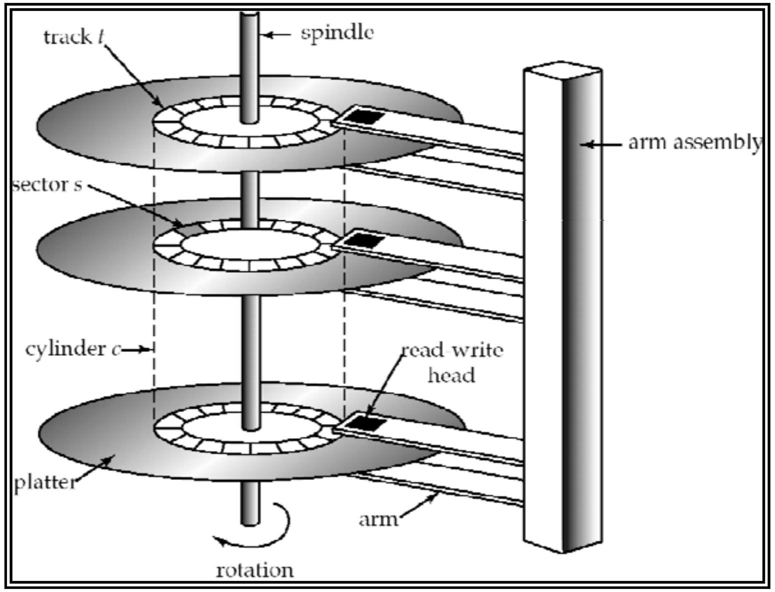
Storage hierarchy - Media

Magnetic Disks

- Read-write head
  - Positioned very close to the platter surface
  - Reads or writes magnetically encoded information
- Surface of platter divided into circular ***tracks***
  - Over 50K-100K tracks per platter on typical hard disks
- Each track is divided into ***sectors.***
  - Sector size typically 512 bytes
  - Typical sectors per track: 500 (on inner tracks) to 1000 (on outer tracks)
- To read/write a sector
  - disk arm swings to position head on right track
  - platter spins continually; data is read/written as sector passes under head

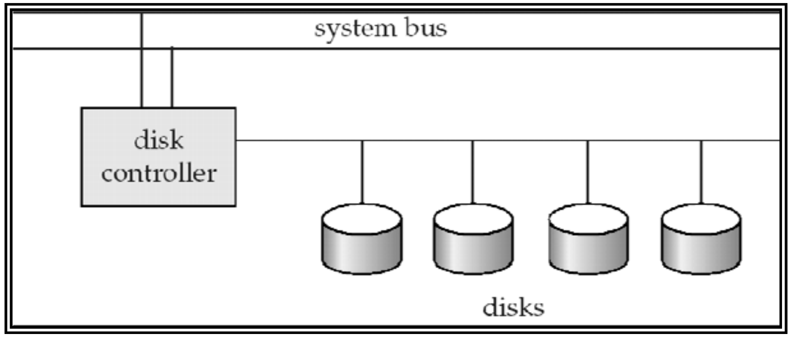
Magnetic Disks (Cont.)

- Head-disk assemblies
  - multiple disk platters on a single spindle (1 to 5 usually)
  - one head per platter, mounted on a common arm.
- ***Cylinder*** *i* consists of *i*th track of all the platters
- Earlier generation disks were susceptible to "head-crashes" leading to loss of all data on disk
  - Current generation disks are less susceptible to such disastrous failures, but individual sectors may get corrupted



Disk Controller

- **Disk controller** interfaces between the computer system and the disk drive hardware.
  - accepts high-level commands to read or write a sector
  - controls actions such as moving the disk arm to the right track and actually reading or writing the data
- Manages quality & robustness
  - computes and attaches **checksums** to each sector to verify that data is read back correctly
  - Ensures successful writing by reading back sector after writing it
  - Performs remapping of bad sectors



Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins
  - **Seek time** – time it takes to reposition the arm over the correct track.
    - average seek time is 1/2 the worst case seek time
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to come under the head
    - average latency is 1/2 of the worst case latency
    - 4 to 11 milliseconds on typical disks (5400 to 15000 rpm)
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
  - 25 to 100 MB per second max rate, lower for inner tracks
  - Multiple disks may share a controller, so rate that controller can handle is also important
    - E.g. ATA-5: 66 MB/sec, SATA: 150 MB/sec, Ultra 320, SCSI: 320 MB/s
    - Fiber Channel (FC2Gb): 256 MB/s

Performance Measures (Cont.)

- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.
  - Typically 3 to 5 years
  - Probability of failure of new disks is quite low
    - "theoretical MTTF" of 500,000 to 1,200,000 hours for a new disk
    - E.g., an MTTF of 1,200,000 hours for a new disk means that given 1000 relatively new disks, on an average one will fail every 1200 hours
  - MTTF decreases as disk ages

Optimization of Disk-Block Access

- **Block** – a contiguous sequence of sectors from a single track
  - data is transferred between disk and main memory in blocks
  - sizes range from 512 bytes to several kilobytes
    - Smaller blocks: more transfers from disk
    - Larger blocks: more space wasted due to partially filled blocks
    - Typical block sizes today range from 4 to 16 kilobytes
- **Disk-arm-scheduling**
  - order pending accesses to tracks so that disk arm movement is minimized
  - **elevator algorithm**
    - move disk arm in one direction (from outer to inner tracks or vice versa), processing next request in that direction, till no more requests in that direction, then reverse direction and repeat

Optimization of Disk-Block Access (Cont.)

- **File organization**
  - optimize block access time by organizing the blocks to correspond to how data will be accessed
  - E.g. Store related information on the same or nearby blocks/cylinders.
    - File systems attempt to allocate contiguous chunks of blocks (8~16 blocks) to a file
  - Files may get **fragmented** over time
- **Nonvolatile write buffers**
  - write blocks to a non-volatile RAM buffer immediately
  - controller then writes to disk whenever the disk is free
  - DB operations can continue without waiting for data to be written to disk
  - writes can be reordered to minimize disk arm movement
  - non-volatile RAM: battery backed up RAM or flash memory
- **Log disk** – a disk devoted to writing a sequential log of block updates
  - used exactly like nonvolatile RAM (no need for special HW)
  - write to log disk is very fast since no seeks are required

RAID

- Redundant Arrays of Independent Disks
- manage a large numbers of disks, providing a view of a single disk of
  - high capacity and high speed
  - high reliability
- "I" in RAID
  - Originally a cost-effective alternative to large, expensive disks
    - "I" stood for *inexpensive*
  - Today RAIDs are used for their higher reliability and bandwidth.
    - The "I" is interpreted as *independent*

Reliability via Redundancy

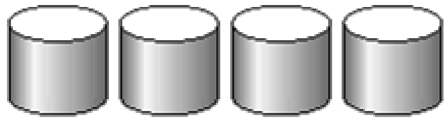
- Redundancy
  - store extra information that can be used to rebuild information lost in a disk failure
- Mirroring (or shadowing)
  - Duplicate every disk
  - Write on both disks / Read either disk
  - If one disk fails, data still available in the other
  - MTTF for mirrored disk: 500,000,000 hours (57,000 years)
    - MTTF (Mean Time To Failure) of single disk: 100,000 hours (approx. 11 years)
    - depends on mean time to repair & independence of failure
- Parity
  - Store parity bit (block) for every *n* data bits (blocks)

Performance via Parallelism

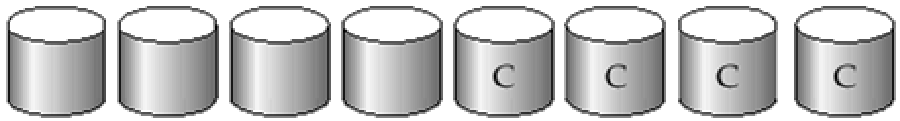
- Stripe data across multiple disks
  - Improve transfer rate
- *Bit-level striping* – split the bits of each byte across multiple disks
  - In an array of eight disks, write bit *i* of each byte to disk *i*.
  - Each access can read data at eight times the rate of a single disk.
  - But seek/access time worse than for a single disk
    - Bit level striping is not used much any more
- *Block-level striping* – with *n* disks, block *i* of a file goes to disk (*i* mod *n*) + 1
  - Requests for different blocks can run in parallel if the blocks reside on different disks
  - A request for a long sequence of blocks can utilize all disks in parallel

RAID Levels

- Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
  - Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics
- RAID Level 0: Block striping; non-redundant
  - Used in high-performance applications where data loss is not critical
- RAID Level 1: Mirrored disks with block striping
  - Offers best write performance
  - Popular for applications such as storing log files in a database system



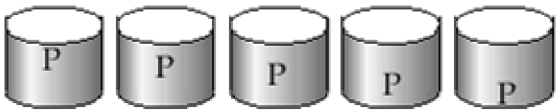
(a) RAID 0: nonredundant striping



(b) RAID 1: mirrored disks

RAID Levels (Cont.)

- RAID Level 5: Block-interleaved distributed parity
  - partition data and parity among all *N* + 1 disks, rather than storing data in *N* disks and parity in 1 disk
  - Compared to level 1, lower storage overhead but higher time overhead for writes: popular for applications with frequent reads with rare writes



(f) RAID 5: block-interleaved distributed parity

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

Storage Access

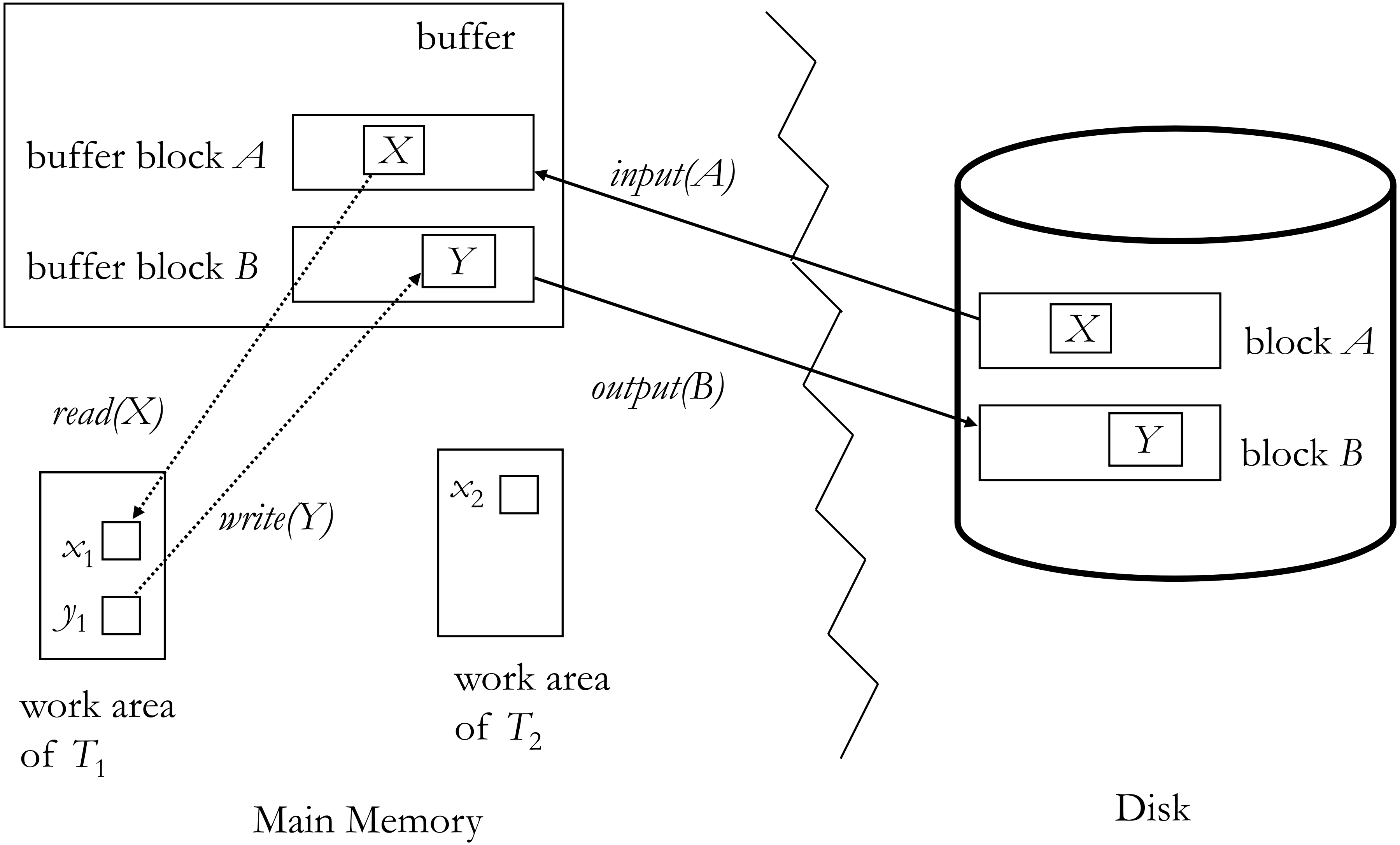
- A database file is partitioned into fixed-length storage units called **blocks**
  - Blocks are units of both storage allocation and data transfer
- Need to minimize the number of block transfers between the disk and memory
  - reduce the number of disk accesses by keeping as many blocks as possible 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

Buffer Manager

Programs call on the buffer manager when they need a block from disk

- If the block is already in the buffer
  - Return pointer to the block in main memory
- Else
  1. allocate space in the buffer for the block
     - If required replace (throw out) some other block
     - needs to be written back to disk only if it was modified
  2. read the block from the disk to the buffer
     - return the pointer of the block in main memory
- Buffer Replacement Policy
  - Most OS use LRU
  - DBMS uses mixed strategy
    - has more information on access patterns for data blocks

Example of Data Access



File Organization

- The database is stored as a collection of *files*
  - Each file is a sequence of *records*
  - A record is a sequence of fields
  - These are stored in units of blocks!
- Fixed length records
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations
- Variable-length records
  - Storage of multiple record types in a file
  - Record types that allow variable lengths for one or more fields
  - Record types that allow repeating fields
  - Byte string representation, pointer based methods, ⋯

Fixed Length Records

record 0	A-102	Perryridge	400
record 1	A-305	Round Hill	350
record 2	A-215	Mianus	700
record 3	A-101	Downtown	500
record 4	A-222	Redwood	700
record 5	A-201	Perryridge	900
record 6	A-217	Brighton	750
record 7	A-110	Downtown	600
record 8	A-218	Perryridge	700

Delete record 2

record 0	A-102	Perryridge	400
record 1	A-305	Round Hill	350
record 3	A-101	Downtown	500
record 4	A-222	Redwood	700
record 5	A-201	Perryridge	900
record 6	A-217	Brighton	750
record 7	A-110	Downtown	600
record 8	A-218	Perryridge	700

record 0	A-102	Perryridge	400
record 1	A-305	Round Hill	350
record 8	A-218	Perryridge	700
record 3	A-101	Downtown	500
record 4	A-222	Redwood	700
record 5	A-201	Perryridge	900
record 6	A-217	Brighton	750
record 7	A-110	Downtown	600

header				
record 0	A-102	Perryridge	400	
record 1				
record 2	A-215	Mianus	700	
record 3	A-101	Downtown	500	
record 4				
record 5	A-201	Perryridge	900	
record 6				
record 7	A-110	Downtown	600	
record 8	A-218	Perryridge	700	

Variable Length Records

0	Perryridge	A-102	400	A-201	900	A-218	700	⊥
1	Round Hill	A-305	350	⊥				
2	Mianus	A-215	700	⊥				
3	Downtown	A-101	500	A-110	600	⊥		
4	Redwood	A-222	700	⊥				
5	Brighton	A-217	750	⊥				

Byte-string representation

Reserved-space rep.

0	Perryridge	A-102	400	A-201	900	A-218	700
1	Round Hill	A-305	350	⊥	⊥	⊥	⊥
2	Mianus	A-215	700	⊥	⊥	⊥	⊥
3	Downtown	A-101	500	A-110	600	⊥	⊥
4	Redwood	A-222	700	⊥	⊥	⊥	⊥
5	Brighton	A-217	750	⊥	⊥	⊥	⊥

Pointer methods

0	Perryridge	A-102	400	
1	Round Hill	A-305	350	
2	Mianus	A-215	700	
3	Downtown	A-101	500	
4	Redwood	A-222	700	
5		A-201	900	
6	Brighton	A-217	750	
7		A-110	600	
8		A-218	700	

anchor block

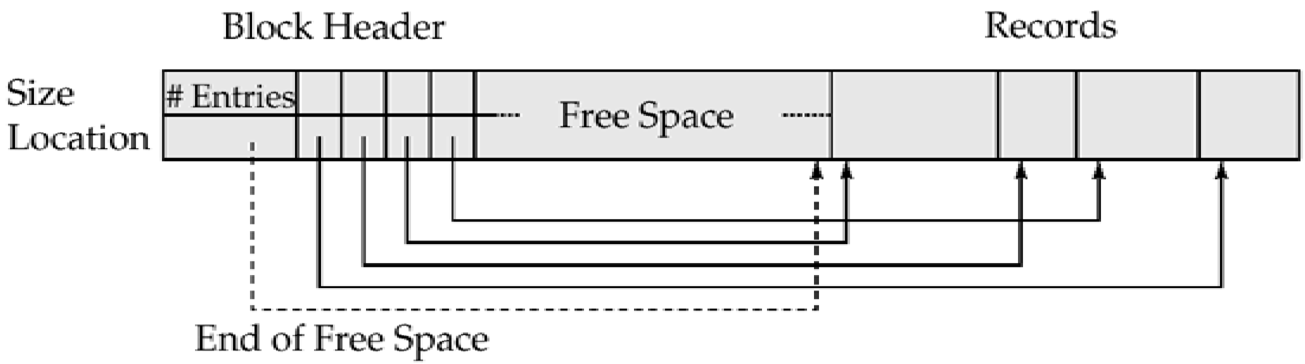
Perryridge	A-102	400	
Round Hill	A-305	350	
Mianus	A-215	700	
Downtown	A-101	500	
Redwood	A-222	700	
Brighton	A-217	750	

overflow block

A-201	900	
A-218	700	
A-110	600	

Variable Length Records – Slotted Page Structure

- Byte-string representation for variable-length records
  - assume that records are smaller than a block
- Page header contains:
  - number of record entries
  - end of free space in the block
  - location and size of each record
- Benefits
  - Supports indirect pointers to record
  - Prevents fragmentation of space inside a block



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

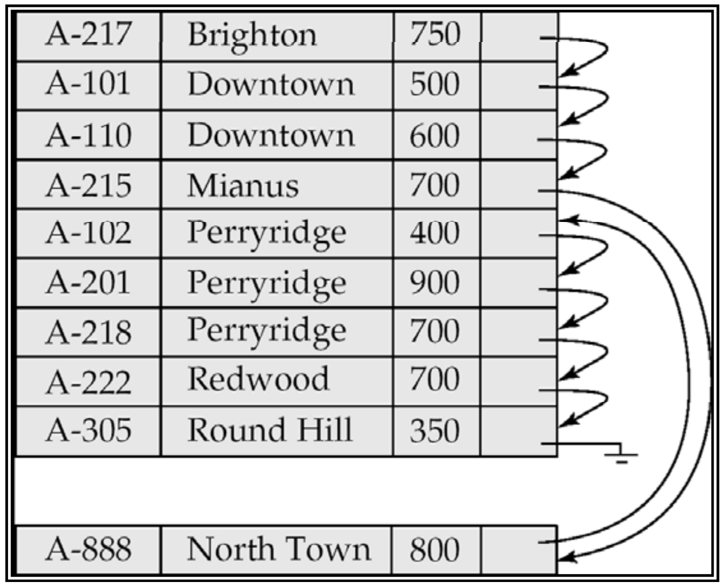
Sequential File Organization

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

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

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



END OF CHAPTER 11