Database II

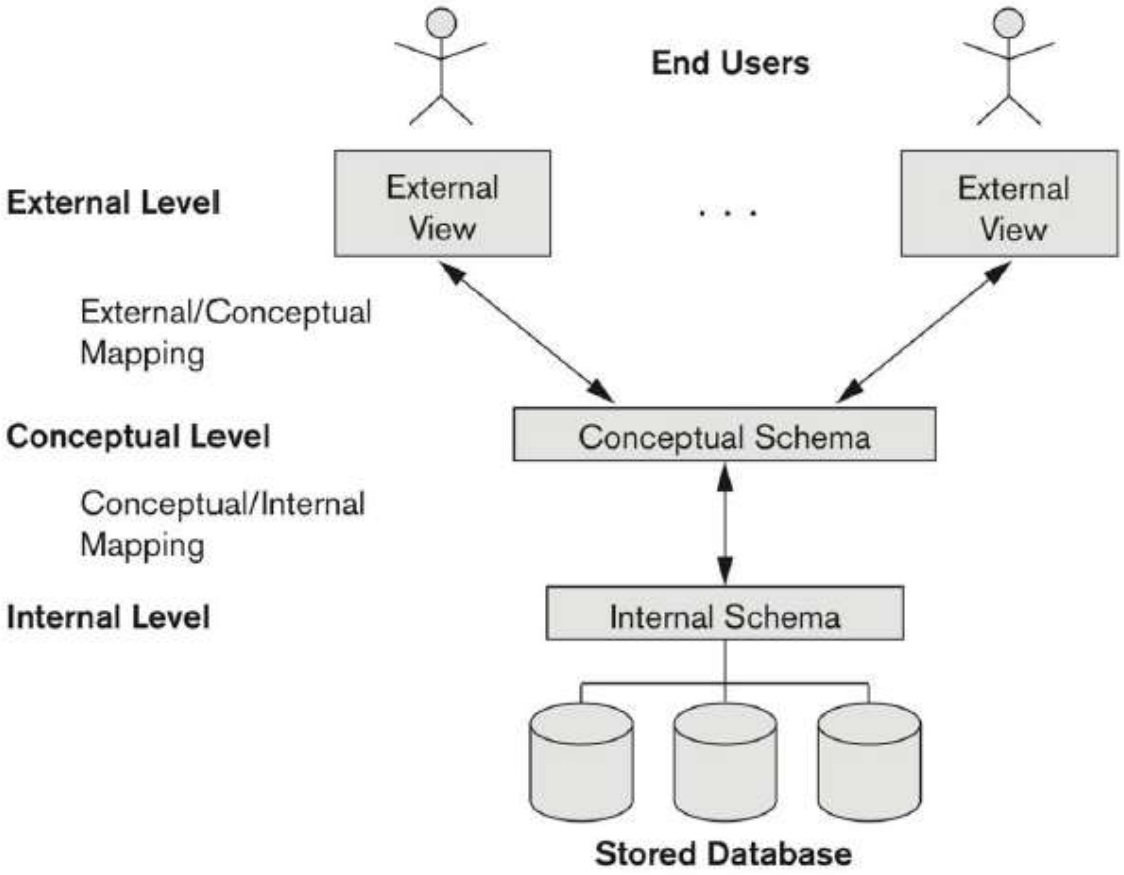
Lecture 3

Disk Storage and Basic File Structures

Dr. Doaa Elzanfaly

Introduction

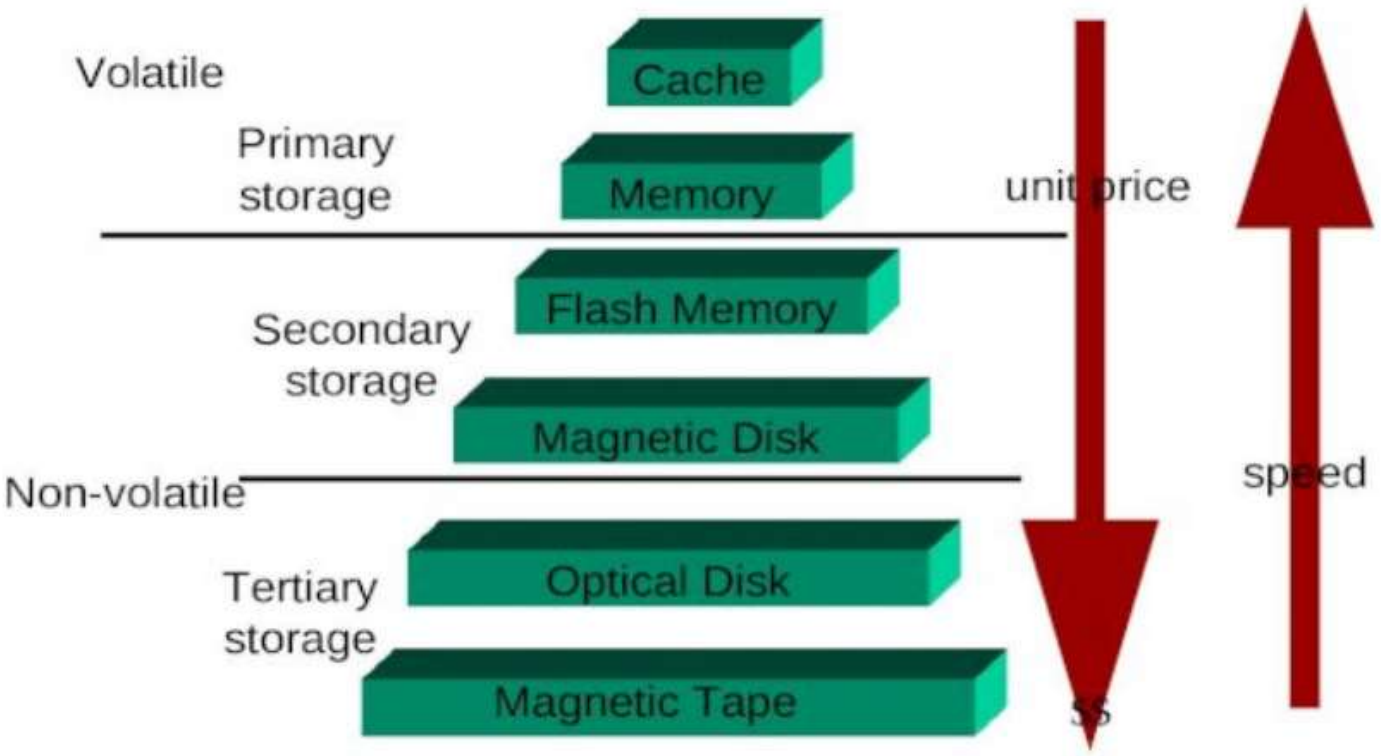
- Databases are stored on disks and accessed according to:
  - Physical database file structures (Physical levels of three schema architecture)



Introduction

- Storage hierarchy
  - Primary storage
    - CPU main memory, cache memory
  - Secondary storage
    - Magnetic disks, flash memory, solid-state drives
  - Tertiary storage
    - Removable media

Storage Hierarchy



- Offline storage, archiving databases (larger capacity, less cost, slower access, not directly accessible by CPU)

Memory Hierarchies and Storage Devices

- Cache memory
  - Static RAM
  - DRAM
- Mass storage
  - Magnetic disks
    - CD-ROM, DVD, tape drives
- Flash memory
  - Nonvolatile

Memory Hierarchies and Storage Devices

- Depending upon the intended use and application requirements, data is kept in one or more levels of hierarchy.
  - Programs are in main memory (DRAM)
  - Permanent databases reside in secondary storage
  - Main memory buffers are used to read and write to secondary storage

Storage Types and Characteristics

Type	Capacity*	Access Time	Max Bandwidth	Commodity Prices (2014)**
Main Memory- RAM	4GB–1TB	30ns	35GB/sec	$100–$20K
Flash Memory- SSD	64 GB–1TB	50µs	750MB/sec	$50–$600
Flash Memory- USB stick	4GB–512GB	100µs	50MB/sec	$2–$200
Magnetic Disk	400 GB–8TB	10ms	200MB/sec	$70–$500
Optical Storage	50GB–100GB	180ms	72MB/sec	$100
Magnetic Tape	2.5TB–8.5TB	10s–80s	40–250MB/sec	$2.5K–$30K
Tape jukebox	25TB–2,100,000TB	10s–80s	250MB/sec–1.2PB/sec	$3K–$1M+

*Capacities are based on commercially available popular units in 2014.

**Costs are based on commodity online marketplaces.

Table 16.1 Types of Storage with Capacity, Access Time, Max Bandwidth (Transfer Speed), and Commodity Cost

Storage Organization of Databases

- Persistent data
  - Most databases
- Transient data
  - Exists only during program execution
- File organization
  - Determines how records are *physically placed* on the disk.
  - Determines how records are *accessed*

Secondary Storage Devices

- Hard disk drive
- Bits (ones and zeros)
  - Grouped into bytes or characters
- Disk capacity measures storage size
- Disks may be single or double-sided
- Concentric circles called tracks
  - Tracks divided into blocks or sectors
- Disk packs
  - Cylinder

Single-Sided Disk and Disk Pack

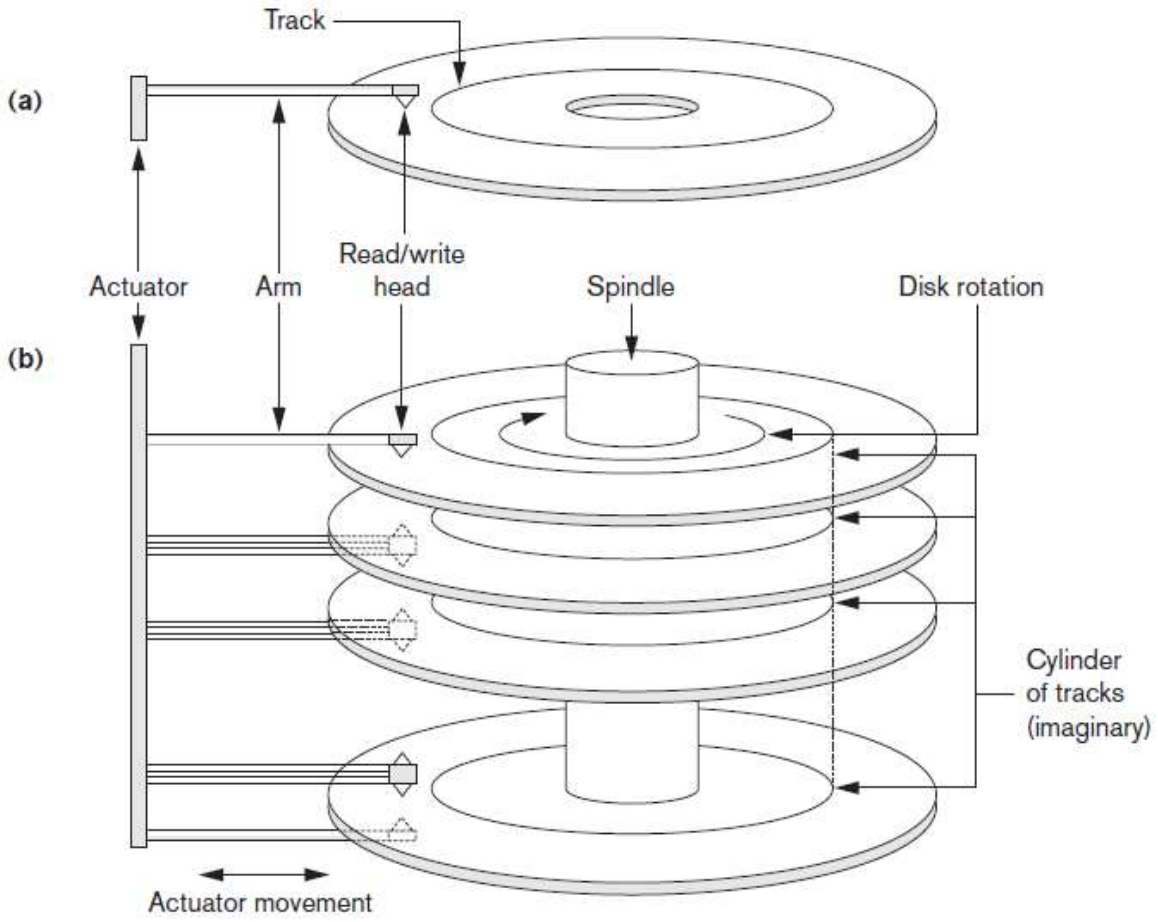


Figure 16.1 (a) A single-sided disk with read/write hardware
(b) A disk pack with read/write hardware

Sectors on a Disk

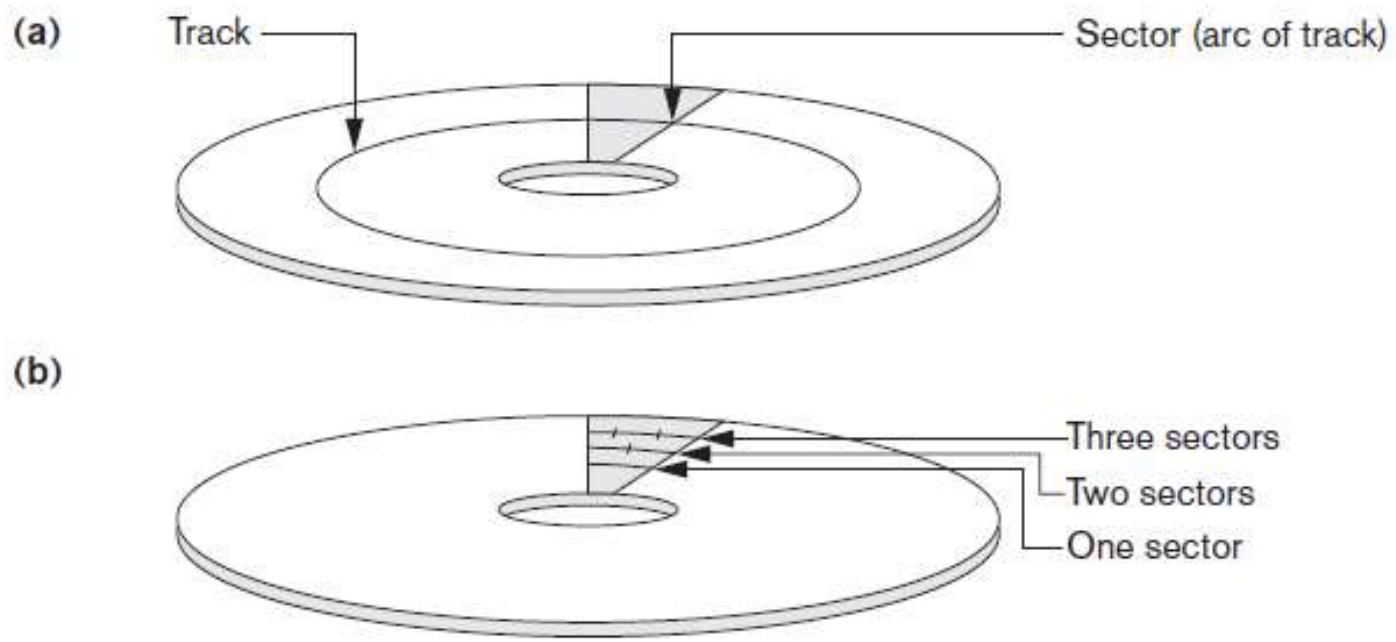
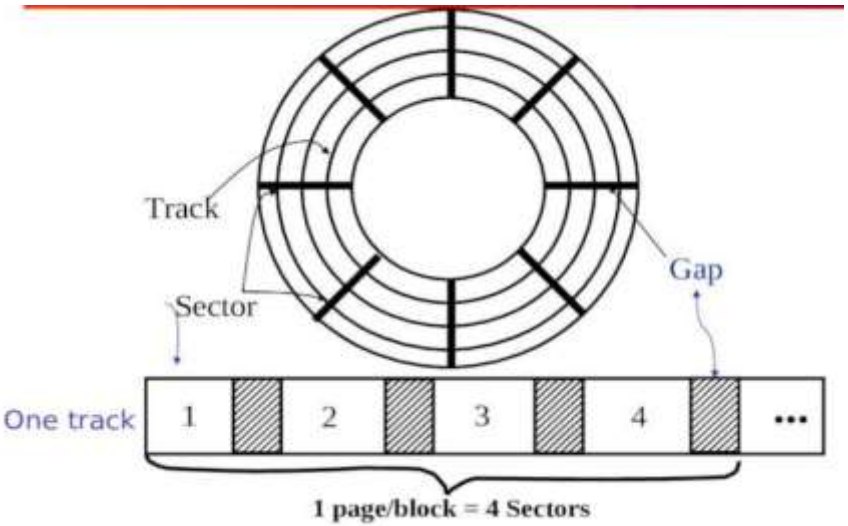


Figure 16.2 Different sector organizations on disk (a) Sectors subtending a fixed angle (b) Sectors maintaining a uniform recording density

Secondary Storage Devices

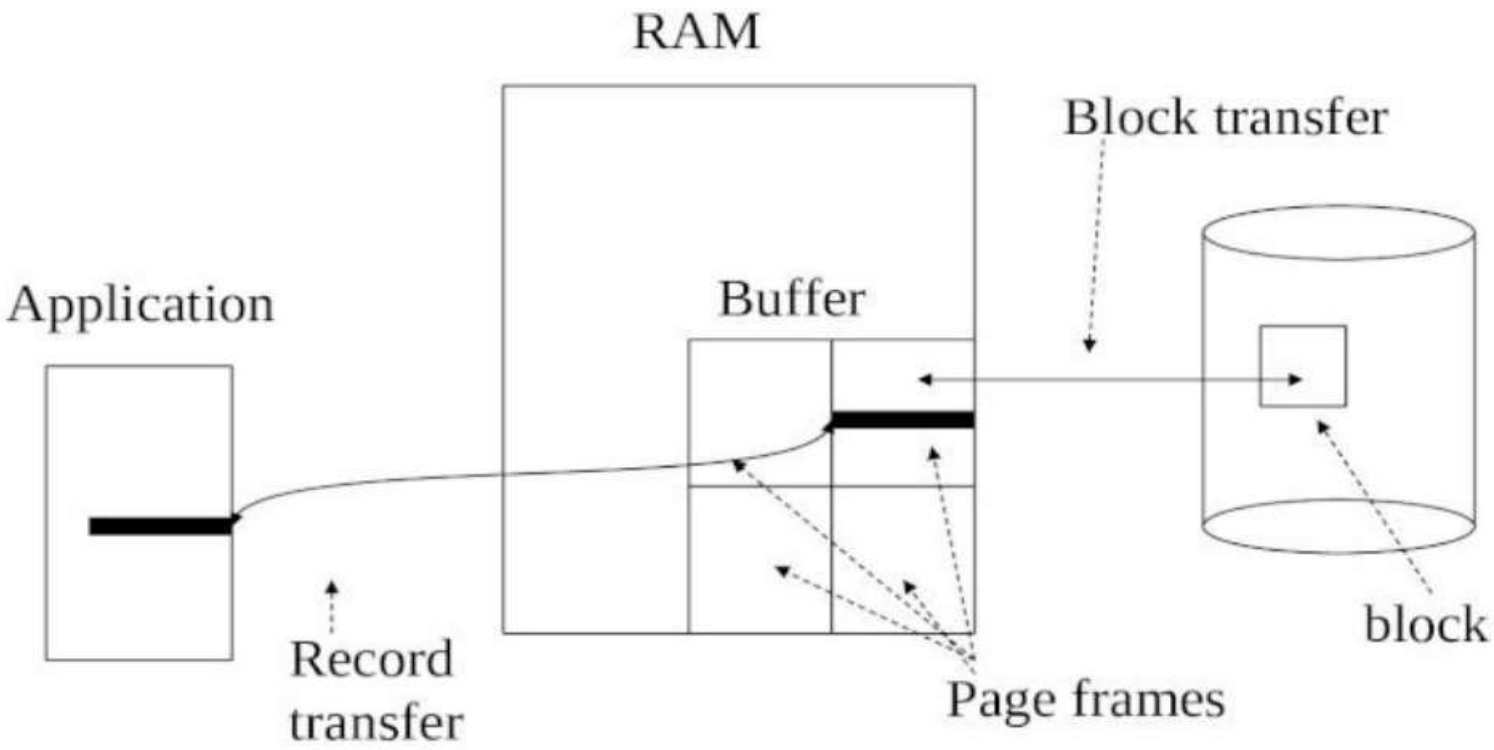
- Formatting
  - Divides tracks into equal-sized disk blocks
  - Blocks separated by interblock gaps
- Data transfer in units of disk blocks
  - Hardware address supplied to disk I/O hardware



Buffering

- A buffer
  - Is a contiguous reserved area in the memory available for storage of copies of disk blocks to speed up the processes.
- For a read command
  - The block is copied from disk into the buffer
- For write command
  - The contents of the buffer are to be written to the disk.

Accessing Data Through Buffer



Buffer Management

- Programs call the buffer manager when they need a block from disk.
  - If the block is already in the buffer
    - The requesting program is given the address of the block in main memory
  - If the block in not in the buffer
    - The buffer manager allocates space in the buffer to read the required block from the disk and passes the address to the requester.

Buffer Replacement Strategies

- To free space in the buffer for the new required blocks, the buffer manager uses on of the following replacement strategies
  - Least recently used (LRU)
  - Clock policy
  - First-in-first-out (FIFO)
- The replaced blocks are written back to disk if they were modified.
- The placement policy has an impact on the number of I/Os depending on the access patterns.

File Organization

- A database is stored as a collection of *files*.
- Each file is a sequence of *records*.
- A record is a sequence of *fields*.
- Records are stored on desk *block*.
- A file can have a *Fixed-length* records or *variable-length* records

File Organization

- Reasons for variable-length records
  - One or more fields have variable length
  - One or more fields are repeating
  - One or more fields are optional
  - File contains records of different types

Record Blocking and Spanned Versus Unspanned Records

- File of records allocated to disk blocks
- Spanned records
  - Larger than a single block
  - Pointer at end of first block points to block containing remainder of record
- Unspanned
  - Records are not allowed to cross block boundaries

Record Blocking and Spanned Versus Unspanned Records

- Blocking factor
  - Average number of records per block for the file

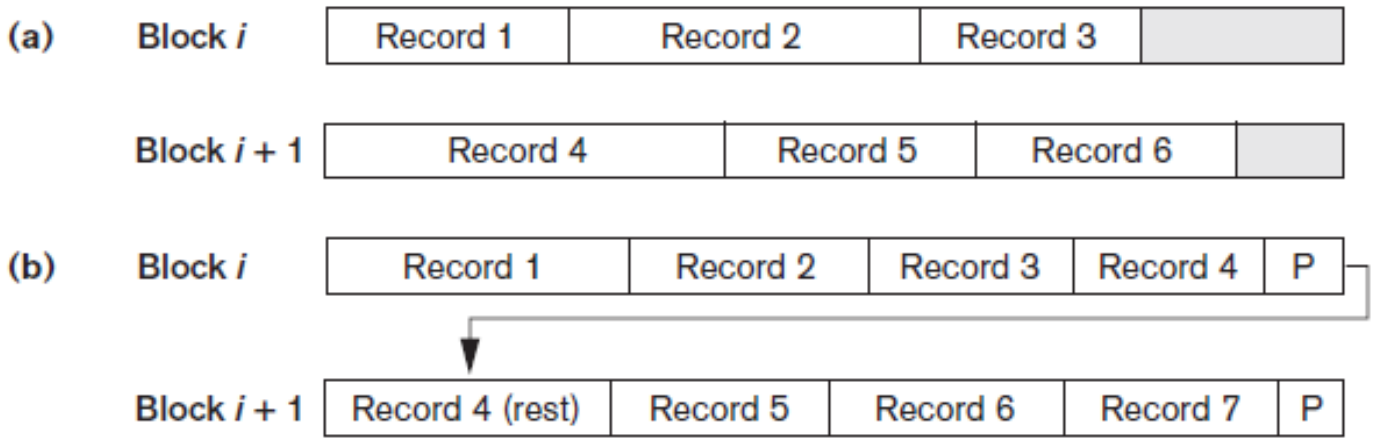


Figure 16.6 Types of record organization (a) Unspanned (b) Spanned

Operations on Files

- Examples of operations for accessing file records
  - Open
  - Find
  - Read
  - FindNext
  - Delete
  - Insert
  - Close
  - Scan

Organization of Records in Files

- Heap/unordered File Organization
- Ordered File Organization
- Hashed File Organization

Files of Unordered Records (Heap Files)

- Heap (or pile) file
  - Records placed in file in order of insertion
- Inserting a new record is very efficient
- Searching for a record requires linear search
- Deletion techniques
  - Rewrite the block
  - Use deletion marker

Files of Ordered Records (Sorted Files)

- Ordered (sequential) file
  - Records sorted by ordering field
    - Called ordering key if ordering field is a key field
- Advantages
  - Reading records in order of ordering key value is extremely efficient
  - Finding next record
  - Binary search technique

Access Times for Various File Organizations

Type of Organization	Access/Search Method	Average Blocks to Access a Specific Record
Heap (unordered)	Sequential scan (linear search)	$b/2$
Ordered	Sequential scan	$b/2$
Ordered	Binary search	$\log_2 b$

Table 16.3 Average access times for a file of *b* blocks under basic file organizations

Hashing Techniques

- Hash function (randomizing function)
  - Applied to hash field value of a record
  - Yields address of the disk block of stored record
- Organization called hash file
  - Search condition is equality condition on the hash field
  - Hash field typically key field.
- Hashing also internal search structure
  - Used when group of records accessed exclusively by one field value

Hashing Techniques

- Collision occurs when a new record hashes to a block that is already full.
- Collision resolution
  - Progressive Overflow (Open Addressing)
  - Chaining
  - Multiple hashing

Progressive Overflow

- Proceed from the occupied position specified by the hash address, checks the subsequent positions in order until an unused (empty) position is found.

Example :

key k	Home address - h(k)
COLE	20
BATES	21
ADAMS	21
DEAN	22
EVANS	20

Complete Table:

0	
1	
2	
⋮	⋮
19	
20	
21	
22	

Table size = 23

Chaining Progressive Overflow

- Various overflow locations are kept, and a collision is resolved by placing the new record in an unused overflow location and setting the pointer of the occupied hash address location to the address of that overflow location.

Key	Home
ADAMS	20
BATES	21
COLES	20
DEAN	21
EVANS	24
FLINT	20

Chained Progressive Overflow

	data	next
⋮	⋮	⋮
20	ADAMS	22
21	BATES	23
22	COLES	25
23	DEAN	-1
24	EVANS	-1
25	FLINT	-1
⋮	⋮	⋮

with separate overflow area:

primary data area

20	ADAMS	0
21	BATES	1
22		
23		
24	EVANS	-1
25		

overflow area

0	COLES	2
1	DEAN	-1
2	FLINT	-1
3		
	⋮	⋮

Example X: Search lengths:

Key	Home	Progressive Overflow	Chained Progr. Overflow
ADAMS	20	1	1
BATES	21	1	1
COLES	20	3	2
DEAN	21	3	2
EVANS	24	1	1
FLINT	20	6	3
Average Search Length :		2.5	1.7

Progressive Overflow

	data
⋮	⋮
20	ADAMS
21	BATES
22	COLES
23	DEAN
24	EVANS
25	FLINT
⋮	⋮

Chained Progressive Overflow

	data	next
⋮	⋮	⋮
20	ADAMS	22
21	BATES	23
22	COLES	25
23	DEAN	-1
24	EVANS	-1
25	FLINT	-1
⋮	⋮	⋮

Multiple Hashing

- The first hash function determine the home address.
- If the home address is occupied, apply a second hash function to get a number *c*.
- *c* is added to the home address to produce an overflow addresses; if occupied, proceed by adding *c* to the overflow address until an empty spot is found.

Example:

k (key)	ADAMS	JONES	MORRIS	SMITH
$h_1(k)$ (home address)	5	6	6	5
$h_2(k) = c$	2	3	4	3

Hashed file using double hashing:

0	
1	
2	
3	
4	
5	ADAMS
6	JONES
7	
8	SMITH
9	
10	MORRIS

Hashing Techniques

- External hashing for disk files
  - Target address space made of buckets
  - Bucket: one disk block or contiguous blocks
- Hashing function maps a key into relative bucket
  - Table in file header converts bucket number to disk block address
- Collision problem less severe with buckets
- Static hashing
  - Fixed number of buckets allocated

SELF-STUDY PART

Parallelizing Disk Access Using RAID Technology

- Redundant arrays of independent disks (RAID)
  - Goal: improve disk speed and access time
- Set of RAID architectures (0 through 6)
- Data striping
  - Bit-level striping
  - Block-level striping
- Improving Performance with RAID
  - Data striping achieves higher transfer rates

Parallelizing Disk Access Using RAID Technology

- Improving reliability with RAID
  - Redundancy techniques: mirroring and shadowing
- RAID organizations and levels
  - Level 0
    - Data striping, no redundant data
    - Spits data evenly across two or more disks
  - Level 1
    - Uses mirrored disks

Parallelizing Disk Access Using RAID Technology

- RAID organizations and levels (cont'd.)
  - Level 2
    - Hamming codes for memory-style redundancy
    - Error detection and correction
  - Level 3
    - Single parity disk relying on disk controller
  - Levels 4 and 5
    - Block-level data striping
    - Data distribution across all disks (level 5)

Parallelizing Disk Access Using RAID Technology

- RAID organizations and levels (cont'd.)
  - Level 6
    - Applies P+Q redundancy scheme
    - Protects against up to two disk failures by using just two redundant disks
- Rebuilding easiest for RAID level 1
  - Other levels require reconstruction by reading multiple disks
- RAID levels 3 and 5 preferred for large volume storage

RAID Levels

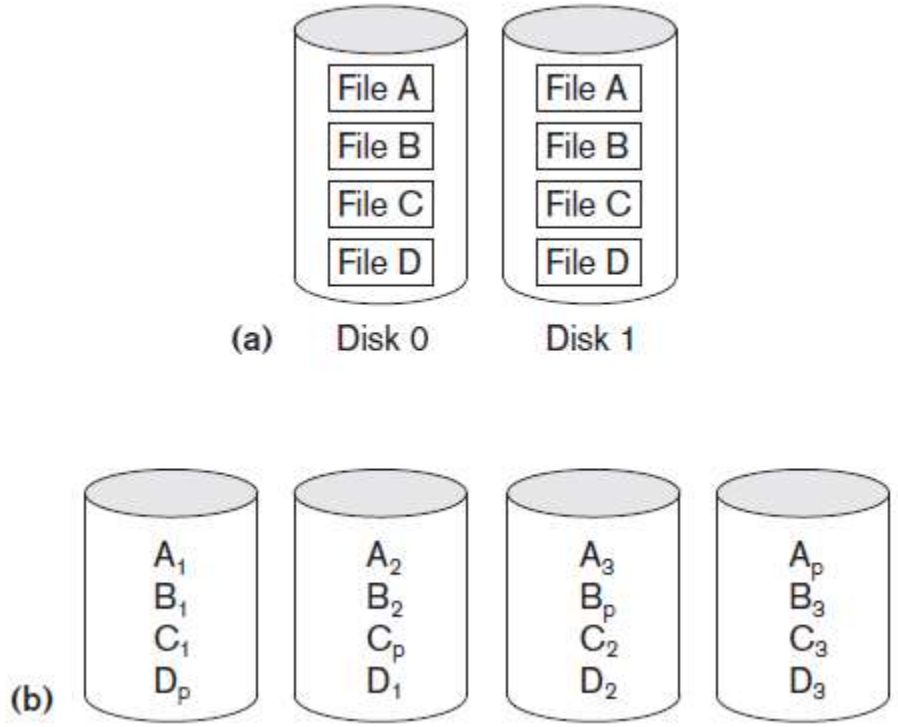


Figure 16.14 Some popular levels of RAID (a) RAID level 1: Mirroring of data on two disks (b) RAID level 5: Striping of data with distributed parity across four disks

Modern Storage Architectures

- Storage area networks
  - Online storage peripherals configured as nodes on high-speed network
- Network-attached storage
  - Servers used for file sharing
  - High degree of scalability, reliability, flexibility, performance.