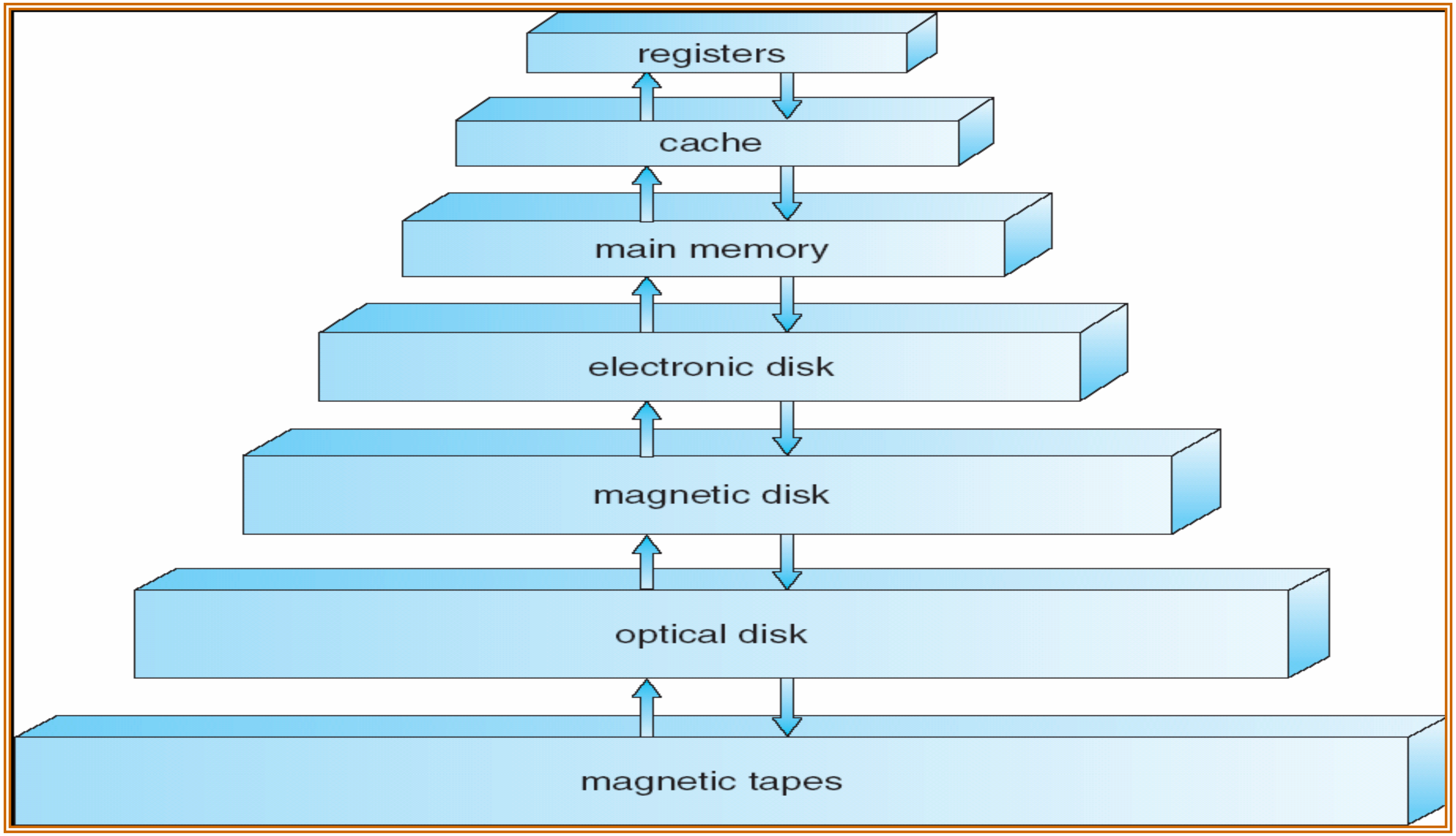
# Disk Storage, Basic File Structures, and Hashing.

# Chapter Outline

- The Storage Hierarchy
- How Far is Your Data
- Disk Storage Devices
- Records
- Blocking
- Files of Records
- Unordered Files
- Ordered Files
- Hashed Files
- RAID Technology
- Storage Area Network

# - The Storage Hierarchy



# - How far away is data?

| Location | Cycles |
| --- | --- |
| Registers | 1 |
| On-chip cache | 2 |
| On-board cache | 10 |
| Memory | 100 |
| Disk | 10**6 |
| Tape | 10**9 |

| Location | Time |
| --- | --- |
| My head | 1 min |
| This room | 2 min |
| KFUPM Campus | 10 min |
| Dhahran | 1.5 hours |
| Pluto | 2 years |
| Andromeda | 2000 years |

(Source: AlphaSort paper, 1995)

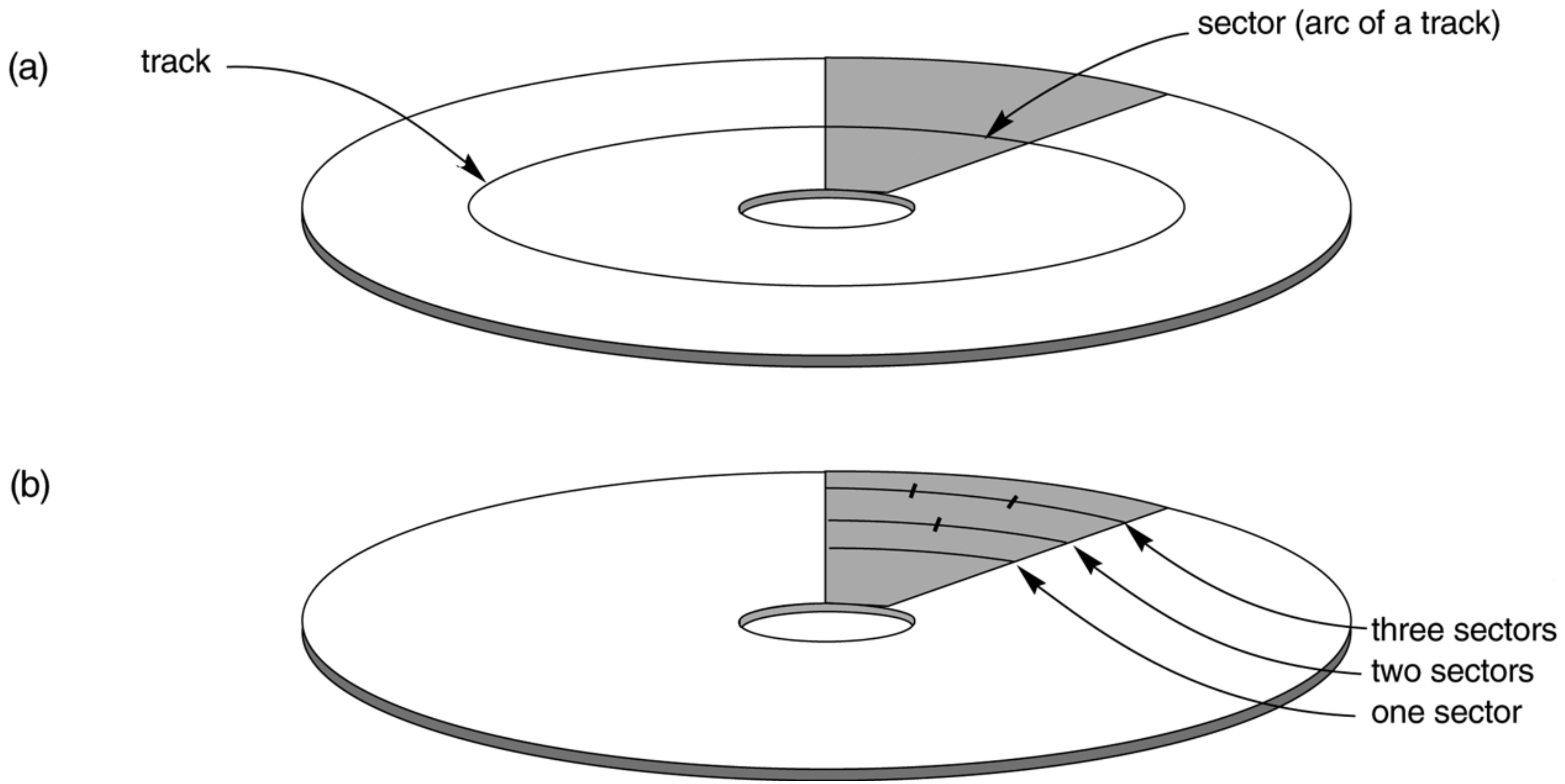
# - Disk Storage Devices ...

- Preferred secondary storage device for high storage capacity and low cost.
- Data stored as magnetized areas on magnetic disk surfaces.
- A *disk pack* contains several magnetic disks connected to a rotating spindle.
- Disks are divided into concentric circular *tracks* on each disk *surface*. Track capacities vary typically from 4 to 50 Kbytes.

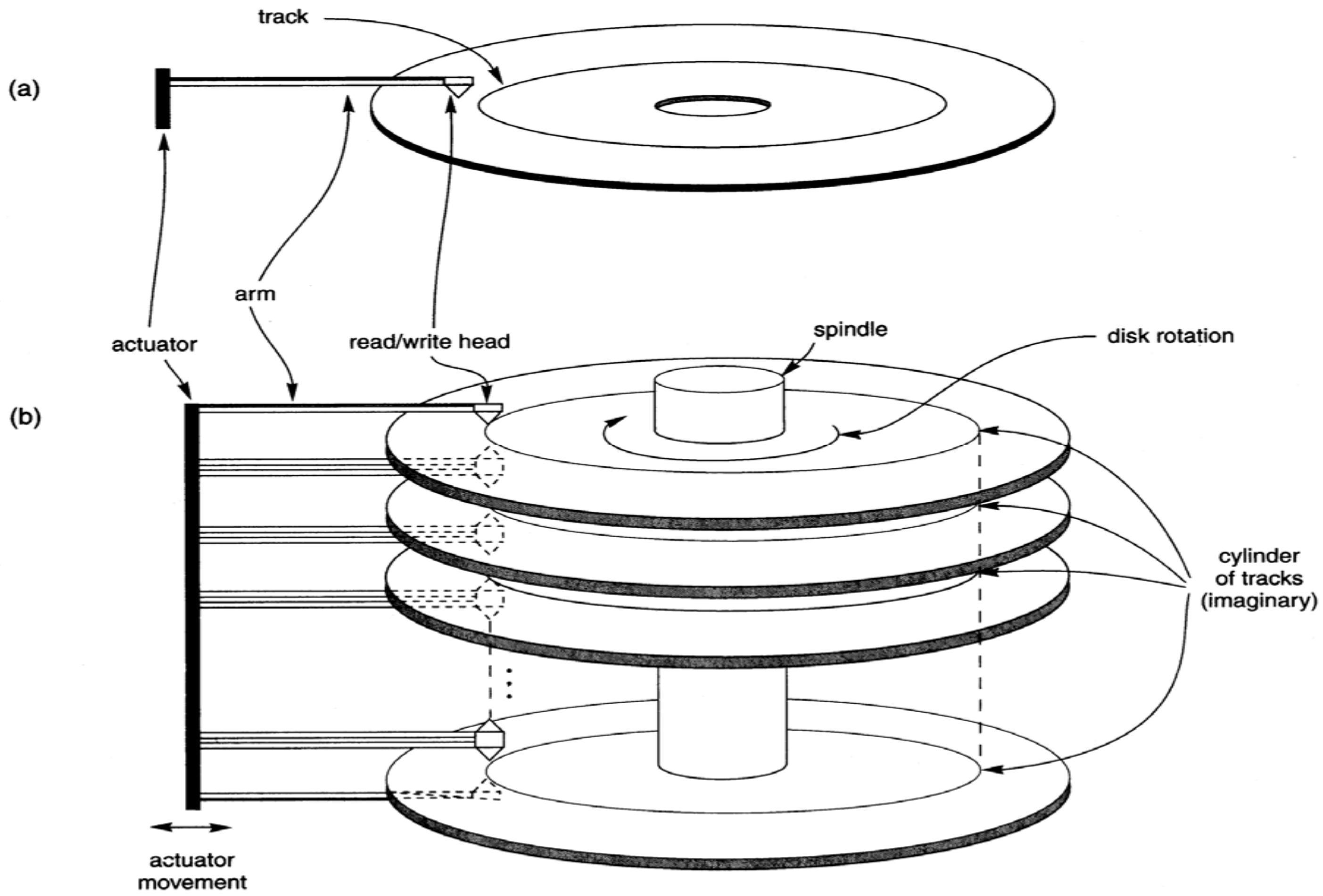
# ... - Disk Storage Devices ...

- Because a track usually contains a large amount of information, it is divided into smaller *blocks* or *sectors*.
- The division of a track into *sectors* is hard-coded on the disk surface and cannot be changed. One type of sector organization calls a portion of a track that subtends a fixed angle at the center as a sector.
- A track is divided into *blocks*. The block size B is fixed for each system. Typical block sizes range from B=512 bytes to B=4096 bytes. Whole blocks are transferred between disk and main memory for processing.

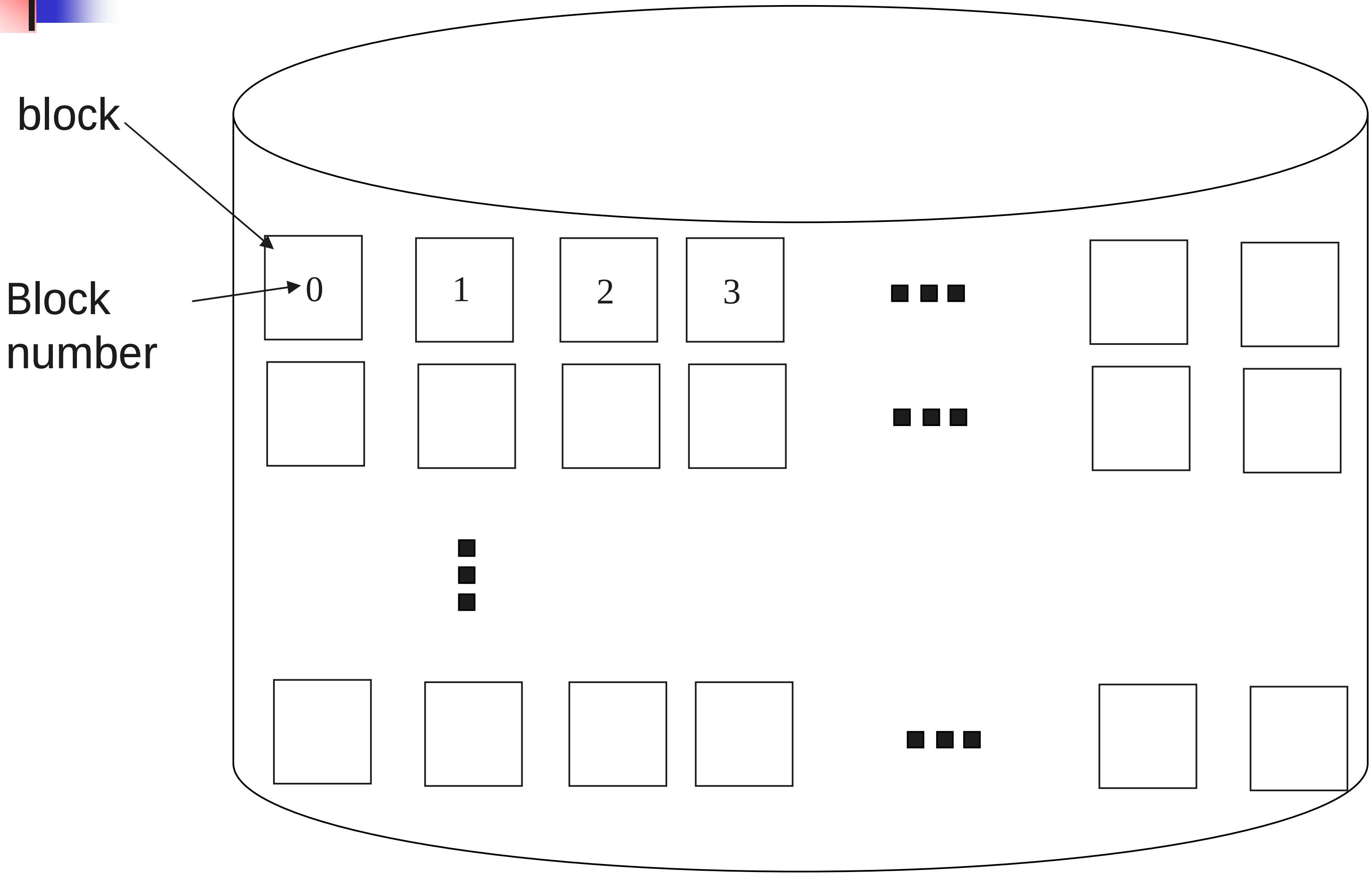
# ... - Disk Storage Devices ...



# ... - Disk Storage Devices ...



# -- Envision Disk



# - Records

- Fixed and variable length records
- Records contain fields which have values of a particular type (e.g., amount, date, time, age)
- Fields themselves may be fixed length or variable length
- Variable length fields can be mixed into one record: separator characters or length fields are needed so that the record can be "parsed".

## - Blocking

- Blocking: refers to storing a number of records in one blo ck on the disk.
- Blocking factor (*bfr*) refers to the number of records per block.
- There may be empty space in a block if an integral number of records do not fit in one block.
- *Spanned Records*: refer to records that exceed the size of one or more blocks and hence span a number of blocks.

## - Files of Records ...

- A file is a *sequence* of records, where each record is a collection of data values (or data items).
- A *file descriptor* (or *file header*) includes information that describes the file, such as the *field names* and their *data types*, and the addresses of the file blocks on disk.
- Records are stored on disk blocks. The *blocking factor bfr* for a file is the (average) number of file records stored in a disk block.
- A file can have *fixed-length* records or *variable-length* records.

# ... - Files of Records ...

- File records can be *unspanned* (no record can span two blocks) or *spanned* (a record can be stored in more than one block).
- The physical disk blocks that are allocated to hold the records of a file can be *contiguous*, *linked*, or *indexed*.
- In a file of fixed-length records, all records have the same format.
- Files of variable-length records require additional information to be stored in each record, such as *separator characters* and *field types*.

# - Unordered Files

- Also called a *heap* or a *pile* file.
- New records are inserted at the end of the file.
- To search for a record, a *linear search* through the file records is necessary. This requires reading and searching half the file blocks on the average, and is hence quite expensive.
- Record insertion is quite efficient.
- Reading the records in order of a particular field requires sorting the file records.

## - Ordered Files ...

- Also called a *sequential file*.
- File records are kept sorted by the values of an *ordering field*.
- Insertion is expensive: records must be inserted in the *correct order*.
- A *binary search* can be used to search for a record on its *ordering field value*. This requires reading and searching $\log_2$ of the file blocks on the average, an improvement over linear search.
- Reading the records in order of the ordering field is quite efficient.

# ... - Ordered Files ...

| | NAME | SSN | BIRTHDATE | JOB | SALARY | SEX |
|---|---|---|---|---|---|---|
| block 1 | Aaron, Ed | | | | | |
| | Abbott, Diane | | | | | |
| | ⋮ | | | | | |
| | Acosta, Marc | | | | | |
| block 2 | Adams, John | | | | | |
| | Adams, Robin | | | | | |
| | ⋮ | | | | | |
| | Akers, Jan | | | | | |
| block 3 | Alexander, Ed | | | | | |
| | Alfred, Bob | | | | | |
| | ⋮ | | | | | |
| | Allen, Sam | | | | | |
| block 4 | Allen, Troy | | | | | |
| | Anders, Keith | | | | | |
| | ⋮ | | | | | |
| | Anderson, Rob | | | | | |
| block 5 | Anderson, Zach | | | | | |
| | Angeli, Joe | | | | | |
| | ⋮ | | | | | |
| | Archer, Sue | | | | | |
| block 6 | Arnold, Mack | | | | | |
| | Arnold, Steven | | | | | |
| | ⋮ | | | | | |
| | Atkins, Timothy | | | | | |
| ⋮ | | | | | | |
| block n −1 | Wong, James | | | | | |
| | Wood, Donald | | | | | |
| | ⋮ | | | | | |
| | Woods, Manny | | | | | |
| block n | Wright, Pam | | | | | |
| | Wyatt, Charles | | | | | |
| | ⋮ | | | | | |
| | Zimmer, Byron | | | | | |

# -- Average Access Times

The following table shows the average access time to access a specific record for a given type of file

**TABLE 13.2 AVERAGE ACCESS TIMES FOR BASIC FILE ORGANIZATIONS**

| TYPE OF ORGANIZATION | ACCESS/SEARCH METHOD | AVERAGE TIME TO ACCESS A SPECIFIC RECORD |
|---|---|---|
| Heap (Unordered) | Sequential scan (Linear Search) | $b/2$ |
| Ordered | Sequential scan | $b/2$ |
| Ordered | Binary Search | $\log_2 b$ |

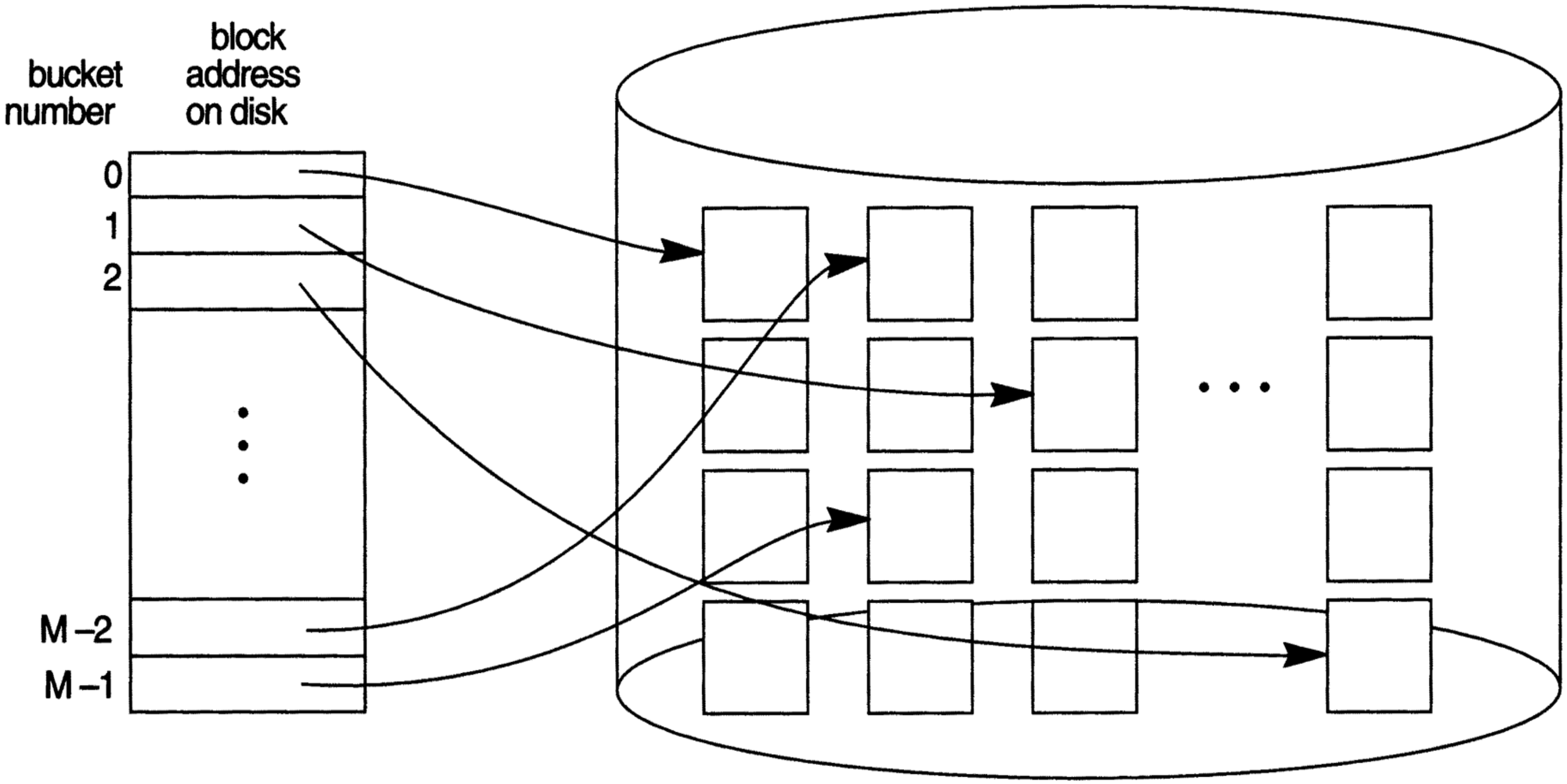
# - Hashed Files ...

- Hashing for disk files is called *External Hashing*
- The file blocks are divided into M equal-sized *buckets*, numbered $bucket_0$, $bucket_1$, ..., $bucket_{M-1}$. Typically, a bucket corresponds to one (or a fixed number of) disk block.
- One of the file fields is designated to be the hash key of the file.
- The record with hash key value K is stored in bucket i, where i=h(K), and h is the *hashing function*.
- Search is very efficient on the hash key.
- Collisions occur when a new record hashes to a bucket that is already full. An overflow file is kept for storing such records. Overflow records that hash to each bucket can be linked together.

# ... - Hashed Files ...

- There are numerous methods for collision resolution, including the following:
  - ***Open addressing:*** Proceeding from the occupied position specified by the hash address, the program checks the subsequent positions in order until an unused (empty) position is found.
  - ***Chaining:*** For this method, various overflow locations are kept, usually by extending the array with a number of overflow positions. In addition, a pointer field is added to each record location. 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.
  - ***Multiple hashing:*** The program applies a second hash function if the first results in a collision. If another collision results, the program uses open addressing or applies a third hash function and then uses open addressing if necessary.

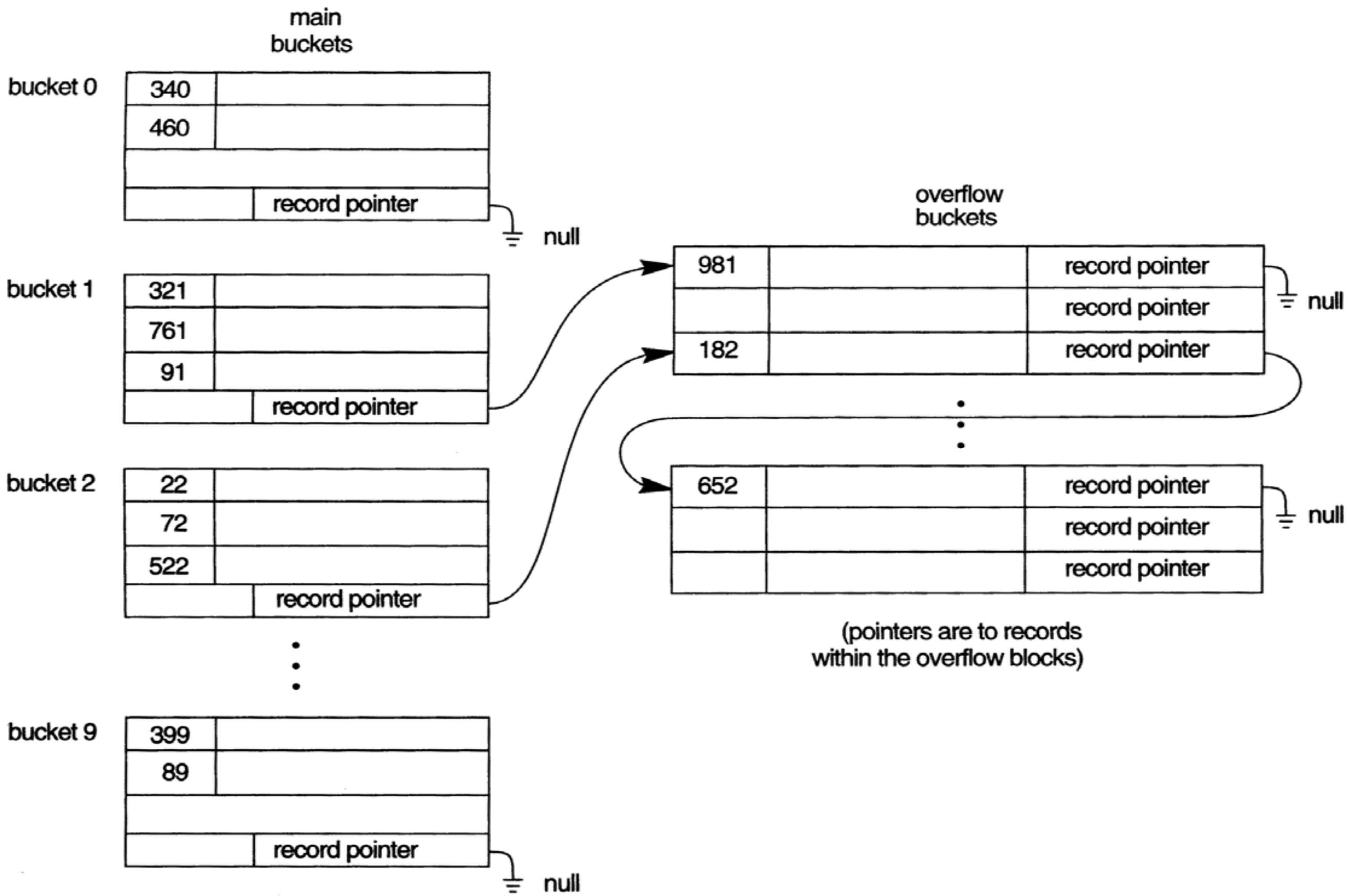
# ... - Hashed Files ...



# ... - Hashed Files

- To reduce overflow records, a hash file is typically kept 70-80% full.
- The hash function h should distribute the records uniformly among the buckets; otherwise, search time will be increased because many overflow records will exist.
- Main disadvantages of static external hashing:
  - Fixed number of buckets M is a problem if the number of records in the file grows or shrinks.
  - Ordered access on the hash key is quite inefficient (requires sorting the records).

# -- Hashed Files - Overflow handling



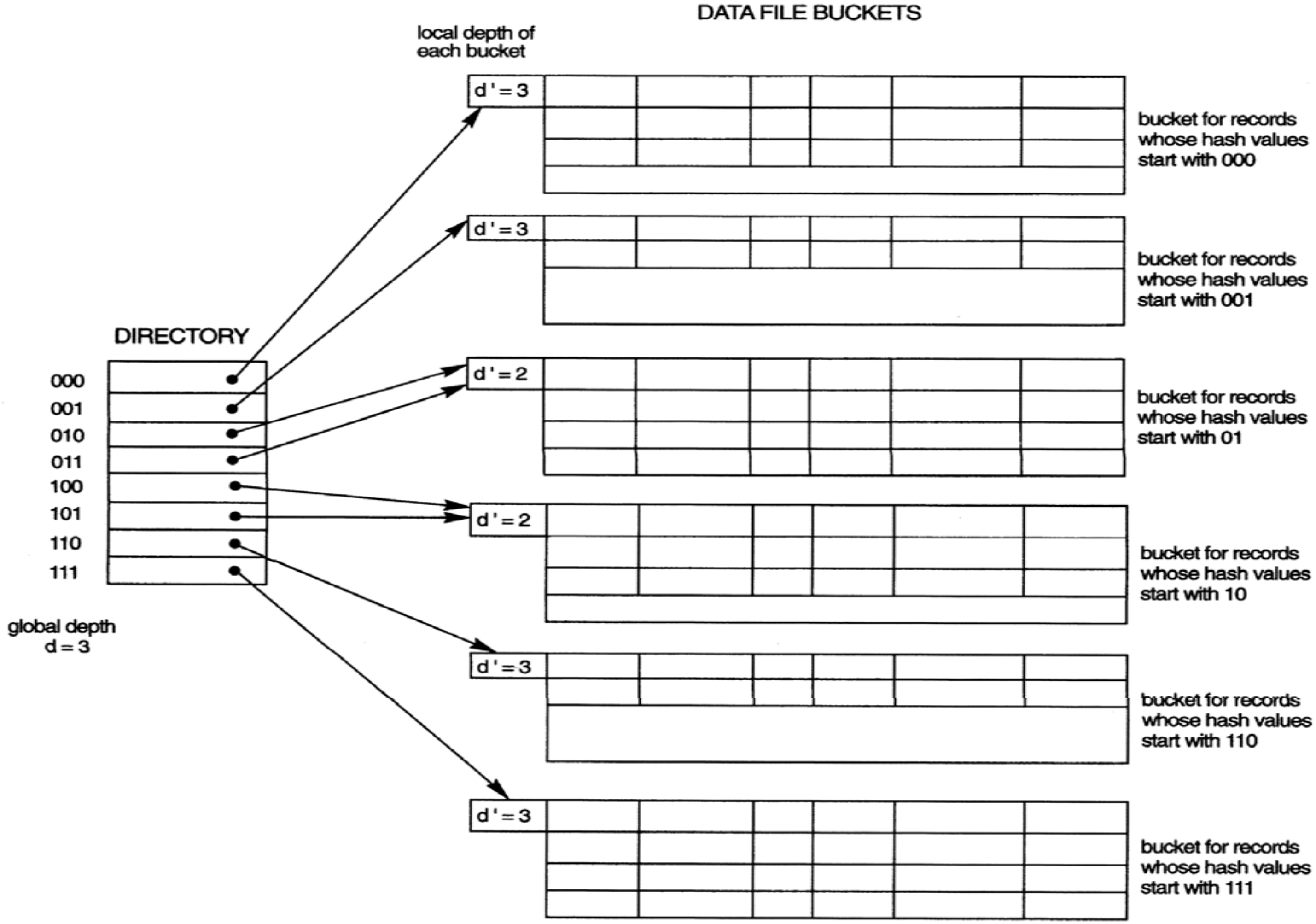
# -- Dynamic And Extendible Hashed Files ...

- Dynamic and Extendible Hashing Techniques
  - Hashing techniques are adapted to allow the dynamic growth and shrinking of the number of file records.
  - These techniques include the following: *dynamic hashing* , *extendible hashing* , and *linear hashing* .
  - Both dynamic and extendible hashing use the *binary representation* of the hash value h(K) in order to access a *directory*. In dynamic hashing the directory is a binary tree. In extendible hashing the directory is an array of size $2^d$ where d is called the *global depth.*

# … -- Dynamic And Extendible Hashing

- The directories can be stored on disk, and they expand or shrink dynamically. Directory entries point to the disk blocks that contain the stored records.
- An insertion in a disk block that is full causes the block to split into two blocks and the records are redistributed among the two blocks. The directory is updated appropriately.
- Dynamic and extendible hashing do not require an overflow area.
- Linear hashing does require an overflow area but <u>does not</u> use a directory. Blocks are split in *linear order* as the file expands.

# --- Extendible Hashing

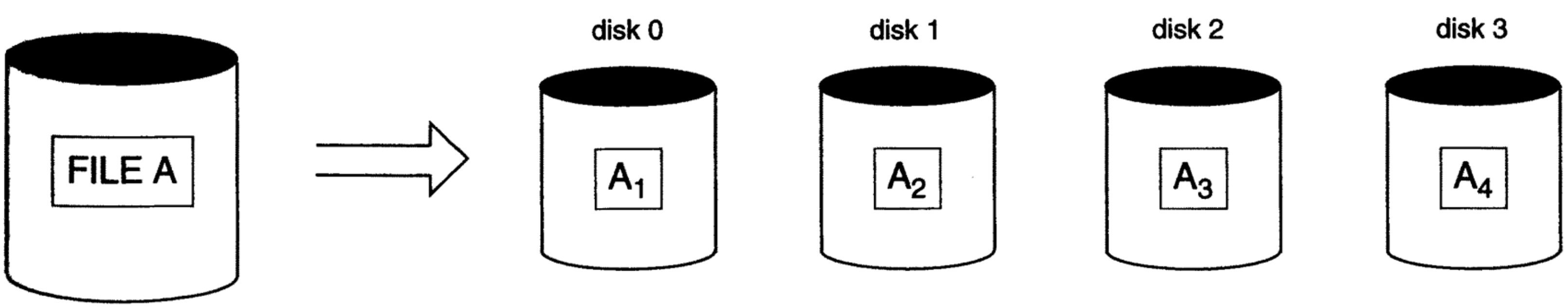


# - RAID Technology ...

- Secondary storage technology must take steps to keep up in performance and reliability with processor technology.
- A major advance in secondary storage technology is represented by the development of **RAID**, which originally stood for **Redundant Arrays of Inexpensive Disks.**
- The main goal of RAID is to even out the widely different rates of performance improvement of disks against those in memory and microprocessors.

# ... - RAID Technology

- A natural solution is a large array of small independent disks acting as a single higher-performance logical disk. A concept called **data striping** is used, which utilizes *parallelism* to improve disk performance.
- Data striping distributes data transparently over multiple disks to make them appear as a single large, fast disk.



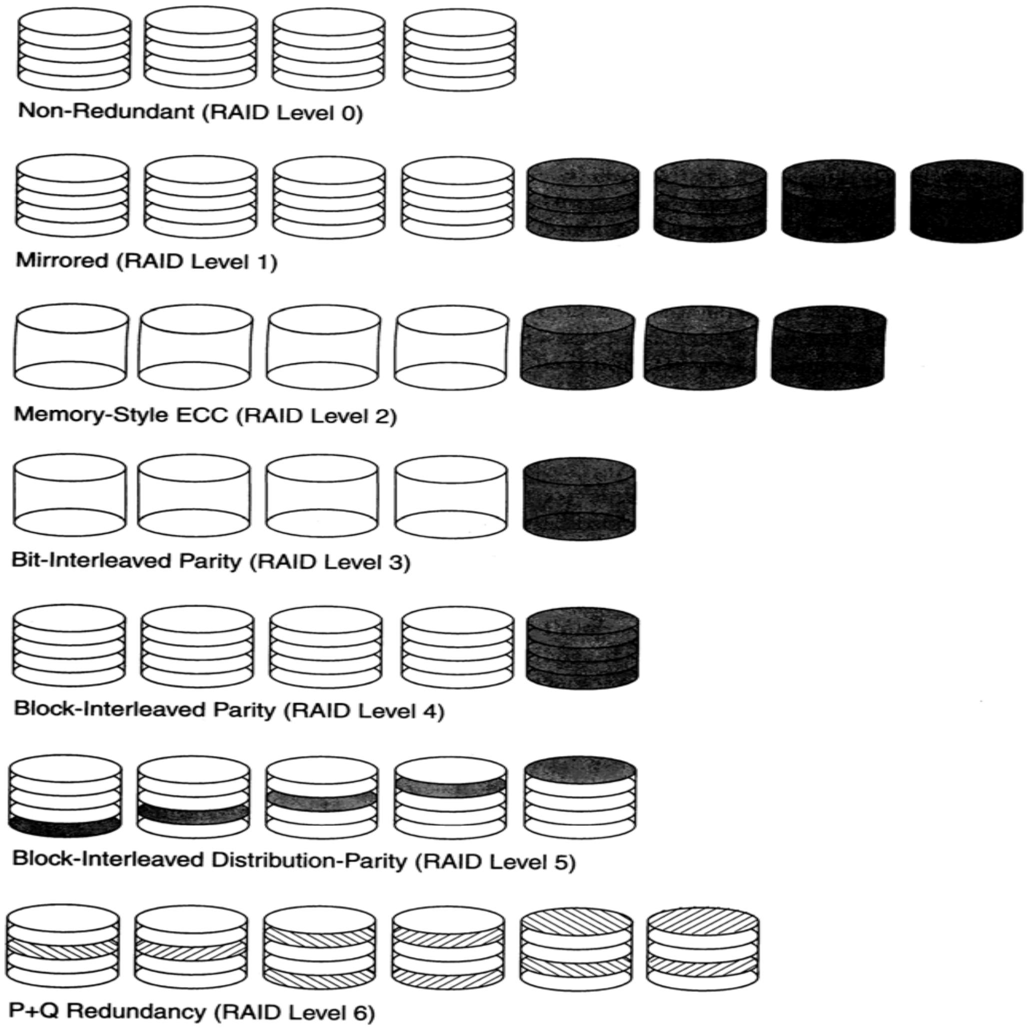
# ... - RAID Technology ...

- Different raid organizations were defined based on different combinations of the two factors of granularity of data interleaving (striping) and pattern used to compute redundant information.
  - Raid level 0 has no redundant data and hence has the best write performance.
  - Raid level 1 uses mirrored disks.
  - Raid level 2 uses memory-style redundancy by using Hamming codes, which contain parity bits for distinct overlapping subsets of components. Level 2 includes both error detection and correction.
  - Raid level 3 uses a single parity disk relying on the disk controller to figure out which disk has failed.
  - Raid Levels 4 and 5 use block-level data striping, with level 5 distributing data and parity information across all disks.
  - Raid level 6 applies the so-called $P + Q$ redundancy scheme using Reed-Soloman codes to protect against up to two disk failures by using just two redundant disks.

# ... - RAID Technology ...

- Different raid organizations are being used under different situations
  - Raid level 1 (mirrored disks)is the easiest for rebuild of a disk from other disks
    - It is used for critical applications like logs
  - Raid level 2 uses memory-style redundancy by using Hamming codes, which contain parity bits for distinct overlapping subsets of components. Level 2 includes both error detection and correction.
  - Raid level 3 ( single parity disks relying on the disk controller to figure out which disk has failed) and level 5 (block-level data striping) are preferred for Large volume storage, with level 3 giving higher transfer rates.
  - Most popular uses of the RAID technology currently are: Level 0 (with striping), Level 1 (with mirroring) and Level 5 with an extra drive for parity.
  - Design Decisions for RAID include – level of RAID, number of disks, choice of parity schemes, and grouping of disks for block-level striping.

# ... - RAID Technology ...



# - Storage Area Networks ...

- The demand for higher storage has risen considerably in recent times.
- Organizations have a need to move from a static fixed data center oriented operation to a more flexible and dynamic infrastructure for information processing.
- Thus they are moving to a concept of Storage Area Networks (SANs). In a SAN, online storage peripherals are configured as nodes on a high-speed network and can be attached and detached from servers in a very flexible manner.
- This allows storage systems to be placed at longer distances from the servers and provide different performance and connectivity options.

# ... - Storage Area Networks

- Advantages of SANs are:
  - Flexible many-to-many connectivity among servers and storage devices using fiber channel hubs and switches.
  - Up to 10km separation between a server and a storage system using appropriate fiber optic cables.
  - Better isolation capabilities allowing nondisruptive addition of new peripherals and servers.
- SANs face the problem of combining storage options from multiple vendors and dealing with evolving standards of storage management software and hardware.

# END