IF3140 – Sistem Basis Data

Storage and File Structure

SEMESTER II TAHUN AJARAN 2024/2025

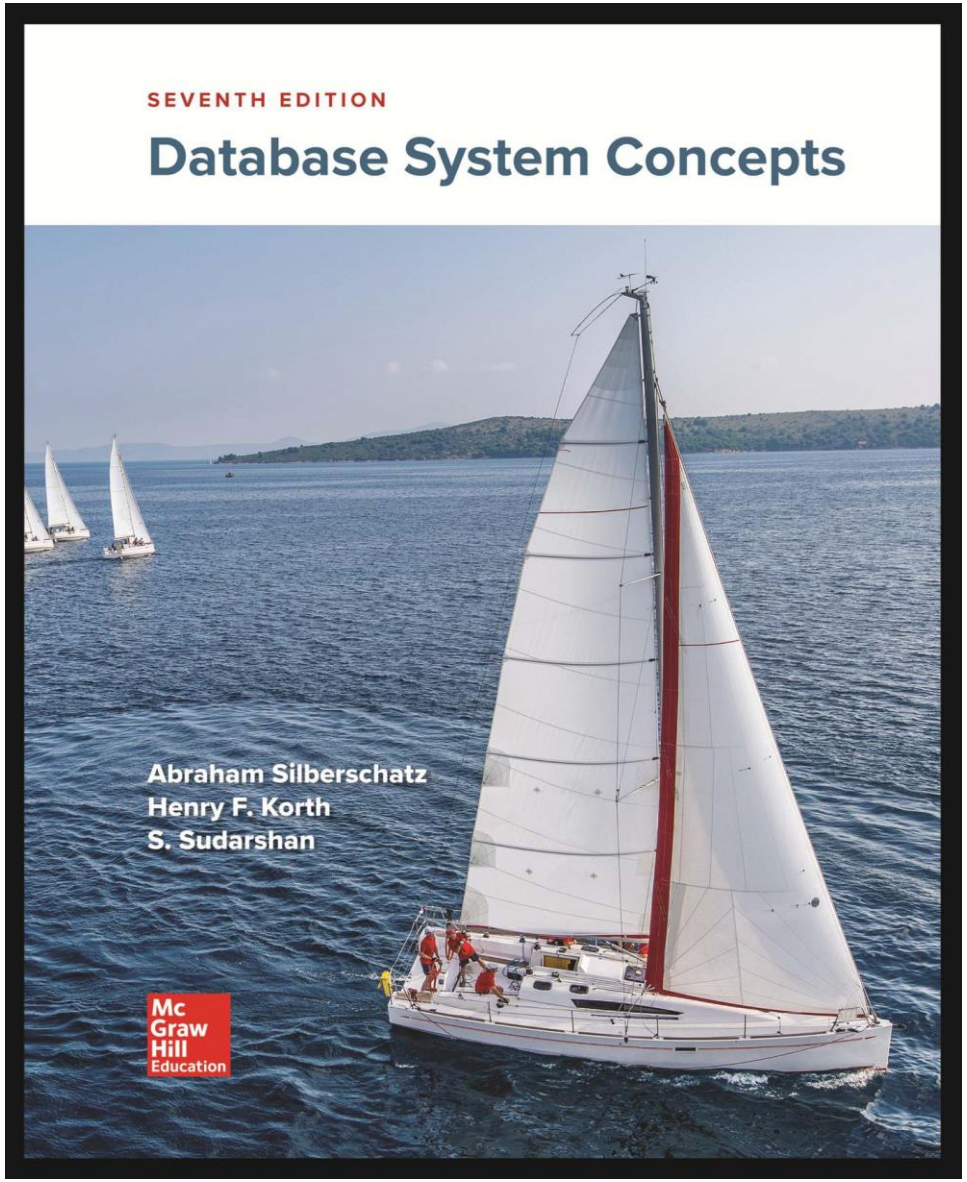
KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

Sumber

Silberschatz, Korth, Sudarshan: "Database System Concepts", 7th Edition

- **Chapter 12:** Physical Storage Systems
- **Chapter 13:** Data Storage Structures



KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides, "Database System Concepts", 7th ed.

Objectives

Mahasiswa mampu:

- menjelaskan media penyimpanan data secara fisik
- menjelaskan organisasi file, bagaimana *record* disimpan dalam *file*,
- Menjelaskan tantangan terhadap kinerja basis data berkaitan dengan bagaimana data disimpan.

KNOWLEDGE & SOFTWARE ENGINEERING

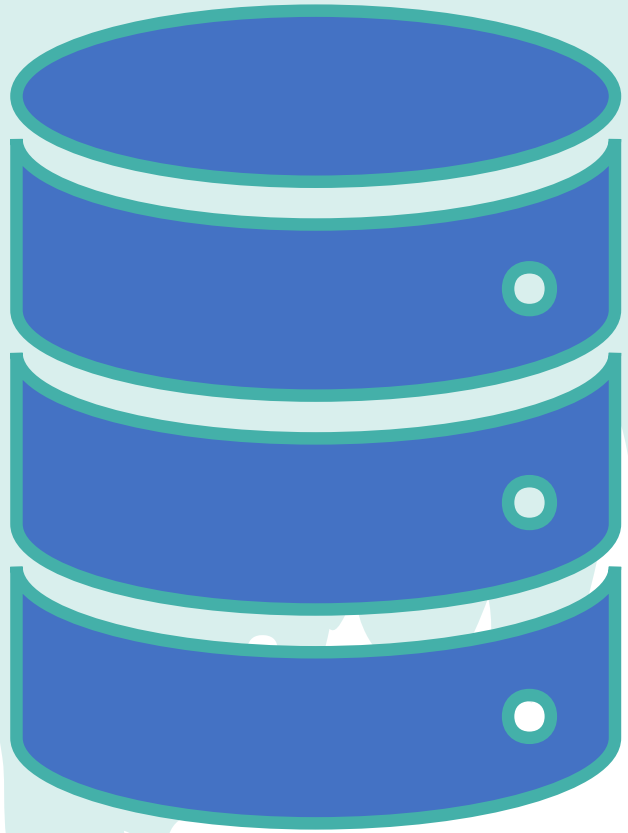
Outline

- Classification of Physical Storage Media
- Performance Measures of Disks
- Disk Block Access
- File Organization
- Organization of Records in Files
- Data Dictionary Storage

KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides, "Database System Concepts", 7th ed.

Physical Storage Systems



KNOWLEDGE & SOFTWARE ENGINEERING

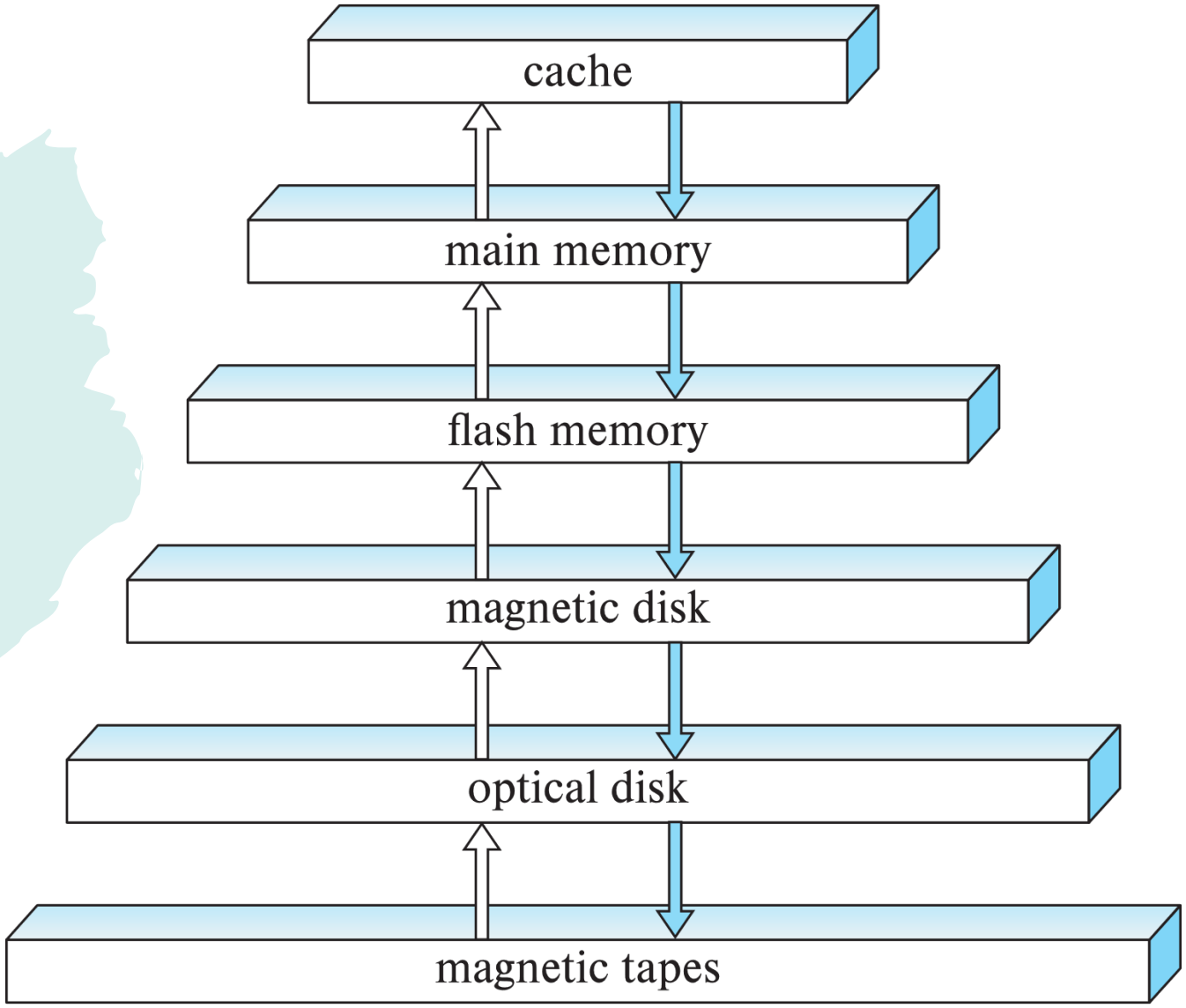
Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

Classification of Physical Storage Media

- Different types of storage:
  - **volatile storage**: loses contents when power is switched off (ex : RAM)
  - **non-volatile storage**: (ex: SSD)
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as battery-backed up main-memory.
- Factors affecting choice of storage media include
  - Speed with which data can be accessed
  - Cost per unit of data
  - Reliability

KNOWLEDGE & SOFTWARE ENGINEERING

Storage Hierarchy

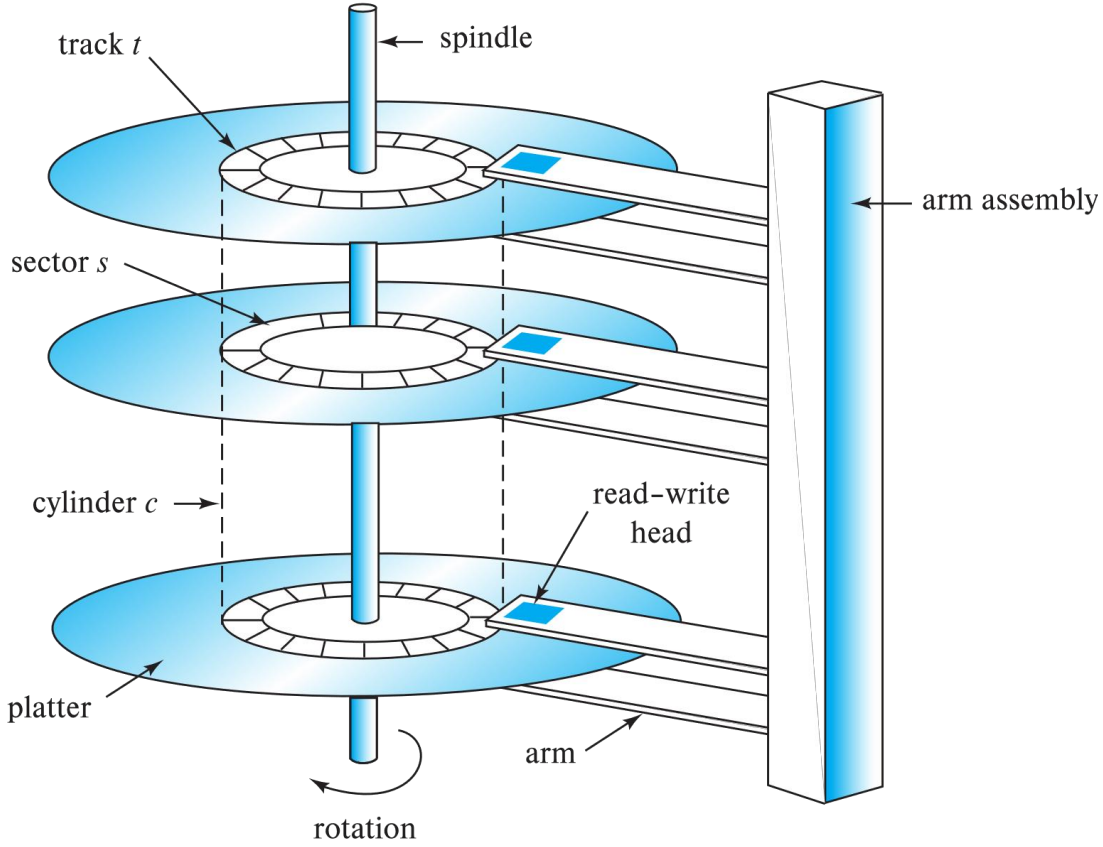


KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

Magnetic Hard Disk Mechanism

- Surface of platter divided into circular **tracks**
- Each track is divided into **sectors**
  - A sector is the smallest unit of data that can be read or written
- Read-Write Head
- Head-disk assemblies
- **Cylinder *i*** consists of ***i***-th track of all the platters

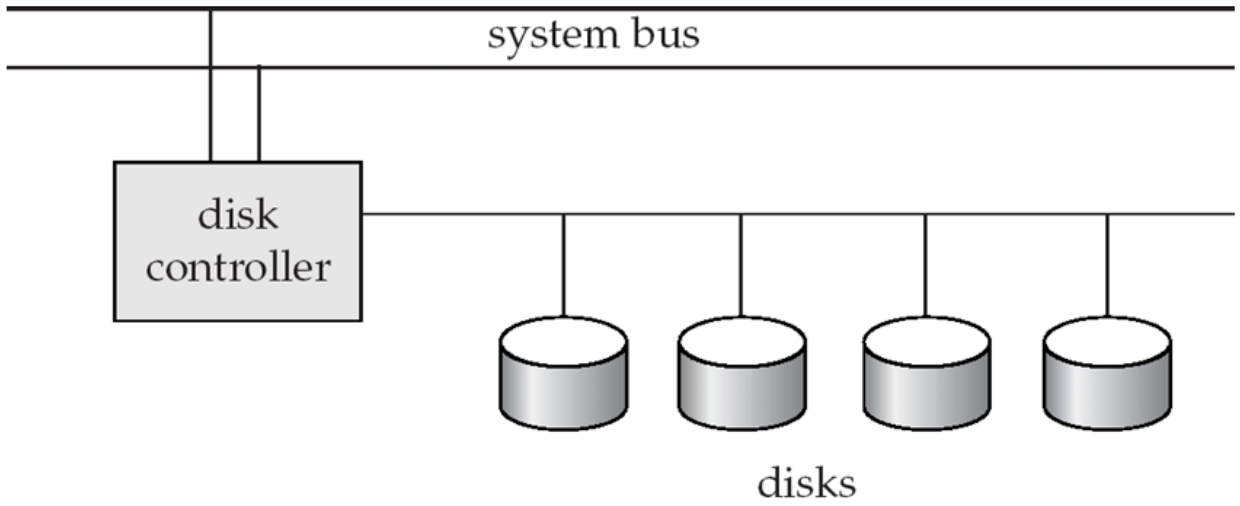


Schematic diagram of magnetic disk drive

KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

Disk Subsystem



- **Disk controller** – interfaces between the computer system and the disk drive hardware.
  - Accepts high-level commands to read or write a sector
  - Initiates actions such as moving the disk arm to the right track and actually reading or writing the data
  - Computes and attaches **checksums** to each sector to verify that data is read back correctly
  - Performs **remapping of bad sectors**
- Multiple disks connected to a computer system through a disk controller

KNOWLEDGE & SOFTWARE ENGINEERING

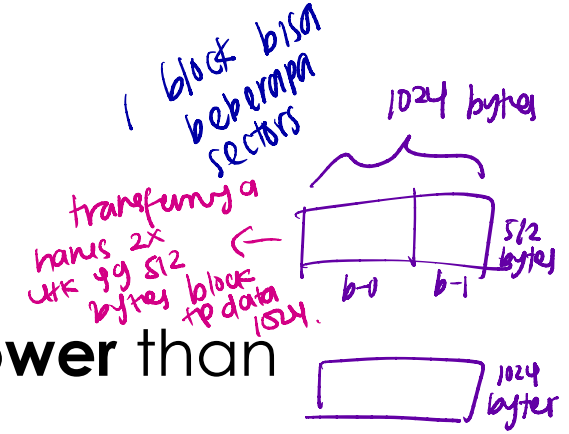
Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins, consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk
- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.
  - Typically 3 to 5 years
  - MTTF decreases as disk ages

KNOWLEDGE & SOFTWARE ENGINEERING

Disk Block Access

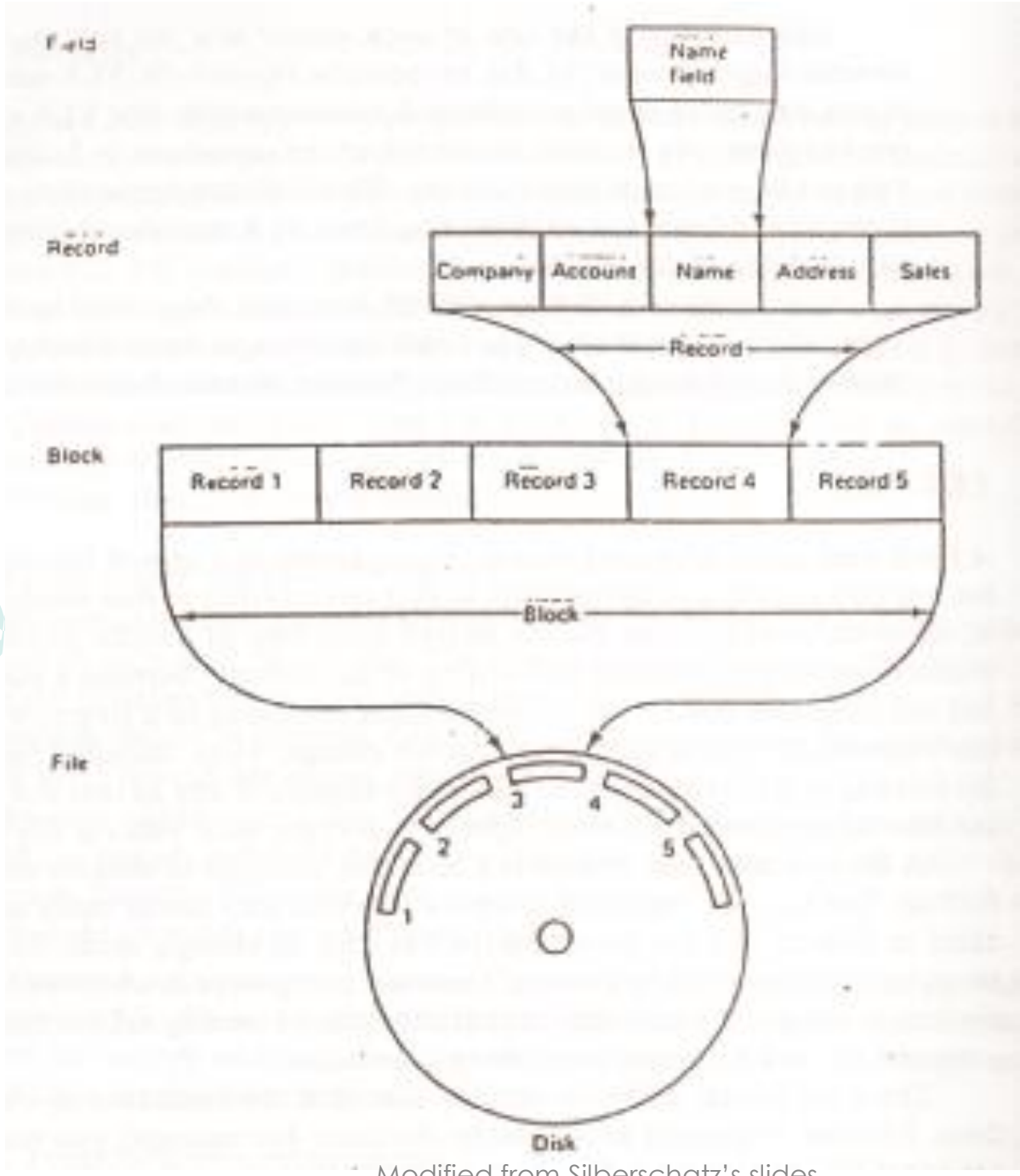
- A database file is partitioned into fixed-length storage units called **blocks**. Blocks are units of both storage allocation and data transfer.
- **Block** – a contiguous sequence of sectors from a single track
  - data is transferred between disk and main memory in blocks
  - sizes range from 512 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



- Access to data on disk is several orders of magnitude **slower** than access to data in main memory
- 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

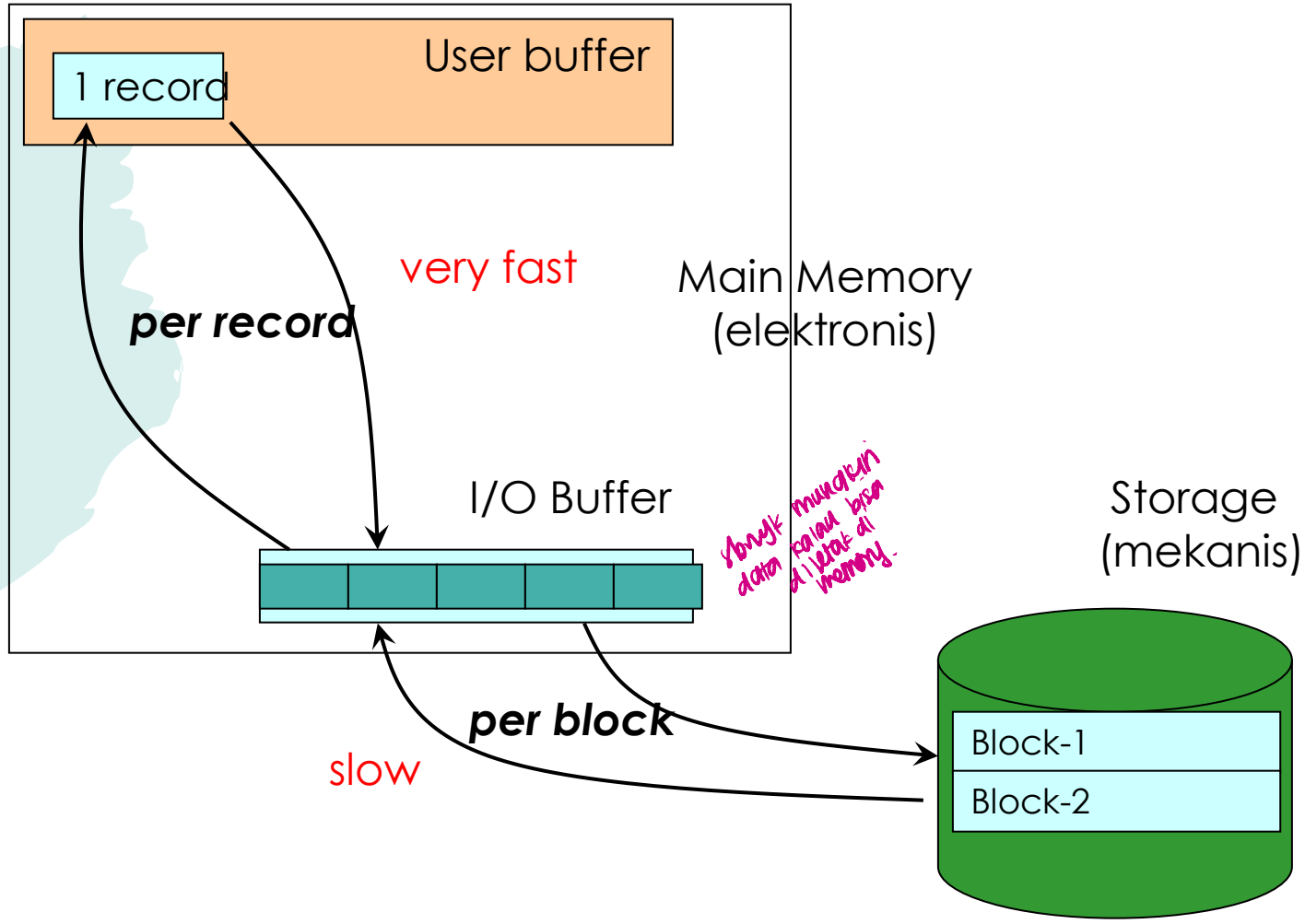
KNOWLEDGE & SOFTWARE ENGINEERING

Fields, Records, and Files



Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

Record Access Mechanism



KNOWLEDGE & SOFTWARE ENGINEERING

Optimization of Disk Block Access

- **Buffering of blocks**
- Disk arm scheduling algorithms
- **File organization**
- Nonvolatile write buffers
- Log disk

KNOWLEDGE & SOFTWARE ENGINEERING

Optimization of Disk Block Access

- **Buffering of blocks** in memory to satisfy future requests
  - **Buffer** – portion of main memory available to store copies of disk blocks.
  - **Buffer manager** – subsystem responsible for allocating buffer space in main memory.
- Programs call on the buffer manager when they need a block from disk.
  - If the block is already in the buffer, buffer manager returns the address of the block in main memory
  - If the block is not in the buffer, the buffer manager:
    - Allocates space in the buffer for the block
      - Replacing (throwing out) some other block, if required, to make space for the new block.
      - 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.
    - Reads the block from the disk to the buffer and returns the address of the block in main memory to requester

KNOWLEDGE & SOFTWARE ENGINEERING

Optimization of Disk Block Access

- **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 cylinders.
  - Files may get fragmented over time
    - E.g. if data is inserted to/deleted from the file
    - Or free blocks on disk are scattered, and newly created file has its blocks scattered over the disk
    - Sequential access to a fragmented file results in increased disk arm movement
  - Some systems have utilities to defragment the file system, in order to speed up file access
  - File organization… next.

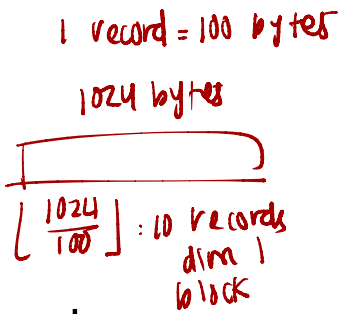
File Organization

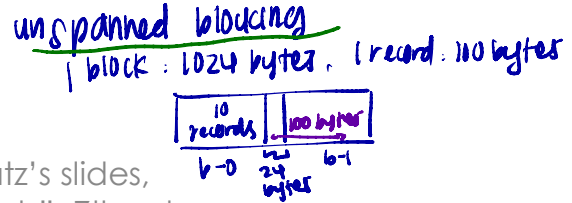
KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides,
"Database System Concepts", 7th ed.

File Organization

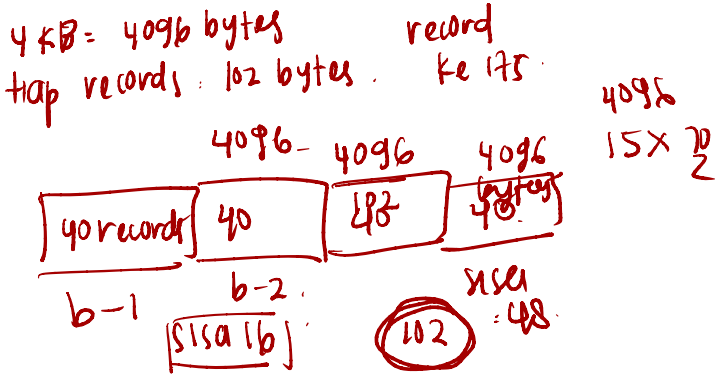
- The database is stored as a collection of files. Each file is a sequence of records. A record is a sequence of fields.
- We assume that records are smaller than a disk block
- The fitting of records into blocks is referred to as **blocking**.
  - The blocking can be *spanned* or *unspanned*. Unspanned blocking requires all records fit within blocks.
  - **Blocking factor**, denoted by **Bfr**, is the number of records expected within a block
  - For unspanned fixed blocking: Bfr = $\lfloor B/R \rfloor$ where B is the block size and R is the record size





most of the time pakai unspanned

Fixed-Length Records



- Simple approach:
  - Store record i starting from byte n * i, where n is the size of each record and i starts from 0.
  - 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, …, n to i, …, 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	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

Fixed-Length Records

- Deletion of record i: alternative-1
  - move records i + 1, …, n to i, …, n –1
- Deletion of record i: alternative-2
  - move record n to i

Record 3 deleted

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

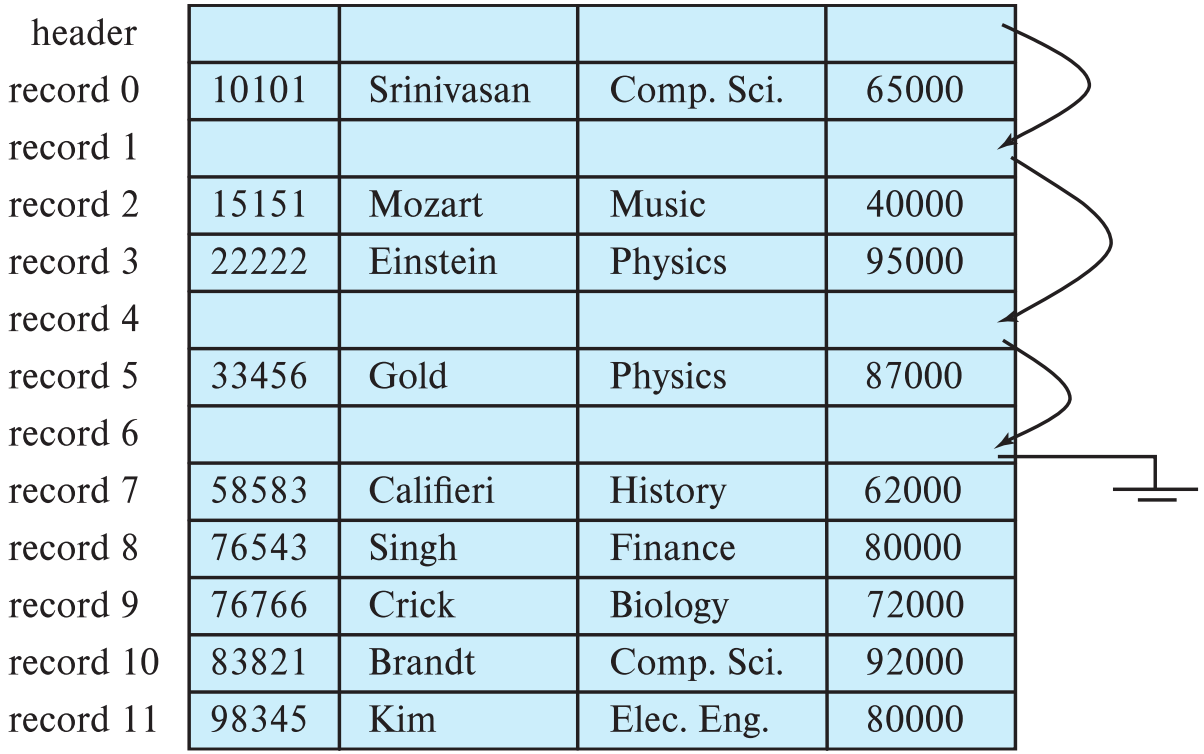
Record 3 deleted and replaced by record 11

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
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

KNOWLEDGE & SOFTWARE ENGINEERING

Fixed-Length Records

- Deletion of record i: alternative-3:
  - do not move records, but link all free records on a free list
  - At the beginning of the file, we allocate a certain number of bytes as a file **header**
    - Contains information e.g. the address of the first record whose contents are deleted
  - The deleted records form a linked list often referred to as a **free list**

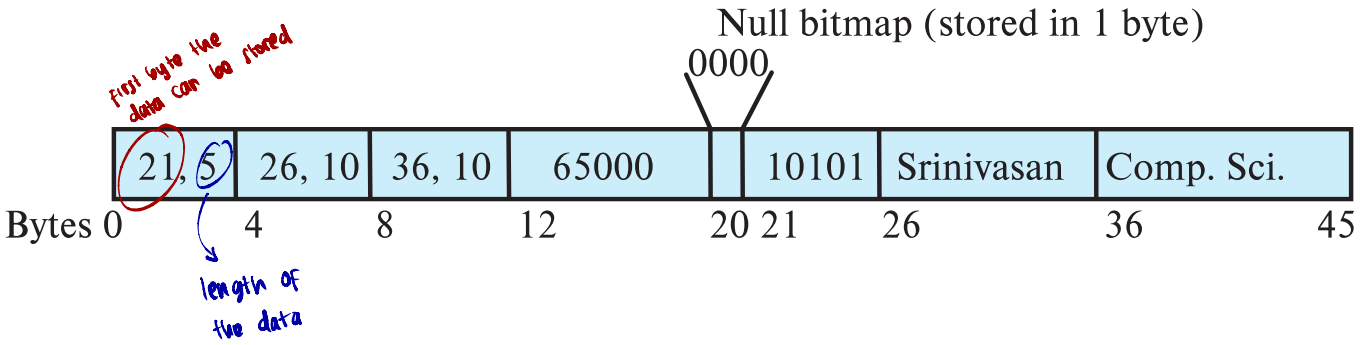


header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology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

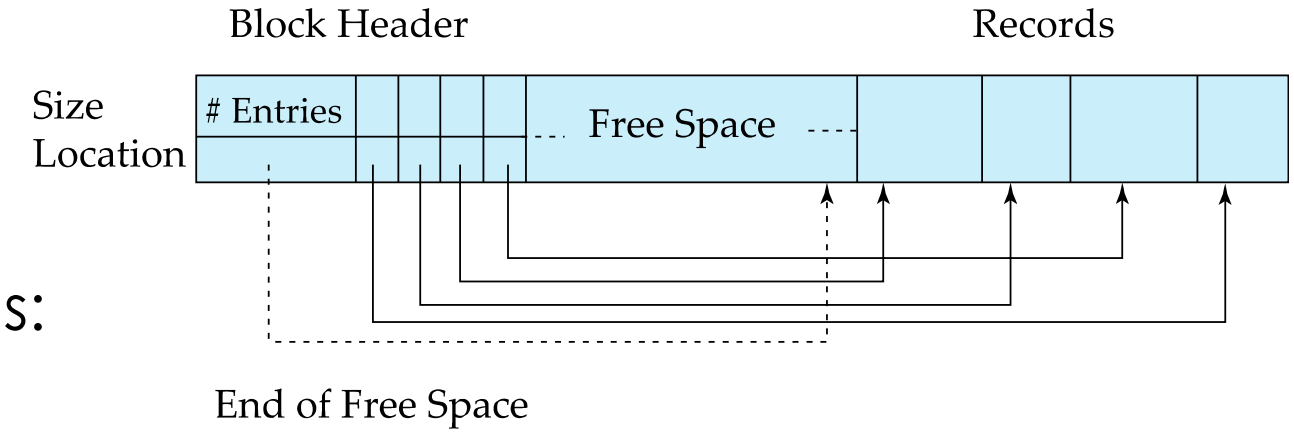
KNOWLEDGE & SOFTWARE ENGINEERING

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



Variable-Length Records: Slotted Page Structure



- **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.

KNOWLEDGE & SOFTWARE ENGINEERING

File Performance Parameters

- R: The amount of storage required for a record
- T_F: The time needed to fetch an arbitrary record from the file
- T_N: The time needed to get the next record within the file
- T_I: The time required to update the file by inserting a record
- T_U: The time required to update the file by changing a record
- T_X: The time needed for exhaustive reading of the entire file
- T_Y: The time needed for reorganization of the file

Record Size

- Beside the data, a record may also stores descriptive and access information.
- Descriptive information is advantageous when storing heterogeneous data.

Fetch A Record

To be able to use data from a file, a record containing the data has to be read into the memory.

- Fetch is an *associative* retrieval of a data element based on a key value.
- Fetching a record consists of 2 steps:
  - Locating the position of the record
  - The actual reading
- In general, the records of a file cannot be directly located on the basis of a subscript value or record number
  - Therefore, a simple address computation cannot be performed

Get the Next Record

Information is mainly generated by relating one fact with another; this implies getting the next record according to some criterion.

- Get-Next is a retrieval using a *structural dependency*.
- A successor record can be obtained most rapidly when related data is kept together; that is, when the locality of these data is strong.
  - When the records are not physically ordered according to the criterion, Get-Next is time consuming.

Insert A Record

Most applications require regular insertion of new data records into their files to remain up to date.

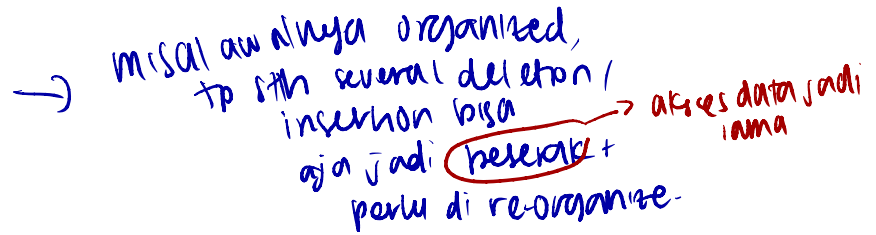
- Inserting a record comprises of:
  - Locating the block where the record will be inserted
  - Reading the block
  - Rewriting the changed block
- When a record has to be put in a specific place within the file, other records may have to be shifted or modified.
- Insertions to the end of the file, Append, are easier to handle.
  - If the address of the last block of the file is known, the performance of this operation can be improved.

Update A Record

All changes of data do not require insertion of a new record.

- When data within a stored record must be changed, the new, updated record is created.
  - The old data and the changes are merged to create a new record,
  - The new record is inserted into the position of the old record within the block, and
  - The block is rewritten into the file.
- If the record has grown in size, it may not be possible to use the old position → delete the old record and insert a new one.
- Deleting a record is not performed immediately, but instead the record to be deleted is marked with an indicator that the record is now invalid → the delete is now converted to an update.

Reorganize the File



Reorganization removes deleted and invalidated records, reclaims the space for new data, and restores clustering.

- It is especially important for file organizations which create tombstones for deletion and updates.
- Its frequency depends on the type of file organization used but varies greatly with the application.
- It has many similarities with "garbage collection".

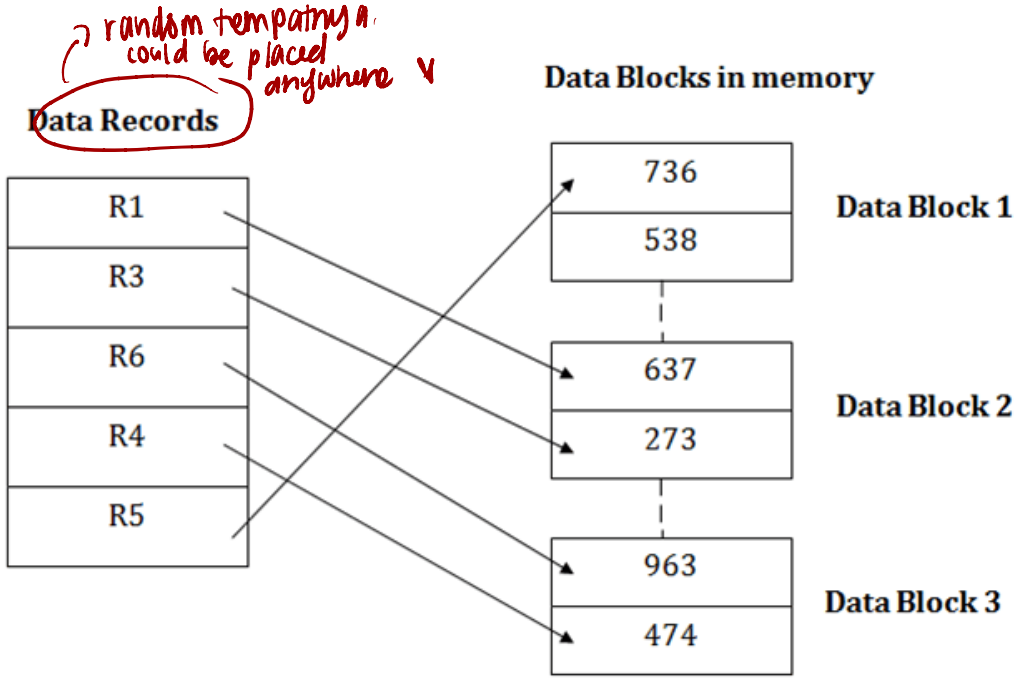
Organization of Records in Files

- **Heap** – 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
- 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
- B^+-tree file organization
  - Ordered storage even with inserts/deletes
  - More on this in next meeting
- Hashing – a hash function computed on search key; the result specifies in which block of the file the record should be placed
  - More on this in next meeting

KNOWLEDGE & SOFTWARE ENGINEERING

Heap File Organization

- Heap files are lists of unordered records of variable size.
- Records can be placed anywhere in the file where there is free space
- Records usually do not move once allocated
- Important to be able to efficiently find free space within file



KNOWLEDGE & SOFTWARE ENGINEERING

Heap File Organization

- Advantages
  - efficient for bulk loading data
  - efficient for relatively small relations as indexing overheads are avoided
  - efficient when retrievals involve large proportion of stored records
- Disadvantages
  - not efficient for selective retrieval using key values, especially if large
  - sorting may be time-consuming
  - not suitable for volatile tables

tabel yg cuman dibutuhkan sementara, abistu diapus

susah for searching, bcs dia acak ^^

KNOWLEDGE & SOFTWARE ENGINEERING

Sequential File Organization

Efficient for system that mostly does sequential data searching

- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key
  - Need not be a primary key or even a superkey

biasanya primary key, tapi gn harus.

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	

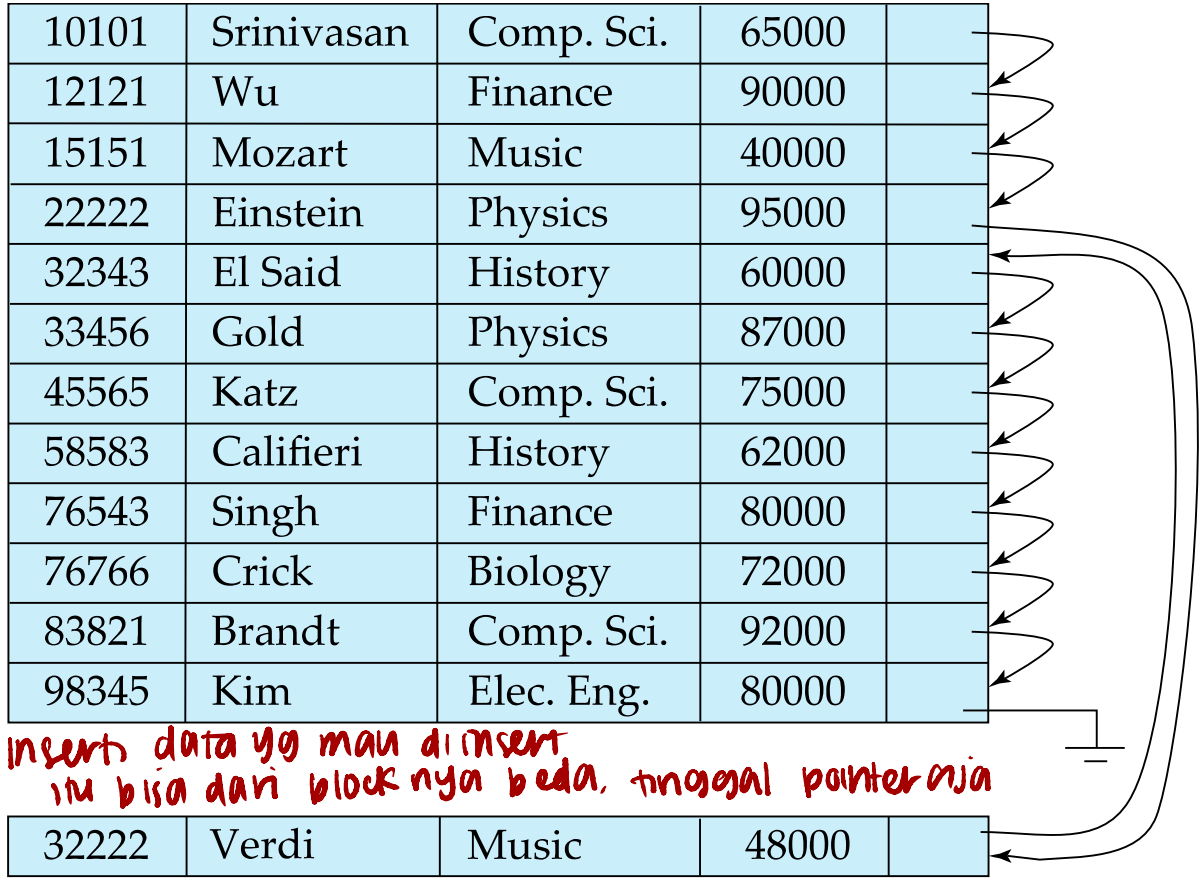
Kenut di dalam disk, bukan di dalam memory.

KNOWLEDGE & SOFTWARE ENGINEERING

Sequential File Organization (Cont.)

literally linked list :)

- **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	Verdi	Music	48000	

Jadi kalau insert data yg mau di insert itu bisa dari block nya beda, tinggal pointer aja

KNOWLEDGE & SOFTWARE ENGINEERING

Multitable Clustering File Organization

↳ kalau 2 data sering di join, pakai multitable clustering biar dalam 1 block.

- Store several relations in one file using a **multitable clustering** file organization
- good for queries involving: → terlalu complex kalo datanya aneh.
  - department ⋈ instructor, and
  - one single department and its instructors
- bad for queries involving only department → karena harus ambil 1 block padhl butuh department doang.
- results in variable size records
- Can add pointer chains to link records of a particular relation

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	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

KNOWLEDGE & SOFTWARE ENGINEERING

Data Dictionary Storage

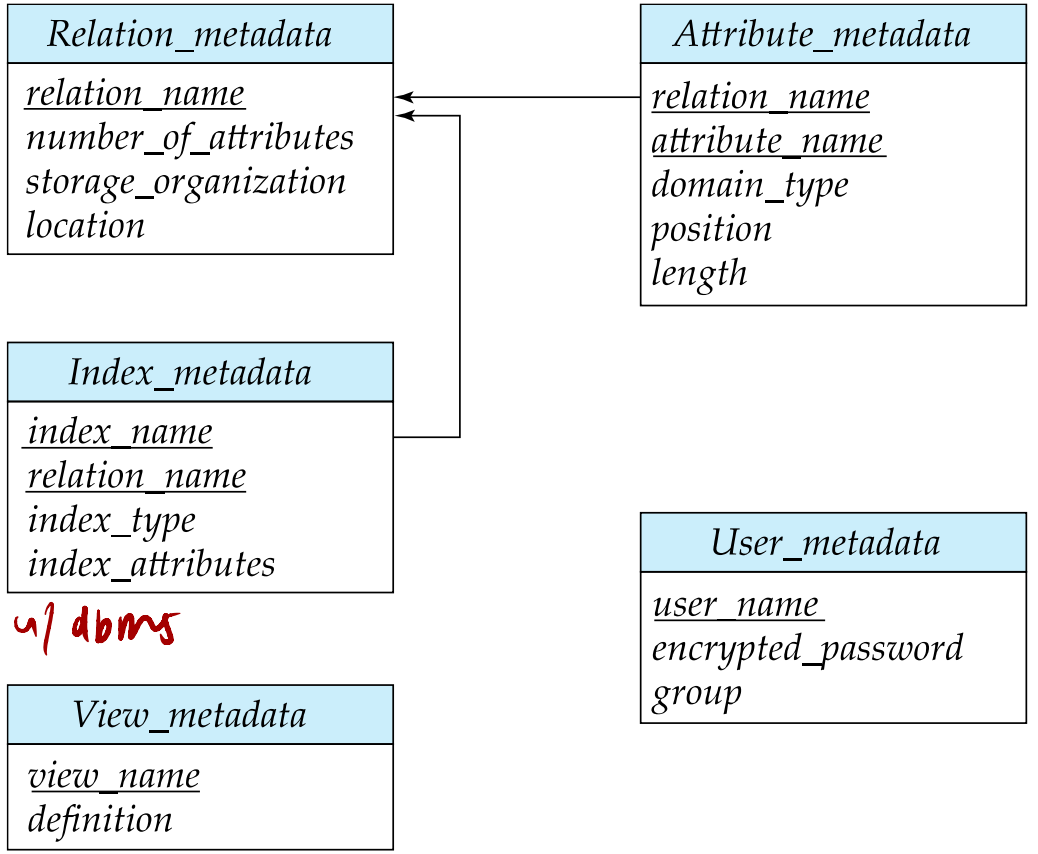
- The **Data dictionary** (also called **system catalog**) stores metadata; that is, data about data, such as
  - Information about relations
    - names of relations
    - names, types and lengths of attributes of each relation
    - names and definitions of views
    - integrity constraints
  - User 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 (next meeting)

KNOWLEDGE & SOFTWARE ENGINEERING

Relational Representation of System Metadata

→ mapping relasi ke dlm relasi

- Relational representation on disk
- Specialized data structures designed for efficient access, in memory



information schema

↳ punya > 1 view yg data bisa diambil u/ dbms

ex : select * from information_schema.tables

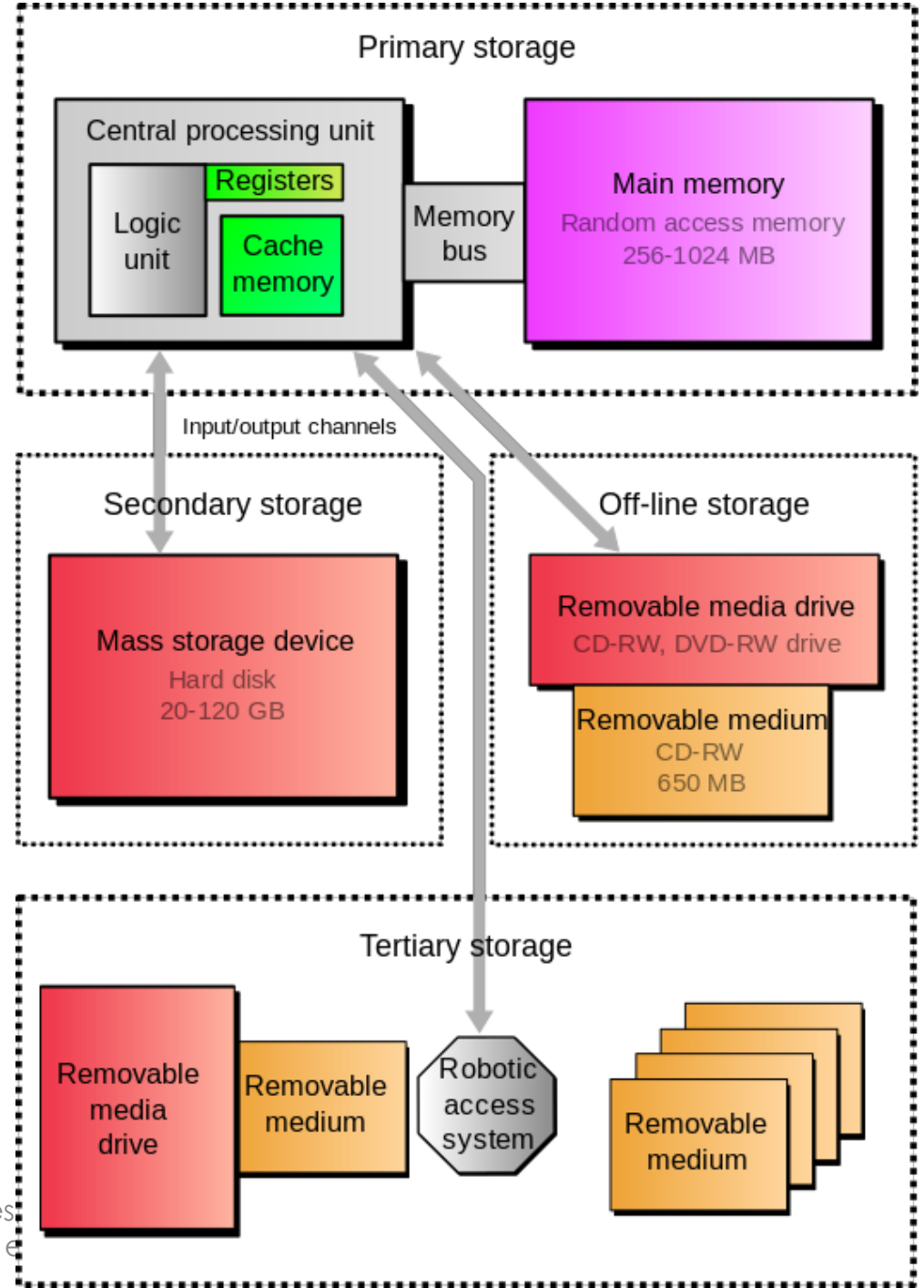
KNOWLEDGE & SOFTWARE ENGINEERING

Additional Materials

KNOWLEDGE & SOFTWARE ENGINEERING

Storage Hierarchy

- **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** and used for archival storage
  - e.g., magnetic tape, optical storage
  - Magnetic tape
    - Sequential access, 1 to 12 TB capacity
    - A few drives with many tapes
    - Juke boxes with petabytes (1000's of TB) of storage



KNOWLEDGE & SOFTWARE ENGINEERING

Modified from Silberschatz's slides
"Database System Concepts", 7th e

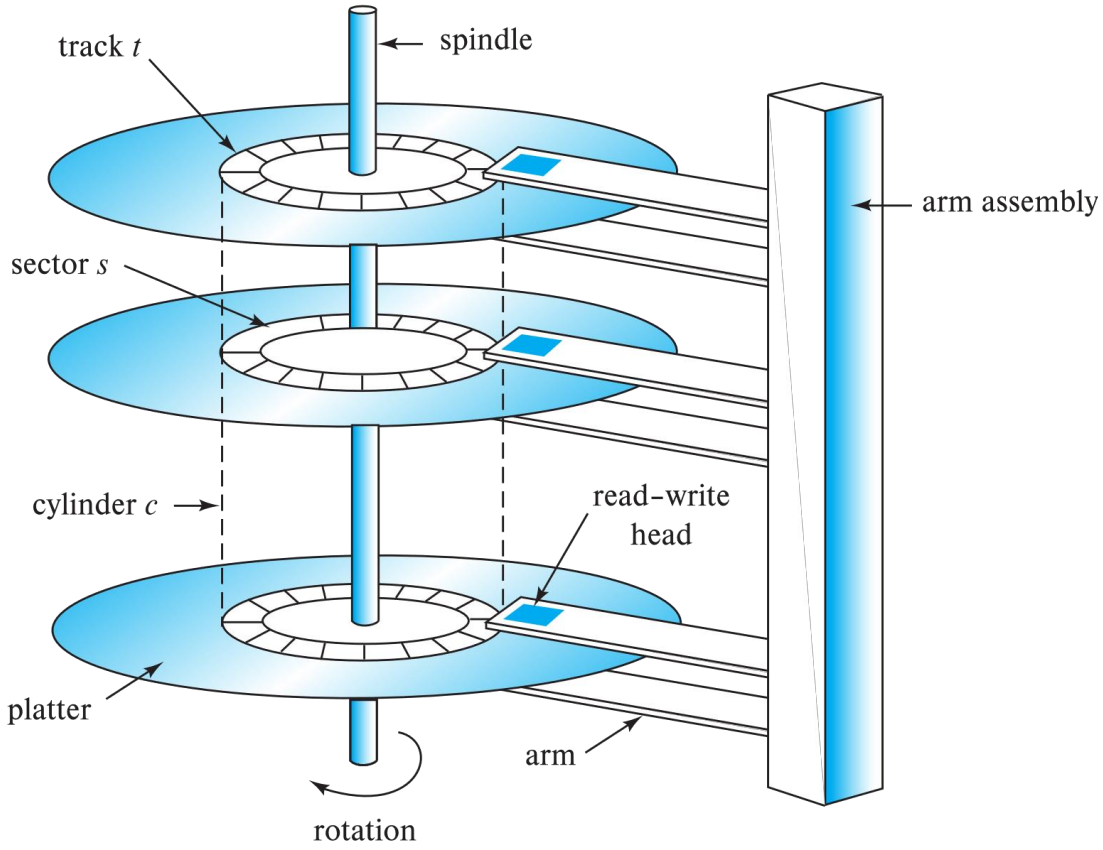
A hard disk drive with protective cover removed
By Evan-Amos - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27941467

A large tape library, with tape cartridges placed on shelves in the front, and a robotic arm moving in the back. Visible height of the library is about 180 cm.
Austin Mills from Austin, TX, USA [CC BY-SA] (https://creativecommons.org/licenses/by-sa/2.0)]

KNOWLEDGE & SOFTWARE ENGINEERING

Magnetic Hard Disk Mechanism

- Surface of platter divided into circular **tracks**
- Each track is divided into **sectors**
  - A sector is the smallest unit of data that can be read or written
- Read-Write Head
- Head-disk assemblies
- **Cylinder *i*** consists of ***i***-th track of all the platters



Schematic diagram of magnetic disk drive

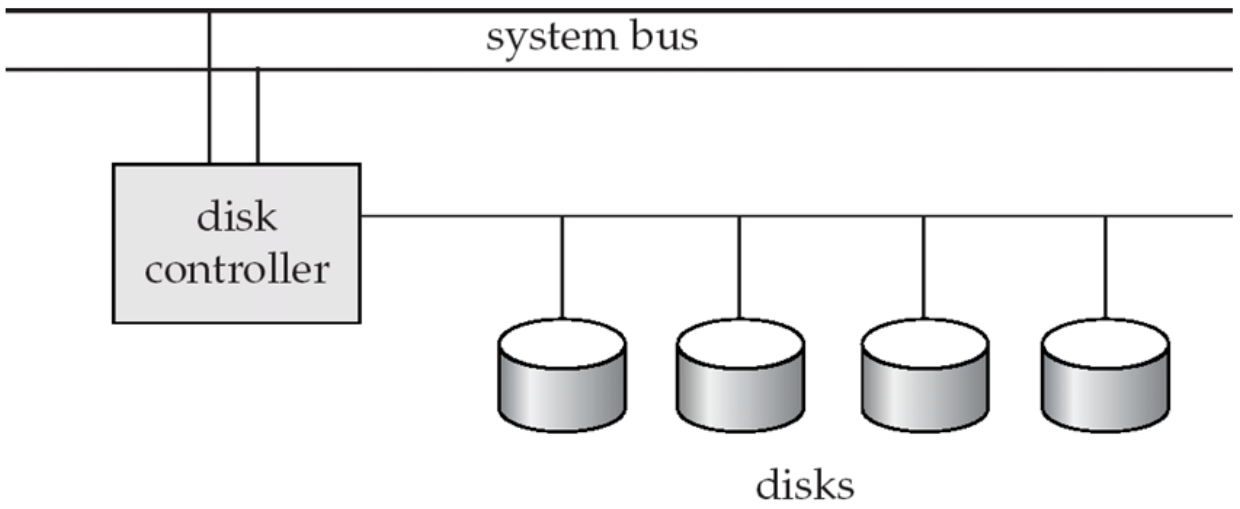
KNOWLEDGE & SOFTWARE ENGINEERING

Magnetic Disks

- **Disk controller** – interfaces between the computer system and the disk drive hardware.
  - accepts high-level commands to read or write a sector
  - initiates actions such as moving the disk arm to the right track and actually reading or writing the data
  - Computes and attaches **checksums** to each sector to verify that data is read back correctly
    - If data is corrupted, with very high probability stored checksum won't match recomputed checksum
    - Ensures successful writing by reading back sector after writing it
  - Performs **remapping of bad sectors**

KNOWLEDGE & SOFTWARE ENGINEERING

Disk Subsystem



- Multiple disks connected to a computer system through a disk controller
  - Controllers functionality (checksum, bad sector remapping) often carried out by individual disks; reduces load on controller
- Disk interface standards families
  - **ATA** (AT Attachment) range of standards
  - **SATA** (Serial ATA)
  - **SCSI** (Small Computer System Interconnect) range of standards
  - **SAS** (Serial Attached SCSI)
  - Several variants of each standard (different speeds and capabilities)

KNOWLEDGE & SOFTWARE ENGINEERING

Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins, consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track
    - Average seek time is 1/2 the worst case seek time.
      - Would be 1/3 if all tracks had the same number of sectors, and we ignore the time to start and stop arm movement
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head
    - 4 to 11 milliseconds on typical disks (5400 to 15000 r.p.m.)
    - Average latency is 1/2 of the above latency.
  - Overall latency is 5 to 20 msec depending on disk model
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk
  - 25 to 200 MB per second max rate, lower for inner tracks

KNOWLEDGE & SOFTWARE ENGINEERING

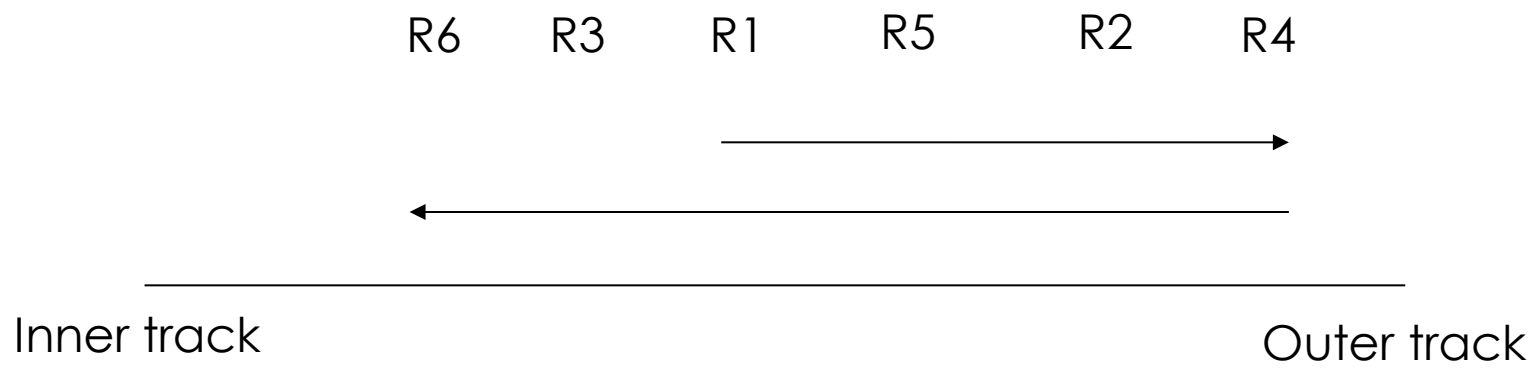
Performance Measures of Disks

- **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, corresponding to a "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, but does not imply that the disks are expected to function 136 years!
  - MTTF decreases as disk ages

KNOWLEDGE & SOFTWARE ENGINEERING

Optimization of Disk Block Access

- **Disk-arm-scheduling algorithms** order pending accesses to tracks so that disk arm movement is minimized
  - elevator algorithm:



Optimization of Disk Block Access

- **Nonvolatile write buffers** speed up disk writes by writing blocks to a non-volatile RAM buffer immediately
  - Non-volatile RAM (NV-RAM): battery backed up RAM or flash memory
    - Even if power fails, the data is safe and will be written to disk when power returns
  - Controller then writes to disk whenever the disk has no other requests or request has been pending for some time
  - Database operations that require data to be safely stored before continuing can continue without waiting for data to be written to disk
  - Writes can be reordered to minimize disk arm movement
- **Log disk** – a disk devoted to writing a sequential log of block updates
  - Used exactly like nonvolatile RAM
    - Write to log disk is very fast since no seeks are required
    - No need for special hardware (NV-RAM)
- File systems typically reorder writes to disk to improve performance
  - **Journaling file systems** write data in safe order to NV-RAM or log disk
  - Reordering without journaling: risk of corruption of file system data

KNOWLEDGE & SOFTWARE ENGINEERING