Chapter 6.

File & Storage

CSIS0278 / COMP3278

Introduction to

Database Management Systems

Department of Computer Science, The University of Hong Kong

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ for students in CSIS0278 / COMP3278

For other uses, please email : ckchui@cs.hku.hk

In this chapter…

- Outcome 1. **Information Modeling**
  - Able to understand the modeling of real life information in a database system.
- Outcome 2. **Query Languages**
  - Able to understand and use the languages designed for data access.
- Outcome 3. **System Design**
  - Able to understand the design of an efficient and reliable database system.
- Outcome 4. **Application Development**
  - Able to implement a practical application on a real database.

Content

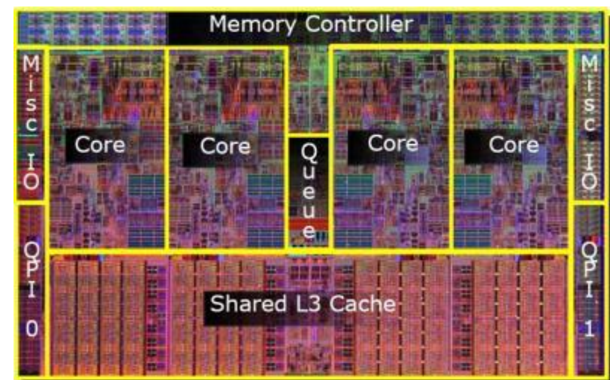
- Storage Media
- Storage Hierarchy
- Reliability and Efficiency
- File organization
- Buffer

Section 1

Storage Media

CPU Cache

Intel core i7 CPU



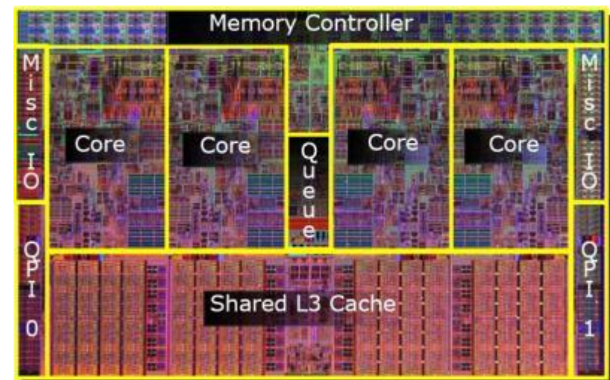
- Cache memory is **extremely fast memory** that is built into a computer's central processing unit (CPU).
- Volatile storage.

Question: Do you know the cache size (L3 cache) of an Intel Core i7 CPU?

Answer :
64KB L1 cache per core.
256KB L2 cache per core.
8MB L3 cache.

CPU Cache

Intel core i7 CPU



- Its use is managed by the computer system hardware. We shall not be concerned about managing CPU cache storage in the database system.
- However, it is worth noting that database implementers do pay attention to cache effects **when designing query processing data structure and algorithms**.

Main Memory

A DDR3 RAM

- Volatile storage.
- Fast access (in nanoseconds : 10^{-9} seconds).
- Generally **too small (or too expensive) to store the entire database** (of an enterprise).
  - Capacities of a few Gigabytes.
  - Capacities have gone up and cost-per-byte has decreased steadily and rapidly.

Question: What is the normal RAM size that can be bought nowadays?

Answer : There are 2GB, 4GB and 8GB (around $500 HKD in 2014) DDR3 RAM.

Magnetic Disk

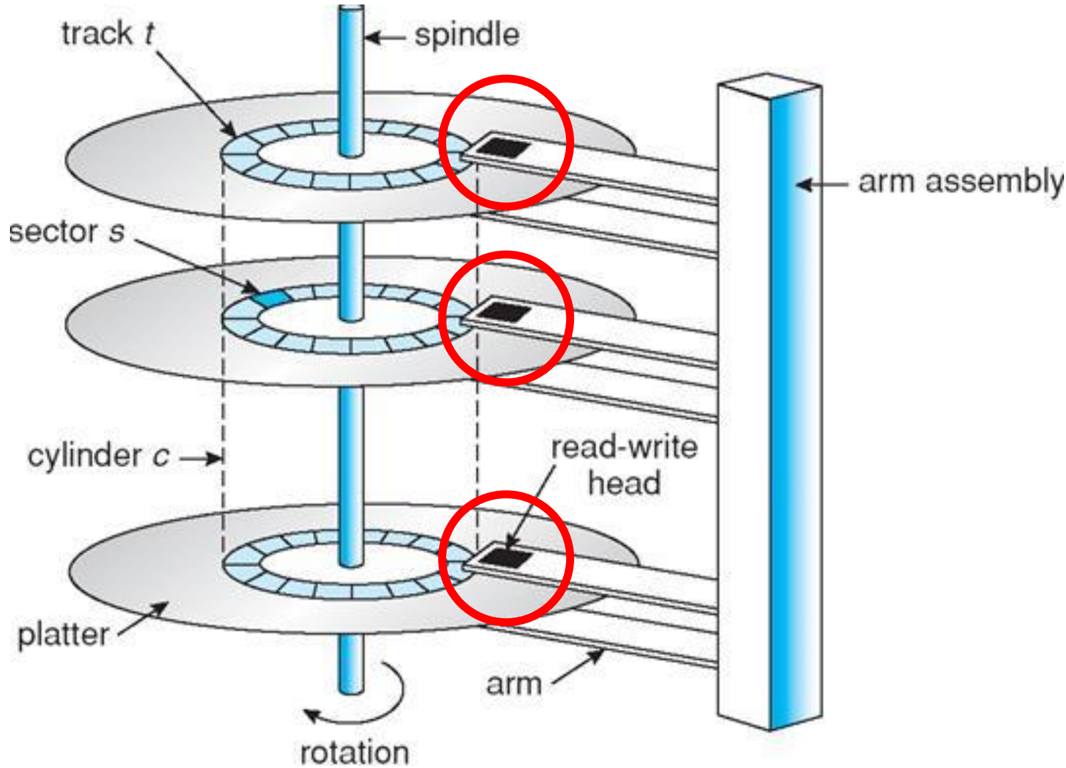
An internal Hard Disk

External Hard Disk

- Primary medium for the long-term storage of data; typically stores the entire database.
- Non-volatile storage.
- Access time: much slower than main memory.
  - **Data are loaded into memory (a buffer) before accessed by DBMS.**

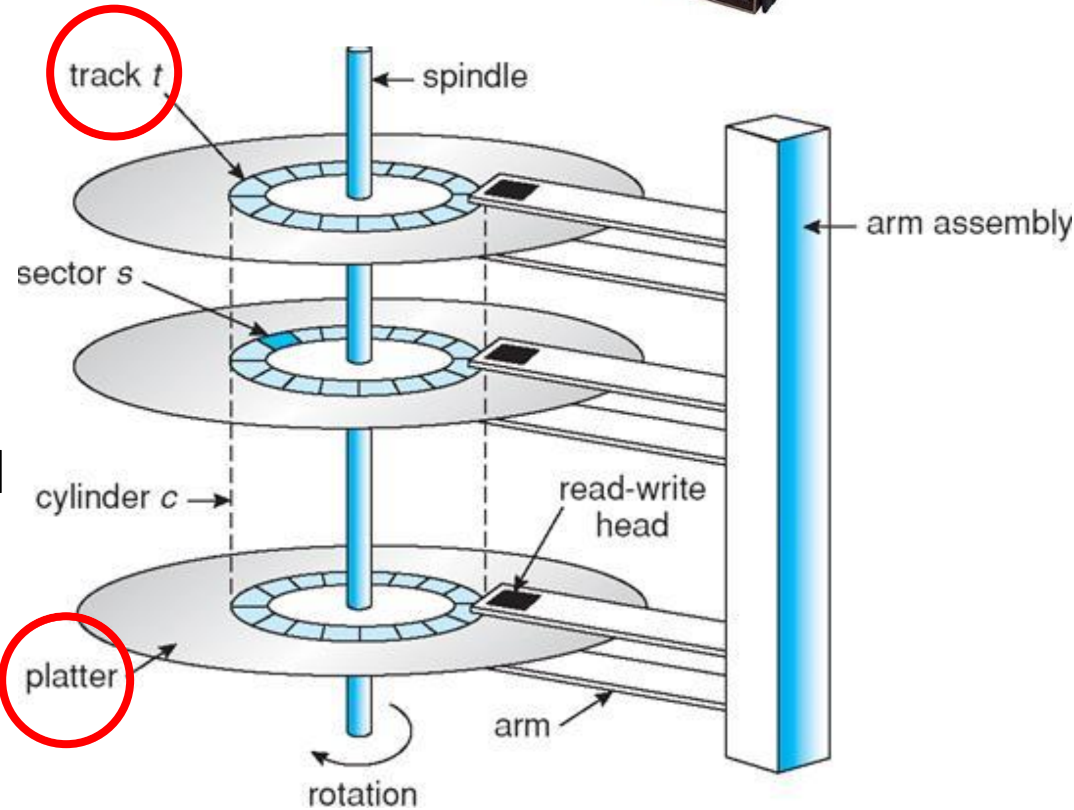
Magnetic Disk

- **Read-Write Heads.**
  - Positioned very closely to the platter surface.
  - Reads or writes magnetically encoded information.



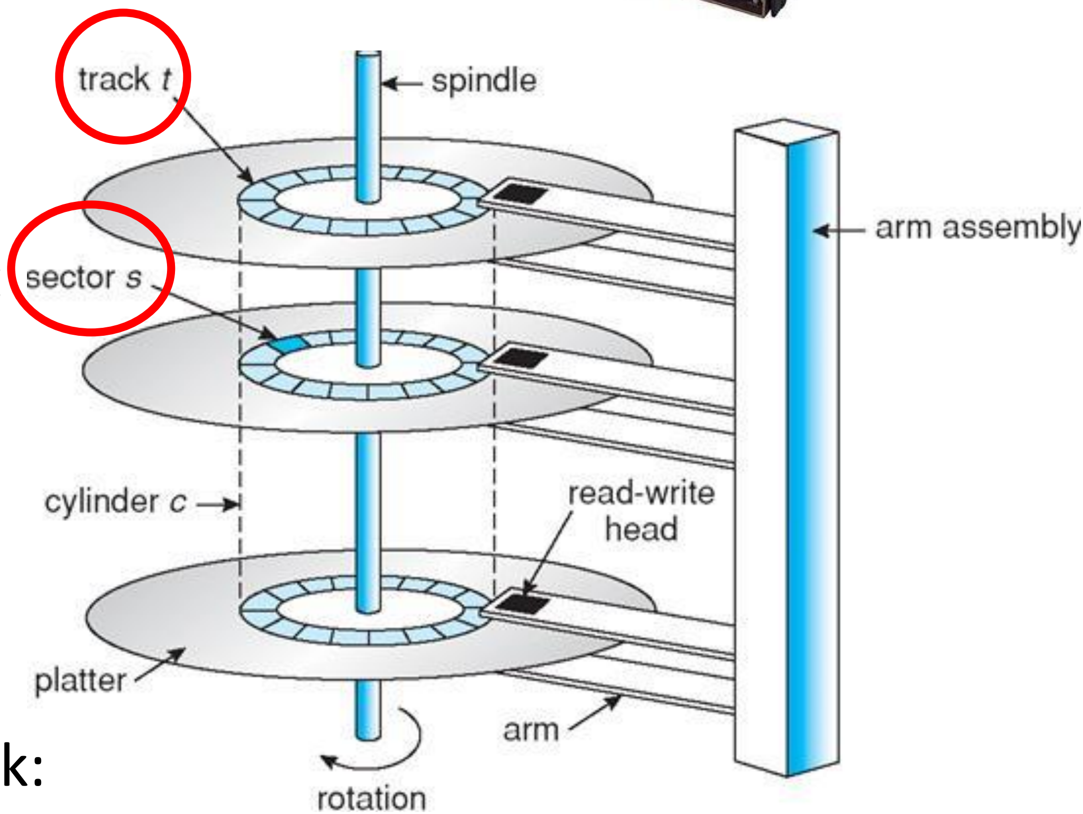
Magnetic Disk

- **A disk has many platters.**
  - Each platter has **two surfaces** covered with magnetic materials, information is recorded on the surfaces.
  - Each platter is divided into circular **tracks**.
  - There are about 50,000 to 100,000 **tracks** per platter. (very dense)



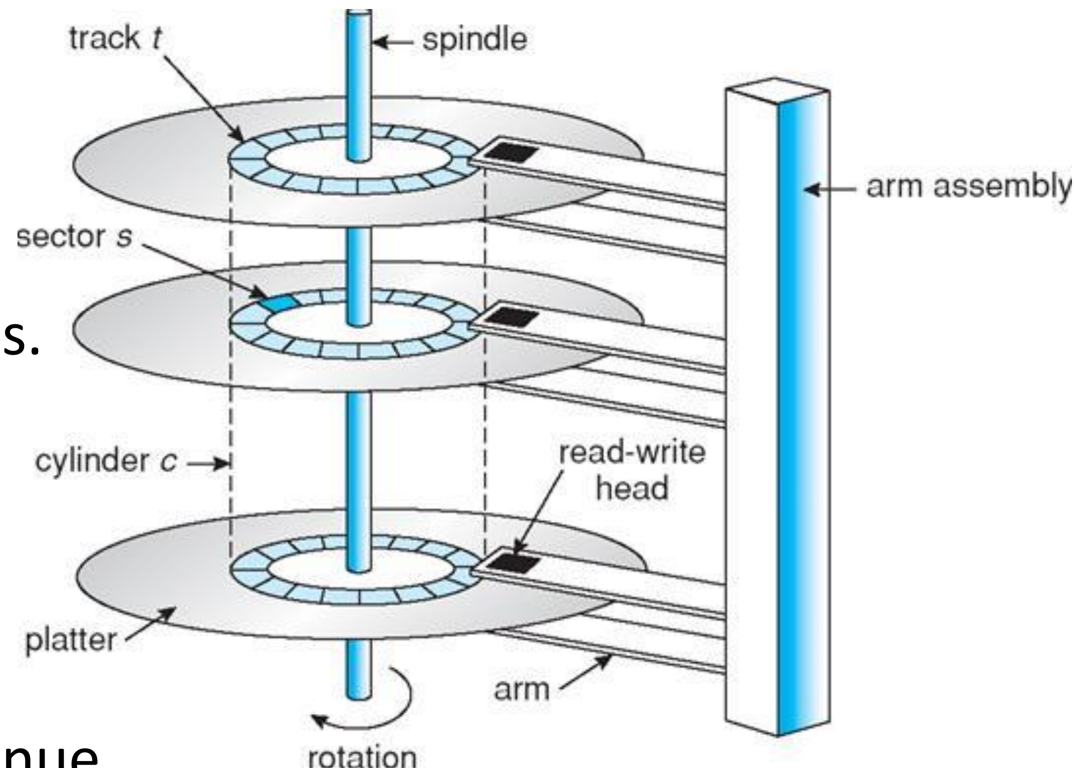
Magnetic Disk

- **Each track is divided into sectors.**
  - A sector is the smallest unit of data that can be read/written.
  - Sector size is typically 512 bytes.
  - Typical sectors per track: 500-1000 (inner tracks) 1000-2000 (outer tracks).



Magnetic Disk

- **To read/write data**
  - **1. [Seek]** Position the head on the right track by moving the disk arms.
  - **2. [Rotation]** Spin the disk so that the start of data is under the head.
  - **3. [Transfer data]** Continue spinning and transfer the data.

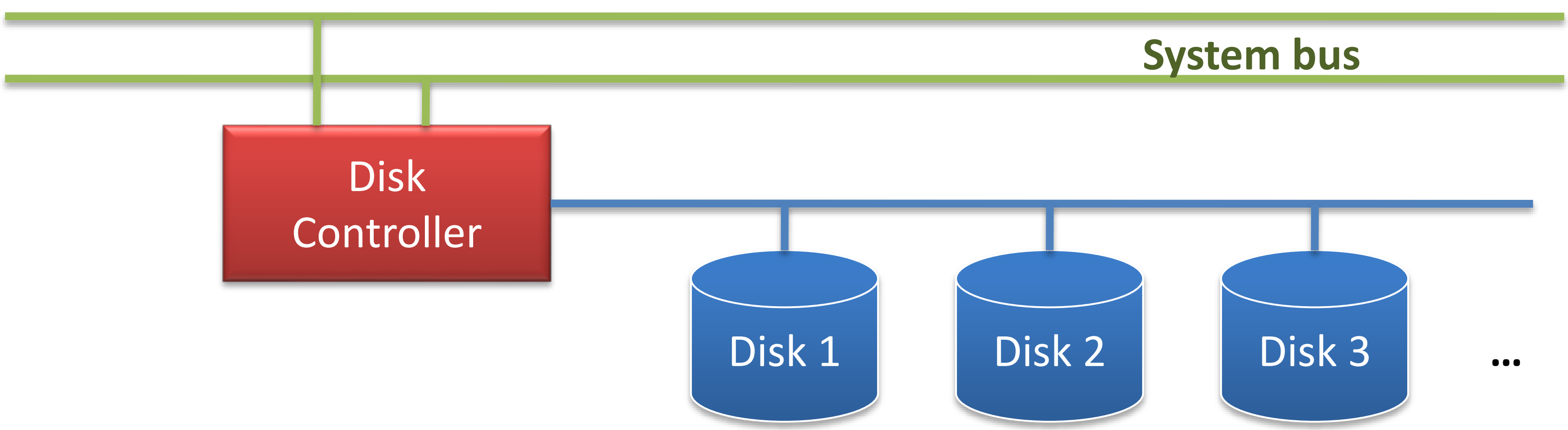


Magnetic Disk

- **Access Time** – the time between the request and the start of data transfer. This consists of:
  - **Seek time**
    - The time required to reposition the arm over the correct track.
    - Around 2 to 30 milliseconds on typical disks, depend on the physical location of the data.
  - **Rotational Latency**
    - The time required to rotate the platter until the required sector is under the disk head.
    - Around 4 to 11 milliseconds per rotation (5400 revolutions per minute (rpm) to 15000 rpm)

Magnetic Disk

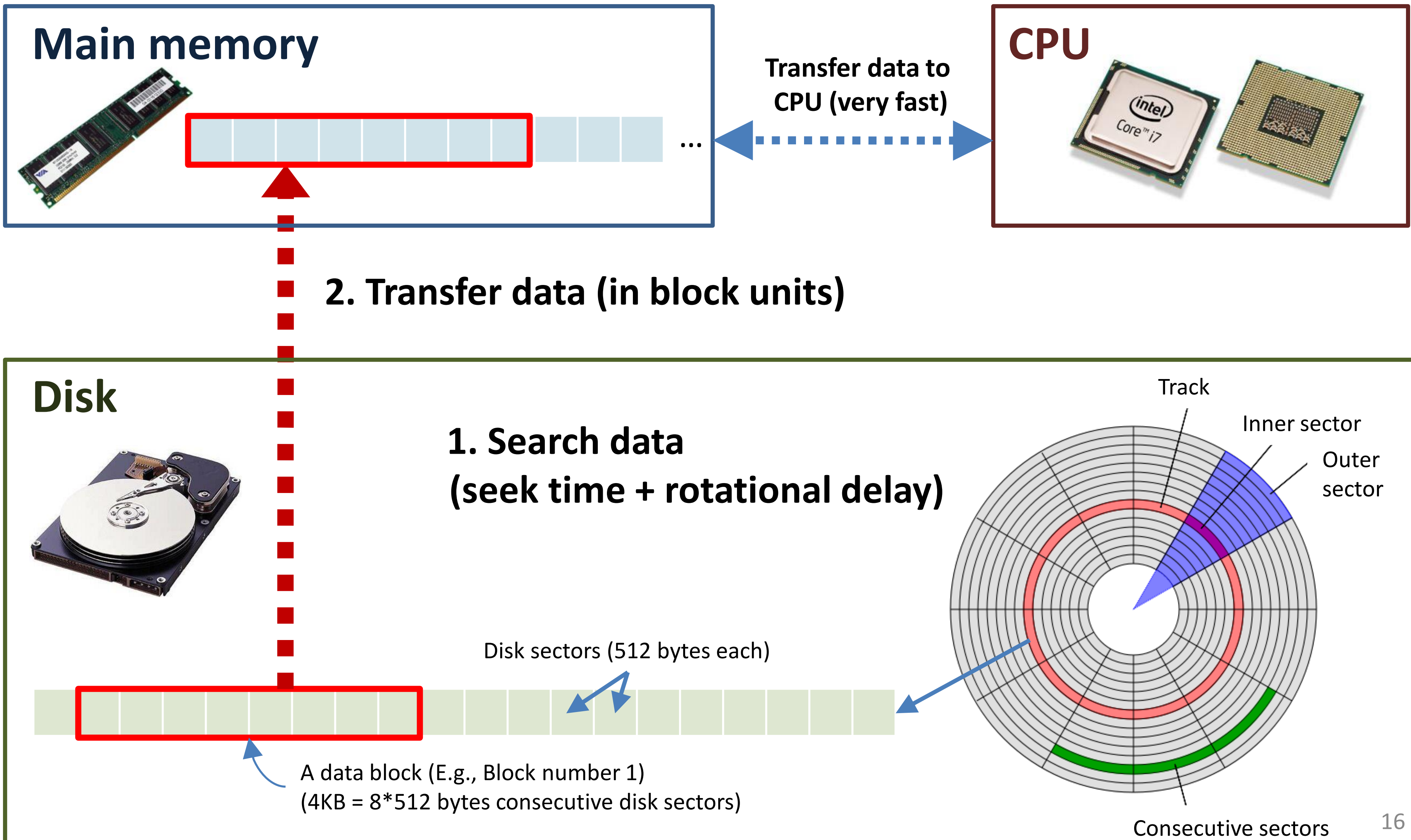
- **Data Transfer Rate** – the rate at which data is retrieved from or stored to disks:
  - Typical value : 25 to 300 megabytes per second (MBps).
  - As multiple disks may share the same controller, we have to **be aware of the controller's processing speed**.



Data block

- **Data must be first transferred to main memory (*buffer) before* the DBMS can operate on them.**
- The data transfer unit between disk and memory is called a **Data block**.
  - Usually with size 4KB to 16KB (spans multiple sectors).
  - When a single item is needed (e.g., an attribute value of a specific tuple), **the whole block that contains the item is transferred**.
  - Reading / writing of a disk block is called **an I/O operation**.

I/O Operation



I/O Operation

- The time required to read/write a block depends on the block's location on disk

Time for one I/O operation
= seek time + rotational delay + transfer time

- **Efficiency issue:** Time to move data from/to disk usually **dominates the cost of processing a query** (CPU actions are in nano-seconds, and a block access is in milli-seconds!)

Magnetic Tape

- Used primarily for **backup** (to recover from disk failures) and archival purpose.
- Non-volatile storage.
- Cost: very low.
- Access speed: slow, and **only sequential access**.
  - As of 2011, the highest capacity tape cartridges (T10000C) can store 5 TB of uncompressed data.

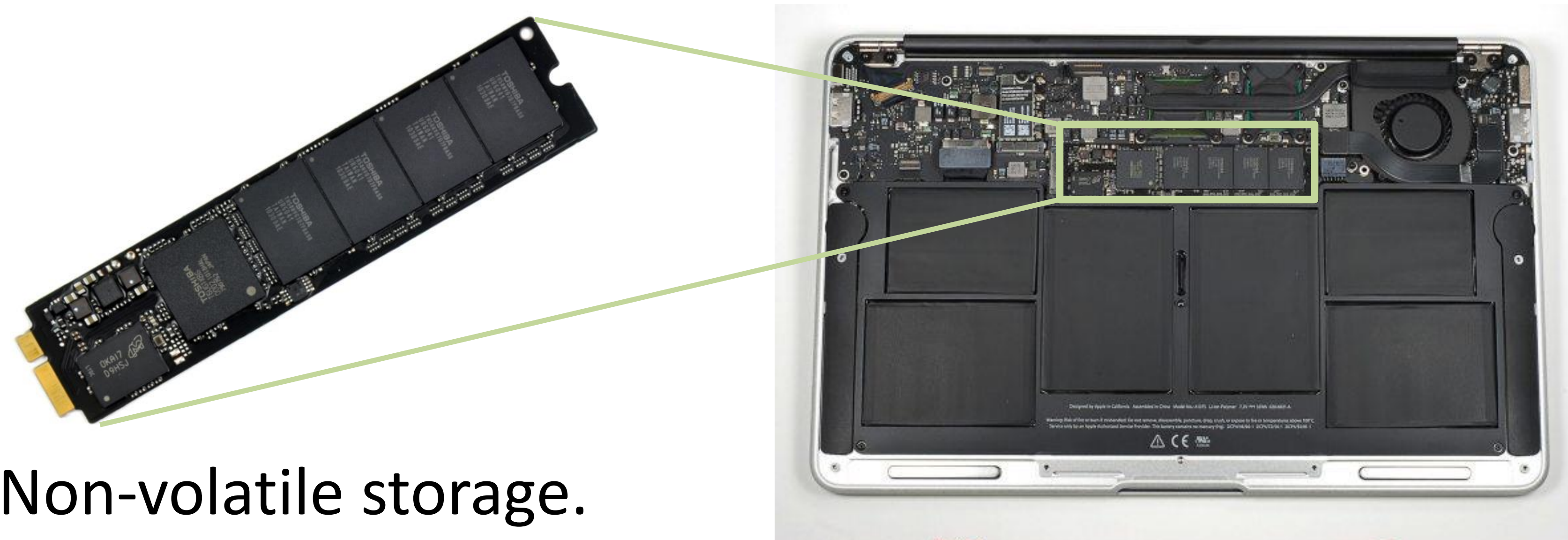
IBM System Storage
TS1140 Tape Drive
Cartridge capacity : 4TB.

IBM System Storage
TS3500 Tape Library

Optical Storage

- Non-volatile storage.
- Access Speed:
  - Read – much slower than magnetic disks (especially on seeks, i.e., random access);
  - Write – even more slower than magnetic disks.
- Capacity: CD-ROM (640MB), and DVD (4.7GB to 17GB)
- Usually write-once, read many (WORM) optical disks are used for archival storage.

Flash Memory



The flash storage in MacBook Air (released in 2010).
16GB flash chips on the board making 64 GB in total.

- Non-volatile storage.
- Access Speed:
  - Read: roughly as fast as main memory.
  - Write: much slower; each location can be written only once. Subsequent writes require “erase” of an entire bank of memory first.
  - **There is a limit of the number of erasing at the same location (typically around 100,000 to 1,000,000 times)**.

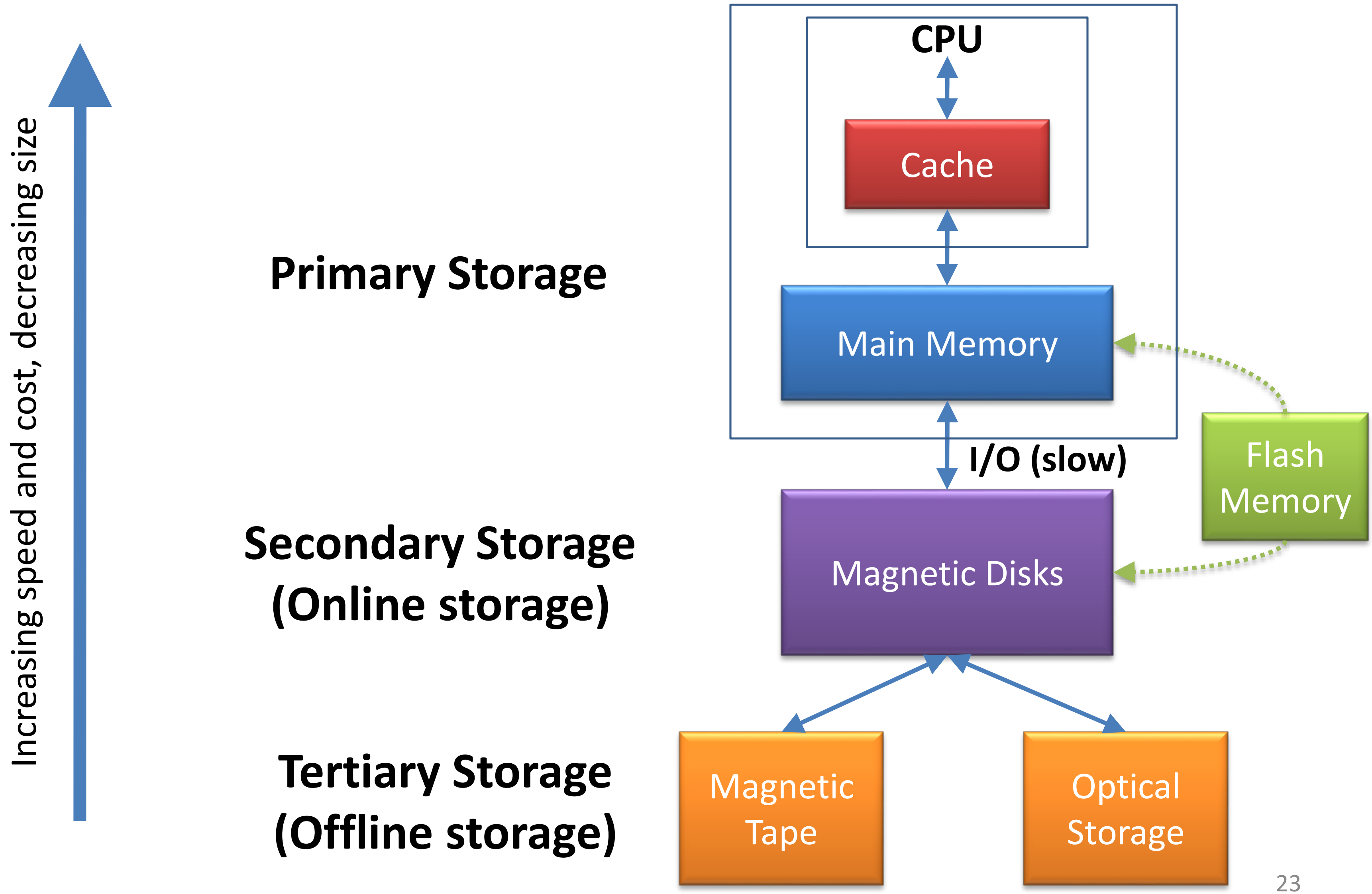
Section 2

Storage Hierarchy

Storage Hierarchy

- **Put all data on disks**
  - Data must be maintained after power failure.
  - Storage capacity must be big enough for all data.
- **Use memory for temporary storage and manipulation of data during queries**
  - Selected data are transferred to memory for fast processing.
  - Updates are first performed in memory and later written back to disk.
- **Backup data on tertiary storage**
  - Periodically backup the contents of DB on tapes.

Storage Hierarchy



Section 3

Reliability & Efficiency

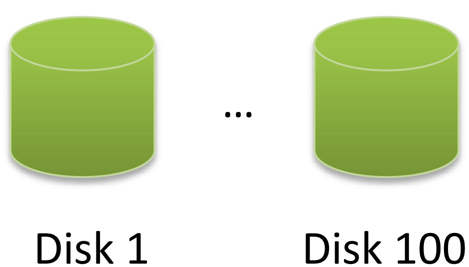
Reliability and Efficiency

- **Reliability -** Hard disks may fail, but we don’t want to lost our data.
- **Efficiency -** Disks are slow compare with the speed of CPU.
- **Solutions**
  - Mirroring.
  - Data striping.
  - Error-correction schemes.

Disk Failure

- **MTTF (Mean time to failure) –** average time the disk is expect to run continuously without failure.
- **Mean time to data loss -** depends on MTTF, and how disks are organized.

Surely not!!!
With multiple disks, the mean time to data lost will be shorten! **Think about a cluster of 100 disks, what is the mean time to data lost?**



Question
Suppose vendor claims that the MTTF of a disk is 100,000 hours (11 years).
Does it mean that it is unlikely to encounter disk failure in an enterprise database system?

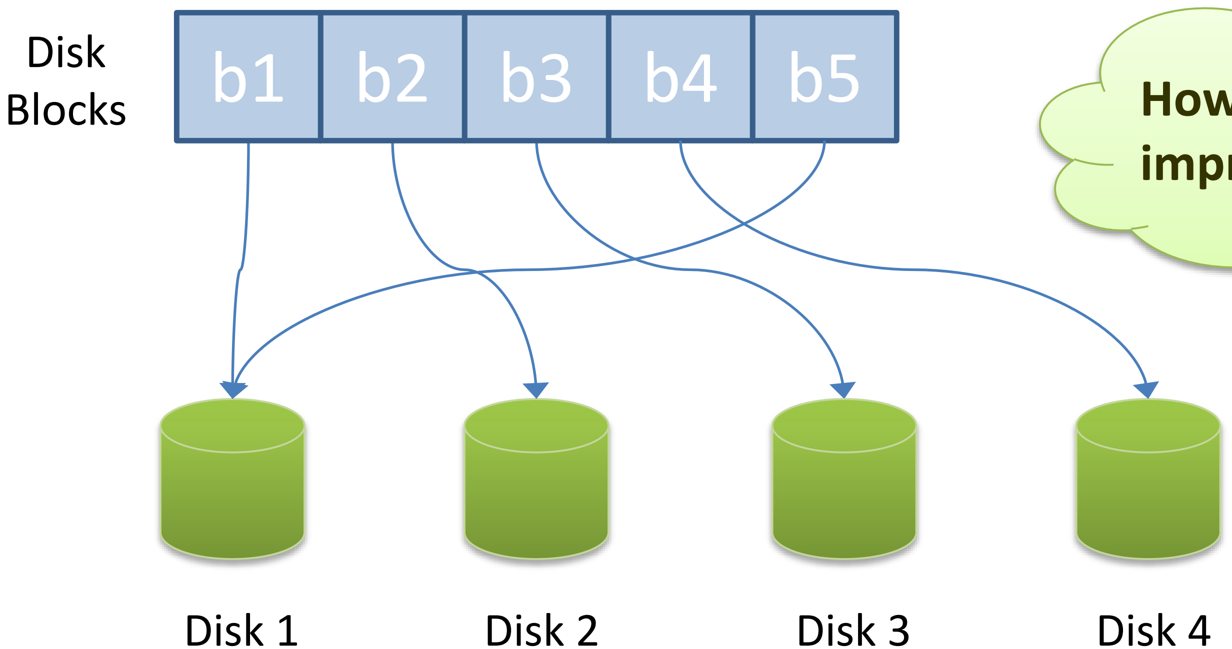
Mirroring

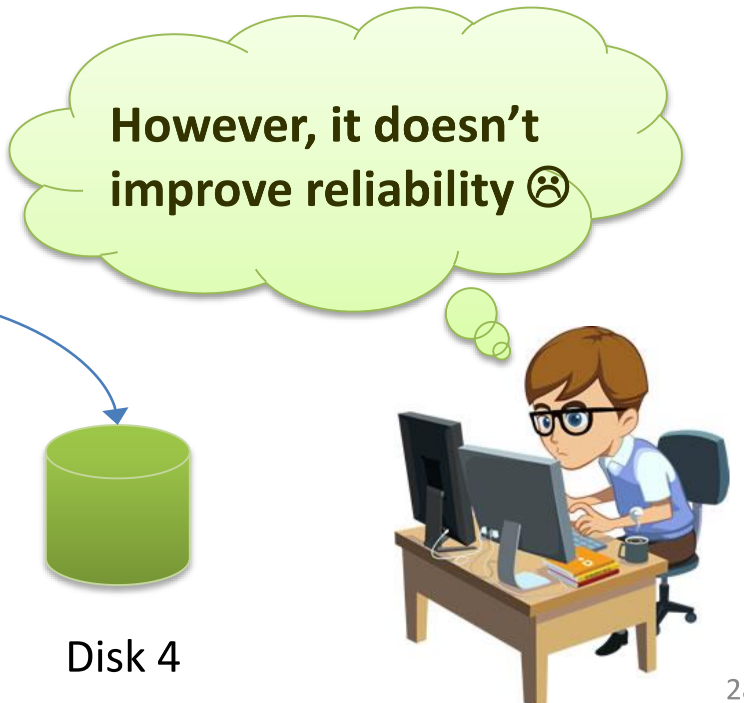
- **Storing a redundant copy of data in another disk(s).**
  - Mean time to data loss will be much longer.
- **Efficiency :**
  - **The rate at which read requests can be handled is doubled -** read requests can be set to all disks.
  - **Note: The speed of each read (or query) is the same as in a single-disk system.**

If I have more than one disks, can I also increase the speed of processing each read request ?

Data Striping

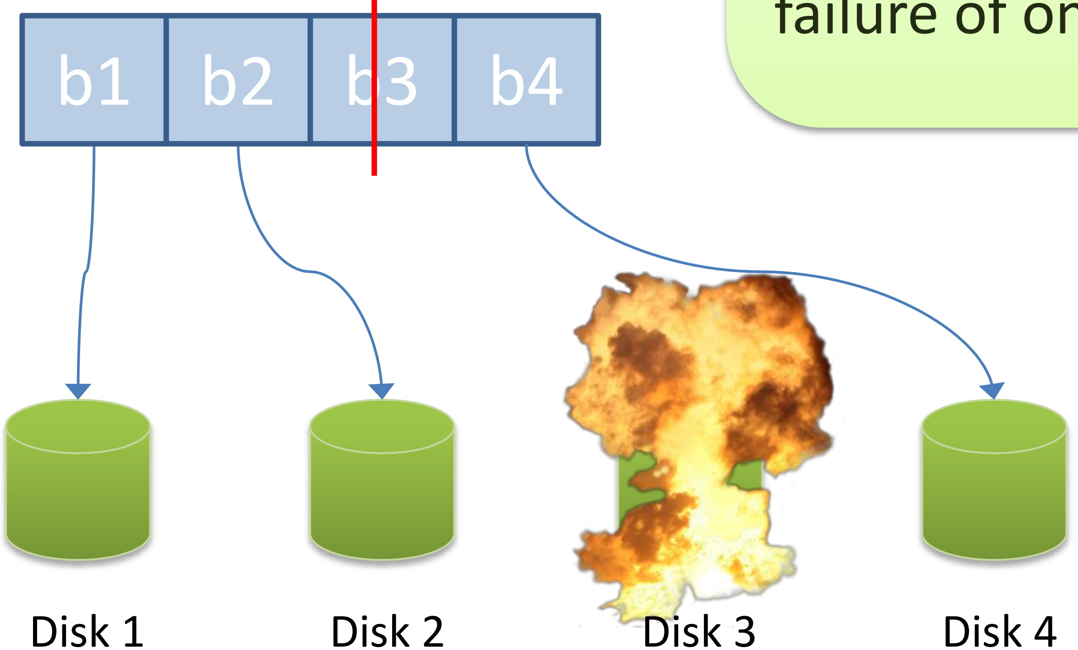
- **Data are partitioned to several disks (e.g., first block to disk 1, second block to disk 2, etc.)**
- **Faster read can be achieved by parallel read.**





Parity check

1	0	0	1
0	0	0	1
1	1	1	0
1	0	0	1
0	0	0	0
1	1	1	1
1	0	1	1



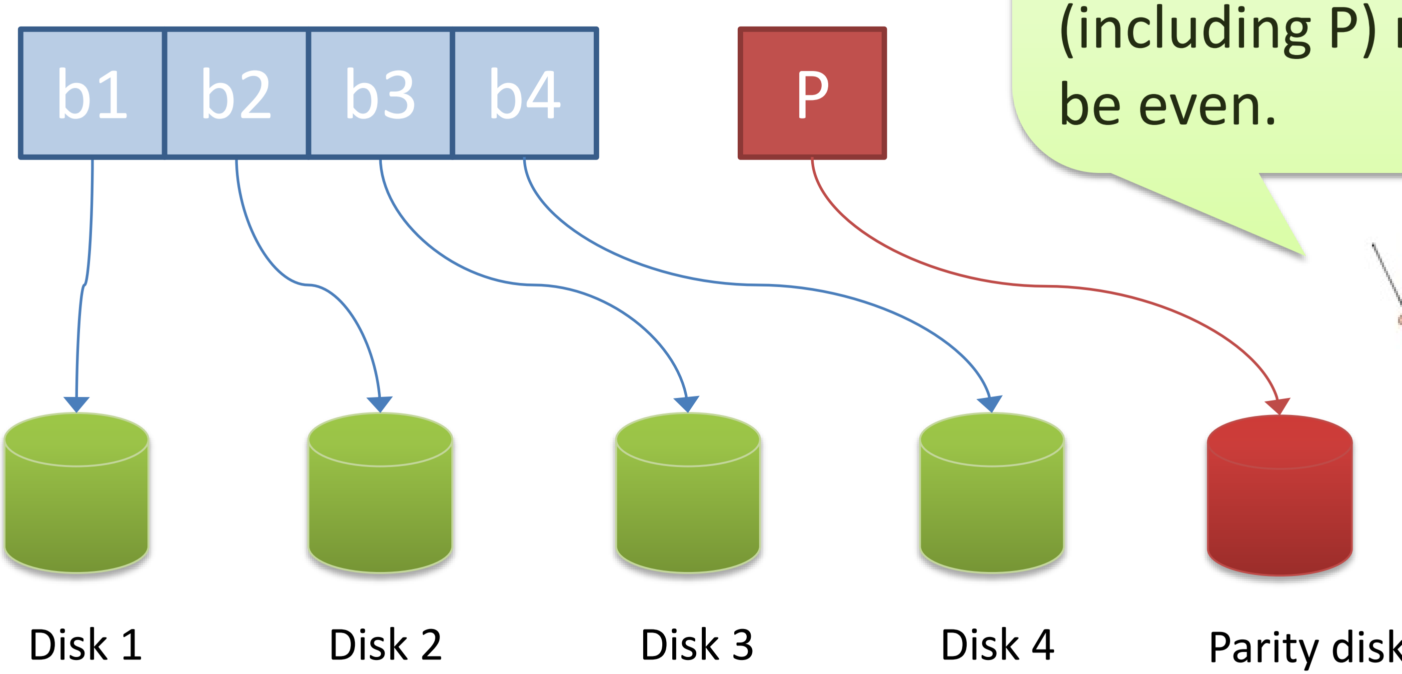
Think about it:
Can you use one more disk to store some extra information so that the database tolerates the failure of one disk?

Parity check

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive or (XOR)

1	0	0	1	0
0	0	0	1	1
1	1	1	0	1
1	0	0	1	0
0	0	0	0	0
1	1	1	1	0
1	0	1	1	1

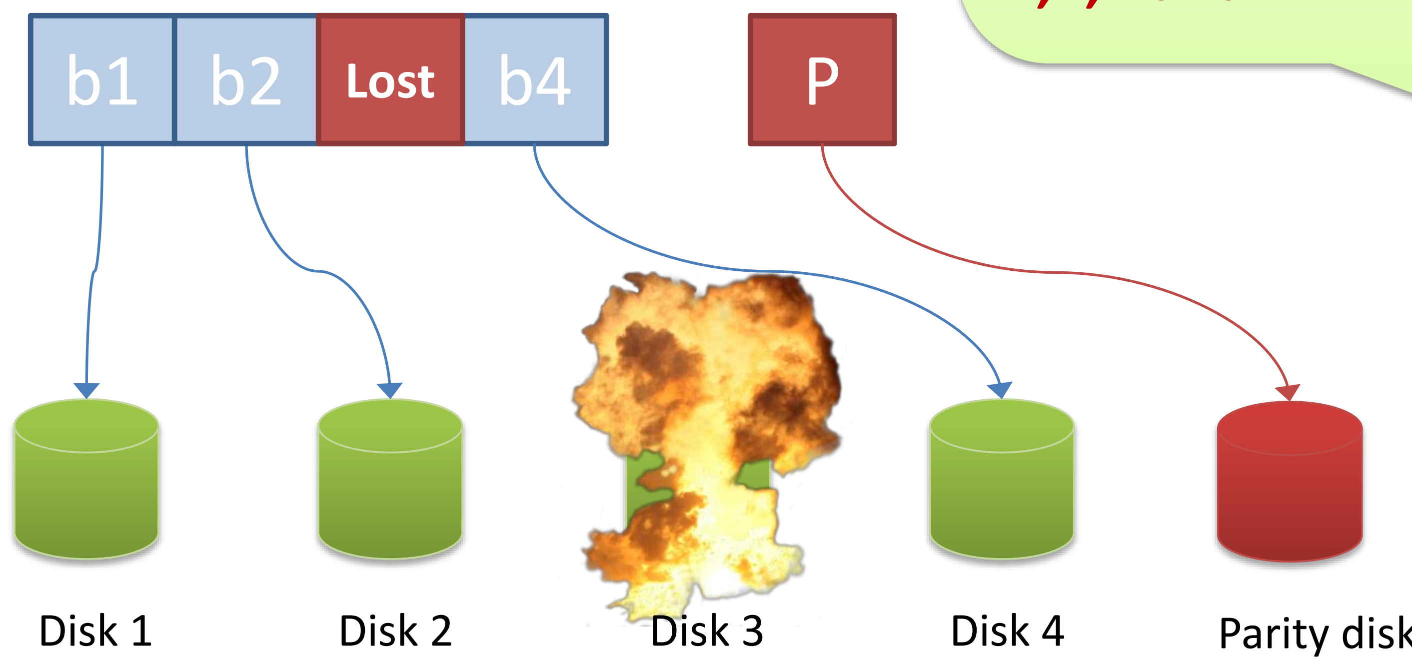


Observation
The number of "1"s among the bit values in each row (including P) must be even.

Parity check

1	0	?	1	0
0	0	?	1	1
1	1	?	0	1
1	0	?	1	0
0	0	?	0	0
1	1	?	1	0
1	0	?	1	1

This bit must be 0! Just apply XOR among the bits of the other disks.



Although the data on Disk 3 is lost, we can **reconstruct** the data in Disk 3 **based on the data on Disk 1,2,4 and P**!

RAID

- **Redundant Arrays of Independent Disks.**
  - Mirroring provides high reliability, but it is **expensive**.
  - Striping provides high data-transfer rate, but **does not improve reliability**.
- **RAID has various levels – with different combinations of mirroring, striping and error-correction strategies.**
  - **RAID 0** – Striping only, no mirroring.
  - **RAID 1/ RAID 10/ RAID 1+0** – Mirroring with striping.
  - Others: RAID level 2,3,4,5,6.

RAID 5

- **By striping also the Parity disk, all disks can share the workload of read requests.**

A block of parity bits

Disk block containing data

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**

Disk 1 Disk 2 Disk 3 Disk 4 Disk 5

Section 4

File Organization

Slides prepared by - **Dr. Chui Chun Kit**, http://www.cs.hku.hk/~ckchui/ for students in CSIS0278 / COMP3278
For other uses, please email : ckchui@cs.hku.hk

File Organization

- **A database store the records in file(s).**
  - Many large-scale database systems **do not rely directly on the underlying operating system for file management**.
  - Instead, **one large operating system file** is allocated to the database system. The database system stores all relations in this one file, and manages the file itself.

File Organization

- **Each file is logically partitioned into fixed-length storage units called blocks, which are the units of both storage and data transfer.**
- **A block may contain several records.**
- **Assumption : No record is larger than a block.**
    - **This assumption is realistic for most data-processing applications.** Large data items (e.g., images), can be stored separately, and storing a pointer to the data item in the record.

1. Records

- **There are two different ways of storing records in a block:**
  - **1a.** Fixed-length records.
  - **1b.** Variable-length records.

```
Instructor (
    ID VARCHAR(5),
    name VARCHAR(20),
    dept_name VARCHAR(20),
    salary INT
)
```

How the database stores the records of tables?

1a. Fixed-length records

- **Fixed-length records**
  - The length of every record is fixed.

A block (e.g., 4KB)

1004	Kit	CS	15000	1012	Ben	CS	18000	…

One record (49 bytes) One record (49 bytes)

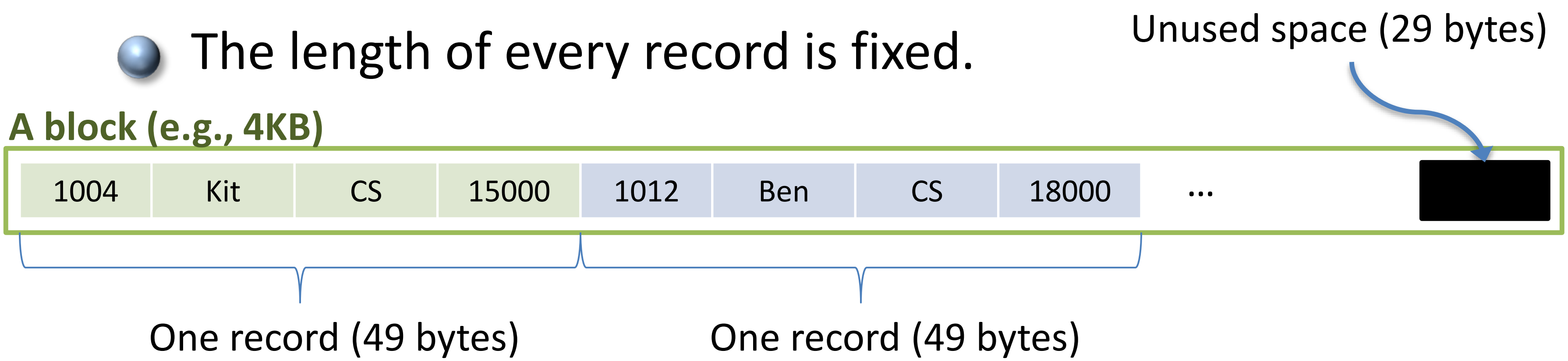
Instructor (
    ID **VARCHAR(5)**,
    name **VARCHAR(20)**,
    dept_name **VARCHAR(20)**,
    salary **INT**
)

Suppose each **INT** value takes **4 bytes, and each CHAR** takes **1** byte. Each Instructor record then has a **maximum** length of **49** bytes**! How many records can fit in a block of 4KB size?**

Floor((1024 * 4) / 49) = 83 records**!**

1a. Fixed-length records

- **Fixed-length records**
  - The length of every record is fixed.

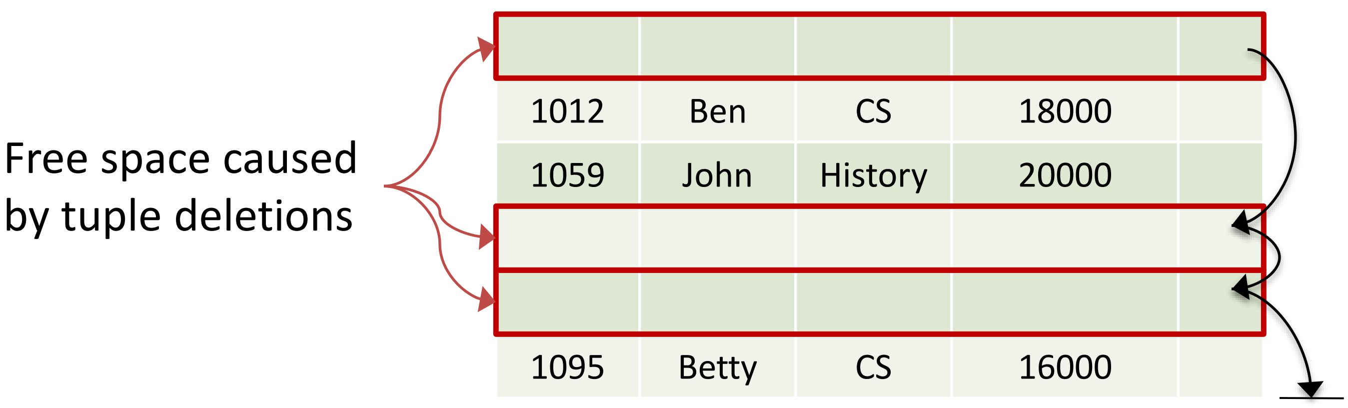


  - Record access is simple, but records may cross blocks.
    - Modification: **Do not allow records to cross blocks, let those areas as unused area.** (Why?)

Answer
Retrieving one record across two blocks **requires two I/Os**, which doubles the amount of disk access time (if no buffer is used)

Free list

- Store the address of the first deleted record in the file header.
- Use the first record to store the address of the second deleted record, and so on.

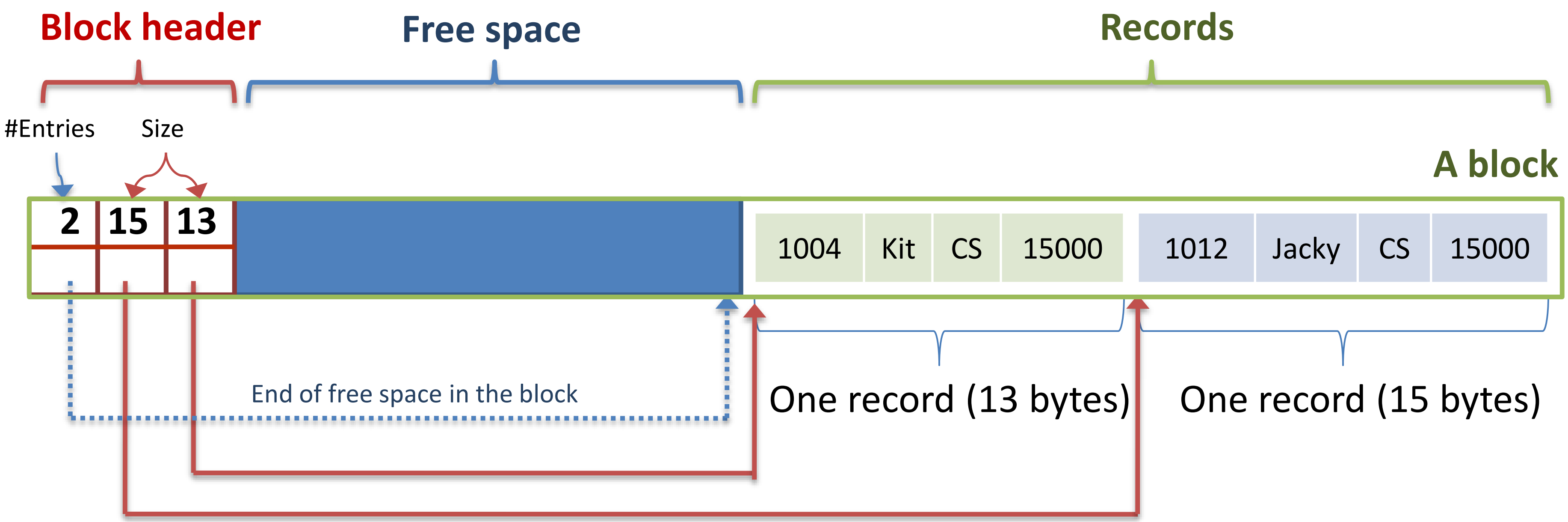


1b. Variable-length records

- **Variable-length records arise in database systems in several ways:**
  - **Storage of multiple records types in the same block** (e.g, Some tuples of the **Instructor** table, and some tuples of the **Department** table stored together in one block).
  - **Record types that allow variable lengths for one or more fields.** (e.g., **VARCHAR**(**250**), **TEXT** …etc)

1b. Variable-length records

- **Slotted-page structure** is commonly used for organizing variable-length records within a block.

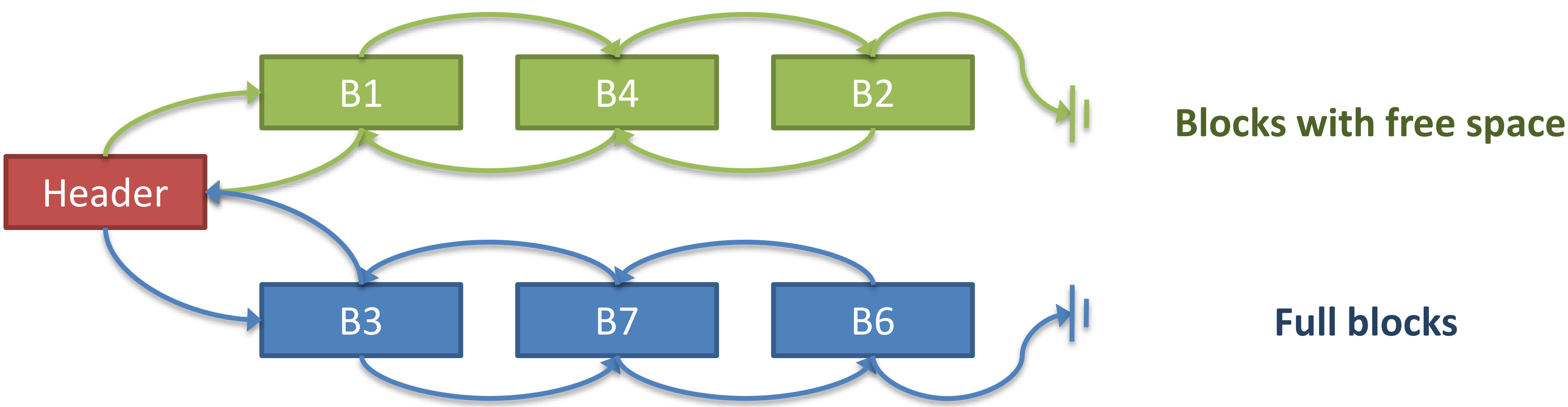


2. Organizing records in files

- **A file contains a number of blocks.**
  - Each block stores a number of records
- **How are the blocks organized in a file?**
- **Which record should be stored in which block?**
- **File organization**
  - **2a.** Heap file
  - **2b.** Sequential file
  - **2c.** Hashing
  - **2d.** Multitable clustering

2a. Heap file

- **No ordering of records, can place anywhere**
  - **Adv: Simplicity –** stores every record in any empty space in any blocks.
  - **Div:** New blocks are allocated or destroyed dynamically; i.e., **blocks in a file may be scattered over the disk.**



2b. Sequential file

- **Store records in sequential order, based on the value of the search key of each record.**
- **Sequential file is designed for efficient processing of records in sorted order based on some search key.**

Note that this sequential ordering according to the Lecturer IDs can help the processing of the following query.

One block

1004	Kit	CS	15000
1012	Ben	CS	18000
1059	John	History	20000
1066	Peter	CS	24000
1084	Billy	Civil	21000
1095	Betty	CS	16000

SELECT * **FROM** Lecturer **WHERE** ID < **1020**;

2b. Sequential file

- **Has to maintain the order during record insertion.**
  - Locate the record in the file that comes before the record to be inserted in search-key order.
  - If there is a **free slot** (maybe after previous deletion) within the same block as this record, insert the new record there.
  - Otherwise, insert the new record in an **overflow block**.

One block

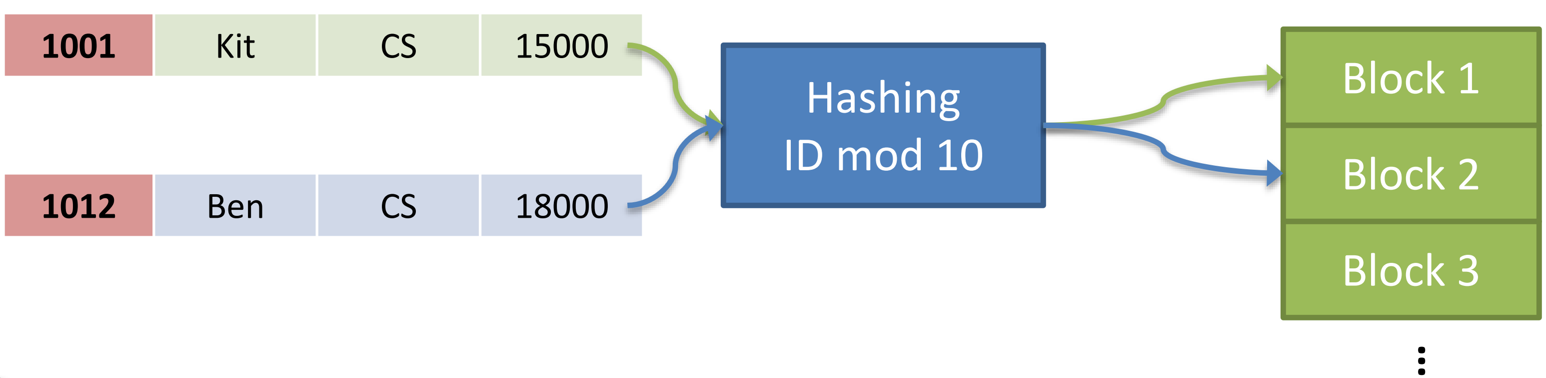
1004	Kit	CS	15000	
1012	Ben	CS	18000	
1059	John	History	20000	
1066	Peter	CS	24000	
1084	Billy	Civil	21000	
1095	Betty	CS	16000	

Overflow block

1011	**Ken**	**CS**	**19000**	

2c. Hashing

- **A hash function is computed on some attribute of each record.**
- **The result of the hash function specifies in which block of the file the record should be placed.**



- **Will be elaborated in the next Chapter.**

2d. Multitable clustering

- We may put ≥ 2 related relations in the same file, to achieve faster joins.

Logical Level

dptID	name	budget
1	CS	300000
2	Civil	200000

lecturerID	name	dptID	salary
1001	Ben	1	20000
1008	Jacky	2	30000
1016	John	1	25000
1005	Betty	1	22000

```
SELECT * FROM Lecturer L, Department D
WHERE L.dptID = D.dptID;
```

Physical Level (A file)

Tuples of the Department table

1	CS	300000
1001	Ben	20000
1016	John	25000
1005	Betty	22000
2	Civil	200000
1008	Jacky	30000

Tuples of the Lecturer table group by dptID, and are ordered after the corresponding department record in the file.

A comparison

- **Heap file is the cheapest to maintain.**
  - But have to scan all data to locate a specific record.
- **Sequential file helps query evaluation.**
  - But it is difficult to maintain a sequential file.
- **Clustering file helps joins and finding related records over different relations.**
  - Accessing data on only one relation may suffer.
  - Variable size records may be difficult to handle.

Section 5

Buffer

Buffer manger

- **DBMS seeks to minimize the number of blocks transfers between disks and memory.**
- **Buffer -** Portion of memory available to store copies of disk blocks.
- **Buffer Manager –** A program responsible for allocating buffer space in main memory and moving blocks between disk and memory (so that disk I/O is minimized).

1. Buffer miss

Step 1. Read data in Block 1.

Database Application

Buffer

2	4	7	

Disk

1	2	3	4	5	6	7	8	9	…

A block

1. Buffer miss

Database Application

Step 1. Read data in Block 1.

Buffer 2 4 7 1

Step 2. Block 1 is not in the buffer, so it is retrieved from disk (**1 Disk I/O**).

Disk 1 2 3 4 5 6 7 8 9 …

A block

1. Buffer miss

Database Application

Step 3. The required data in block 1 is sent to the application.

Buffer

2	4	7	1

Disk

1	2	3	4	5	6	7	8	9	…

A block

2. Buffer hit

Step 1. Read data in Block 2.

Database Application

Step 2. The required data is in the buffer, return to users without disk access (**No I/O**).

Buffer

2	4	7	1

Disk

1	2	3	4	5	6	7	8	9	…

A block

3. Write operation

Database Application

Step 1. Update record in block 2.

Buffer

2 | 4 | 7 | 1

Updates are done in memory only. The result will be reflected on disk when the buffer is **flushed** back to disk (or under other reliability requirements).

Disk

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | …

A block

4. Buffer full

Step 1. Read data in block 5.

Database Application

Buffer

2	4	7	1

?

Disk

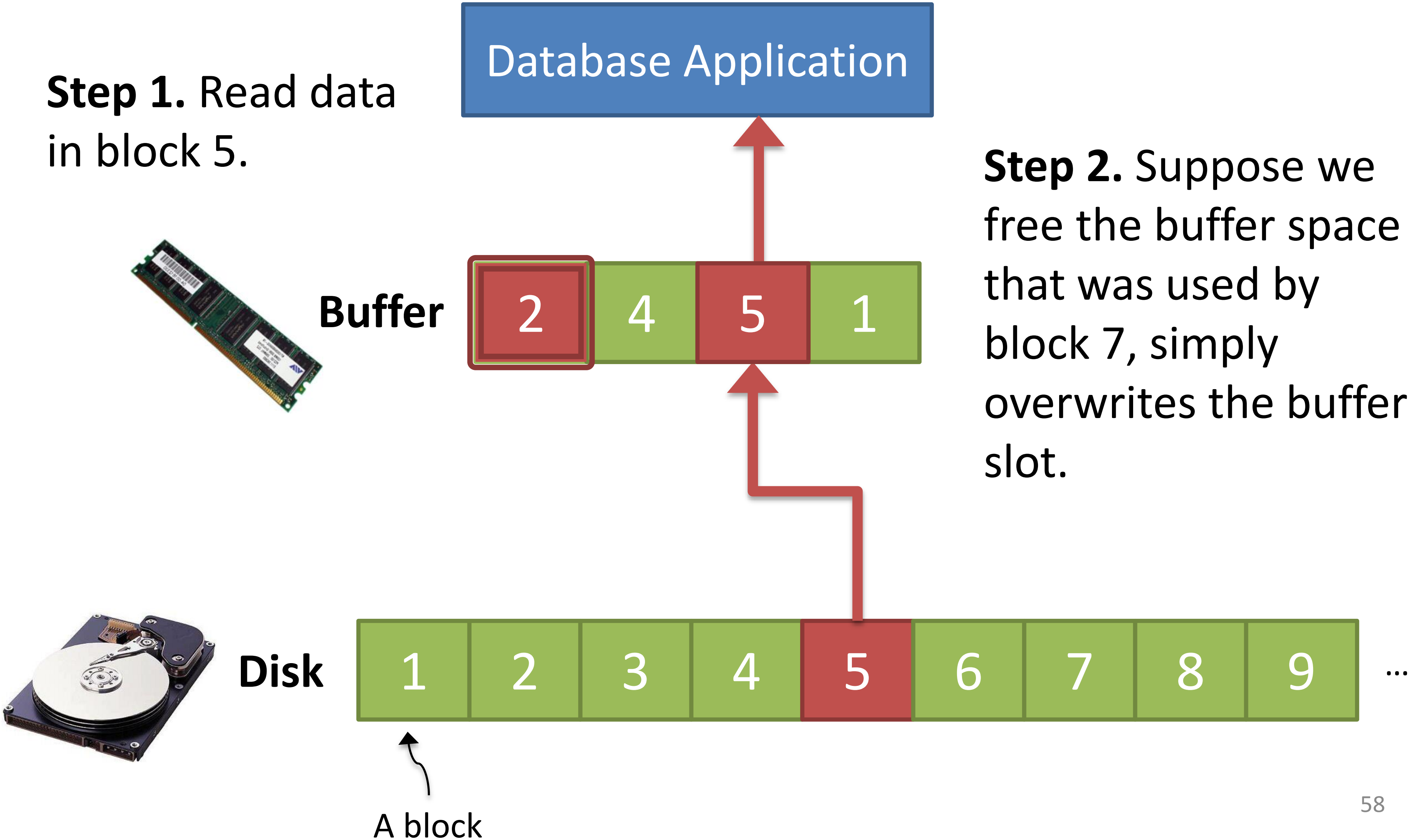
1	2	3	4	5	6	7	8	9	…

A block

Since block 5 is not in buffer, we need to fetch it from disk. But we have **no buffer space** …

4. Buffer full

Step 1. Read data in block 5.



Step 2. Suppose we free the buffer space that was used by block 7, simply overwrites the buffer slot.

4. Buffer full

Database Application

Step 1. Read data in block 5.

Buffer

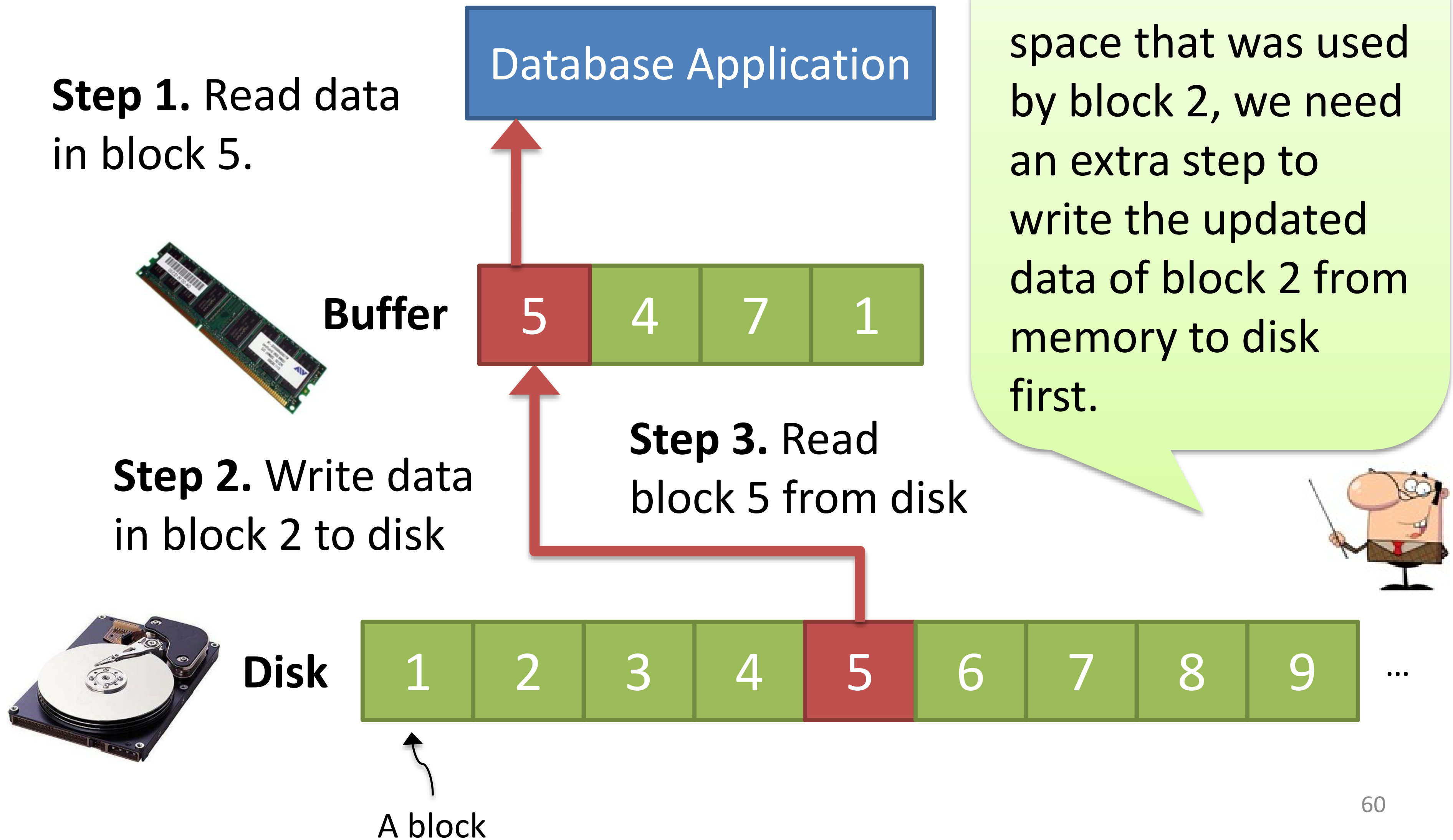
Step 2. Write data in block 2 to disk

If we free the buffer space that was used by block 2, we need an extra step to write the updated data of block 2 from memory to disk first.

Disk

A block

4. Buffer full



Buffer replacement policy

- **Most operating systems replace the block that was the least recently used – LRU strategy**
- **The intuition behind LRU**
  - **If a block is not used for a long time, it is not likely that it will be accessed again very soon.**
  - This uses the past patterns of block accesses as a predictor of future accesses.
- **Other replacement policies**
  - **LFU** – Least frequently used., etc.

Data dictionary

- **Data dictionary (also called system catalog) stores metadata, i.e., data about data. e.g.,**
  - Information about relations (names of relations, names & types of attributes, integrity constraints, views)
  - **Statistical data** (e.g., number of tuples in each relation).
  - Physical file organization (sequential, hashing, etc.)
  - Frequently accessed by the **buffer manager** and **query optimizer** and therefore stays in the memory for fast access.

Chapter 6.

END

CSIS0278 / COMP3278

Introduction to

Database Management Systems

Department of Computer Science, The University of Hong Kong