

21121350

Database System

Lecture 7: Storage and File Structure

Lu Chen (陈璐)

College of Computer Science

Zhejiang University

Spring & Summer 2023

luchen@zju.edu.cn/18868818726

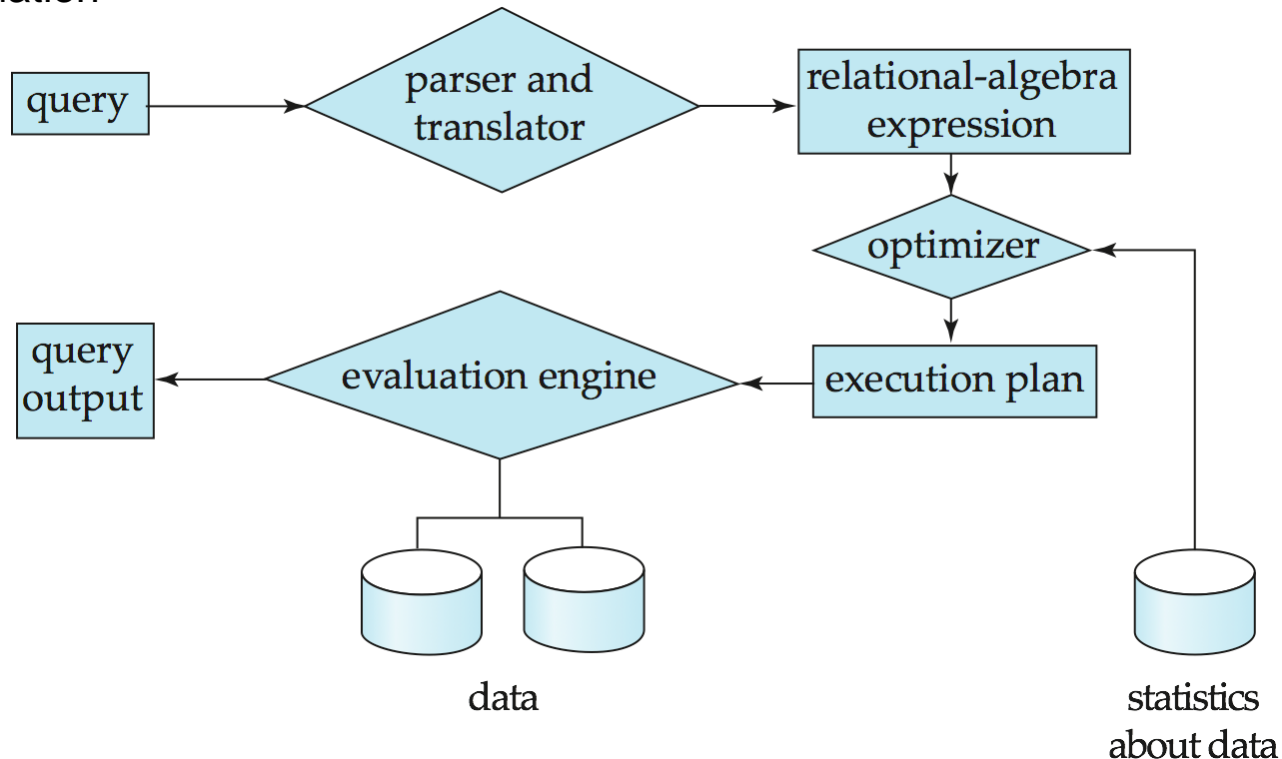
Review: Storage Manager

- ❑ **Storage Manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- ❑ **Storage Manager** is responsible for the following tasks:
 - Interaction with the file manager
 - Efficient storing, retrieving and updating of data
- ❑ **Storage Manager** includes
 - Transaction manager
 - Authorization and integrity manager
 - File manager (interaction with the file system to process data files, data dictionary, and index files)
 - Buffer manager

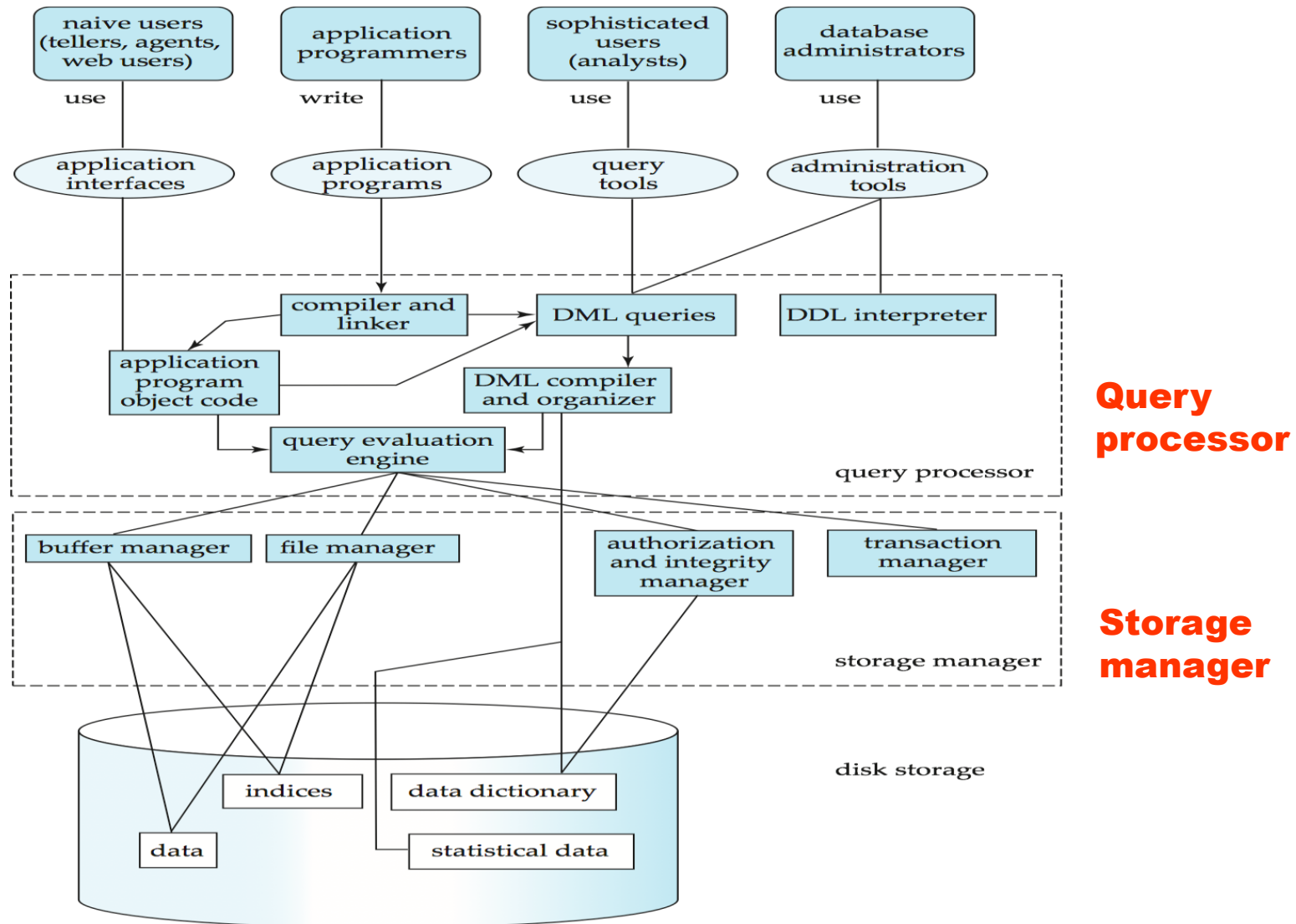
Review: Query Processor

❑ Query Processor includes DDL interpreter, DML compiler, and query processing.

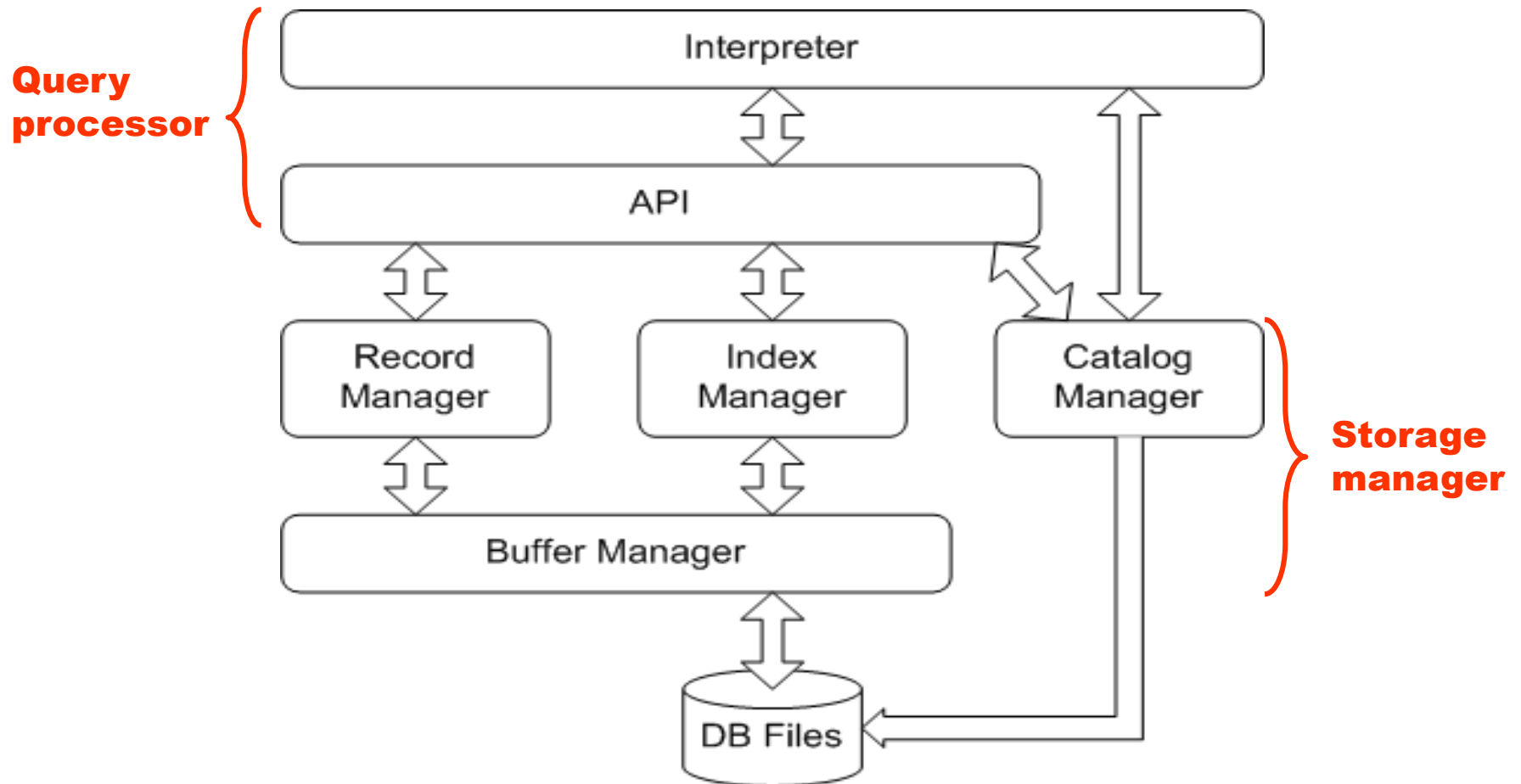
- Parsing and translation
- Optimization
- Evaluation



Review: Database System Internals



MiniSQL Architecture



Outline

- ❑ Overview of Physical Storage Media
- ❑ Magnetic Disks
- ❑ * RAID
- ❑ * Tertiary Storage
- ❑ Storage Access
- ❑ File Organization
- ❑ * Organization of Records in Files
- ❑ * Data-Dictionary Storage
- ❑ Column-Oriented Storage



Classification of Physical Storage Media

❑ The physical level of database:

- Files, storage (e.g., **.mdf**, **.ldf**, **.ora**, **.dbf**)

❑ Storage media can be classified by

- **Speed** with which data can be accessed
- **Cost** per unit of data
- **Reliability**
 - Data loss on power failure or system crash
 - Physical failure of the storage device (RAID)

Classification of Physical Storage Media (Cont.)

❑ Storages classified by **reliability**:

- Volatile storage: loses contents when power is switched off, e.g., DDR2, SDR.
- Non-volatile storage(非易失性存储器): contents persist even when power is switched off.
 - Includes secondary and tertiary storage, as well as **batter-backed up main-memory**.

❑ Storages classified by **speed**:

- Cache
- Main-memory
- Flash memory (快闪存储器)
- Magnetic-disk
- Optical storage
- Tape storage

Physical Storage Media

- ❑ Cache: **Fastest** and most **costly** form of storage, **volatile**, and managed by the computer system hardware.
 - Speed: ≤ 0.5 nanoseconds (ns for short, $1 \text{ ns} = 10^{-9}$ seconds); size: $\sim \text{KB} \sim \text{MB}$

- ❑ Main memory:
 - **Fast access** (10 to 100 ns)
 - **Generally too small** (or **too expensive**) to store the entire database
 - Capacities of up to **a few Gigabytes** ($1\text{GB} = 10^9\text{B}$) widely used currently
 - **Capacities** have **gone up** and **per-byte costs** have **decreased steadily** and rapidly (roughly factor of 2 every 2 to 3 years)
 - **Volatile**: contents of main memory are usually lost if a power failure or system crash occurs.

Physical Storage Media (Cont.)

❑ Flash memory（快闪存储器）：

- EPROM (Erasable Programmable Read-Only Memory)
- Also known as **EEPROM** (Electrically Erasable Programmable Read-Only Memory, 电可擦可编程只读存储器)
- NOR Flash
- NAND Flash
- **Data survives when power failure**
- **Data can be written** at a location only once, but location can be erased and written to again
 - Can support only **a limited number** (10K – 1M) **of write/erase cycles**
 - **Erasing** of memory has to be done to an entire bank of memory
- **Reads are roughly as fast as main memory** (< 100ns)
- But **writes are slow** (~ 10μs), **erase is slower**
- **Cost** per unit of storage roughly similar to main memory
- Widely used in **embedded devices** such as digital cameras, phones, and USB keys

Physical Storage Media (Cont.)

❑ Magnetic-disk:

- Data is stored on spinning disk, and read/written magnetically
- SMR vs PMR
- Primary medium for the long-term storage of data, typically stores entire database
- Data must be moved from disk to main memory for access, and written back for storage
 - Much slower access than main memory (more on this later)
- Direct-access: possible to read data on disk in any order, unlike magnetic tape
- Capacities range up to roughly 1.5 TB as of 2009
 - Much larger capacity and lower cost/byte than main memory/flash memory
 - Growing constantly and rapidly with technology improvements (factor of 2 to 3 every 2 years)
- Survives power failures and system crashes
 - Disk failure can destroy data, but is very rare

Physical Storage Media (Cont.)

❑ Optical storage:

- Non-volatile, data is read optically from a spinning disk using a laser
- CD-ROM (640 MB) and DVD (4.7 to 17 GB) most popular forms
- Write-one, read-many (WORM) optical disks used for archival storage (CD-R and DVD-R)
- Multiple write versions also available (CD-RW, DVD-RW, and DVD-RAM)
- Reads and writes are slower than with magnetic disk.
- Juke-box (自动光盘机) systems, with large numbers of removable disks, a few drives, and a mechanism for automatic loading/unloading of disks available for storing large volumes of data

Physical Storage Media (Cont.)

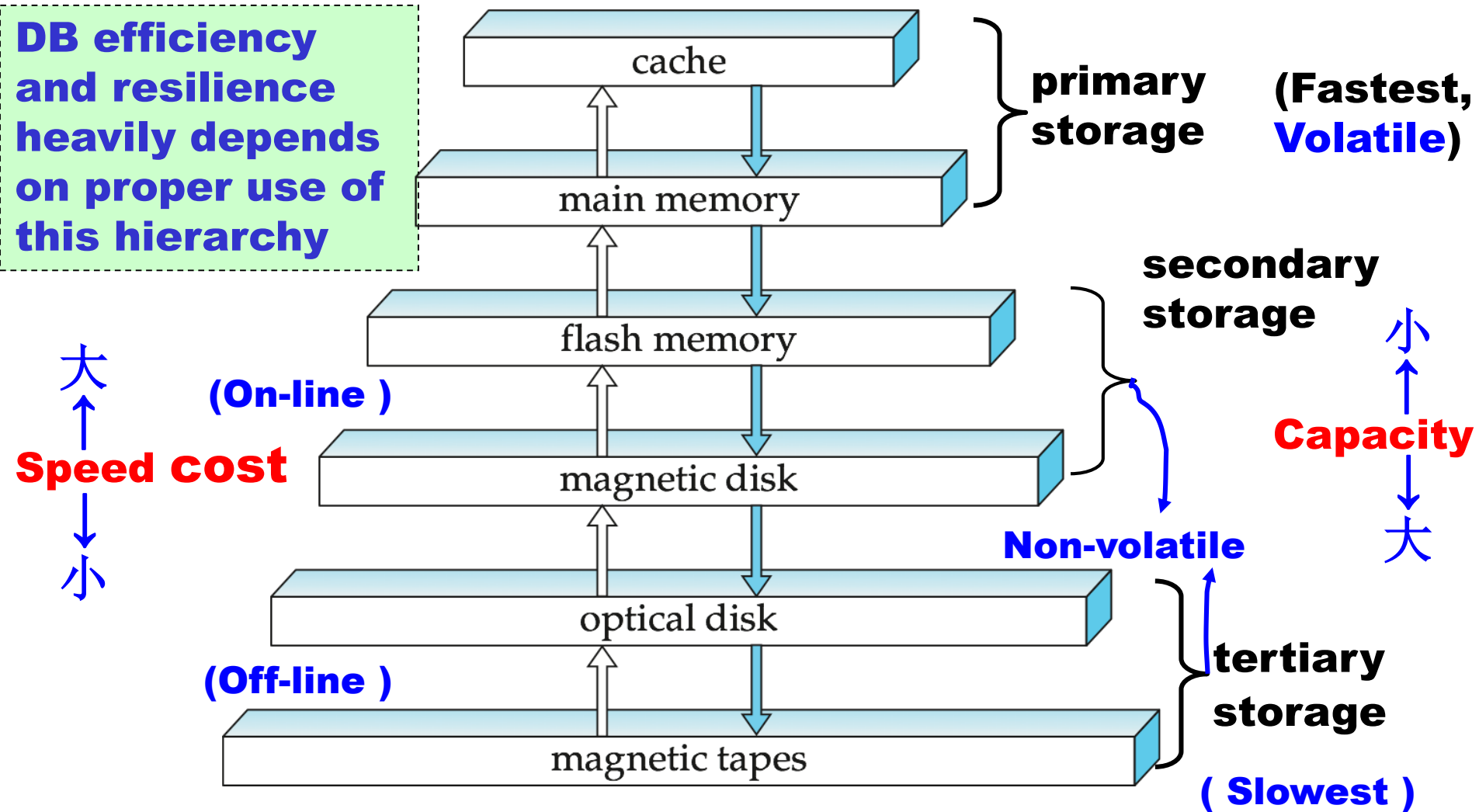
□ Tape storage:

- Non-volatile, used primarily for backup (to recover from disk failure), and for archival data
- Sequential-access – much slower than disk
- Very high capacity (40 to 300 GB tapes available)
- Tape can be removed from drive \Rightarrow storage costs much cheaper than disk, but drives are expensive
- Tape jukeboxes available for storing massive amounts of data
 - Hundreds of terabytes (TB, 1 terabyte = 10^{12} bytes) to even a petabyte (PB, 1 petabyte = 10^{15} bytes)

Jim Gray: Tape is Dead, Disk is Tape, Flash is Disk.

Storage Hierarchy

DB efficiency and resilience heavily depends on proper use of this hierarchy



Storage Hierarchy (Cont.)

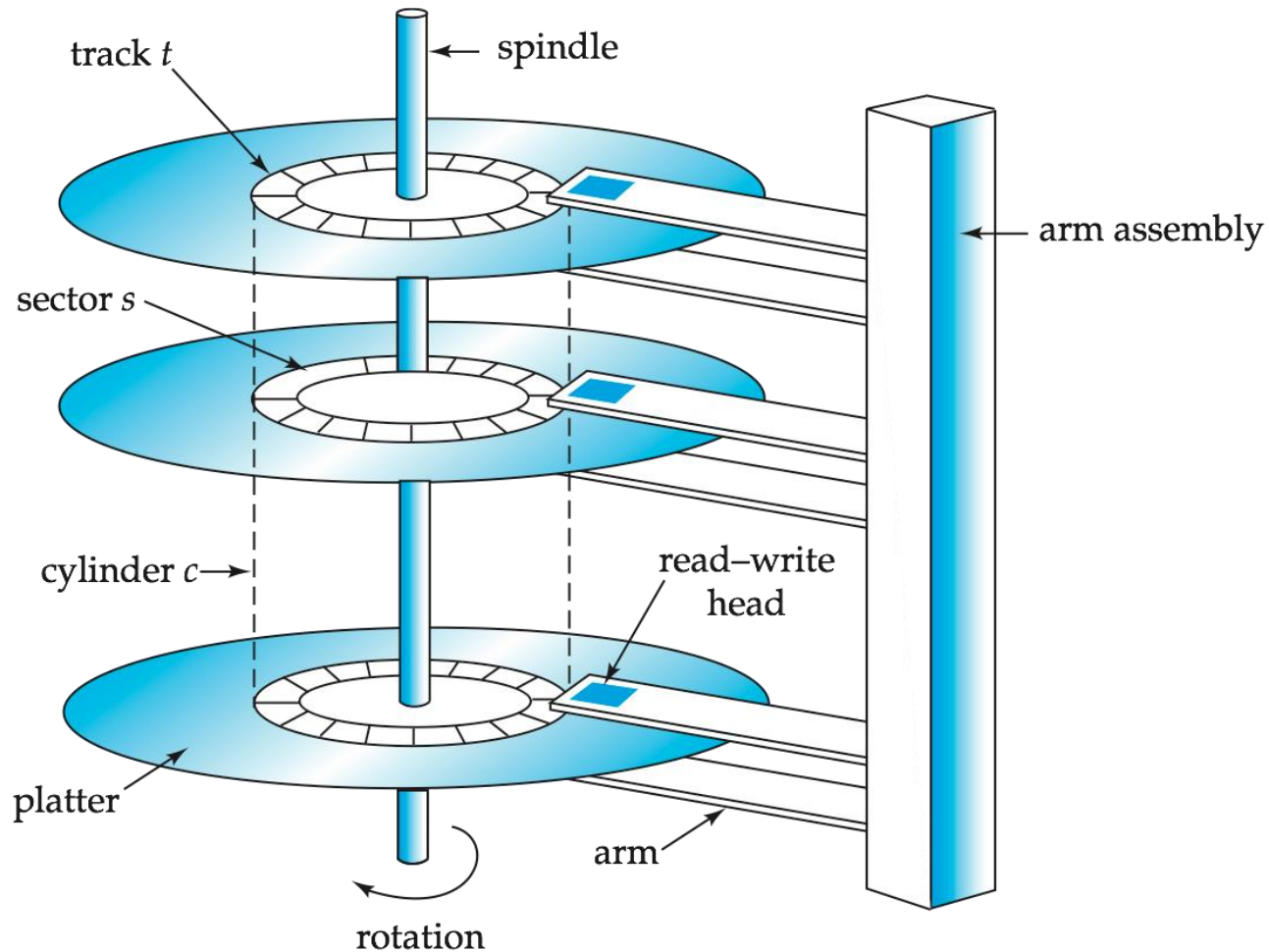
- ❑ **Primary storage**: Fastest media but volatile (cache, main memory).
- ❑ **Secondary storage** (辅助存储器, 联机存储器): next level in hierarchy, non-volatile, moderately fast access time
 - Also called on-line storage
 - E.g., flash memory, magnetic disks
- ❑ **Tertiary storage** (三级存储器, 脱机存储器): lowest level in hierarchy, non-volatile, slow access time
 - also called off-line storage
 - E.g., magnetic tape, optical storage

Outline

- ☐ Overview of Physical Storage Media
- ☐ **Magnetic Disks**
- ☐ * RAID
- ☐ * Tertiary Storage
- ☐ Storage Access
- ☐ File Organization
- ☐ * Organization of Records in Files
- ☐ * Data-Dictionary Storage
- ☐ Column-Oriented Storage



Magnetic Hard Disk Mechanism



NOTE: Diagram is schematic, and simplifies the structure of actual disk drives.

Magnetic Disks

❑ Read-write head

- Positioned **very close to the platter surface** (almost touching it)
- 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

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

Magnetic Disks (Cont.)

❑ 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

❑ Head-disk assemblies

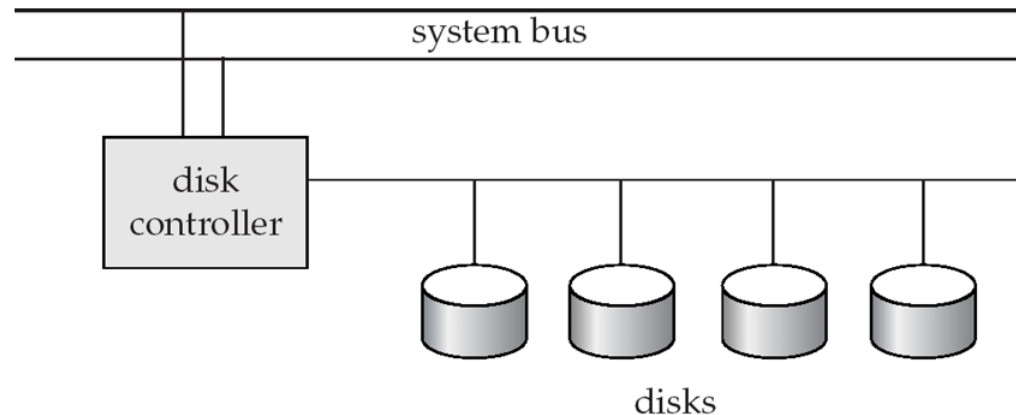
- Multiple disk platters on a single spindle (typically 4 to 16 platters)
- One head per platter, mounted on a common arm

❑ Cylinder i consists of i^{th} track of all the platters

Magnetic Disks (Cont.)

- ❑ 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（坏扇区的重映射：将该扇区从逻辑上映射到预留的物理扇区，并且重映射被记录在磁盘或其他非易失性存储器中）

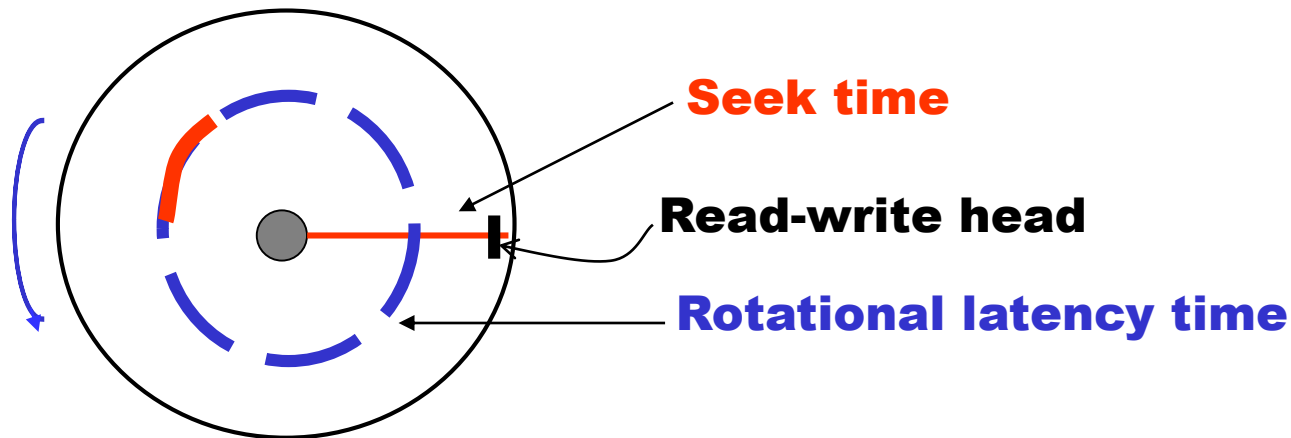
Disk Subsystem



- ❑ Multiple disks connected to a computer system through a controller
 - Controllers functionality (checksum, bad sector remapping) often carried out by individual disks; reduces load on controller
- ❑ Disk interface standards families
 - **ATA** (AT adaptor) 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)

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**(寻道时间) + **Rotational latency**(旋转等待时间)
- **Seek time** – it takes to **reposition the arm over the correct track**
 - 4 to 10 milliseconds **on typical disks**
 - **Rotational latency time** – it takes for the sector to be accessed to appear under the head.
 - Average latency is 1/2 of the worst case latency
 - 4 to 11 milliseconds on typical disks (**5400 to 15000 r.p.m.**)



Performance Measures of Disks (Cont.)

❑ **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. **SATA**: 150 MB/sec, SATA-II 3Gb/sec, SATA-III: 6Gb/sec
 - Ultra 320 **SCSI**: 320 MB/s, SAS (3 to 6 Gb/sec)
 - Fiber Channel (FC2Gb or 4Gb): 256 to 512 MB/s

Example

❑ IBM Deskstar 14GPX

- 3.5 INCH, 7200R/minute
- 容量14.4GB, 含5张双面盘片, 每张约3.35GB
- 平均寻道时间 9.1 ms
 - 相邻磁道寻道时间 2.2ms
 - 最大寻道时间15.5ms
- 平均旋转等待时间 4.17 ms
- 数据传输速率 13Mb/s ($\approx 610\text{ns/B}$)。故磁盘读写主要是寻道时间和旋转等待时间, 相比之下数据传输 (真正的读写)时间可略(0.3ms/sector)
- 故磁盘存取时间 $>10\text{ms}$,
而内存的存取时间 $\approx 10\text{ ns}$ } 相差 ≈ 100 万倍

犹如蜗牛比飞机

23 days vs. 2 seconds
2000 KM vs. 2 M } 100万倍

Many problems are derived from these fact !

Performance Measures of Disks (Cont.)

- ❑ **Mean time to failure (MTTF, 平均故障时间):** 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
 - MTTF decreases as disk ages

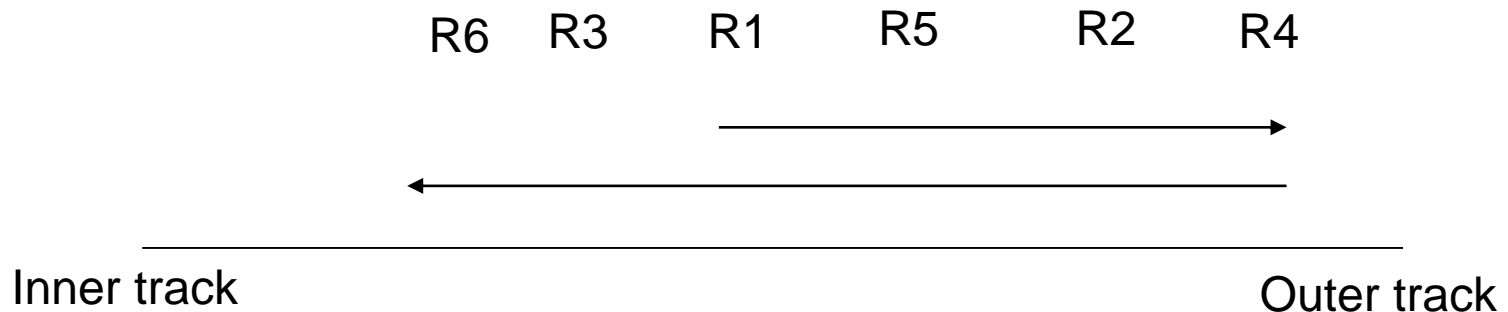
Optimization of Disk-Block Access

❑ **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 algorithms** (磁盘臂调度算法) order pending accesses to **tracks** so that disk arm movement is minimized

- **Elevator algorithm** (电梯算法): 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 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

- ❑ But the system is generally unusable when these utilities are running.

Optimization of Disk-Block Access (Cont.)

- ❑ **Nonvolatile write buffers** (非易失性写缓冲区) speed up disk writes by writing blocks to a non-volatile RAM buffer immediately
 - Non-volatile 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
 - Then *writes can be reordered to minimize disk arm movement*

Optimization of Disk-Block Access (Cont.)

- ❑ **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

Outline

- ☐ 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



*RAID

- ❑ RAID: Redundant Arrays of Independent Disks(独立磁盘冗余阵列)
 - Disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
 - High capacity and high speed by using multiple disks in parallel, and
 - High reliability by storing data redundantly, so that data can be recovered even if a disk fails
- ❑ The chance that some disk out of a set of N disks will fail is much higher than the chance that a specific single disk will fail
 - E.g., a system with 100 disks, each with MTTF of 100,000 hours (approx. 11 years), will have a system MTTF of 1000 hours (approx. 41 days)
 - Techniques for using redundancy to avoid data loss are critical with large numbers of disks
- ❑ Originally a cost-effective alternative to large, expensive disks
 - I in RAID originally stood for “inexpensive”
 - Today RAIDs are used for their higher reliability and bandwidth
 - The “I” is interpreted as independent
- ❑ RAID技术从二个方面改善系统性能：冗余---可靠性、并行性---速度

Reliability Improvement via Redundancy

- ❑ **Redundancy**: Store extra information that can be used to rebuild information lost in a disk failure
- ❑ E.g., **mirroring** (or **shadowing**)
 - Duplicate every disk; Logical disk consists of two physical disks
 - Every write is carried out on both disks
 - Reads can take place from either disk
 - If one disk in a pair fails, data still available in the other
 - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
 - Probability of combined event is very small
 - » Except for dependent failure modes such as fire or building collapse or electrical power surges
- ❑ **Mean time to data loss** depends on mean time to failure, and **mean time to repair**
 - E.g., MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of 500×10^6 hours ($= 100000^2 / (2 \times 10) = 57,000$ years) for a mirrored pair of disks (ignoring dependent failure modes)

Performance Improvement via Parallelism

- ❑ Two main goals of parallelism in a disk system:
 - Load balance multiple small accesses to increase throughput
 - Parallelize large accesses to reduce response time
- ❑ Improve transfer rate by striping data across multiple disks
- ❑ **Bit-level striping** (比特级拆分): 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** (块级拆分): With n disks, block i of a file goes to disk $(i \bmod n) + 1$
 - 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 lost 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 2:** Memory-Style Error-Correcting-Codes (ECC) with bit striping.

❑ **RAID Level 3:** Bit-Interleaved Parity

- A single parity bit is enough for error correction, not just detection, since we know which disk has failed
 - When writing data, corresponding parity bits must also be computed and written to a parity bit disk
 - To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk)
- Faster data transfer than with a single disk, but fewer I/Os per second since every disk has to participate in every I/O.
- Subsumes Level 2 (provides all its benefits, at lower cost).



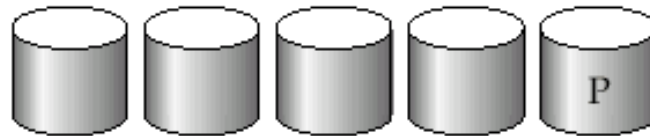
(c) RAID 2: memory-style error-correcting codes



(d) RAID 3: bit-interleaved parity

RAID Levels (Cont.)

- ❑ **RAID Level 4:** Block-Interleaved Parity; uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from N other disks.
 - When writing data block, corresponding block of parity bits must also be computed and written to parity disk
 - To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks.



(e) RAID 4: block-interleaved parity

RAID Levels (Cont.)

❑ RAID Level 4 (Cont.):

- Provides higher I/O rates for independent block reads than Level 3
 - Block read goes to a single disk, so blocks stored on different disks can be read in parallel
- Provides high transfer rates for reads of multiple blocks than no-striping
- Before writing a block, parity data must be computed
 - Can be done by using old parity block, old value of current block and new value of current block (2 block reads + 2 block writes)
 - Or by recomputing the parity value using the new values of blocks corresponding to the parity block
 - More efficient for writing large amounts of data sequentially
- Parity block becomes a bottleneck for independent block writes since every block write also writes to parity disk

RAID Levels (Cont.)

- ❑ **RAID Level 5:** Block-Interleaved Distributed Parity; partitions data and parity among all $N + 1$ disks, rather than storing data in N disks and parity in 1 disk.
 - E.g., with 5 disks, parity block for n th set of blocks is stored on disk $(n \bmod 5) + 1$, with the data blocks stored on the other 4 disks.



(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

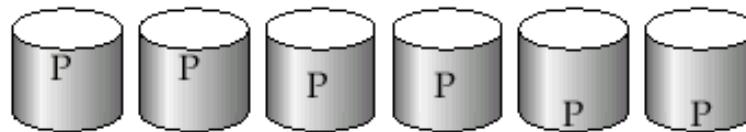
RAID Levels (Cont.)

❑ RAID Level 5 (Cont.)

- Higher I/O rates than Level 4.
 - Block writes occur in parallel if the blocks and their parity blocks are on different disks.
- Subsumes Level 4: provides same benefits, but avoids bottleneck of parity disk.

❑ RAID Level 6: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against **multiple disk failures**.

- Better reliability than Level 5 at a higher cost; not used as widely.



(g) RAID 6: P + Q redundancy

Choice of RAID Level

❑ Factors in choosing RAID level

- Monetary cost
- Performance: Number of I/O operations per second, and bandwidth during normal operation
- Performance during failure
- Performance during rebuild of failed disk
 - Including time taken to rebuild failed disk

❑ RAID 0 is used only when data safety is not important

- E.g., data can be recovered quickly from other sources

❑ Level 2 and 4 never used since they are subsumed by 3 and 5

❑ Level 3 is not used anymore since bit-striping forces single block reads to access all disks, wasting disk arm movement, which block striping (level 5) avoids

❑ Level 6 is rarely used since levels 1 and 5 offer adequate safety for almost all applications

❑ So competition is between 1 and 5 only

Choice of RAID Level (Cont.)

- ❑ Level 1 provides much better write performance than level 5
 - Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes
 - Level 1 preferred for high update environments such as log disks
- ❑ Level 1 had higher storage cost than level 5
 - Disk drive capacities increasing rapidly (50%/year) whereas disk access times have decreased much less (x 3 in 10 years)
 - I/O requirements have increased greatly, e.g. for Web servers
 - When enough disks have been bought to satisfy required rate of I/O, they often have spare storage capacity
 - So there is often no extra monetary cost for Level 1!
- ❑ Level 5 is preferred for applications with low update rate, and large amounts of data.
- ❑ Level 1 is preferred for all other applications.

Hardware Issues

- ❑ **Software RAID:** RAID implementations done entirely in software, with no special hardware support.
- ❑ **Hardware RAID:** RAID implementations with special hardware
 - Use non-volatile RAM to record writes that are being executed
 - Beware: power failure during write can result in corrupted disk
 - E.g., failure after writing one block but before writing the second in a mirrored system
 - Such corrupted data must be detected when power is restored
 - Recovery from corruption is similar to recovery from failed disk
 - NV-RAM helps to efficiently detected potentially corrupted blocks
 - » Otherwise all blocks of disk must be read and compared with mirror/parity block

Hardware Issues (Cont.)

- ❑ **Latent failures:** Data successfully written earlier gets damaged
 - Can result in data loss even if only one disk fails
- ❑ **Data scrubbing:**
 - Continually scan for latent failures, and recover from copy/parity
- ❑ **Hot swapping (热插拔):** Replacement of disk while system is running, without power down
 - Supported by some hardware RAID systems,
 - Reduces time to recovery, and improves availability greatly
- ❑ Many systems maintain **spare disks** which are kept online, and used as replacements for failed disks immediately on detection of failure
 - Reduces time to recovery greatly
- ❑ Many hardware RAID systems ensure that a single point of failure will not stop the functioning of the system by using
 - **Redundant power supplies** with battery backup
 - **Multiple controllers** and **multiple interconnections** to guard against controller/interconnection failures

Optical Disks

❑ Compact disk-read only memory (CD-ROM)

- Removable disks, 640 MB per disk
- Seek time about 100 msec (optical read head is heavier and slower)
- Higher latency (3000 RPM) and lower data-transfer rates (3-6 MB/s) compared to magnetic disks

❑ Digital Video Disk (DVD)

- DVD-5 holds 4.7 GB , and DVD-9 holds 8.5 GB
- DVD-10 and DVD-18 are double sided formats with capacities of 9.4 GB and 17 GB
- Blu-ray DVD: 27 GB (54 GB for double sided disk)
- Slow seek time, for same reasons as CD-ROM

❑ Record once versions (CD-R and DVD-R) are popular

- data can only be written once, and cannot be erased.
- high capacity and long lifetime; used for archival storage
- Multi-write versions (CD-RW, DVD-RW, DVD+RW and DVD-RAM) also available

Magnetic Tapes

- ❑ Hold large volumes of data and provide high transfer rates
 - Few GB for DAT (Digital Audio Tape) format, 10-40 GB with DLT (Digital Linear Tape) format, 100 GB+ with Ultrium format, and 330 GB with Ampex helical scan format
 - Transfer rates from few to 10s of MB/s
- ❑ Tapes are cheap, but cost of drives is very high
- ❑ Very slow access time in comparison to magnetic and optical disks
 - limited to sequential access.
 - Some formats (Accelis) provide faster seek (10s of seconds) at cost of lower capacity
- ❑ Used mainly for backup, for storage of infrequently used information, and as an off-line medium for transferring information from one system to another.
- ❑ Tape jukeboxes used for very large capacity storage
 - Multiple petabytes (10^{15} bytes)

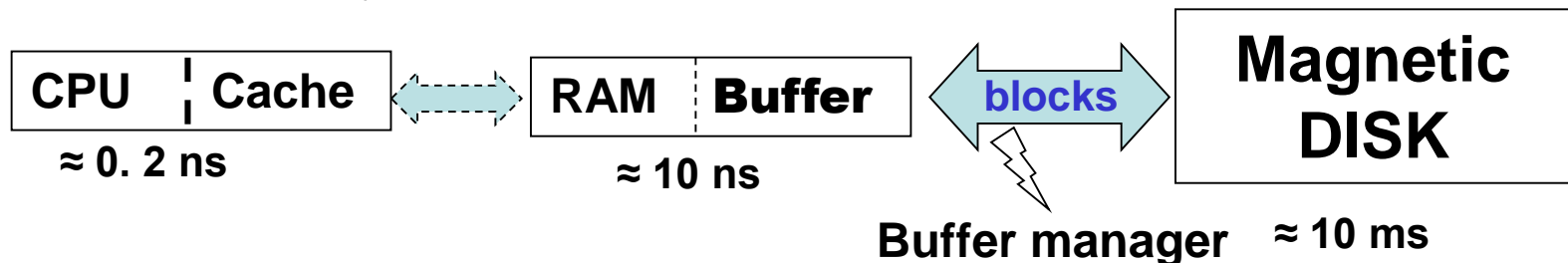
Outline

- ☐ 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

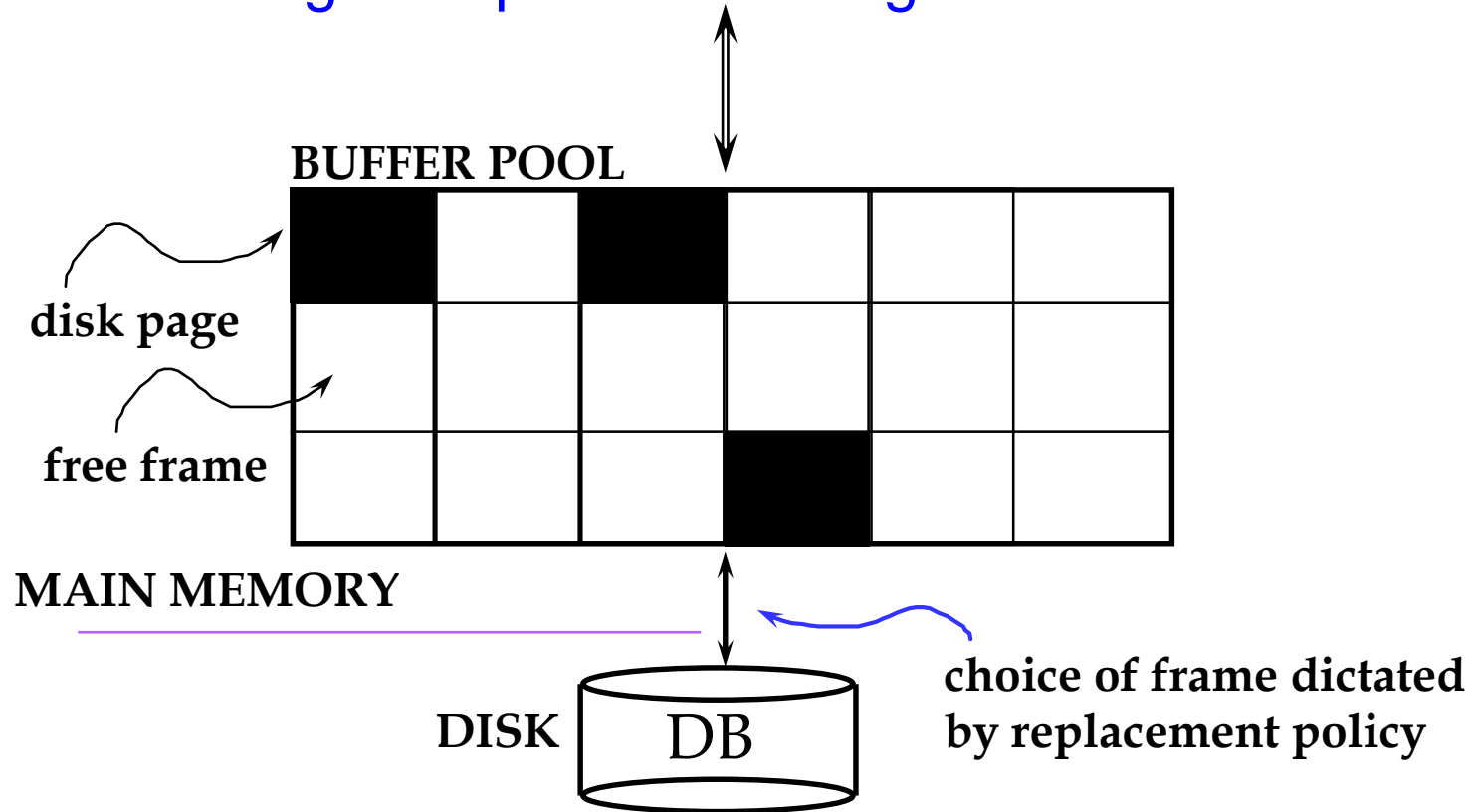


Storage Access

- ❑ A database file is **logically** partitioned into fixed-length storage units called blocks. **Blocks are units** of both **storage allocation and data transfer** in database system.
- ❑ **Buffer**: Portion of main memory available to store copies of disk blocks.
- ❑ Database system seeks to **minimize** the number of block transfers between the disk and memory.
 - To reduce the number of disk accesses: **keeping as many blocks as possible in main memory --- Buffer.**
 - **But buffer is limited in size. How to do with?**
- ❑ **Buffer manager**: Subsystem responsible for allocating buffer space in main memory.



Page Requests from Higher Levels



Data must be in RAM for DBMS to operate on it!

Table of **<frame#, pageid>** pairs is maintained.

Page : a unit of data ; Block: a unit of disk space;
Frame: a unit of buffer pool; in practice block \approx page

Buffer Manager

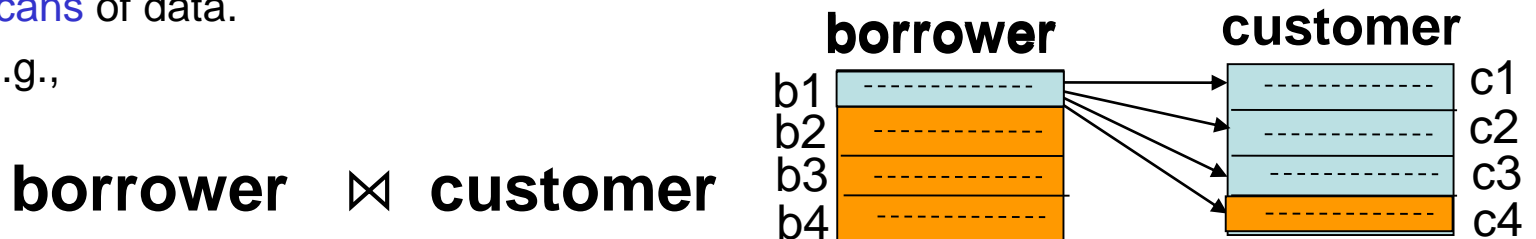
- ❑ Application programs call on the buffer manager when they need a block from disk.
 - If the **block is already in the buffer**, the requesting program is given the address of the block in main memory
 - If the **block is not in the buffer**
 - The buffer manager **allocates space in the buffer for the block, replacing (throwing out) some old pages; if no free spaces**, to make space for the new block. (在buffer中为新页分配空间)
 - The block that is thrown out is written back to disk **only if it was modified** since the most recent time that it was written to/fetched from the disk. (将被覆盖的旧块若已被修改过，则写回磁盘)
 - Once space is allocated in the buffer, the buffer manager reads the block from the disk to the buffer, and passes the address of the block in main memory to requester. (从磁盘读入新块放buffer)

Buffer Manager (Cont.)

- ❑ **Buffer-Replacement Policies:** LRU, MRU
- ❑ **Pinned block** (被钉住的块): Memory block that is not allowed to be written back to disk. (如当前块正在被使用时)
- ❑ **Toss-immediate strategy:** Frees the space occupied by a block as soon as the final tuple of that block has been processed. (用后立即丢弃)
- ❑ **Forced output of blocks.**
 - Requestor of block must unpin it, and indicate whether page has been modified:
 - **Dirty** bit is used for this.
- ❑ **Page in pool may be requested many times** (被多个事务使用)
 - A **pin count** is used, and a page is a candidate for replacement iff *pin count* = 0.

Buffer-Replacement Policies

- ❑ 当Buffer的空闲区不够，不能容下新读入的Block时，需要将Buffer中原有Block覆盖（替换）。主要策略为：
- ❑ (1) **LRU strategy (Least Recently Used, 最近最少使用策略)**: Replace the block which was least recently used.
 - Idea behind LRU: Use past pattern of block references as a predictor for future references. (如：职称评审、高考录取，...)
 - Queries have well-defined access patterns, and a database system can use the information in a user's query to predict future references.
 - But LRU can be a **bad strategy** for certain access patterns involving repeated **scans** of data.
 - E.g.,



设 $M = 5$ (buffer has 5 blocks), 1 for borrower, 3 for customer, 1 for out
对customer, LRU的块可能是下面快要用的块(循环)，而最近刚用过的块则
暂时不用，当空间不够时倒是可以将其覆盖的，故**LRU策略不佳**。

Buffer-Replacement Policies (Cont.)

- ❑ (2) **MRU strategy (Most recently used, 最近最常用策略)**: System must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the **most recently used block**.
- ❑ **Buffer manager** can use **statistical information** regarding the probability that a request will reference a particular relation
 - E.g., the data dictionary is frequently accessed. **Heuristic: Keep data-dictionary blocks in main memory buffer.**
- ❑ **Buffer manager** also support forced output of blocks for the purpose of recovery (more in Chapter 16)
- ❑ **Mixed strategy** with hints on replacement strategy provided by the **query optimizer** is preferable.

Outline

- ☐ Overview of Physical Storage Media
- ☐ Magnetic Disks
- ☐ * RAID
- ☐ * Tertiary Storage
- ☐ Storage Access
- ☐ **File Organization**
- ☐ * Organization of Records in Files
- ☐ * Data-Dictionary Storage
- ☐ Column-Oriented Storage



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.
- ❑ Two kinds of record:
 - **Fixed-length** records.
 - **Variable-length** records.

Fixed-Length Records

❑ Advantage: simple approach:

- Store record i starting from byte $n * (i - 1)$, where n is the size of each record.
- Record access is simple but records may cross blocks
 - Modification: do not allow records to cross block boundaries

❑ Deletion of record i : alternatives:

- **Method 1:** move records $i + 1, \dots, n$ to $i, \dots, n - 1$
- **Method 2:** move record n to i
- **Method 3:** 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

Free Lists

- ❑ Store the address of the first deleted record in the file header (还有其它信息).
- ❑ Use this first record to store the address of the second deleted record, and so on
- ❑ Can think of these stored addresses as **pointers** since they “point” to the location of a record.
- ❑ **Advantage:** more space efficient representation: reuse space for normal attributes of free records to store pointers. (No pointers stored in in-use records.)

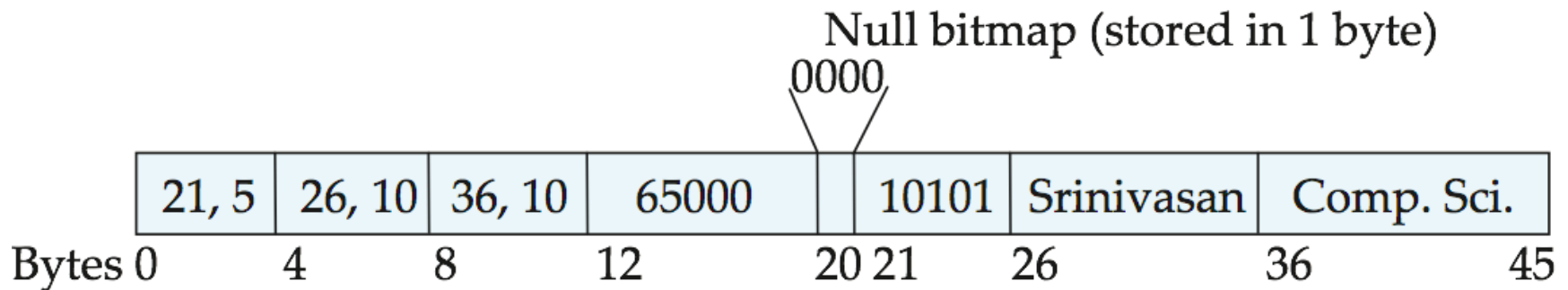
header
record 0
record 1
record 2
record 3
record 4
record 5
record 6
record 7
record 8
record 9
record 10
record 11

10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

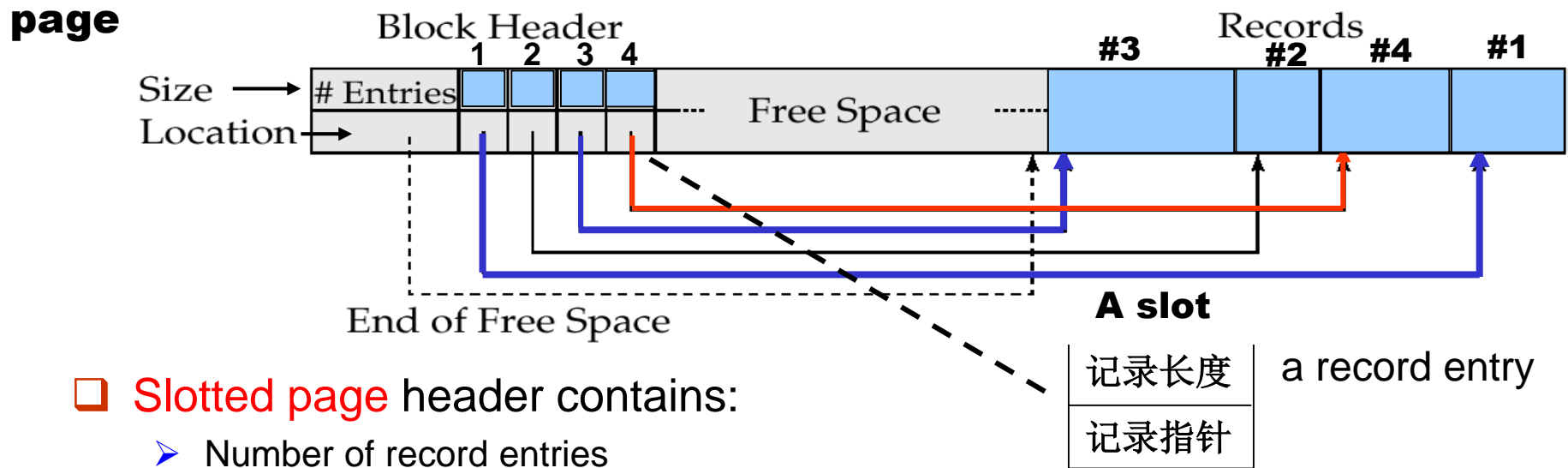
Link for free list

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



Slotted Page Structure



❑ Slotted page header contains:

- Number of record entries
- End of free space in the block
- Location and size of each record

rid = < slot# pid >

- ❑ 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. (页内无碎块: 删除时页内移动记录)
- ❑ Pointers (Index) should not point directly to record — instead they should point to the entry for the record in header---indirect pointer(间接指针)

Fixed-length Representation

❑ Fixed-length representation:

- A) reserved space
- B) pointers

❑ A) **Reserved space**: can use fixed-length records of a known **maximum length**; unused space in shorter records filled with a null or end-of-record symbol.

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	⊥	⊥	⊥	⊥

Branch-name

acc-num

acc-num

acc-num

bal

bal

bal

Pointer Method

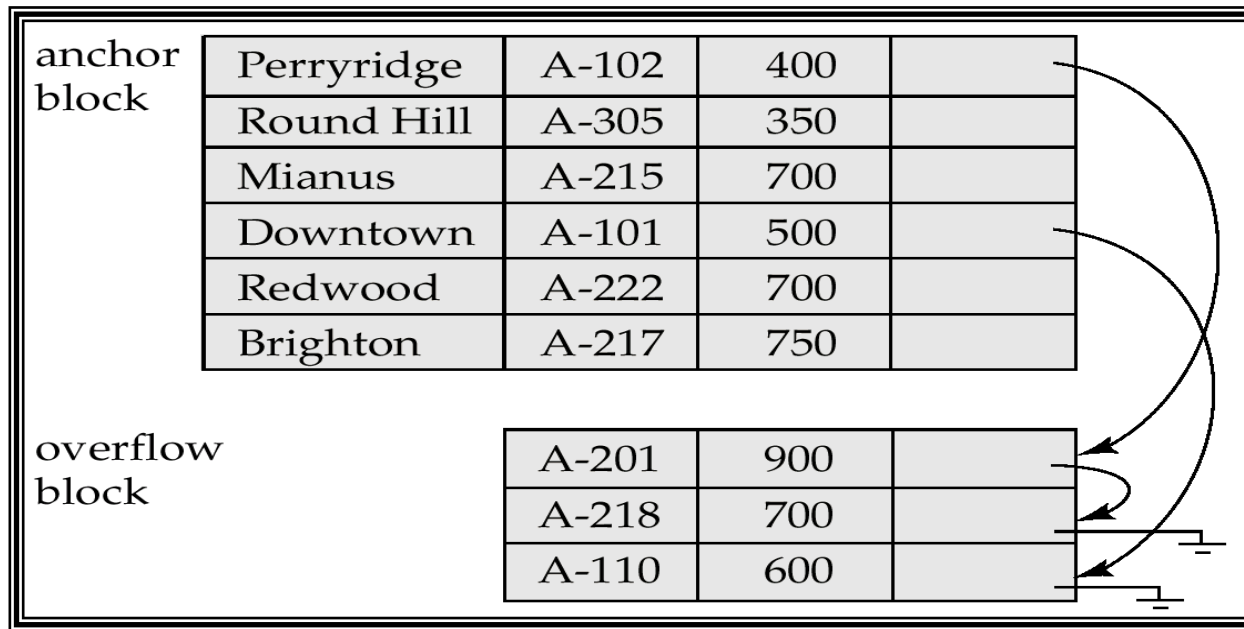
❑ Pointer method

- A variable-length record is represented by a list of fixed-length records, **chained together via pointers**.
- Can be used even if the maximum record length is not known



Pointer Method (Cont.)

- ❑ Disadvantage to pointer structure: space(for branch-name) is wasted in all records except the first in a chain.
- ❑ Solution is to allow two kinds of block in file:
 - Anchor block(锚块) – contains the first records of chain
 - Overflow block(溢出块) – contains records other than those that are the first records of chains.



Outline

- ☐ Overview of Physical Storage Media
- ☐ Magnetic Disks
- ☐ * RAID
- ☐ * Tertiary Storage
- ☐ Storage Access
- ☐ File Organization
- ☐ * Organization of Records in Files
- ☐ * Data-Dictionary Storage
- ☐ Column-Oriented Storage



Organization of Records in Files

- ❑ **Heap file (堆文件, 流水文件)**: a record can be placed anywhere in the file where there is space
- ❑ **Sequential file (顺序文件)**: store records in **sequential order**, based on the value of a **search key** of each record
- ❑ **Hashing file (散列文件)**: a **hash function** computed on **some attribute** of each record; the result specifies in which block of the file the record should be placed
- ❑ **Clustering file organization (聚集文件组织)**: records of several different relations can be stored in the same file
 - Motivation: store related records **in different relations** 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**

Search key:
Branch name



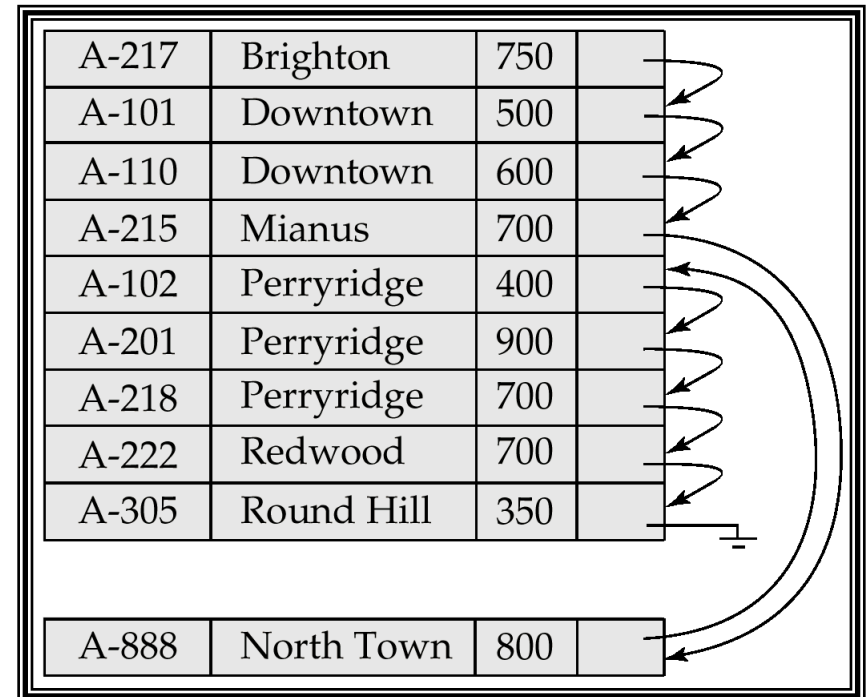
A-217	Brighton	750
A-101	Downtown	500
A-110	Downtown	600
A-215	Mianus	700
A-102	Perryridge	400
A-201	Perryridge	900
A-218	Perryridge	700
A-222	Redwood	700
A-305	Round Hill	350

Account-number / **branch-name** / balance

**A sequential file only
has one search key ?**

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.** (需要定期对文件重新排序)



Multitable Clustering File Organization

Store several relations in one file using a **multitable clustering** file organization

department

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Physics	Watson	70000

instructor

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

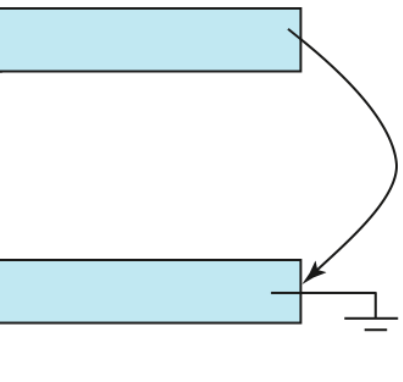
multitable clustering
of *department* and
instructor

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

Multitable Clustering File Organization (Cont.)

- ❑ Good for queries involving *department* ⋈ *instructor*, and for queries involving one single department and its instructors
- ❑ Bad for queries involving only *department*
- ❑ Results in variable size records
- ❑ Can add pointer chains to link records of a particular relation

Comp. Sci.	Taylor	100000	
45564	Katz	75000	
10101	Srinivasan	65000	
83821	Brandt	92000	
Physics	Watson	70000	
33456	Gold	87000	



Outline

- ❑ Overview of Physical Storage Media
- ❑ Magnetic Disks
- ❑ * RAID
- ❑ * Tertiary Storage
- ❑ Storage Access
- ❑ File Organization
- ❑ * Organization of Records in Files
- ❑ * **Data-Dictionary Storage**
- ❑ Column-Oriented Storage



Data Dictionary Storage

Data dictionary (also called system catalog) stores metadata: that is, data about data, such as:

❑ Information about relations

- Names of relations
- Names and types 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
 - Operating system file name or
 - Disk addresses of blocks containing records of the relation

❑ Information about indices (Chapter 14)

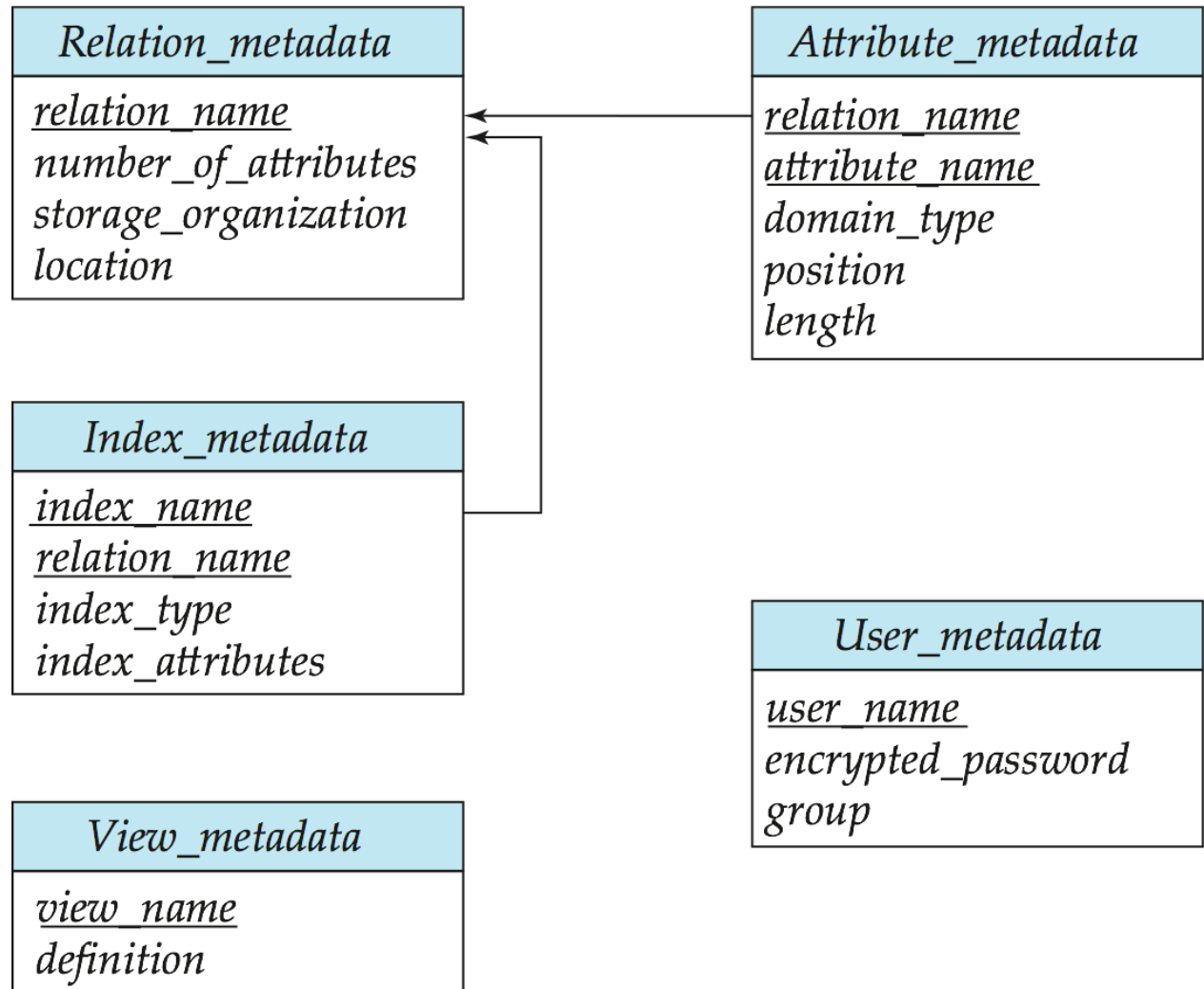
Outline

- ❑ Overview of Physical Storage Media
- ❑ Magnetic Disks
- ❑ * RAID
- ❑ * Tertiary Storage
- ❑ Storage Access
- ❑ File Organization
- ❑ * Organization of Records in Files
- ❑ * Data-Dictionary Storage
- ❑ **Column-Oriented Storage**



Relational Representation of System Metadata

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



Column-Oriented Storage

- ❑ Also known as **columnar representation**
- ❑ Store each attribute of a relation separately
- ❑ Example

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

Columnar Representation

❑ Benefits:

- Reduced IO if only some attributes are accessed
- Improved CPU cache performance
- Improved compression
- **Vector processing** on modern CPU architectures

❑ Drawbacks

- Cost of tuple reconstruction from columnar representation
- Cost of tuple deletion and update
- Cost of decompression

❑ Columnar representation found to be more efficient for decision support than row-oriented representation

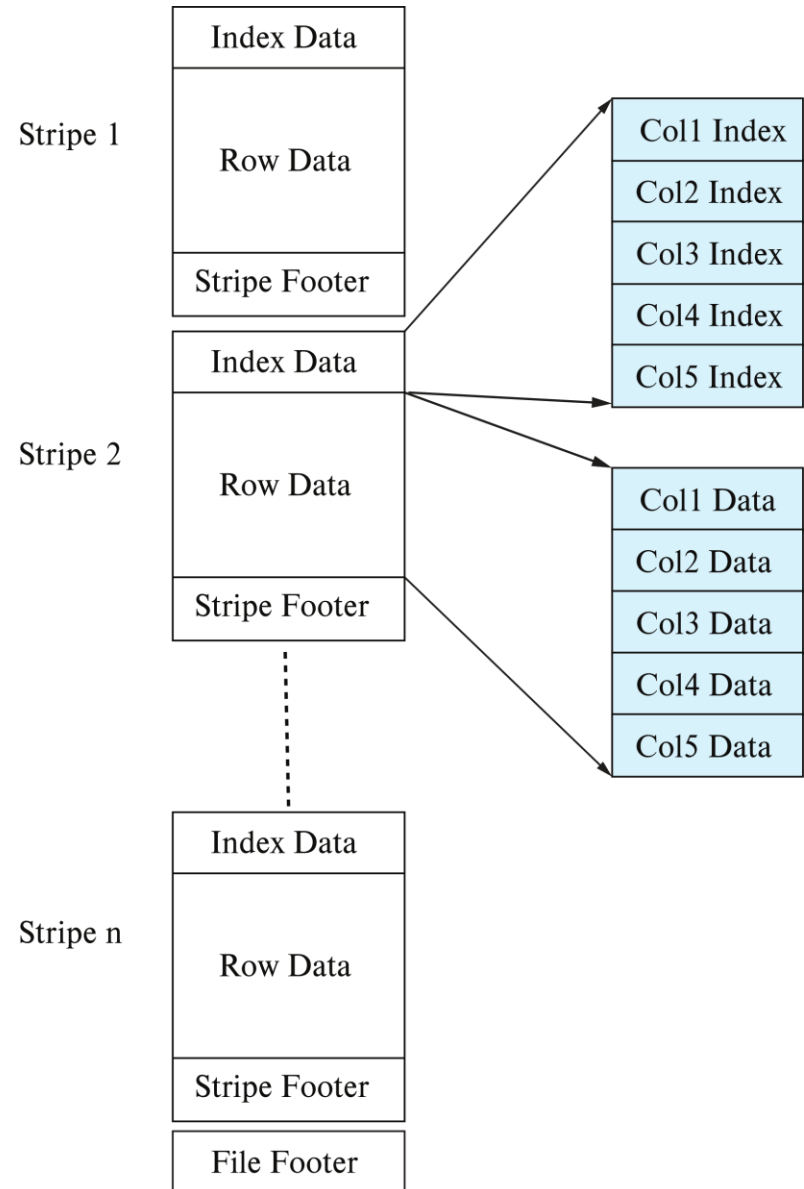
❑ Traditional row-oriented representation preferable for transaction processing

❑ Some databases support both representations

- Called **hybrid row/column stores**

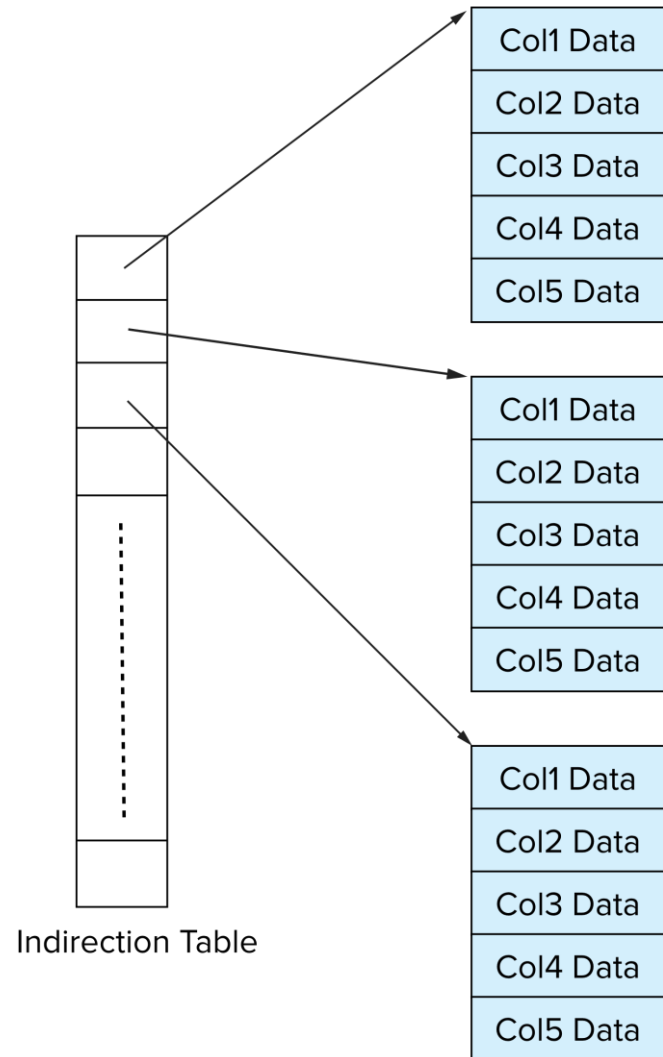
Columnar File Representation

- ❑ ORC and Parquet: file formats with columnar storage inside file
- ❑ Very popular for big-data applications
- ❑ Orc file format shown on right:

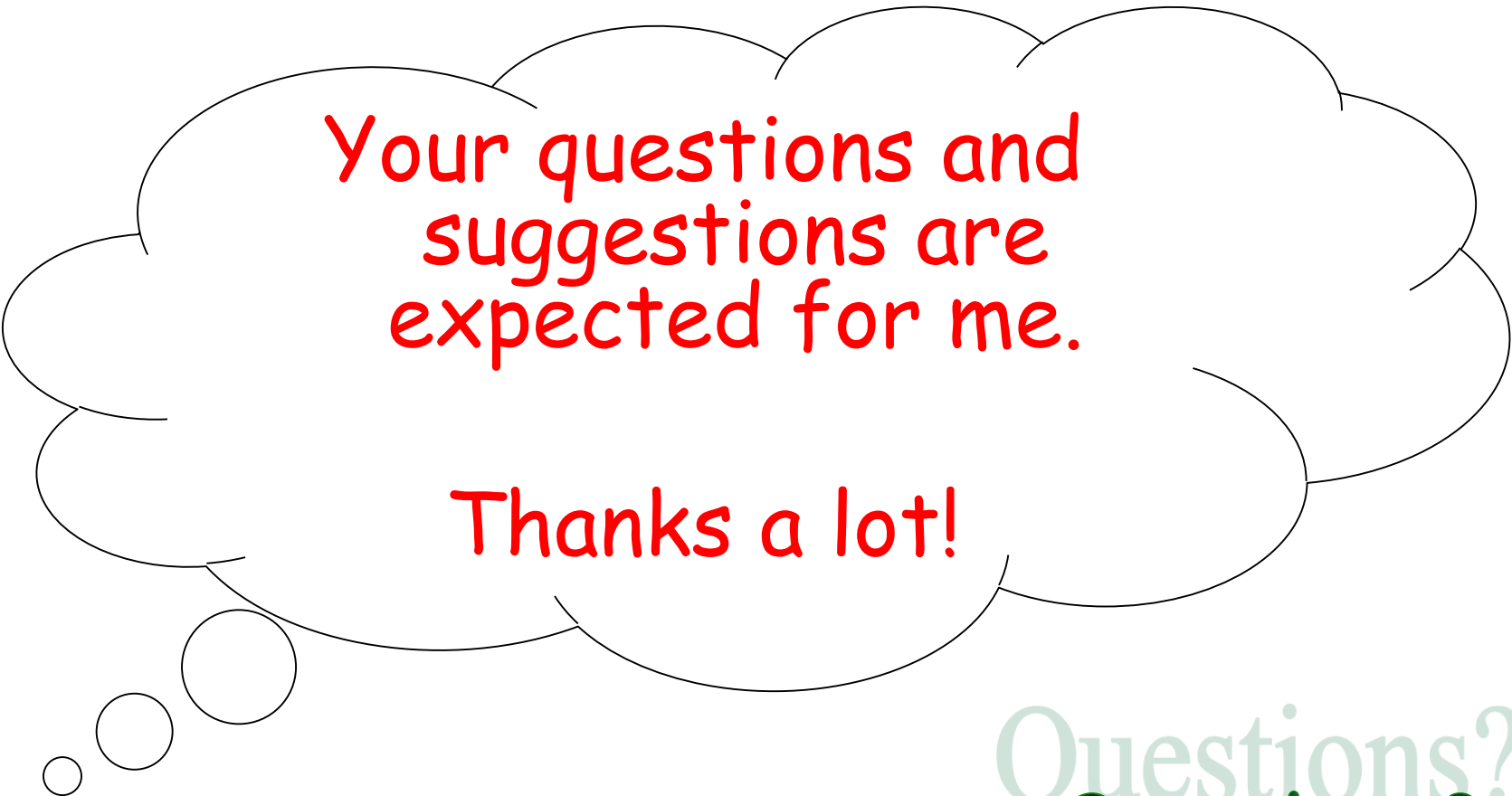


Storage Organization in Main-Memory Databases

- ❑ Can store records directly in memory without a buffer manager
- ❑ Column-oriented storage can be used in-memory for decision support applications
 - Compression reduces memory requirement



Q & A



Your questions and
suggestions are
expected for me.

Thanks a lot!

Questions?
Questions?

Exercises: 12.13 and 13.11 (see Page 584 and
Page 620)