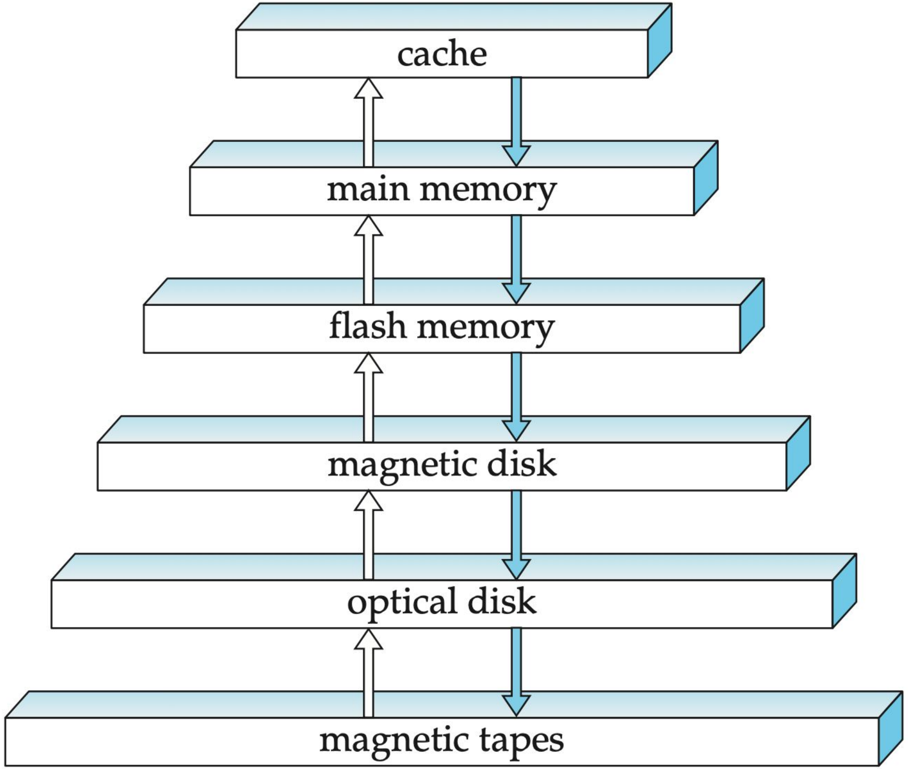
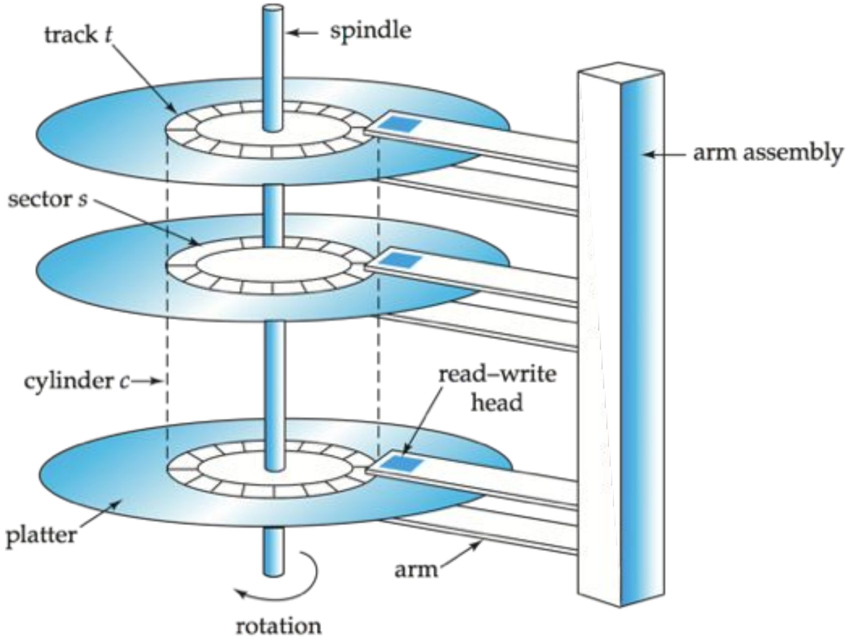
Data Storage and File Structure

Outline

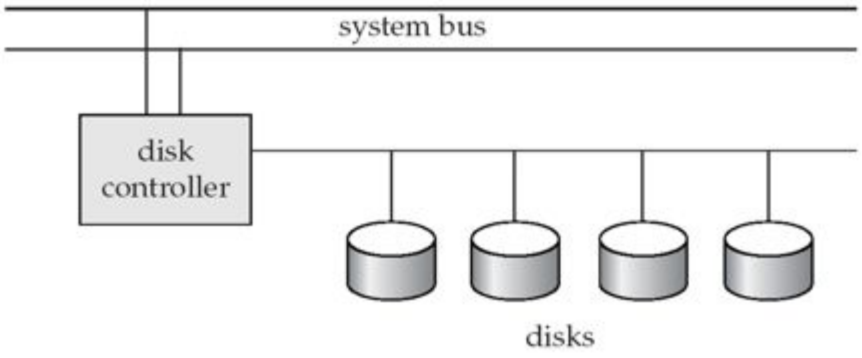
- Storage
- RAID
- File Organization

Storage


cache
main memory
flash memory
magnetic disk
optical disk
magnetic tapes


track t
spindle
arm assembly
sector s
cylinder c
read–write head
platter
arm
rotation

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 (Advanced Technology attachment/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 measure

- Access time
  - Seek time - Time taken to access the track
  - Rotational latency - Time taken to access the sector within a track

- Data-transfer rate

- 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

RAID

- Redundant Arrays of Independent Disks
  - 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,
    - high reliability by storing data redundantly, so that data can be recovered even if a disk fails

- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
  - **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
      - Probability of combined event is very small

RAID

- Performance 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** – 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.

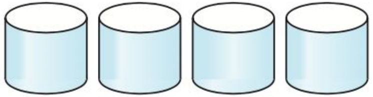
- **Block-level striping** – with *n* disks, block *i* of a file goes to disk (*i* mod *n*) + 1
  - Requests for different blocks can run in parallel if the blocks reside on different disks
  - A request for a long sequence of blocks can utilize all disks in parallel

RAID levels

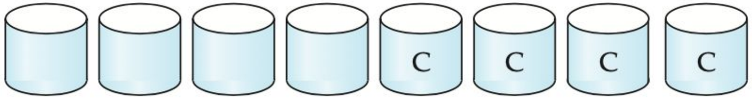
- Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
  - Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics

- **RAID Level 0**: Block striping; non-redundant.
  - Used in high-performance applications where data loss 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



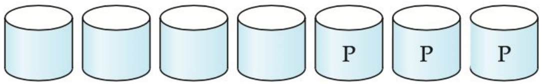
(b) RAID 1: mirrored disks

RAID levels

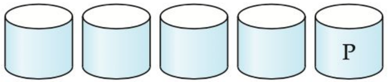
- **RAID Level 2**: Memory-Style Error-Correcting-Codes (ECC) with bit striping.

- **RAID Level 3**: Bit-Interleaved Parity
  - Disk controllers, unlike memory systems, can detect whether a sector has been read correctly, so a single parity bit can be used for error correction, as well as for detection.
  - RAID level 3 is as good as level 2, but is less expensive.

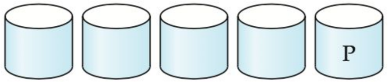
- **RAID Level 4:** Block-Interleaved Parity;
  - uses block-level striping, and keeps a parity block on a se
    from *N* other disks.



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



(d) RAID 3: bit-interleaved parity



(e) RAID 4: block-interleaved parity

RAID levels

- **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.
  - For example, with an array of 5 disks, the parity block, labeled Pk,
    - For logical blocks 4k, 4k + 1, 4k + 2, 4k + 3 is stored in disk k mod 5;
    - The corresponding blocks of the other four disks store the 4 data blocks 4k to 4k + 3

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

(f) RAID 5: block-interleaved distributed parity

File organization

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*.

- One approach:
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations

  This case is easiest to implement; will consider variable length records later.

File organization

type *instructor* = **record**

ID **varchar** (5);

name **varchar**(20);

dept_name **varchar** (20); *salary* **numeric** (8,2);

end

- 53 bytes per record per block;
- Records may cross blocks
  - Modification: do not allow records to cross block boundaries

- Deletion of record *i:*
  a. move records $i + 1, \ldots, n$ to $i, \ldots, n - 1$
  b. move record n to i
  c. 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

Deletion 0f record 3

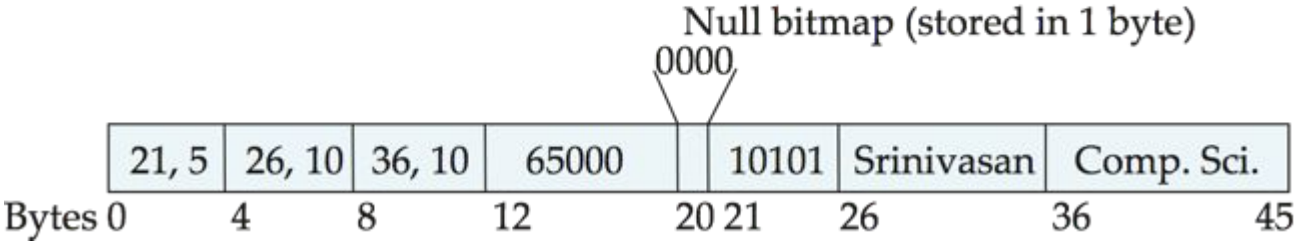
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

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

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

Variable-length record

- 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



Sequential file organization

- Data must be stored in order based on some key.

- Deletion – use pointer chains
- Insertion –
  a. locate the position where the record is to be inserted
     i. if there is free space insert there
     ii. if no free space, insert the record in an overflow block
     iii. 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

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	

Multi-table clustering file organization

- Store several relations in one file using a **multi-table clustering** file organization

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
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

Multi-table clustering file organization

- good for queries involving *department* X *instructor*, and for queries involving one single department and its instructors
- bad for queries involving only a single table, e.g., *department*

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

End

Home - Exercise

- Design a database for keeping track of inventory of items sold in a brick and mortar apparel retail store. Consider that the retail store has multiple outlets, inventory m monitoring is done at each store level and HQ level (by store/region/state/country).
- Normalize your database design to be in 1NF, 2NF, 3NF and BCNF.

- Important Note:
  - *No marks*, *no submission* required. However, you are most welcome to discuss your solution with your course-mates/instructors/TF/TA.