No se puede mostrar la imagen.

Storage and File Structure
Updated: Sonia Ordóñez S.

Database System Concepts, 6th Ed.

Classification of Physical Storage Media

- Main Issues
  - 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

- Can differentiate storage into:
  - **volatile storage**: loses contents when power is switched off
  - **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter-backed up main-memory.

Physical Storage Media

- **Cache**
  - fastest and most costly form of storage
  - Volatile
  - managed by the computer system hardware.

- **Main memory:**
  - fast access (10s to 100s of nanoseconds; 1 nanosecond = 10–9 seconds)
  - generally too small (or too expensive) to store the entire database
    - capacities of up to a few Gigabytes widely used currently (2 – 16 GB)
    - 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**
  - Data survives 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
  - But writes are slow (few microseconds), erase is slower
  - Widely used in embedded devices such as digital cameras, phones, and USB keys
  - also known as EEPROM (Electrically Erasable Programmable Read-Only Memory)

Physical Storage Media (Cont.)

- **Magnetic-disk**
  - Data is stored on spinning disk, and read/written magnetically
  - 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
  - Hard disks vs floppy disks
  - Capacities range up to roughly 1.5 TB as of 2009
    - Much larger capacity and 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 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
  - Blu-ray disks: 27 GB to 54 GB
  - Write-one, read-many (WORM) optical disks used for archival storage (CD-R, DVD-R, DVD+R)
  - Multiple write versions also available (CD-RW, DVD-RW, DVD+RW, and DVD-RAM)
  - Reads and writes are slower than with magnetic disk
  - **Juke-box** systems
    - large numbers of removable disks and a few drives
    - 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
    - storage costs much cheaper than disk, but drives are expensive
  - Tape jukeboxes available for storing massive amounts of data
    - hundreds of terabytes (1 terabyte = 109 bytes) to even multiple petabytes (1 petabyte = 1012 bytes)

Physical Storage Media (Cont.)

- The storage capacity of a solid hard drive is currently up to 8TB and the price per TB is much higher than that of a mechanical hard drive
- It has a very marked programmed obsolescence.
- Generally used for data analysis

Flash Storage

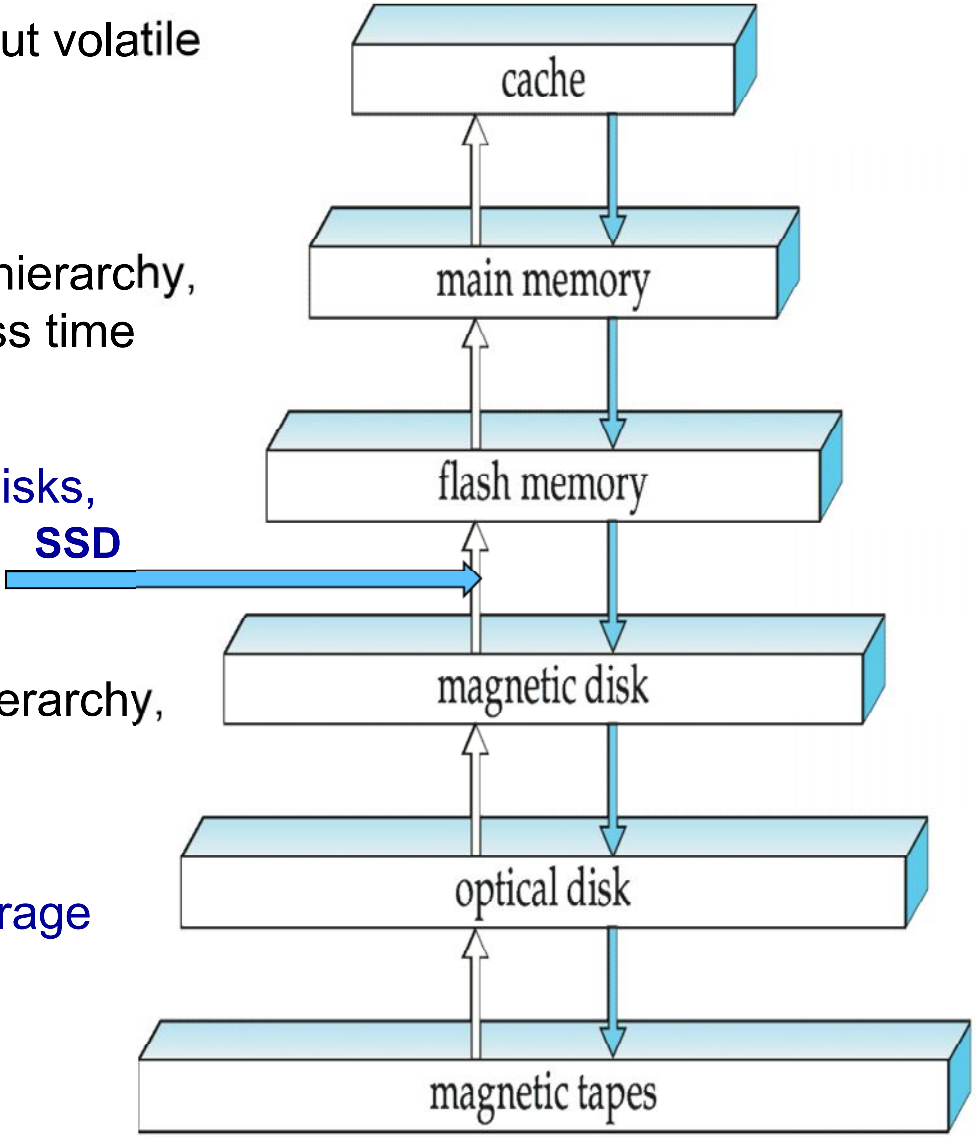
- NOR flash vs NAND flash
- NAND flash
  - used widely for storage, since it is much cheaper than NOR flash
  - requires page-at-a-time read (page: 512 bytes to 4 KB)
  - transfer rate around 20 MB/sec
  - **solid state disks**: use multiple flash storage devices to provide higher transfer rate of 100 to 200 MB/sec
  - erase is very slow (1 to 2 millisecs)
    - erase block contains multiple pages
    - **remapping** of logical page addresses to physical page addresses avoids waiting for erase
      - **translation table** tracks mapping
        - also stored in a label field of flash page
      - remapping carried out by **flash translation layer**
    - after 100,000 to 1,000,000 erases, erase block becomes unreliable and cannot be used
      - **wear leveling**

Storage Hierarchy

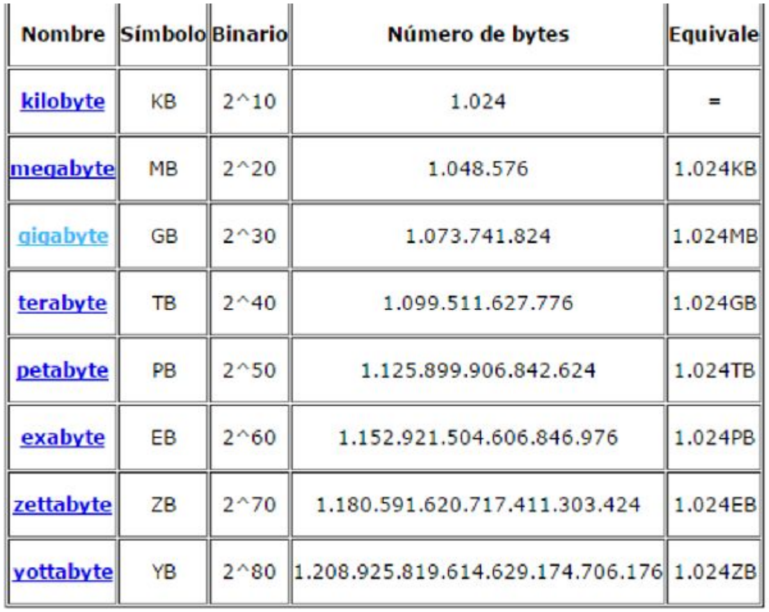
- **primary storage**: Fastest media but volatile
  - E.g. cache, main memory

- **secondary storage**: next level in hierarchy, non-volatile, moderately fast access time
  - also called **on-line storage**
  - E.g. flash memory, magnetic disks, optical disks

- **tertiary storage**: lowest level in hierarchy, non-volatile, slow access time
  - also called **off-line storage**
  - E.g. magnetic tape, optical storage



Units of measure for storage



Nombre	Símbolo	Binario	Número de bytes	Equivale
kilobyte	KB	2^10	1.024	=
megabyte	MB	2^20	1.048.576	1.024KB
gigabyte	GB	2^30	1.073.741.824	1.024MB
terabyte	TB	2^40	1.099.511.627.776	1.024GB
petabyte	PB	2^50	1.125.899.906.842.624	1.024TB
exabyte	EB	2^60	1.152.921.504.606.846.976	1.024PB
zettabyte	ZB	2^70	1.180.591.620.717.411.303.424	1.024EB
yottabyte	YB	2^80	1.208.925.819.614.629.174.706.176	1.024ZB

Unit	Value	Example
Kilobytes (KB)	1,000 bytes	a paragraph of a text document
Megabytes (MB)	1,000 Kilobytes	a small novel
Gigabytes (GB)	1,000 Megabytes	Beethoven's 5th Symphony
Terabytes (TB)	1,000 Gigabytes	All the X-rays in a large hospital
Petabytes (PB)	1,000 Terabytes	Half the contents of all US academic research libraries
Exabytes (EB)	1,000 Petabytes	About one fifth of the words people have ever spoken
Zettabytes (ZB)	1,000 Exabytes	As much information as there are grains of sand on all the world's beaches
Yottabytes (YB)	1,000 Zettabytes	As much information as there are atoms in 7,000 human bodies

Data Store Time line

Data Store Time line

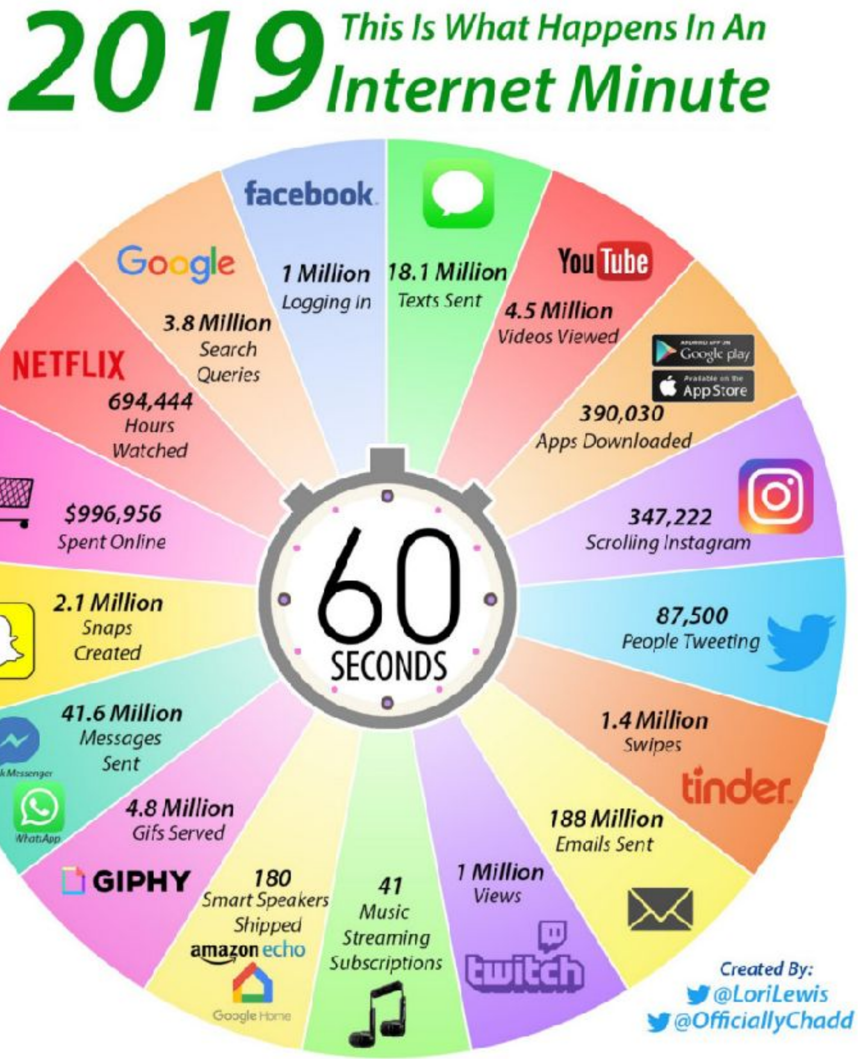
https://www.frontierinternet.com/gateway/data-storage-timeline/ (check)

https://tech.co/news/future-data-storage-technologies-2017-01

“En 2010, se estimó que almacenar un yottabyte en unidades de disco del tamaño de un terabyte requeriría un millón de centros de datos del tamaño de una manzana, tan grande como los estados de Delaware y Rhode Island combinados.

A finales de 2016, la densidad de memoria había aumentado hasta el punto en que se podía almacenar un yottabyte en tarjetas SD que ocupaban aproximadamente el doble del tamaño del Hindenburg (alrededor de 400 mil metros cúbicos)”. https://es.wikipedia.org/wiki/Yottabyte

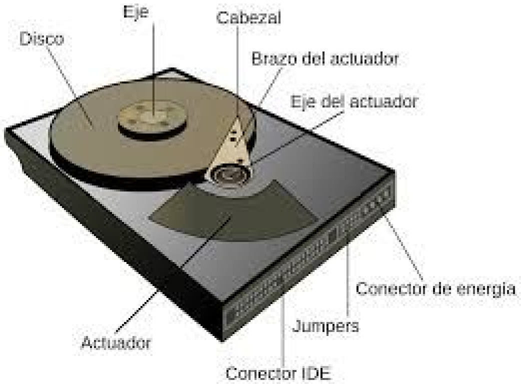
Storage needs



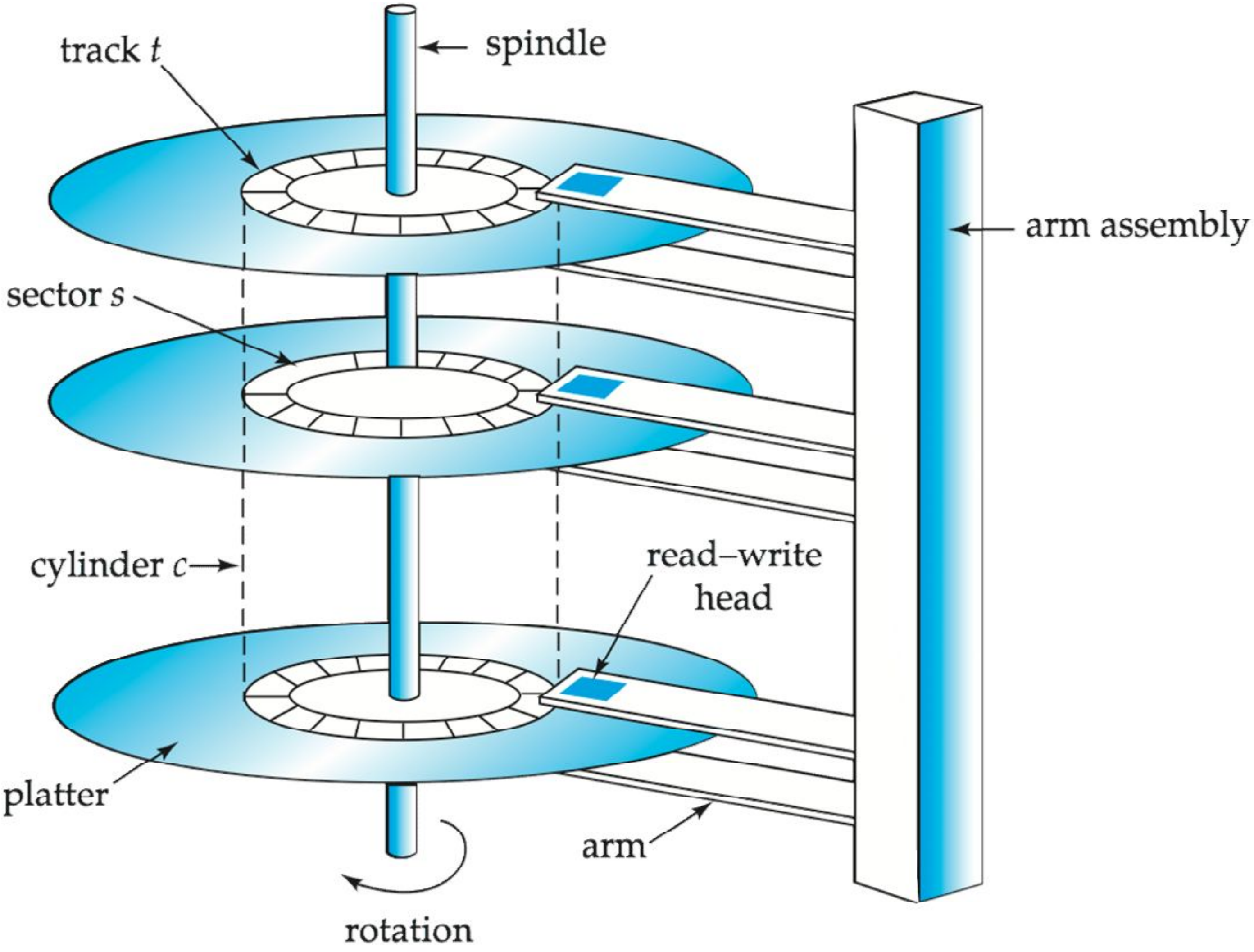
https://www.visualcapitalist.com/what-happens-in-an-internet-minute-in-2019/

Magnetic Hard Disk Mechanism

In 1956, the first magnetic hard drive was developed, a monster weighing several kilograms that was rented for about $ 3,200 a month, and which was first installed in the IBM RAMAC 305.



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)
- 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 (1 to 5 usually)
  - one head per platter, mounted on a common arm.
- **Cylinder** *i* consists of i^{th} track of all the platters

Magnetic Disks (Cont.)

- Earlier generation disks were susceptible to head-crashes
  - Surface of earlier generation disks had metal-oxide coatings which would disintegrate on head crash and damage all data on disk
  - Current generation disks are less susceptible to such disastrous failures, although individual sectors may get corrupted
- **Disk controller**
  - hardware 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
    - Detect bad sector and find a new location for the sector

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.
    - Average latency is 1/2 of the worst case latency.
    - 4 to 11 milliseconds on typical disks (5400 to 15000 r.p.m.)
- **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 (300 MB/sec)
    - Ultra 320 SCSI: 320 MB/s, SAS (3 to 6 Gb/sec)
    - Fiber Channel (FC2Gb or 4Gb): 256 to 512 MB/s

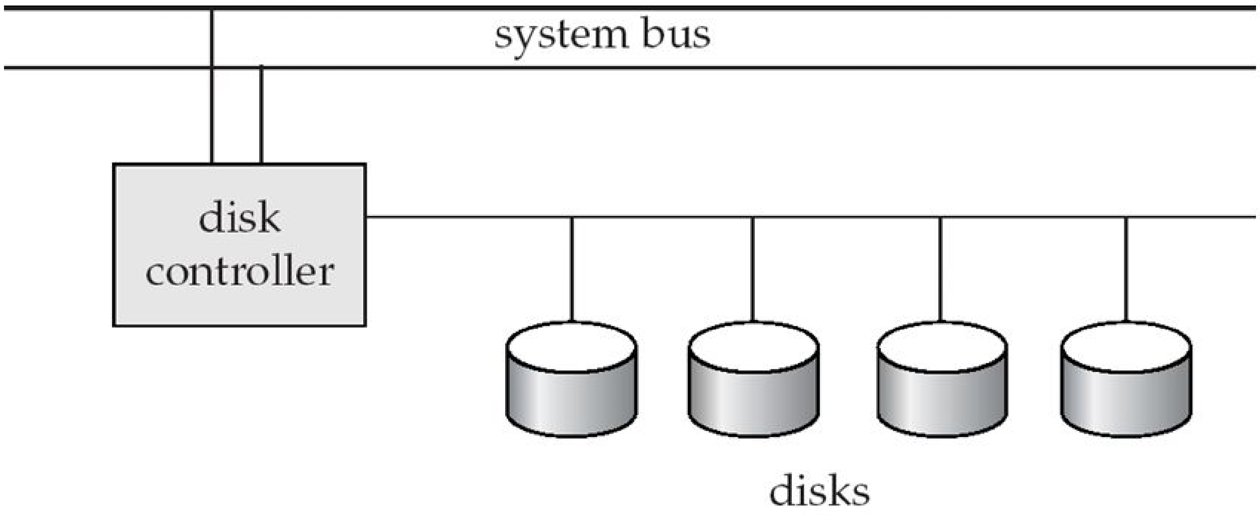
Performance Measures (Cont.)

- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.
  - Typically 3 to 5 years (1 year = 8760 hrs)
  - 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

Disk Subsystem

- Disks usually connected directly to computer system
- In **Storage Area Networks (SAN)**, a large number of disks are connected by a high-speed network to a number of servers
- In **Network Attached Storage (NAS)** networked storage provides a file system interface using networked file system protocol, instead of providing a disk system interface

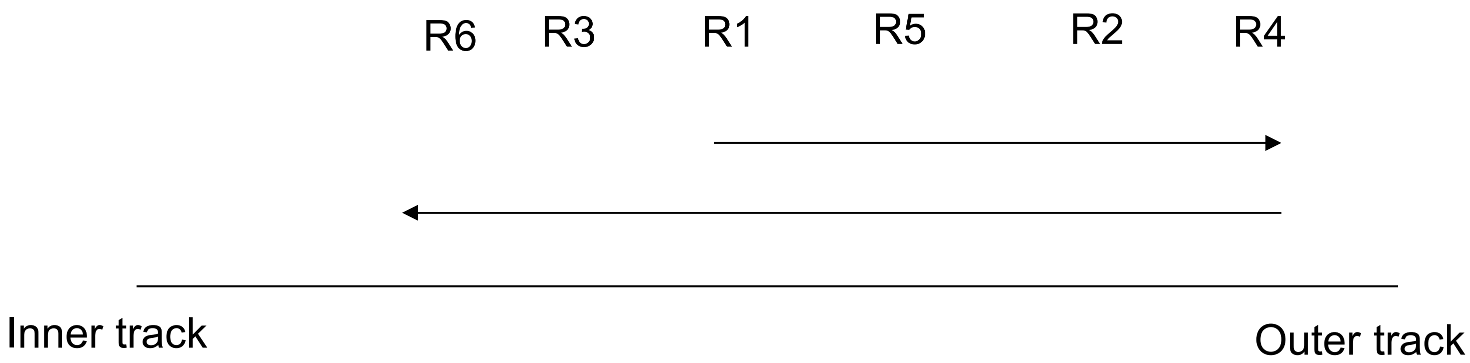
Disk Subsystem



- 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)

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** order pending accesses to tracks so that disk arm movement is minimized
  - **elevator algorithm**: 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**
  - Need to 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.
  - Sequential access to a fragmented file results in increased disk arm movement
    - Files may get fragmented over time if data is often inserted and deleted
    - New file may have scattered blocks over the disk if free blocks are scattered
  - Some systems have utilities to defragment the file system, in order to speed up file access

Optimization of Disk Block Access (Cont.)

- **Non-volatile RAM**: battery backed up RAM or flash memory
  - Speed up disk writes by writing blocks to a non-volatile RAM buffer immediately
  - 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
  - Supported in High-end disks or RAID

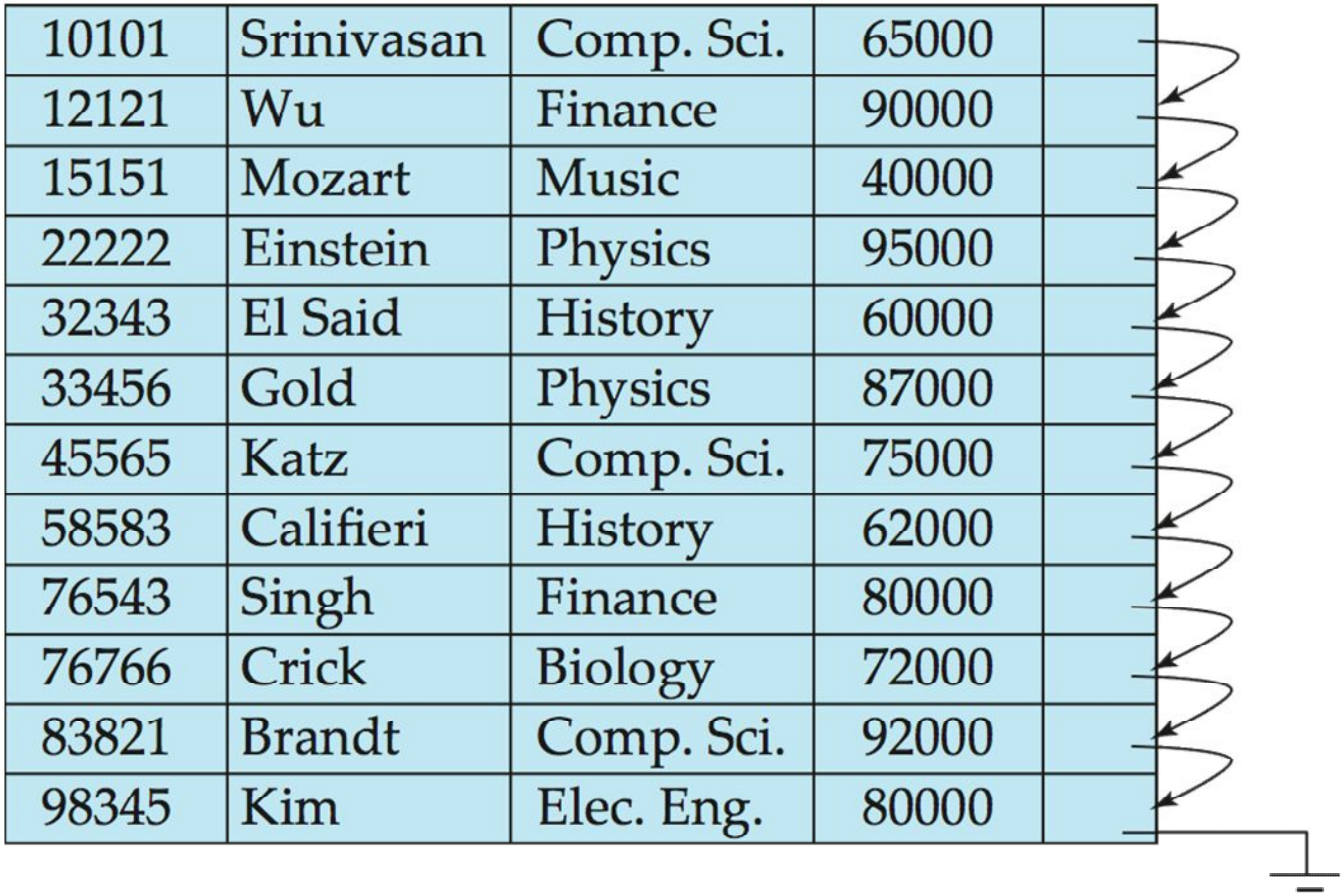
- **Log disk**: a disk devoted to writing a sequential log of block updates
  - First all updates are written to a log disk and later to the actual disk
  - Write to log disk is very fast since no seeks are required
  - No need for special hardware such as NV-RAM
  - **Journaling file systems**: file system supporting log disk
    - Typically reorder writes to disk to improve performance
    - Reordering without journaling may take risk of corruption of file system data
    - Supported in most modern file systems

Organization of Records in Files

- **Heap**
  - a record can be placed anywhere in the file where there is space
  - Sometimes called, Pile
- **Sequential**
  - store records in sequential order, based on the value of the search key of each record
- **Hashing**
  - 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**
  - Records of several different relations can be stored in the same file
  - Motivation: store related records on the same block to minimize I/O

Sequential File Organization

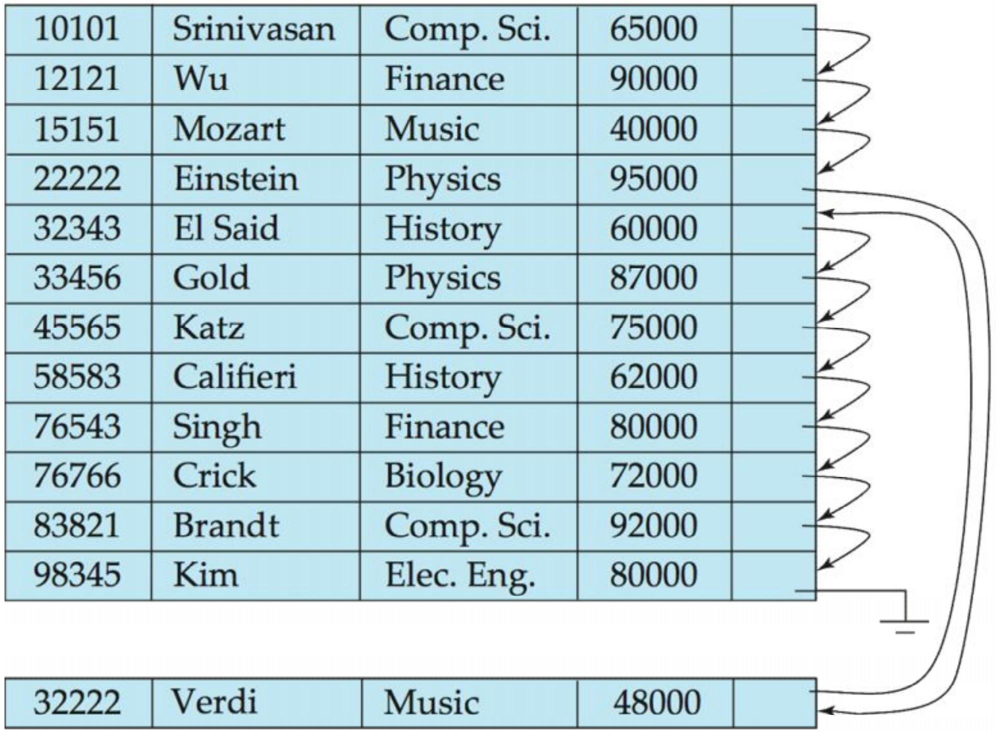
- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key



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	

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



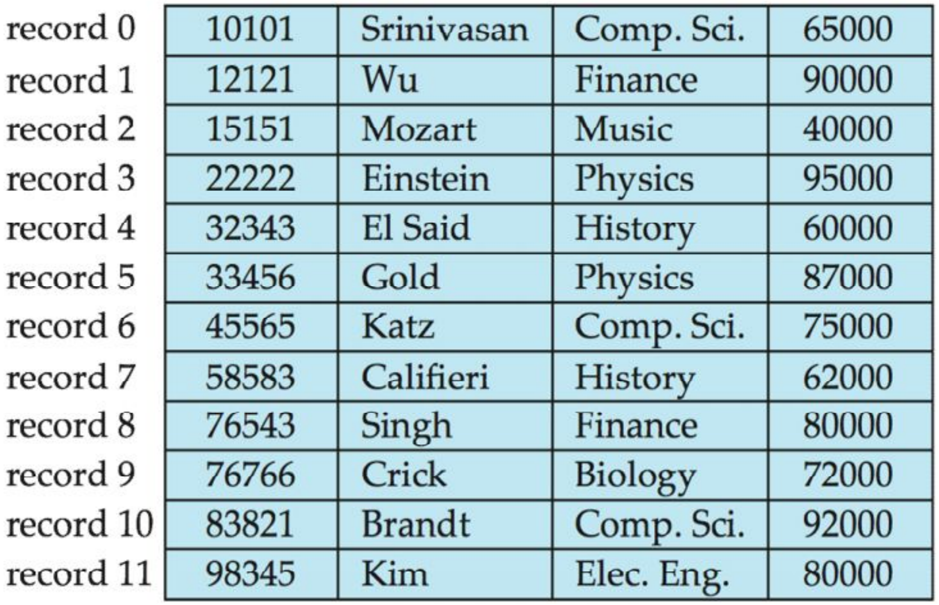
10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	

Fixed-Length Records

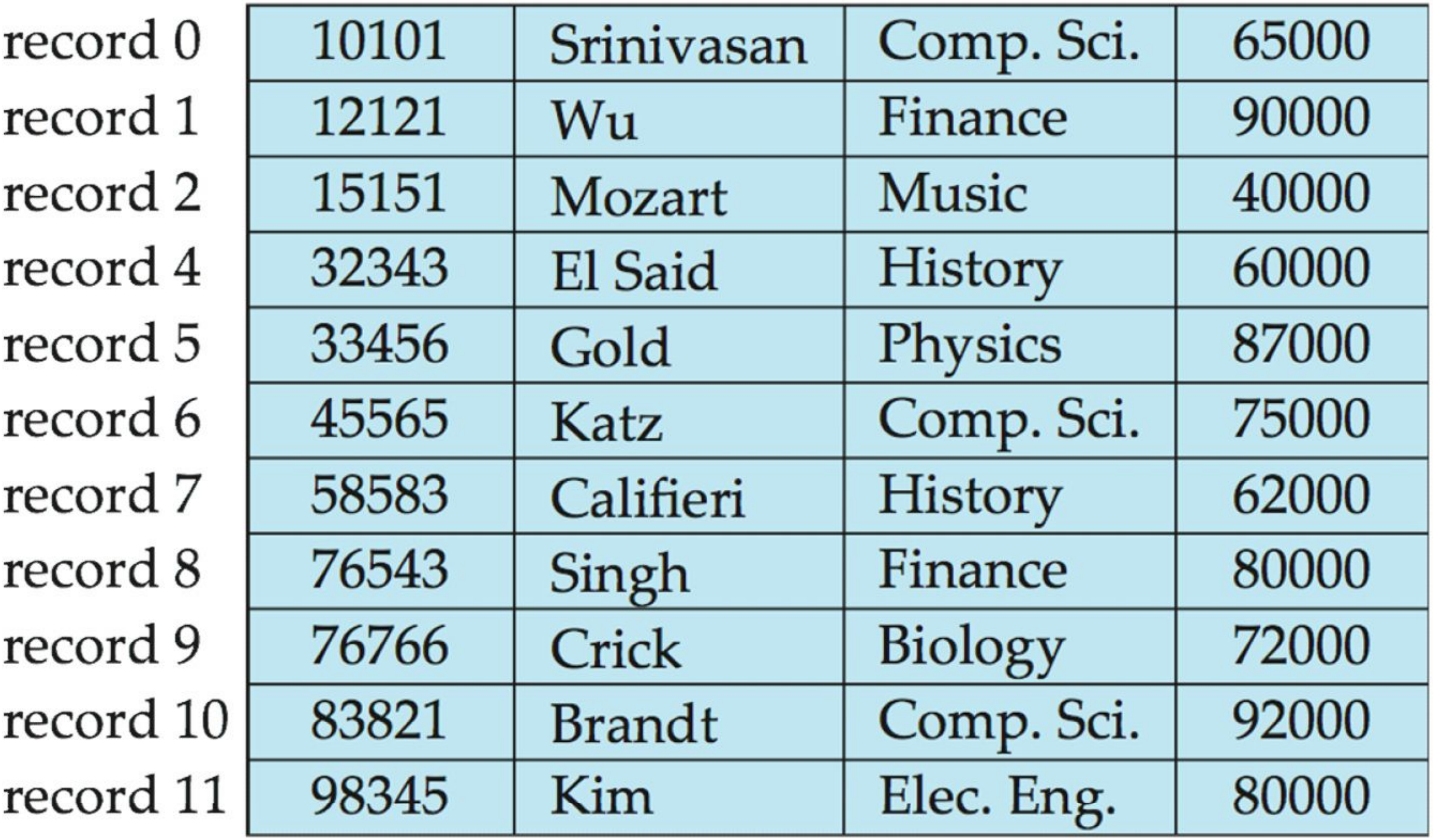
- Simple approach:
  - Store record *i* starting from byte *n* ∗ (*i* – 1), where *n* is the size of each record.
  - Record access is simple but records may cross blocks
    - Modification: do not allow records to cross block boundaries

- Deletion of record *i:* alternatives*:*
  - 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*



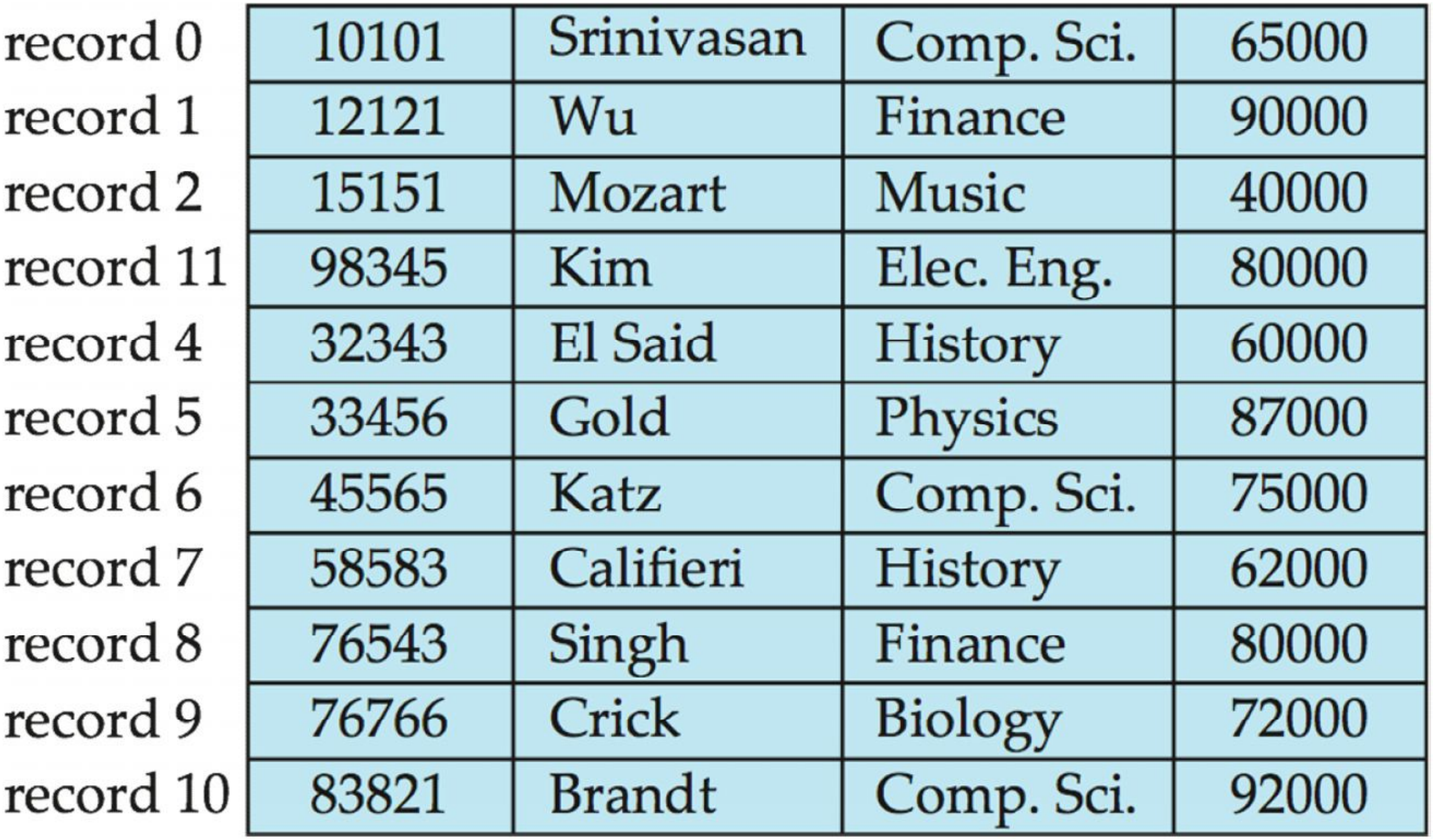
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 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

Deleting record 3 and compacting



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

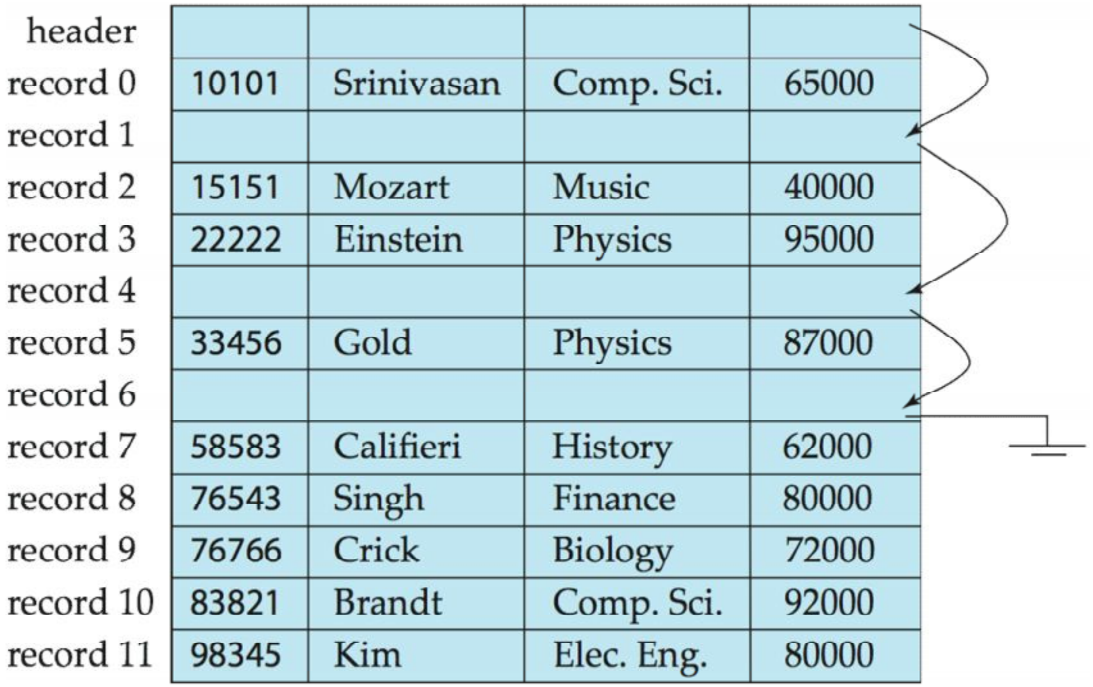
Deleting record 3 and moving last record



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

Free Lists

- Store the address of the first deleted record in the file header.
- 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.
- More space efficient representation: reuse space for normal attributes of free records to store pointers. (No pointers stored in in-use records.)

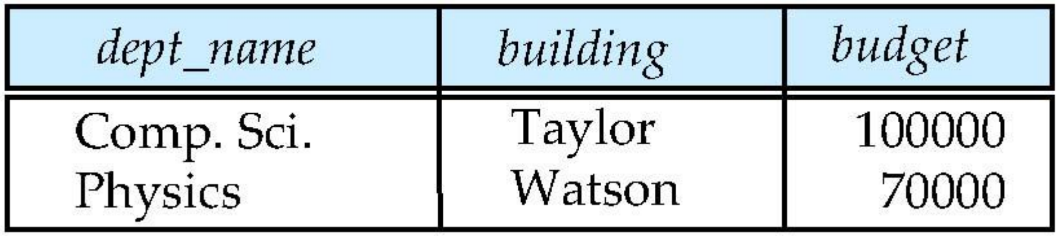


header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology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

Multitable Clustering File Organization

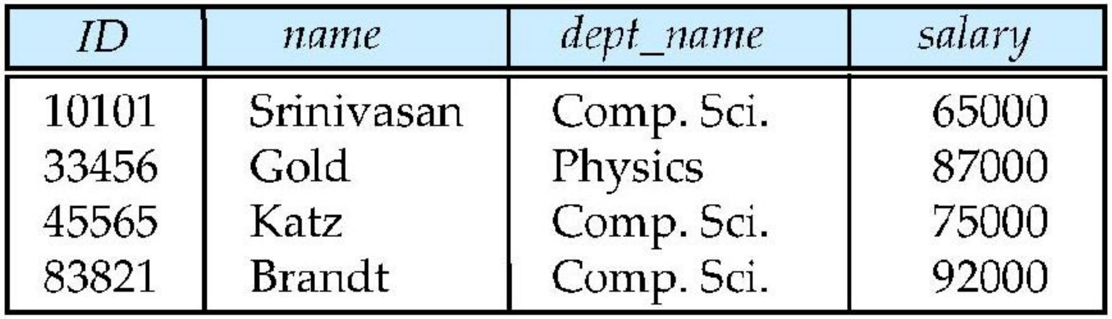
Store several relations in one file using a **multitable clustering** file organization

department



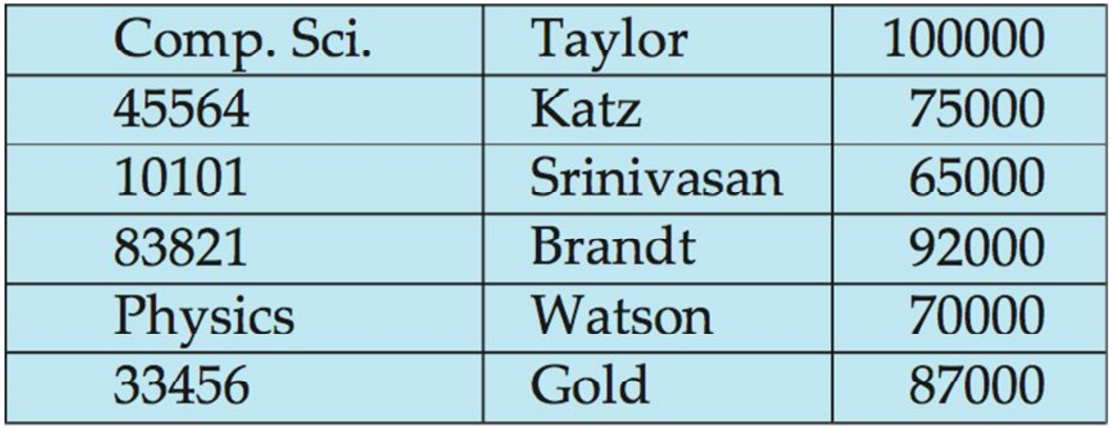
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

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

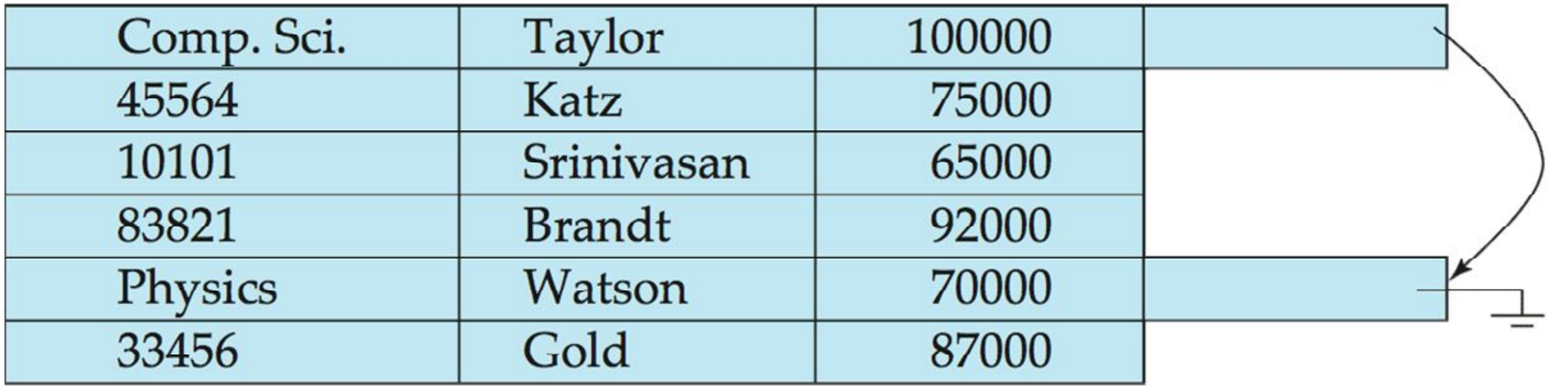
multitable clustering of *department* and *instructor*



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

Multitable Clustering File Organization (cont.)

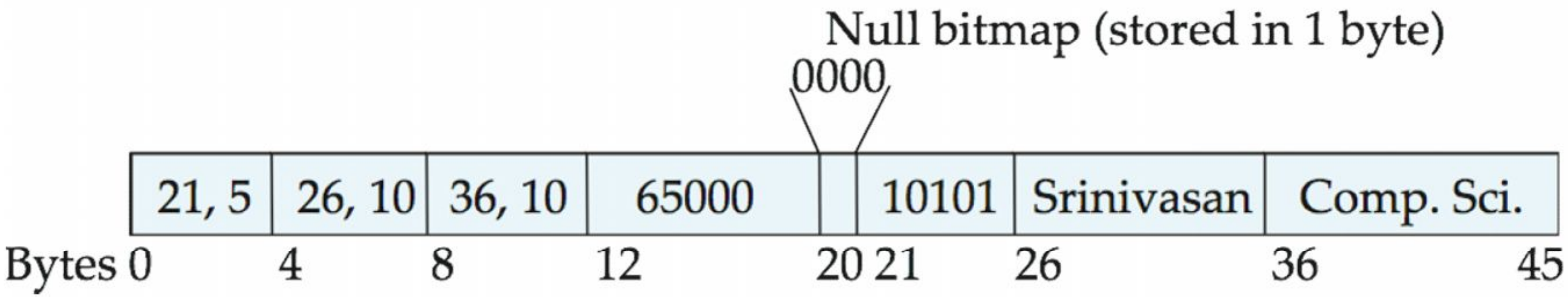
- good for queries involving *department* $\bowtie$ *instructor*, and for queries involving one single department and its instructors
- bad for queries involving only *department*
- results in variable size records
- Can add pointer chains to link records of a particular relation



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

Variable-Length Records

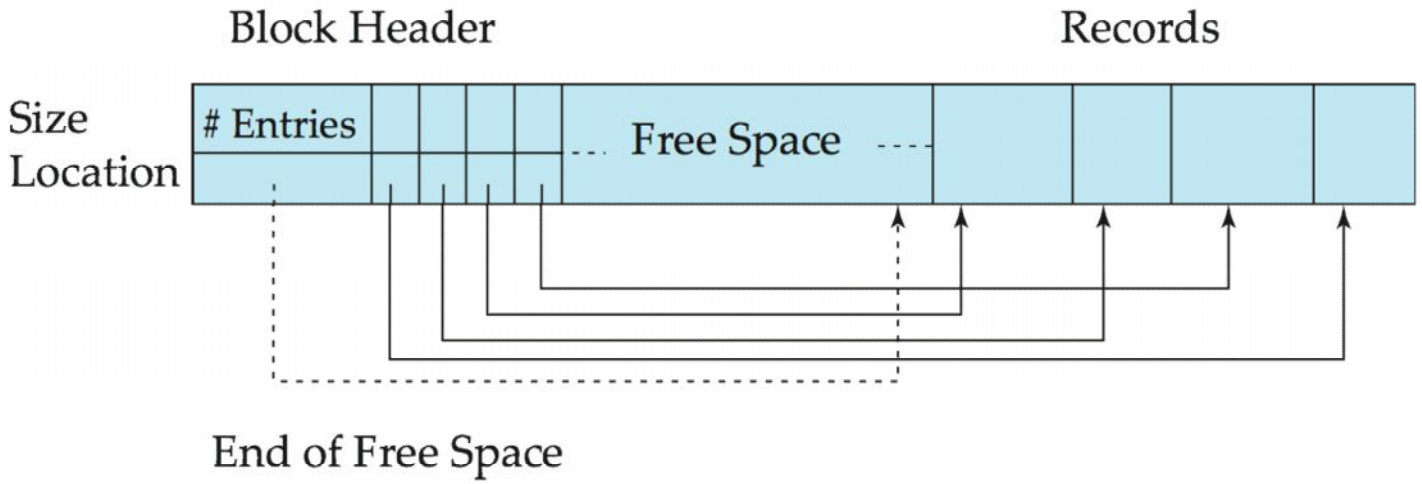
- Variable-length records arise in database systems in several ways:
  - Storage of multiple record types in a file.
  - Record types that allow variable lengths for one or more fields such as strings (**varchar**)
  - Record types that allow repeating fields (used in some older data models).
- Attributes 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 (Cont.)

- Byte string representation
  - Attach an *end-of-record* (⊥) control character to the end of each record
  - Difficulty with deletion / growth
- Slotted page method
  - The most widely used method
- Fixed length representation
  - Reserved space method
  - Pointer method
- 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.

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.

No se puede mostrar la imagen.

End of Chapter

Database System Concepts, 6th Ed.