# Database System Concepts Notes

## Chapter 10

### Physical Storage

To understand how databases store data, it is necessary to understand the physical constraints of our storage. The physical characteristics of storage devices play a major role in the way data are stored. For example: Disk access takes tens of milliseconds, whereas memory access takes a tenth of a microsecond.

Storage media are classified by the speed with which data can be accessed, the cost per unit of data to buy the medium and the reliability of the medium. Typically, we can divide media into the following:

1. **Cache:** Fastest, most costly, volatile, usually managed by computer hardware. We will not concern ourselves with managing this in our database however some implementors do.
2. **Main Memory:** Storage medium used for data that are available to be operated on. Can be up to several gigabytes but entire database is not stored here. Volatile.
3. **Flash Memory:** Non-volatile aka data is retained upon power loss. Mainly two types -> NAND and NOR Flash. Of these, NAND Flash is better and is widely used. Cheaper than Main Memory. Solid-State Drives, USBs etcetera use Flash Memory.
4. **Magnetic-disk storage:** Primary medium for long term storage in 2009. System moves data from here to main memory for operation. Non-volatile but sometimes fails and destroys data, but such failures are rare.
5. **Optical Storage:** Old stuff; Compact Disks (CDs), Digital Video Disks (DVDs), Blu-ray DVDs. Optical Disks are also used in read-only cases (CD-ROM, DVD-ROM). There are also record-once variants (CD-R, DVD-R) These are called write-once, read-many (WORM) disks. Can also have write-many variants (CD-RW, DVD-RW)
6. **Tape Storage:** Archaic stuff, long tapes that are read sequentially and hold enormous amounts of data. Archival in nature.

And therefore, we have a hierarchy of storages, based on the cost-capacity-speed metric, with cache at top for highest speed, cost and lowest capacity and moving all the way down to tape storage.

Cache and Main Memory are called primary storage. Magnetic Disks are referred to as secondary or online storage and the lowest level is referred to as tertiary storage or offline storage.

A second sorting is enforced on the basis of volatility with cache and main memory being the only volatile ones.

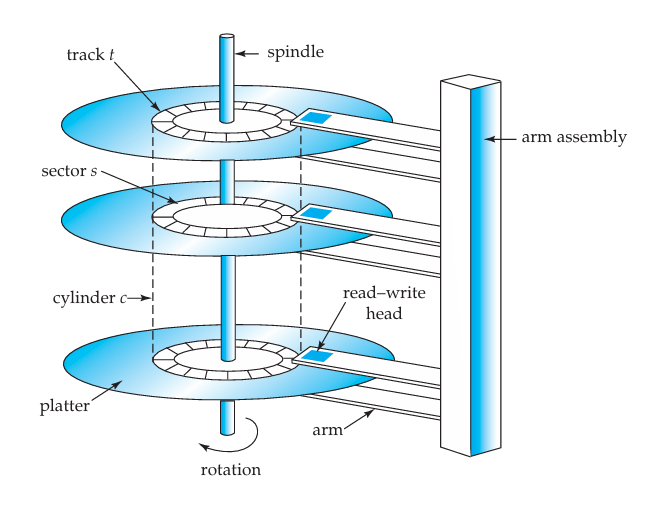
### Magnetic Disk and Flash Storage

#### Physical Characteristics of Disks:

A physical disk has platters. Each platter has a flat, circular shape. Its 2 surfaces are covered with magnetic material and information is recorded on the surfaces.

A disk surface is logically divided into tracks, which are subdivided into sectors.

A sector is the smallest unit of information that can be read from or written to the disk. Sector sizes are typically 512 bytes. There are about 50,000 to 100,000 tracks per platter and 1 to 5 platters per disk. The inner tracks (closer to the spindle) are of smaller length and the outer tracks contain more sectors than the inner tracks; typical numbers are around 500 to 1000 sectors per track in the inner tracks, and around 1000 to 2000sectors per track in the outer tracks.



The read–write head stores information on a sector magnetically as reversals of the direction of magnetization of the magnetic material.

A disk controller interfaces between the computer system and the actual hardware of the disk drive. It accepts high-level commands to read or write a sector, and initiates actions, such as moving the disk arm to the right track and actually reading or writing the data.

Another interesting task that disk controllers perform is remapping of bad sectors. If the controller detects that a sector is damaged when the disk is initially formatted, or when an attempt is made to write the sector, it can logically map the sector to a different physical location

#### Performance Measures of Disks

Access time is the time from when a read or write request is issued to when data transfer begins. The time for repositioning the arm is called the seek time, and it increases with the distance that the arm must move.

Once the head has reached the desired track, the time spent waiting for the sector to be accessed to appear under the head is called the rotational latency time. The access time is then the sum of the seek time and the latency, and ranges from 8 to 20 milliseconds.

The mean time to failure of a disk (or of any other system) is the amount of time that, on average, we can expect the system to run continuously without any failure. According to vendors’ claims, the mean time to failure of disks today ranges from 500,000 to 1,200,000 hours— about 57 to 136 years.

#### Optimization of Disk-Block Access

A block is a logical unit consisting of a fixed number of contiguous sectors. Block sizes range from 512 bytes to several kilobytes.

A sequence of requests for blocks from disk may be classified as a sequential access pattern or a random-access pattern. In a sequential access pattern, successive requests are for successive block numbers, which are on the same track, or on adjacent tracks.

In a random-access pattern, successive requests are for blocks that are randomly located on disk. Each such request would require a seek.

A number of techniques have been developed for improving the speed of access to blocks:

1. **Buffering**: Blocks read from disk are stored temporarily in an in-memory buffer, to satisfy future requests.
2. **Read-ahead**: When a block is accessed, consecutive blocks from the same track are read into an in-memory buffer; minimizes time wasted in disk seeks and rotational latency per block read.
3. **Scheduling**: Disk Arm scheduling algorithms (like the elevator algorithm) to decide what would be the best order for data to be accessed, not necessarily maintain the order in which data were requested.
4. **File Organisation**: Organising the data in a manner such that latency decreases. For example, storing a larger file in continuous blocks.
5. **Non-volatile write buffers:** Use NVRAM to speed up disk writes. For write-heavy systems, write to disk from NVRAM buffer when the buffer is full or the disk is idle. Allows recovery.
6. **Log Disks:** A disk devoted to writing a sequential log of operations. As before, the data have to be written to their actual location on disk as well, but the log disk can do the write later, without the database system having to wait for the write to complete.

#### Flash Memory

There are two types of flash memory, NOR flash and NAND flash. NOR flash allows random access to individual words of memory, and has read time comparable to main memory.

However, unlike NOR flash, reading from NAND flash requires an entire page of data, typically consisting of between 512 and 4096 bytes, to be fetched from NAND flash into main memory. Pages in a NAND flash are thus similar to sectors in a magnetic disk.

But NAND flash is significantly cheaper than NOR flash, and has much higher storage capacity, and is by far the more widely used.

A write to a page of flash memory typically takes a few microseconds. However, once written, a page of flash memory cannot be directly overwritten. Instead, it has to be erased and rewritten subsequently.

### Redundant Array of Independent Disks (RAID)

The data-storage requirements of some applications have been growing so fast that a large number of disks are needed to store their data, even though disk-drive capacities have been growing very fast.

Disk-organization techniques, collectively called redundant arrays of independent disks (RAID), have been proposed to achieve improved performance and reliability.

#### Improvement of Reliability via Redundancy

The chance that at least one disk out of a set of N disks will fail is much higher than the chance that a specific single disk will fail.

The solution to the problem of reliability is to introduce redundancy; that is, we store extra information that is not needed normally, but that can be used in the event of failure of a disk to rebuild the lost information.

The simplest (but most expensive) approach to introducing redundancy is to duplicate every disk. This technique is called mirroring (or, sometimes, shadowing)

#### Improvement in Performance via Parallelism

With disk mirroring, the rate at which read requests can be handled is doubled, since read requests can be sent to either disk.

The transfer rate of each read is the same as in a single-disk system, but the number of reads per unit time has doubled.

With multiple disks, we can improve the transfer rate as well (or instead) by striping data across multiple disks.

**Bit-level striping**: if we have an array of eight disks, we write bit i of each byte to disk i. The array of eight disks can be treated as a single disk with sectors that are eight times the normal size, and, more importantly, that has eight times the transfer rate. In such an organization, every disk participates in every access.

**Block-level striping** stripes blocks across multiple disks. It treats the array of disks as a single large disk, and it gives blocks logical numbers; we assume the block numbers start from 0. With an array of n disks, block-level striping assigns logical block i of the disk array to disk (i mod n) + 1

### RAID Levels

1. **RAID level 0** refers to disk arrays with striping at the level of blocks, but without any redundancy (such as mirroring or parity bits)
2. **RAID level 1** refers to disk mirroring with block striping.
3. **RAID level 2 or memory-style error-correcting-code (ECC) organization** usesparitybits.

Each byte in a memory system may have a parity bit associated with it that records whether the numbers of bits in the byte that are set to 1 is even (parity = 0) or odd (parity = 1). If one of the bits in the byte gets damaged (either a 1 becomes a 0, or a 0 becomes a 1), the parity of the byte changes and thus will not match the stored parity.

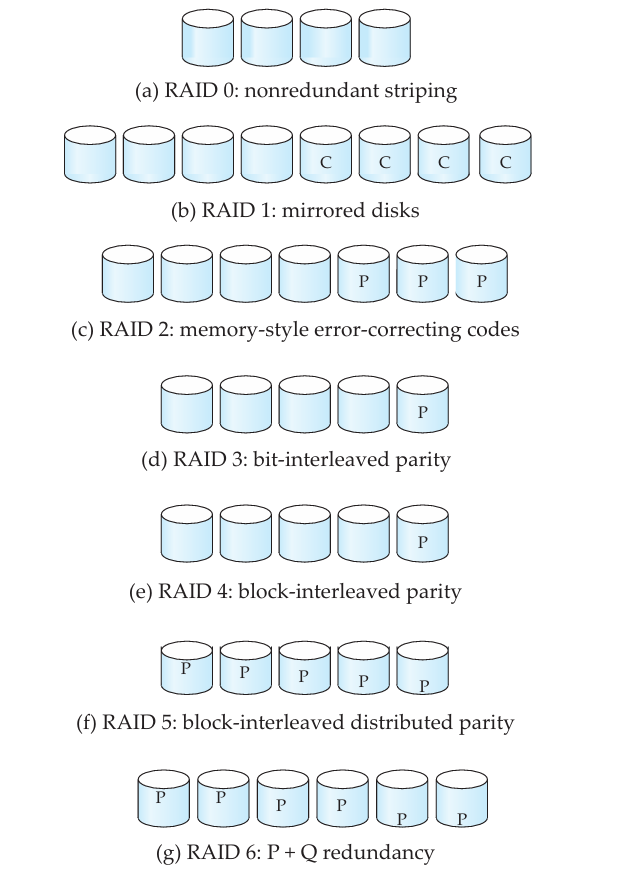
1. **RAID level 3** or **bit-interleaved parity organization** exploits the fact that disk controllers, unlike memory systems, can detect whether a sector has been read correctly, so a single parity bit can be used for error correction, as well as for detection.

If one of the sectors gets damaged, the system knows exactly which sector it is, and, for each bit in the sector, the system can figure out whether it is a 1 or a 0 by computing the parity of the corresponding bits from sectors in the other disks. If the parity of the remaining bits is equal to the stored parity, the missing bit is 0; otherwise, it is 1.

1. **RAID level 4** or **block-interleaved parity organization**, uses block-level striping, like RAID 0, and in addition keeps a parity block on a separate disk for corresponding blocks from N other disks.

A block read accesses only one disk, allowing other requests to be processed by the other disks. Thus, the data-transfer rate for each access is slower, but multiple read accesses can proceed in parallel, leading to a higher overall I/O rate.

1. **RAID level 5** or **block-interleaved distributed parity**, improves on level 4 by partitioning data and parity among all N+1 disks, instead of storing data in N disks and parity in one disk.
2. **RAID level 6, the P + Q redundancy scheme**, is much like RAID level 5, but stores extra redundant information to guard against multiple disk failures. Instead of using parity, level 6 uses error-correcting codes such as the Reed Solomon codes (see the bibliographical notes). In the scheme in the Figure, 2 bits of redundant data are stored for every 4 bits of data—unlike 1 parity bit in level 5—and the system can tolerate two disk failures



### Choice of RAID Level

The factors to be taken into account in choosing a RAID level are:

1. Monetary cost of extra disk-storage requirements.
2. Performance requirements in terms of number of I/O operations.
3. Performance when a disk has failed.
4. Performance during rebuild (that is, while the data in a failed disk are being rebuilt on a new disk)

Rebuilding is easiest for RAID level 1, since data can be copied from another disk; for the other levels, we need to access all the other disks in the array to rebuild data of a failed disk.

The choice of a RAID level is also highly dependent on the application of the user. If continuous availability of data is required, then the rebuild performance becomes a key factor.

RAID level 0 is used in high-performance applications where data safety is not critical.

RAID levels 2 and 4 are subsumed by RAID levels 3 and 5.

Bit striping (level 3) is inferior to block striping (level 5), since block striping gives as good data-transfer rates for large transfers, while using fewer disks for small transfers. For small transfers, the disk access time dominates anyway, so the benefit of parallel reads diminishes.

RAID level 1 is popular for applications such as storage of log files in a database system, since it offers the best write performance. RAID level 5 has a lower storage overhead than level 1, but has a higher time overhead for writes. For applications where data are read frequently, and written rarely, level 5 is the preferred choice.

RAID level 1 is the RAID level of choice for many applications with moderate storage requirements and high I/O requirements.

There is also an array of hardware issues that need to be addressed by a RAID system.

Loss of data that were successfully written earlier is sometimes referred to as a latent failure, or as bit rot.

Good RAID controllers perform scrubbing; that is, during periods when disks are idle, every sector of every disk is read, and if any sector is found to be unreadable, the data are recovered from the remaining disks in the RAID organization.

Some hardware RAID implementations permit hot swapping; that is, faulty disks can be removed and replaced by new ones without turning power off.

### Tertiary Storage

The two most common types of tertiary storage are Optical Disks and Magnetic Tapes

#### Optical Disks

Compact Disks (CDs) have a storage capacity of 640 to 700 megabytes, and they are cheap to mass-produce.

Digital video disks (DVDs) have now replaced compact disks in applications that require larger amounts of data. DVD-5 format can store 4.7 gigabytes of data (in one recording layer), while disks in the DVD-9 format can store 8.5 gigabytes of data.

The Blu-ray DVD format has a significantly higher capacity of 27 to 54 gigabytes per disk.

The transfer rate of optical drives is characterized as n×, which means the drive supports transfers at n times the standard rate; rates of around 50× for CD and 16× for DVD are now common.

Jukeboxes are devices that store a large number of optical disks (up to several hundred) and load them automatically on demand to one of a small number of drives (usually 1 to 10)

#### Magnetic Tapes

Magnetic tapes are limited to sequential access.

Tapes are 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 capacities range from a few gigabytes with the Digital Audio Tape (DAT) format, 10 to 40 gigabytes with the Digital Linear Tape (DLT) format, 100 gigabytes and higher with the Ultrium format, to 330 gigabytes with Ampex helical scan tape formats

### File Organisation

A database is mapped into a number of different files that are maintained by the underlying operating system. A file is organized logically as a sequence of records. These records are mapped onto disk blocks.

Each file is also logically partitioned into fixed-length storage units called blocks, which are the units of both storage allocation and data transfer.

We will operate under the assumptions that:

*No record is larger than a block* and *each record is entirely contained in a single block*

Tuples of distinct relations are generally of different sizes. One approach to mapping the database to files is to use several files, and to store records of only one fixed length in any given file. An alternative is to structure our files so that we can accommodate multiple lengths for records.

#### Fixed Length Records

Let us assume that a record is 53 bytes long. A simple approach to store it is to use the first 53 bytes for the first record, the next 53 bytes for the second record, and so on.

However, there are two problems with this simple approach:

1. Unless the block size happens to be a multiple of 53 (which is unlikely), some records will cross block boundaries.
2. It is difficult to delete a record from this structure. The space occupied by the record to be deleted must be filled with some other record of the file, or we must have a way of marking deleted records so that they can be ignored.

For the first problem, the first problem, we allocate only as many records to a block as would fit entirely in the block. Any remaining bytes would be unused.

For the second one, we could move the record that came after it into the space formerly occupied by the deleted record, and so on, until every record following the deleted record has been moved ahead.

It might be easier simply to move the final record of the file into the space occupied by the deleted record. Since insertions tend to be more frequent than deletions, it is acceptable to leave open the space.

To solve this, at the beginning of the file, we allocate a certain number of bytes as a file header. For now, all we need to store there is the address of the first record whose contents are deleted. We use this first record to store the address of the second available record, and so on. The deleted records form a linked list, which is often referred to as a free list.

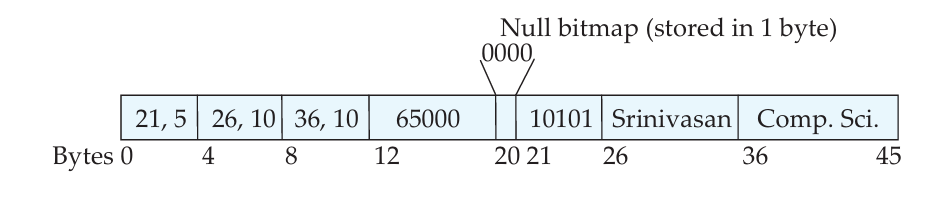
#### Variable Length Records

Variable-length records arise in database systems in several ways:

1. Storage of multiple record types in a file.
2. Record types that allow variable lengths for one or more fields.
3. Record types that allow repeating fields, such as arrays or multisets

The representation of a record with variable-length attributes typically has two parts: an initial part with fixed length attributes, followed by data for variable length attributes.

Thus, the initial part of the record stores a fixed size of information about each attribute, whether it is fixed-length or variable-length.



The figure also illustrates the use of a null bitmap, which indicates which attributes of the record have a null value.

We next address the problem of storing variable-length records in a block. The slotted-page structure is commonly used for organizing records within a block.

There is a header at the beginning of each block, containing the following information:

1. The number of record entries in the header.
2. The end of free space in the block.
3. An array whose entries contain the location and size of each record

records are allocated contiguously in the block, starting from the end of the block. The free space in the block is contiguous, between the final entry in the header array, and the first record.

If a record is deleted, the space that it occupies is freed, and its entry is set to deleted (its size is set to −1, for example).

#### Large Files

Databases often store data that can be much larger than a disk block. For instance, an image or an audio recording.

Most relational databases restrict the size of a record to be no larger than the size of a block, to simplify buffer management and free-space management.

Large objects are often stored in a special file (or collection of files) instead of being stored with the other (short) attributes of records in which they occur. A (logical) pointer to the object is then stored in the record containing the large object