Storage

R&G Chapter 9

(slides adapted from content by J.Gehrke, J.Shanmugasundaram, and/or C.Koch)

Review

Operation	**Sym**	**Meaning**
Selection	σ	Select a subset of the input rows
Projection	π	Delete unwanted columns
Cross-product	x	Combine two relations
Set-difference	-	Tuples in Rel 1, but not Rel 2
Union	U	Tuples either in Rel 1 or in Rel 2

Also: Intersection, **Join**, Division, Renaming
(Not essential, but very useful)

Review

Find the Last Names of all captains of a ship located in Federation Territories

Affiliation

Location,	Affiliation
[Earth,	Federation]
[Risa,	Federation]
[Bajor,	Bajor]

Review

Find the Last Names of all captains of a ship located in Federation Territories

Affiliation

Location,	Affiliation
[Earth,	Federation]
[Risa,	Federation]
[Bajor,	Bajor]

$$\pi_{\text{LastName}}(\sigma_{\text{Affiliation= 'Federation'}}(\text{Loc}) \bowtie \text{Affil} \bowtie \text{Cap})$$

Review

Find the Last Names of all captains of a ship located in Federation Territories

Affiliation

Location,	Affiliation
[Earth,	Federation]
[Risa,	Federation]
[Bajor,	Bajor]

$\pi_{\text{LastName}}(\sigma_{\text{Affiliation= 'Federation'}}(\text{Loc}) \bowtie \text{Affil} \bowtie \text{Cap})$

But we can do this more efficiently:

Review

Find the Last Names of all captains of a ship located in Federation Territories

Affiliation

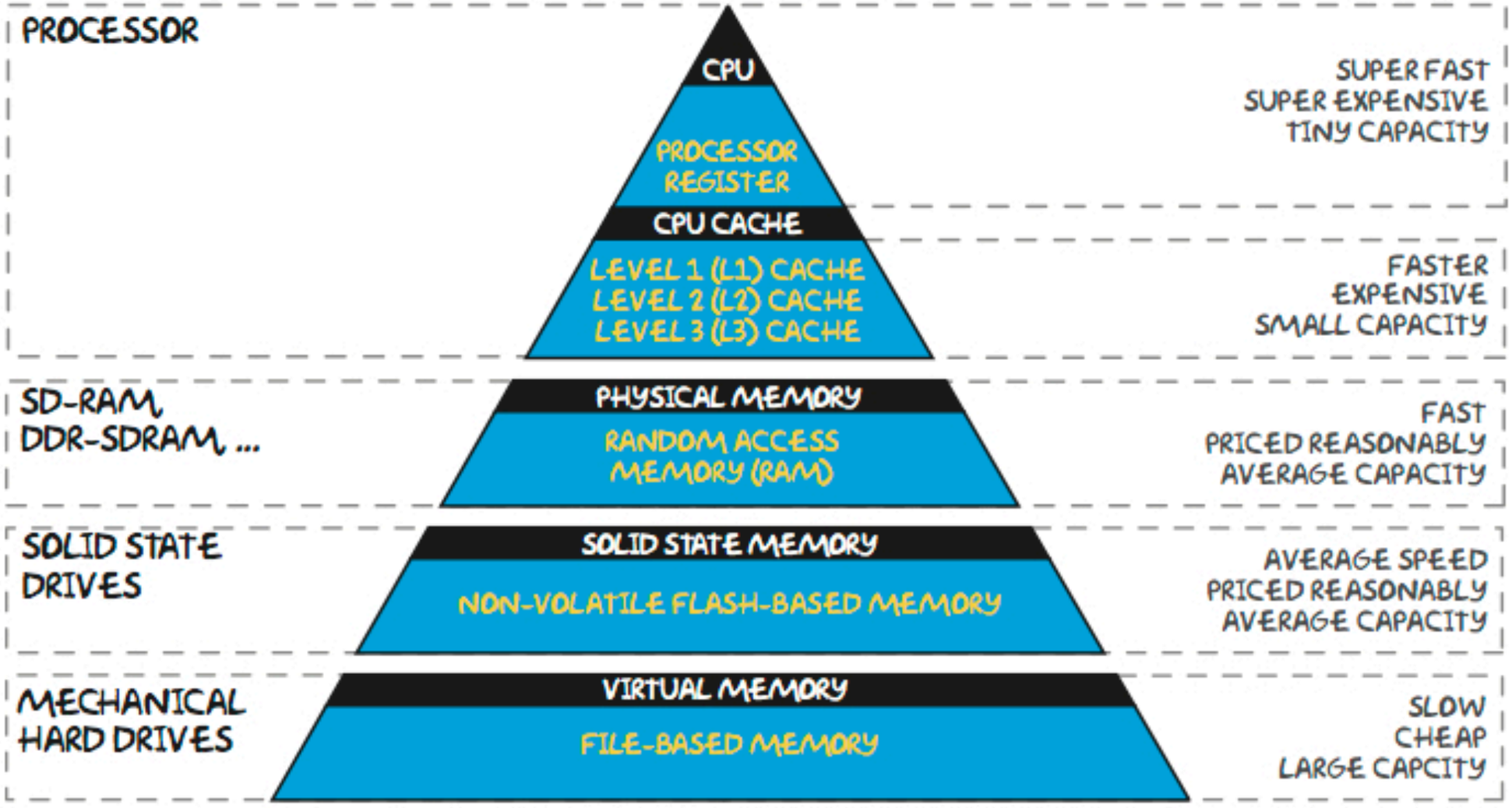
Location,	Affiliation
[Earth,	Federation]
[Risa,	Federation]
[Bajor,	Bajor]

$$\pi_{\text{LastName}}(\pi_{\text{Ship}}(\pi_{\text{Location}}(\sigma_{\text{Affiliation= 'Federation'}}(\text{Loc}))) \bowtie \text{Affil}) \bowtie \text{Cap})$$

A query optimizer can find this, given the first solution

The Memory Hierarchy

Fast (but small)



Big (but slow)

image credit: teachbook

The Memory Hierarchy

Fast (but small)

RAM

Flash Memory

Hard Disks

Tape Drives

Big (but slow)

The Memory Hierarchy

Fast (but small)

RAM

What are some characteristics of each form of data storage?

Flash Memory

Hard Disks

Tape Drives

Big (but slow)

The Memory Hierarchy

Fast (but small)

RAM

What are some characteristics of each form of data storage?

Flash Memory

Hard Disks

Tape Drives

Which layers are needed by a DBMS?

Big (but slow)

Storage

- How do we…
  - …go deeper into the memory hierarchy?
  - …use the right data access pattern for the storage medium we're using?
  - …organize data to minimize access costs?
- These ideas are incorporated into the **Buffer Manager** of a DBMS.

Data Organization

Heap	**Sorted**	**Indexed**
Records stored in any order	Records stored in sorted order	Secondary file used to organize data records

Data Organization

Heap	**Sorted**	**Indexed**
Records stored in any order	Records stored in sorted order	Secondary file used to organize data records

What are the benefits/drawbacks of each method?

Data Organization

Heap	**Sorted**	**Indexed**
Records stored in any order	Records stored in sorted order	Secondary file used to organize data records

What are the benefits/drawbacks of each method?

Does it matter what medium the data is being stored on?

Data Organization

Heap	**Sorted**	**Indexed**
Records stored in any order	Records stored in sorted order	Secondary file used to organize data records

What are the benefits/drawbacks of each method?

Does it matter what medium the data is being stored on?

When do we use each method?

The IO Problem

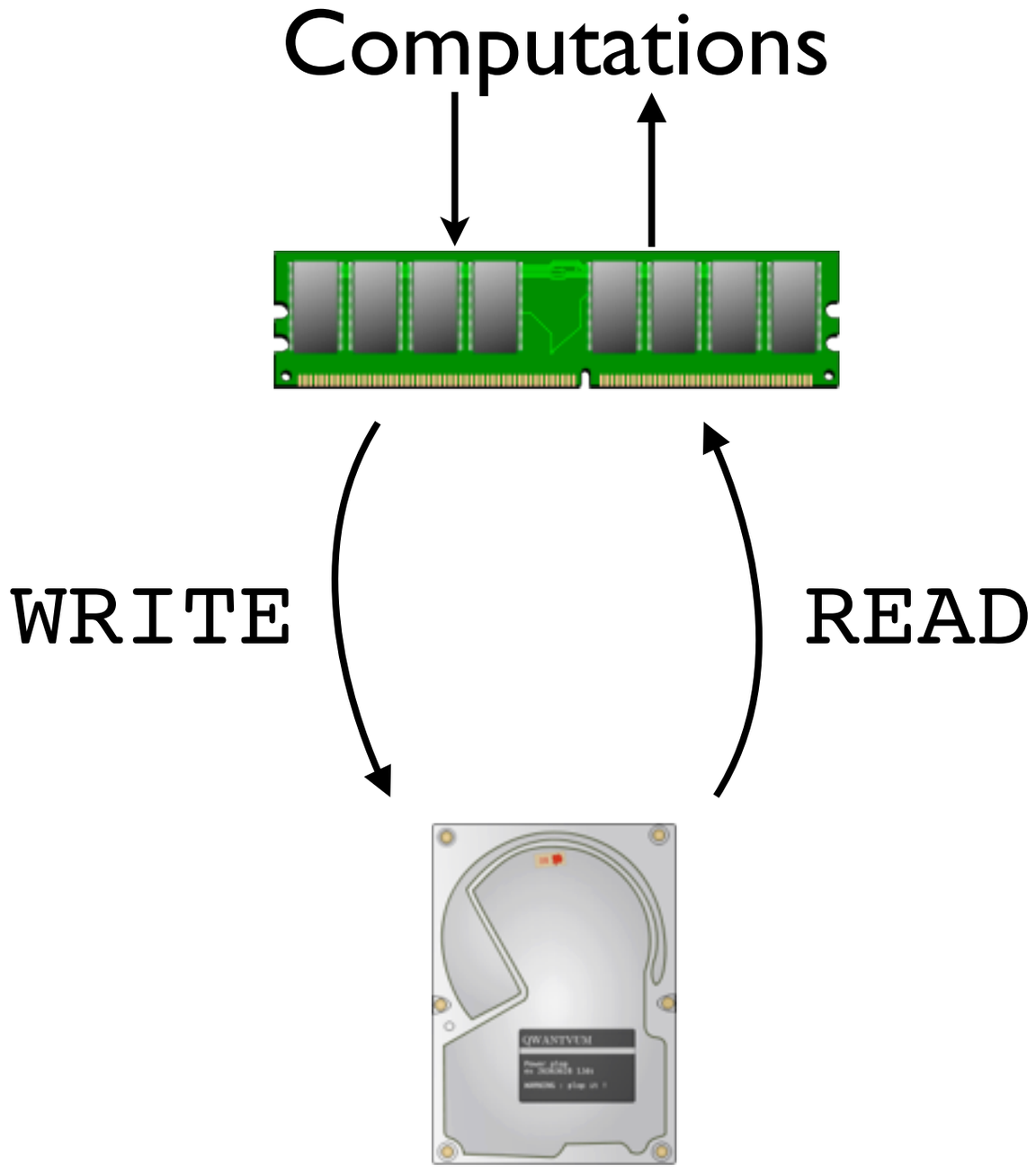


image credit: openclipart.org

The IO Problem

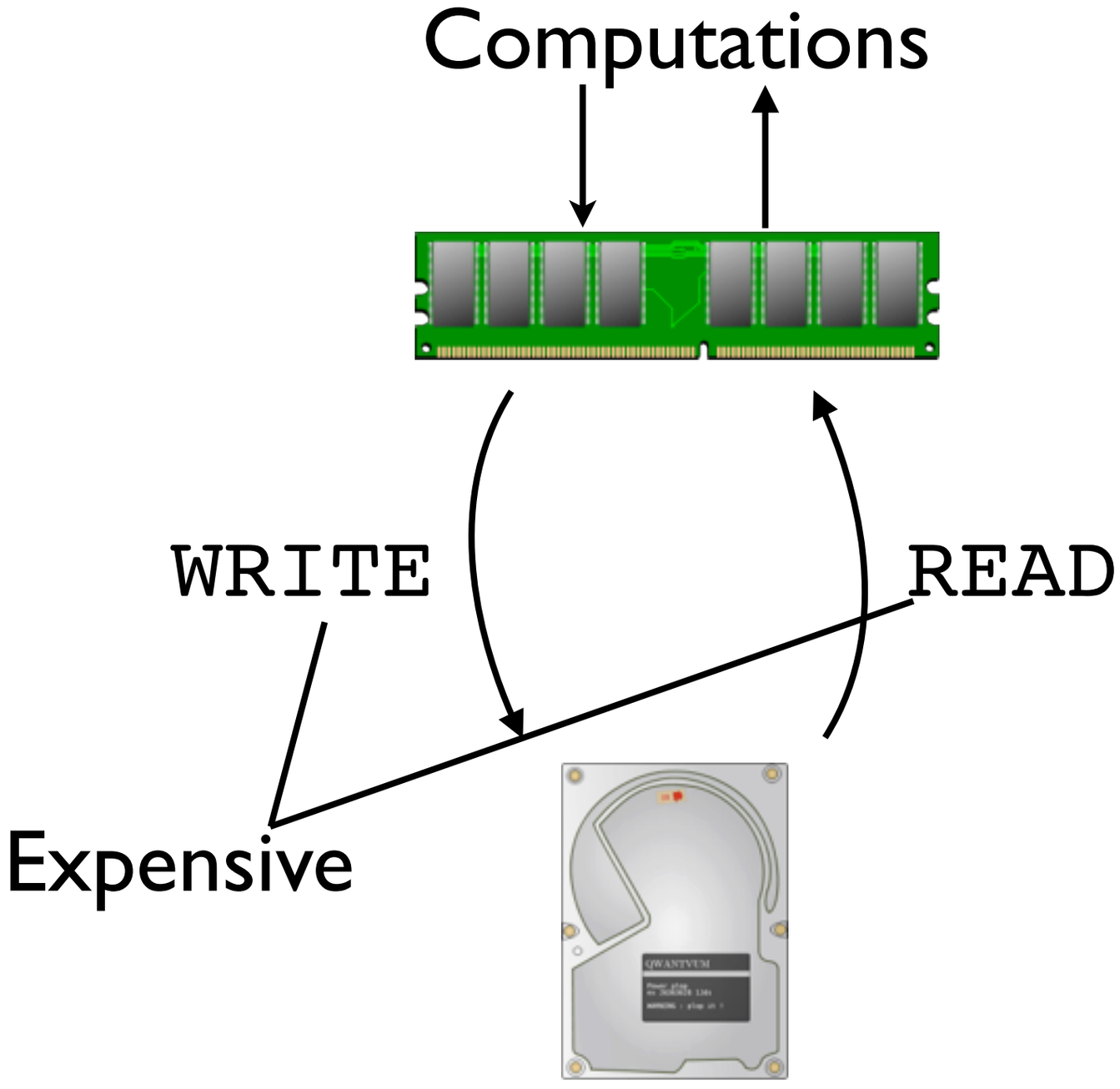


image credit: openclipart.org

Why not use just RAM?

- RAM is more expensive than HD
  - 200 MB/$ vs 10 GB/$
- RAM is smaller
  - 128 GB vs 10 TB
- RAM is volatile

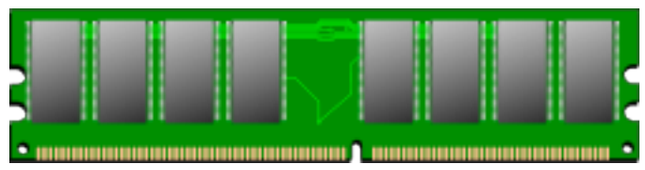
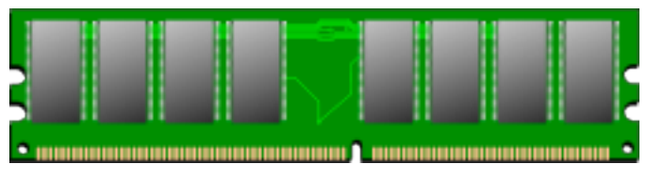


image credit: openclipart.org

Why not use just RAM?

- RAM is more expensive than HD
  - 200 MB/$ vs 10 GB/$
- RAM is smaller
  - 128 GB vs 10 TB
- RAM is volatile



Are in-memory databases still viable? (Hint: Yes)

image credit: openclipart.org

In-Memory DBs

- Why use In-Memory DBs?
  - Faster processing (especially for random access)
- How can we provide persistence?
  - … with respect to local failures (crashes)
  - … with respect to global failures (hurricanes)
- How do we provide scale?
  - Some DBs need TB/PB of space.

Hard Disks (and Flash)

- Data is stored on ~4KB (or more) **pages**
- Tuples are typically < 4KB
- Need to store multiple tuples per page.
- Why is this expensive?
- How can we mitigate the cost?

image credit: openclipart.org

Components of a Disk

- Platters spin (e.g., at 5400 rpm).
- The arm assembly moves in/out to a desired track (like a record player).
- A cylinder is the set of tracks in the same place on each platter.
- One head at time reads/writes.
- One block is read/written to/ from the disk at a time.

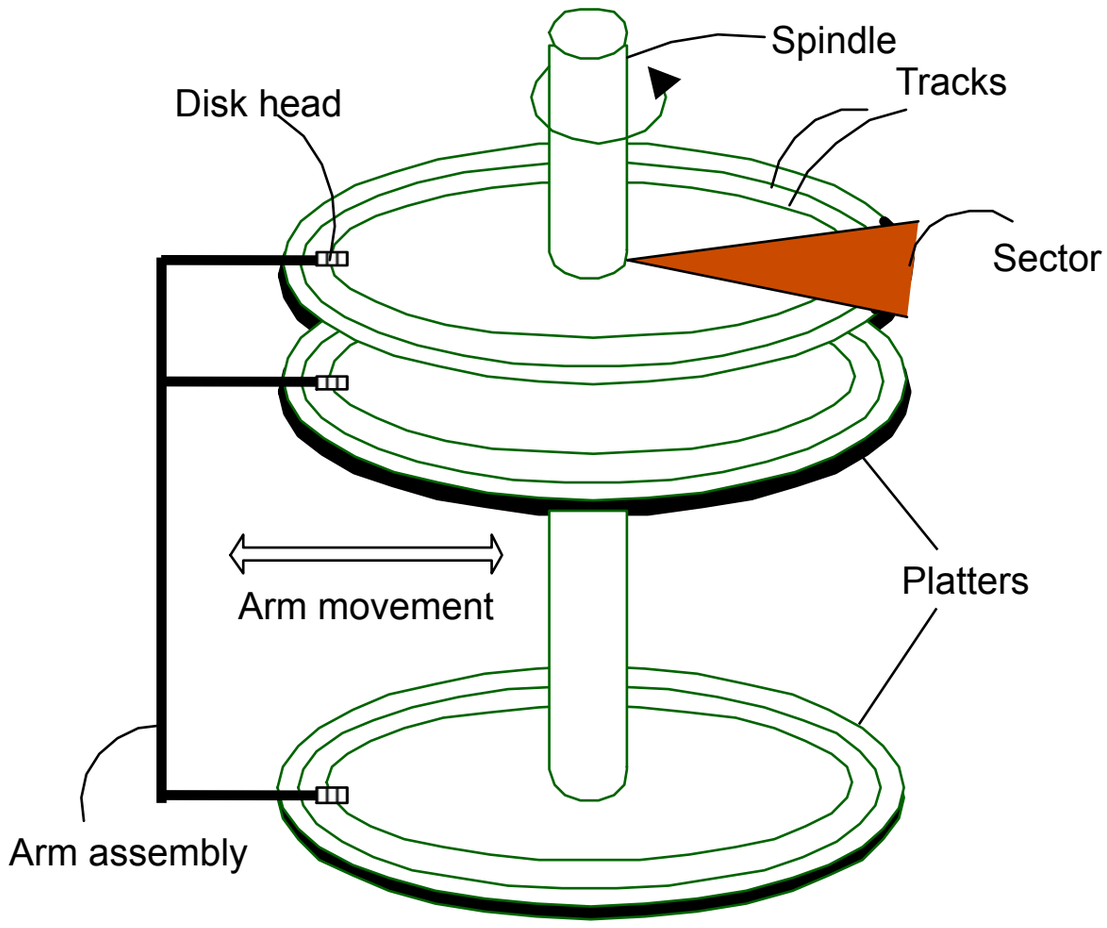


image credit: R&G

Accessing the Disk

- **Seek Time**: The time to move the arm assembly/disk head to the right track (1-20 ms).
- **Rotational Delay**: Waiting for the block to rotate under the disk head (0-10ms).
- **Transfer Time**: Time to read/write the block from/to disk (1ms/4kb page).

Accessing the Disk

- **Seek Time**: The time to move the arm assembly/disk head to the right track (1-20 ms).
- **Rotational Delay**: Waiting for the block to rotate under the disk head (0-10ms).
- **Transfer Time**: Time to read/write the block from/to disk (1ms/4kb page).

How do we make reads/writes more efficient?

“Nearby” Blocks

- Blocks in a file should be kept nearby…
  - …within a track.
  - …in the same cylinder.
    - …in adjacent cylinders.
- Minimize seek/rotational delay to get nearby blocks in a file.
- Bulk transfers of adjacent blocks are cheap!

Flash Memory

(The same stuff that's in your cellphone/camera)

- Two types: NOR/NAND-based
  - NAND is denser and provides bit-addressing
- Bit-at-a-time writes (unset operation on bits)
  - Block-at-a-time erasure (set operation on blocks)
- Erasure wears out the chip
  - 100,000-1,000,000 read/write cycles per block.
  - "Wear leveling" spreads writes between blocks.

Accessing Flash

- Different Cost Structure
  - Random Reads == Sequential Reads
  - Append to Block < Write to Block
- What kind of data-structures would work well with flash?

Hard Disks vs Flash

- Flash has faster read/write times
- Flash supports random read access
- Hard disks are more durable
- Hard disks are bigger
- Erasing flash is slow (can make writes slow)

RAID

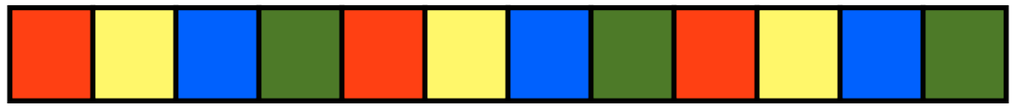
- Redundant Array of Independent Disks
- Combine several disks into a single ‘virtual’ disk.
- Goals: Increased performance and redundancy.
- **Striping**: spread data across multiple disks.
  - Allows for parallel data access.
- **Mirroring**: Copies of data allow for redundancy.

RAID

- **Level 0**: Striping only
  - N disks = Reads/Writes @ N x disk bandwidth
- **Level 1**: Mirroring only (Exactly 2 disks)
  - Parallel reads, Serial writes
  - 2 disks: Reads @ 2x disk bandwidth, Writes @ 1x
- **Level 0+1**: Stripe over mirrored pairs
  - Parallel reads, Writes serial on 2 disks
  - N disks: Reads @ N x disk bandwidth, Writes @ (N/2)x

RAID

- **Level 2-4**: Striped with parity bits, bytes, or blocks (respectively)



Can lose any 3 drives and still reconstruct all data

image credit: openclipart.org

RAID

- **Level 2-4**: Striped with parity bits(Lvl 2), bytes(Lvl 3), or blocks(Lvl 4)

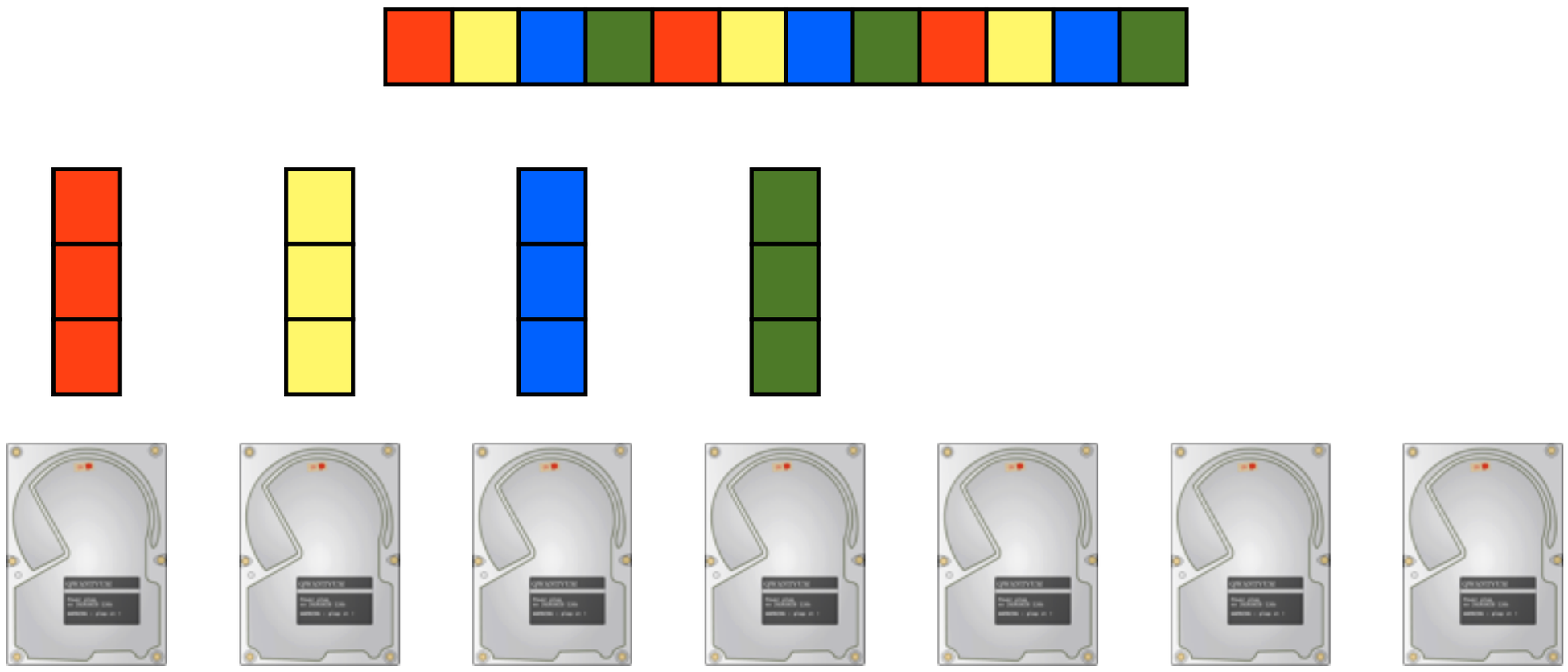


image credit: openclipart.org

RAID

- **Level 2-4**: Striped with parity bits(Lvl 2), bytes(Lvl 3), or blocks(Lvl 4)

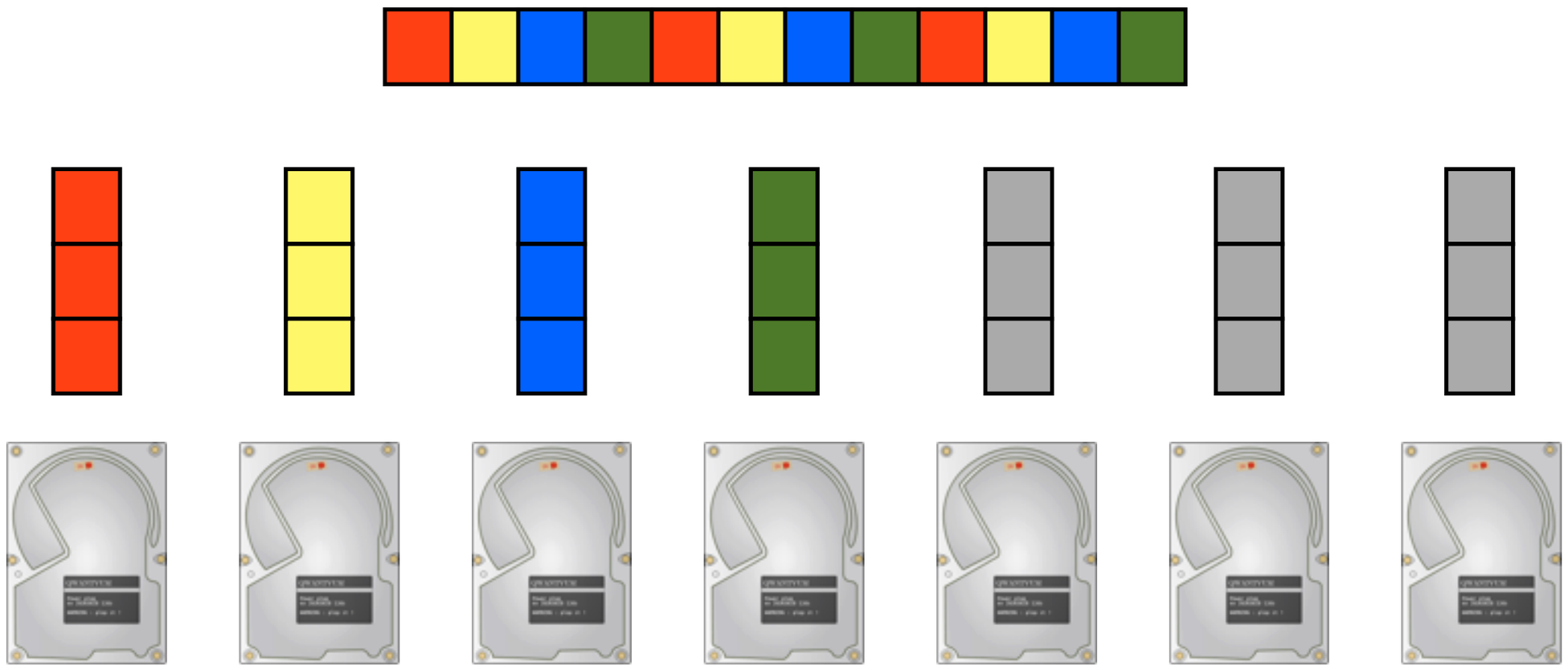
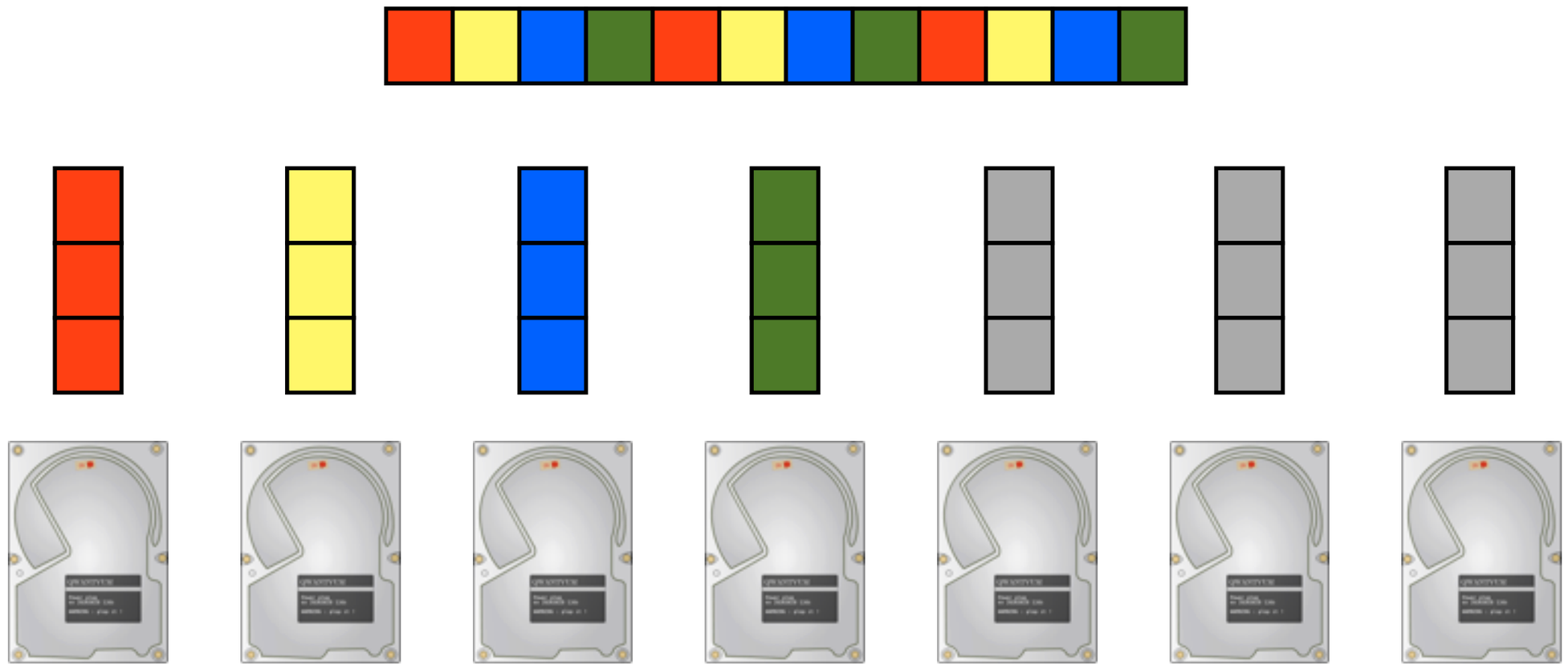


image credit: openclipart.org

RAID

- **Level 2-4**: Striped with parity bits(Lvl 2), bytes(Lvl 3), or blocks(Lvl 4)



Can lose any 3 drives and still reconstruct all data

image credit: openclipart.org

Error Correcting Codes

- e.g. Hamming, Tornado, Reed-Solomin
- Parity Bits

- Store N data values in N+K ‘slots’
- Can lose up to K ‘slots’ of data and still reconstruct the original N data values
- Reconstruction requires any N ‘slots’

Error Correcting Codes

- e.g. Hamming, Tornado, Reed-Solomin
- Parity Bits

- Store N data values in N+K ‘slots’
- Can lose up to K ‘slots’ of data and still reconstruct the original N data values
- Reconstruction requires any N ‘slots’

Why are Raid Levels 2-4 suboptimal?

RAID

- **Level 5**: Like Level 4, but parity blocks are spread out over each disk.

More read bandwidth available

image credit: openclipart.org

Summary

- Disks provide cheap, non-volatile storage
  - Support random access, but sequential reads are faster
- DBMS vs OS filesystem
  - DBMS has a better idea of its access patterns, and can exploit this to get better filesystem performance.