

Introduction to DBMS Internals

DBMS Architecture

Data storage

Alvin Cheung
Aditya Parameswaran
Reading: R & G Chapter 9



Course Overview

- Unit 1: Relational model and SQL
- Unit 2: Storage and indexing
- Unit 3: Query execution
- Unit 4: Query optimization
- Unit 5: Transactions
- Unit 6: Recovery
- Unit 7: Conceptual design
- Unit 8: Advanced topics (time permitting)



DBMS Architecture

Architecture of a DBMS: SQL Client



- How is a SQL query executed?



DBMS: Parsing & Optimization



Purpose:

Parse, check, and verify the SQL

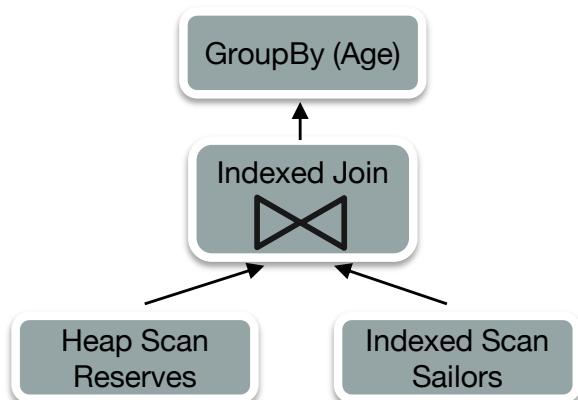
```
SELECT S.sid, S.sname, R.bid  
FROM Sailors R, Reserves R  
WHERE S.sid = R.sid and S.age > 30  
GROUP BY age
```

And translate into an efficient relational query plan



DBMS: Relational Operators

Purpose: Execute query plan by operating on **records** and **files**

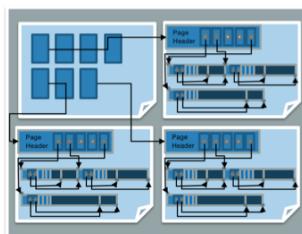


DBMS: Files and Index Management



Purpose: Organize tables and Records as groups of pages in a logical file

SSN	Last Name	First Name	Age	Salary
123	Adams	Elmo	31	\$400
443	Grouch	Oscar	32	\$300
244	Oz	Bert	55	\$140
134	Sanders	Ernie	55	\$400

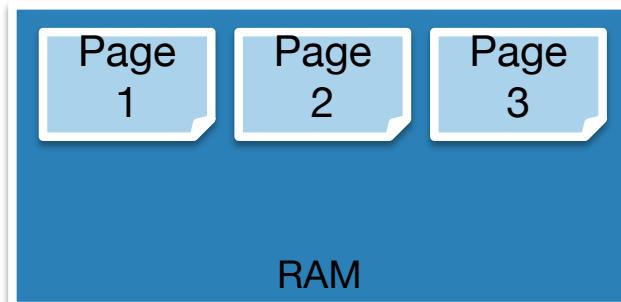


DBMS: Buffer Management



Purpose:

Provide the illusion of operating
in memory



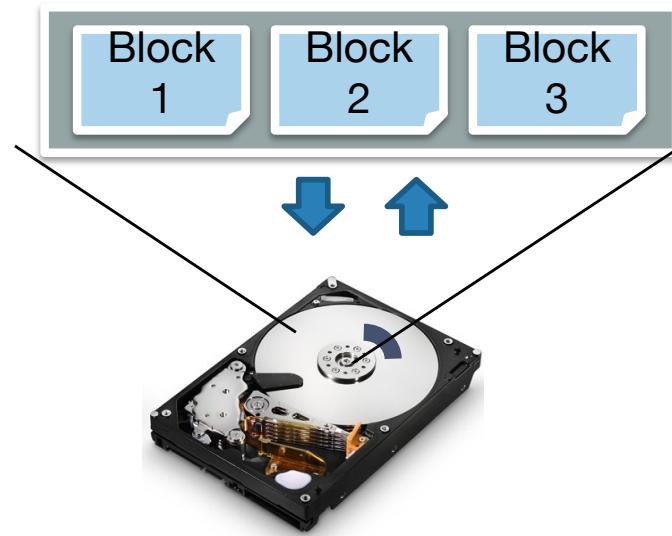
Disk Space Management



DBMS: Disk Space Management



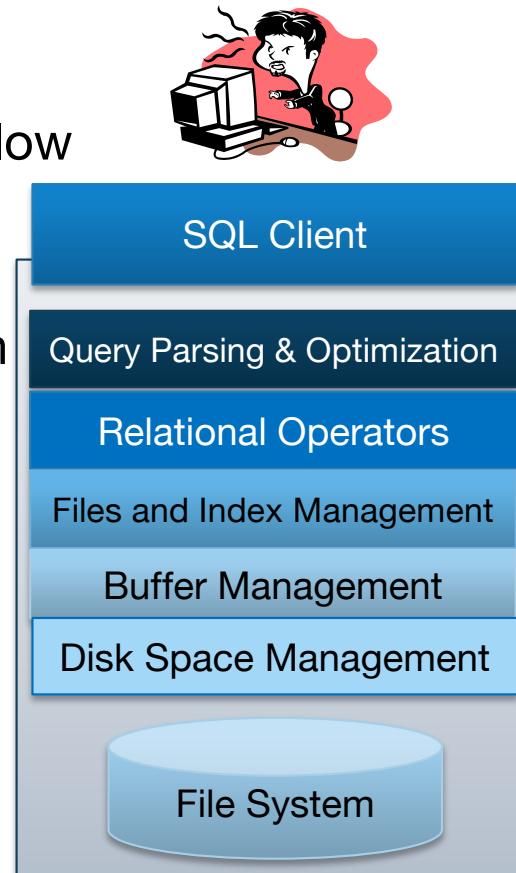
Purpose: Translate page requests into physical bytes on one or more device(s)



Architecture of a DBMS



- Organized in layers
- Each layer abstracts the layer below
 - Manage complexity
 - Performance assumptions
- Example of good systems design
- Many non-relational DBMSs are structured similarly



DBMS: Concurrency & Recovery

Two cross-cutting issues related to storage and memory management:



STORAGE MEDIA

Disks

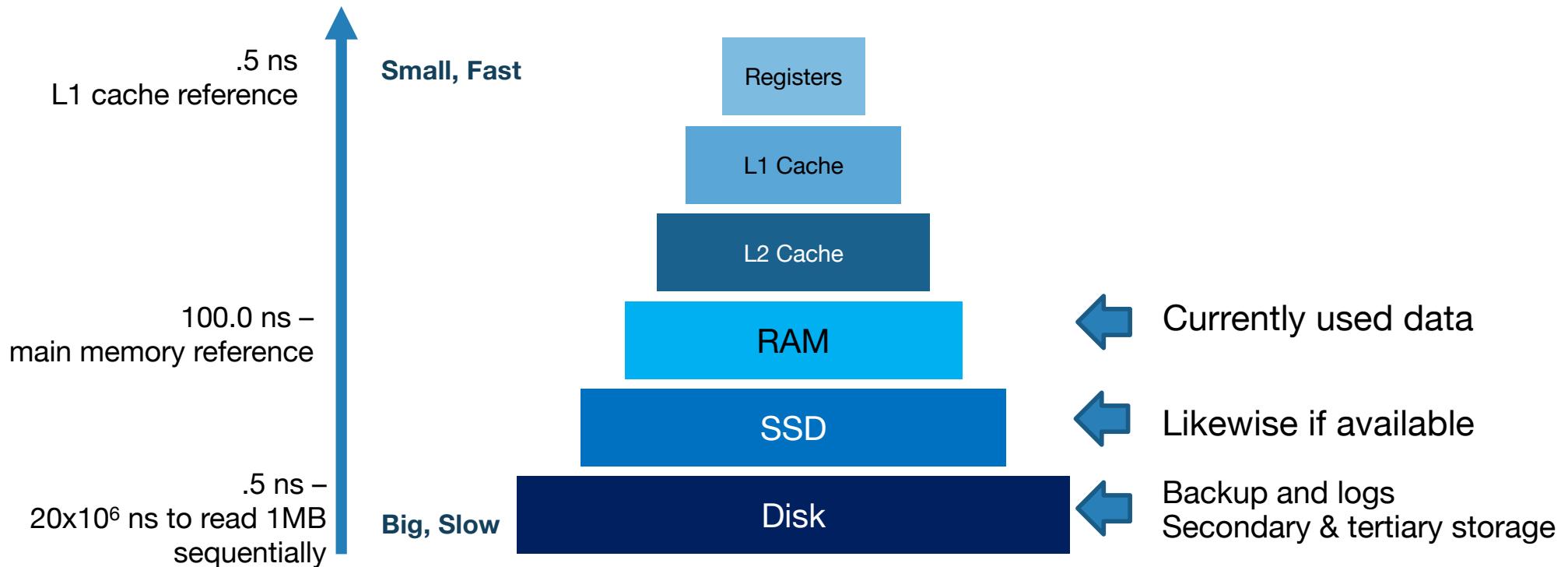


- Most database systems were originally designed for magnetic “spinning” disks
 - Disk are a mechanical anachronism!
 - Instilled design ideas that apply to using solid state disks as well
- Major implications:
 - Disk API:
 - READ: transfer “page” of data from disk to RAM.
 - WRITE: transfer “page” of data from RAM to disk.
 - No random reads / writes!!
 - Both API calls are very, **very slow!**
 - Plan carefully!



CS 162: Operating Systems
and System Programming

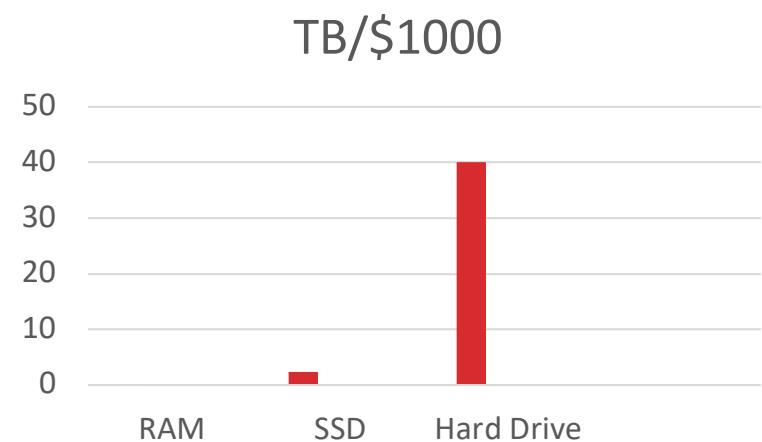
Storage Hierarchy



Economics



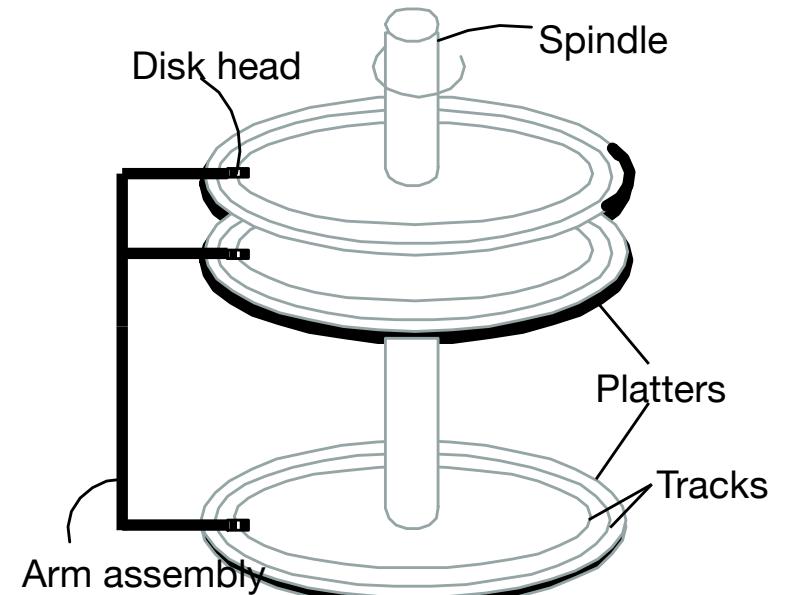
- \$1000 at NewEgg 2018:
 - Mag Disk: ~40TB for \$1000
 - SSD: ~2.3TB for \$1000
 - RAM: 80GB for \$1000



Components of a Disk



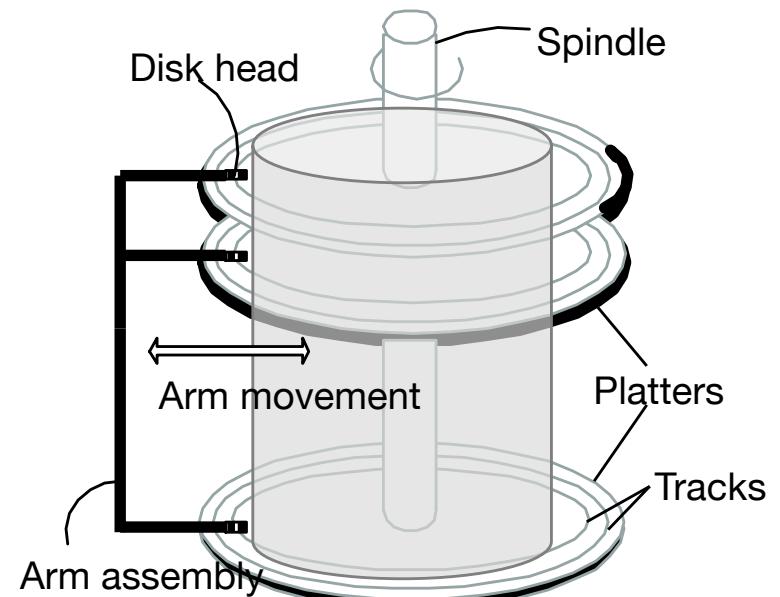
- **Platters** spin (say 15000 rpm)
- **Arm assembly** moved in or out to position a **head** on a desired **track**
 - Tracks under heads make a “cylinder”



Components of a Disk



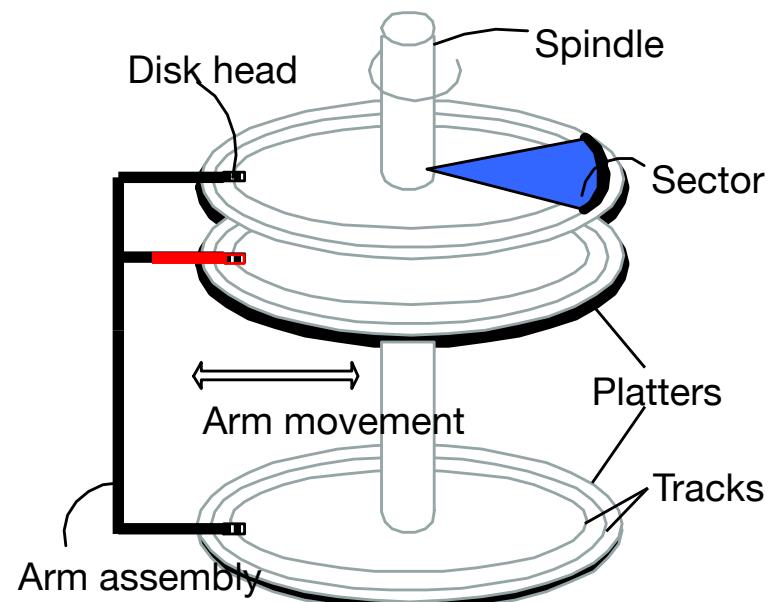
- **Platters** spin (say 15000 rpm)
- **Arm assembly** moved in or out to position a **head** on a desired **track**
 - Tracks under heads make a “cylinder”



Components of a Disk



- **Platters** spin (say 15000 rpm)
- **Arm assembly** moved in or out to position a **head** on a desired **track**
 - Tracks under heads make a “cylinder”
- Only one head reads/writes at any one time
- Block/page size is a multiple of (fixed) **sector** size



An Analogy



Accessing a Disk page

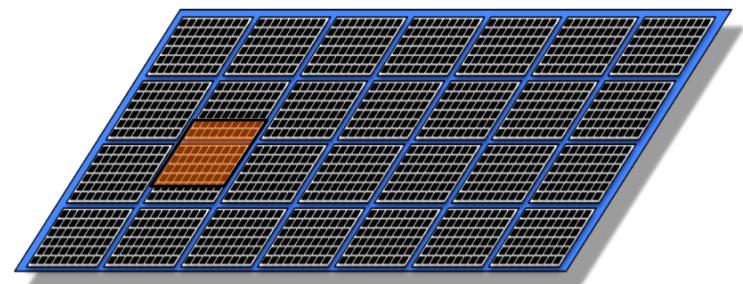


- Time to access (read/write) a disk block:
 - **seek time** (moving arms to position disk head on track)
 - ~2-3 ms on average
 - **rotational delay** (waiting for block to rotate under head)
 - ~0-4 ms (15000 RPM)
 - **transfer time** (actually moving data to/from disk surface)
 - ~0.25 ms per 64KB page
- Key to lower I/O cost: reduce seek/rotational delays

Flash (SSD)



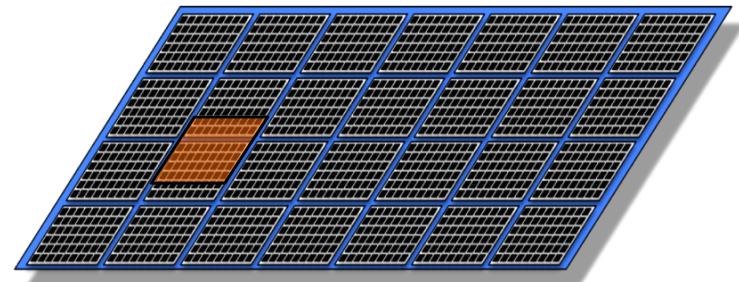
- Organized into “cells”
- Current generation (NAND)
 - Random reads and writes, but:
 - Fine-grain reads (4-8K reads), coarse-grain writes (1-2MB writes)



Flash (SSD), Pt. 2



- So... read is fast and predictable
 - 4KB random reads: ~500MB/sec
- But write is not!
 - 4KB random writes: ~120 MB/sec
 - Why? Only 2k-3k erasures before failure
 - so keep moving write units around (“wear leveling”)



DISK SPACE MANAGEMENT

Block Level Storage



- Read and Write **large chunks of sequential bytes**
- *Sequentially*: “Next” disk block is fastest
- Maximize usage of data per Read/Write
 - “Amortize” seek delays (HDDs) and writes (SSDs):
if you’re going all the way to Pluto, pack the spaceship full!
- Predict future behavior
 - Cache popular blocks
 - Pre-fetch likely-to-be-accessed blocks
 - Buffer writes to sequential blocks
 - More on these as we go

A Note on Terminology

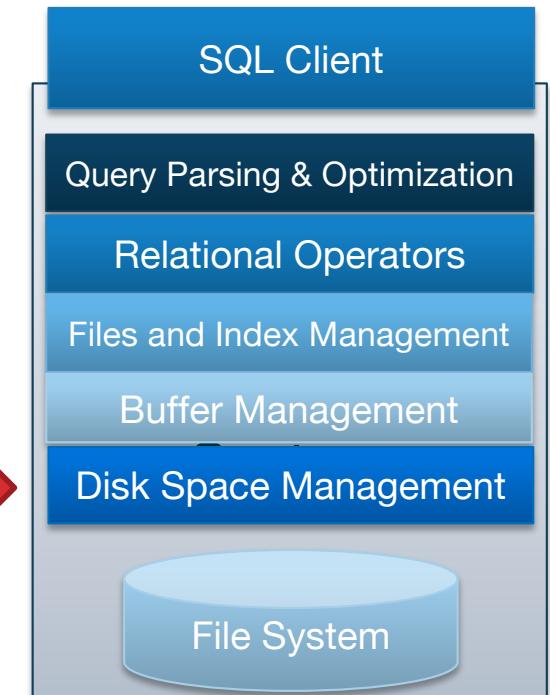


- **Block** = Unit of transfer for disk read/write
 - 64KB – 128KB is a good number today
 - Book says 4KB
 - We'll use this unit for all storage devices
- **Page**: a common synonym for “block”
 - In some texts, “page” = a block-sized chunk of RAM
- We'll treat “block” and “page” as synonyms

Disk Space Management



- Lowest layer of DBMS, manages space on disk
- Purpose:
 - Map pages to locations on disk
 - Load pages from disk to memory
 - Save pages back to disk & ensuring writes
- Higher levels call upon this layer to:
 - Read/write a page
 - Allocate/de-allocate logical pages



Disk Space Management: Requesting Pages



- ```
page = getFirstPage("Sailors");
while (!done) {
 process(page);
 page = page.nextPage();
}
```
- Physical details hidden from higher levels of system
- Higher levels may “safely” assume nextPage is fast
  - Hence sequential runs of pages are quick to scan

# Disk Space Management: Implementation



- **Proposal 1:** Talk to the storage device directly
  - Could be very fast if you knew the device well
  - Hard to program when each device has its own API
  - What happens when devices change?

# Disk Space Management: Implementation



- **Proposal 2:** Run our own over filesystem (FS)
  - Bypass the OS, allocate single large “contiguous” file on an empty disk
    - assume sequential/nearby byte access are fast
  - Most FS optimize disk layout for sequential access
    - Gives us more or less what we want if we start with an empty disk
  - DBMS “file” may span multiple FS files on multiple disks/machines

# Disks and Files: Summary



- Magnetic (hard) disks and SSDs
  - Basic HDD and SSD mechanics
  - Concept of “near” pages and how it relates to cost of access
- Relative cost of
  - Random vs. sequential disk access (10x)
  - Disk (Pluto) vs RAM (Sacramento) vs. registers (your head)
    - Big, big differences!

# Files: Summary



- DB File storage
  - Typically over FS file(s)
- Disk space manager loads and stores pages
  - Block level reasoning
  - Abstracts device and file system; provides fast “next page”