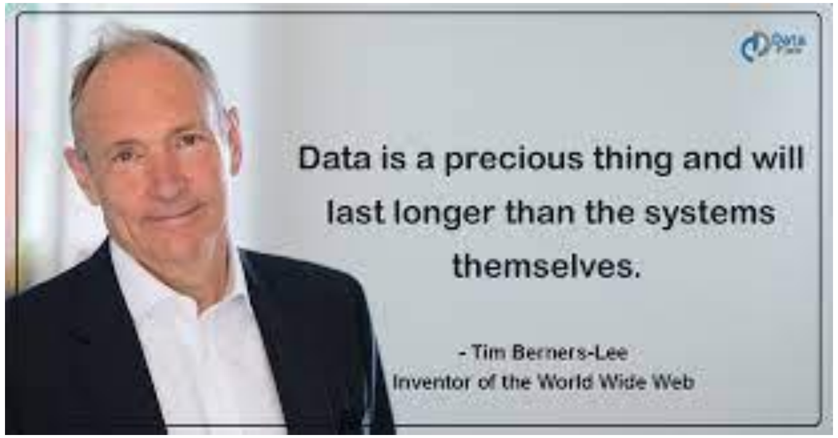
CS460
Systems for Data Management and Data Science

Prof. Anastasia Ailamaki
Prof. Anne-Marie Kermarrec

Introduction and Storage Management

February 17, 2025

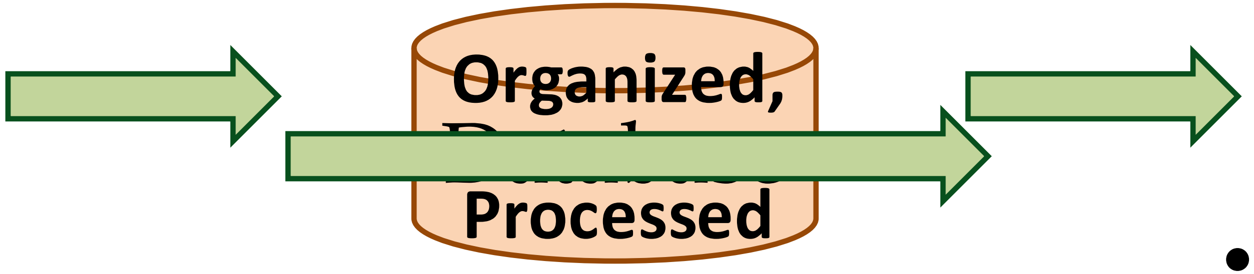
Data: an **extremely** valuable resource



What is data?

Organized, Processed

Data



Information

- Facts
- Basis for reasoning/discussion/ calculation
- Useful or irrelevant or redundant
- Must be processed to be meaningful

- Has meaning
- Relevant to the problem
- Actionable – leads to a solution

Knowledge

Wisdom

What is a database?

- A **large**, **integrated**, **structured collection** of data
- Usually intended to model some real-world enterprise
- Example: University
  - Courses
  - Students
  - Professors
  - Enrollment
  - Teaching

Entities (Courses, Students, Professors)

Relationships (Enrollment, Teaching)

What is a Database Management System (DBMS)?

- A software system designed to **store**, **manage,** and **facilitate access** to databases
- **DBMS** = Interrelated data (database) + set of programs to access it (software)

What does a DBMS do?

Protects data from failures: h/w, s/w, power; malicious users

Thousands of queries / updates per second

24X7 availability

Physical data independence, declarative high-level query languages

Provides efficient, reliable, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

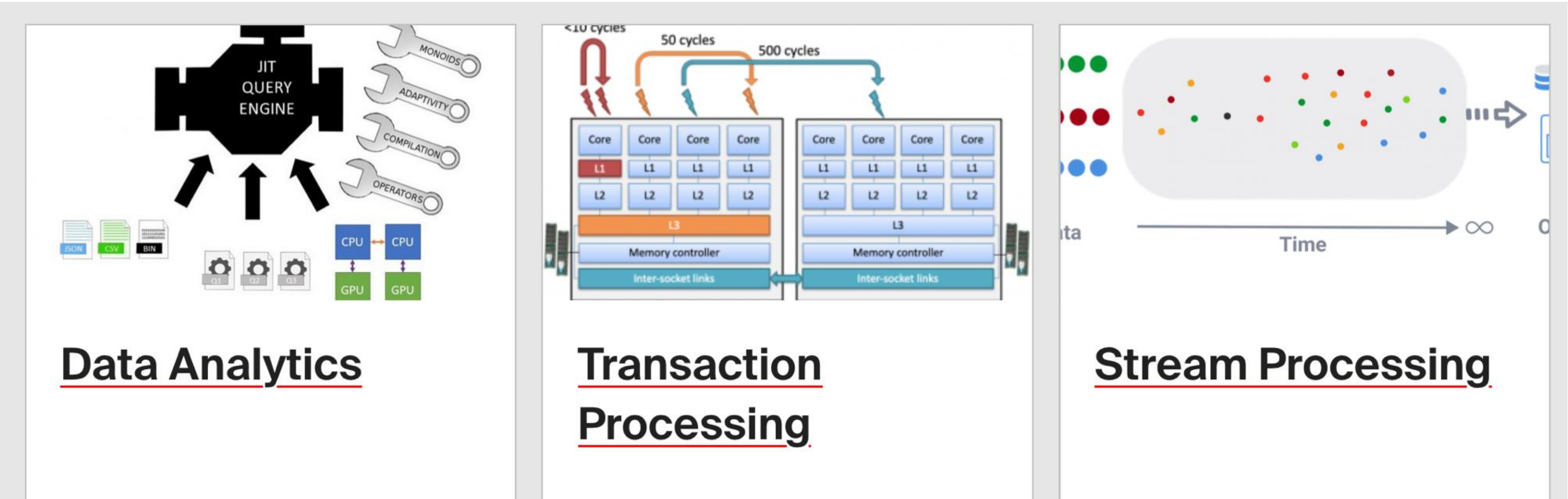
Concurrency control

Extremely large (often Exabytes every day)

Data outlives the programs that operate on it

Data-intensive applications & systems

- Data-intensive vs compute-intensive
  - Volume, complexity, velocity, volatility, variety…
- Hardware/software codesign
  – Optimizing for memory hierarchy, and hardware accelerators
- Scientific applications



Data science

A data-driven approach to problem solving by analyzing and exploring large volumes of possibly multi-modal data.

It involves collecting, preparing, managing, processing, analyzing, and explaining the data and analysis results.

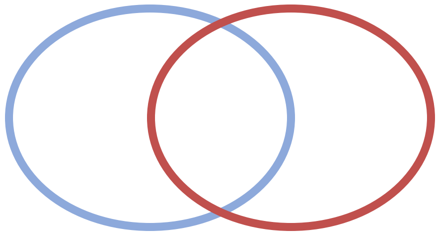
Data science is interdisciplinary (statistics, computer science, information science, mathematics, social science, visualization, etc.).

Debunking some myths

- Data Science <> Big Data

- Data science <> Machine Learning

© marketoonist.com



Related but not the same!

Data science's *raison d'existence*: its applications

- Fraud and Risk detection
- Recommendation Systems
- Digital advertisement
- Finance
- Image and speech recognition
- Gaming
- Healthcare
- Smart cities
- Sustainability
- Augmented reality

- Real Time
- Scalability
- Accuracy
- Latency
- Failure Resilience
- Security/ Privacy

The many faces of data science

Data engineering

Big data management
Data preparation
Large-scale deployment

Data analytics

Data exploration (mining)
Models and algorithms (ML)
Visualizations

Data Security / Privacy

Data integrity
Differential privacy
Cryptography

Data Ethics

Biases (data and algorithms)
Impact on society
Regulations

CS460 landscape

- Gossip Protocols
- Consistency protocols
  CAP Theorem
- Distributed/decentralized systems

Data science software stack

Data Processing

- **Transaction Management**
- **Graph Data** (Pregel, GraphLab, X-Streem, Chaos)
- **Structured Data** (Spark SQL)
- **Machine Learning**
- **Query Execution**
- **Batch Data** (Map Reduce, Dryad, Spark)
- **Streaming Data** (Storm, Naiad, Flink, Spark Streaming Google Data Flow)

Data Storage

- **Storage Hierarchies & Data Layout**
- **Distributed File Systems** (GFS)
- **NoSQL DB** (Dynamo Big Table Cassandra)
- **Distributed Messaging systems** (Kafka)

Ressource Management & Optimization

- **Query optimization**
- **Scheduling (Mesos, YARN)**

Week	Date	Topic
1	17/02	Introduction and Storage hierarchy
2	24/02	Query execution
3	03/03	Query Optimization
4	10/03	Transactions
5	17/03	Distributed transactions
6	24/03	Distributed Query Execution
7	31/03	**Midterm exam (not graded)**
8	7/04	Gossip Protocols
9	14/04	Distributed hash tables + consistency models
10	28/04	Key-value stores + CAP theorem
11	5/05	Scheduling
12	12/05	Stream Processing
13	12/05	Distributed Learning Systems
14	19/05	Invited Industry Lecture

CS460 learning experience

Lecture

Learn the internals of a (distributed) platform for data science

Breadth coverage

Exercises

Put the course in practice

Programming skills

Exam preparation

Background for the project

Project

Acquaintance with a real platform

Going in depth

Intended as a practical work

May not be related to every part of the course

TA/AE Team

Martijn (head TA)

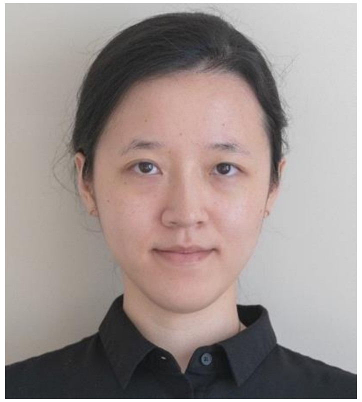
Milos (TA)

Diana (TA)

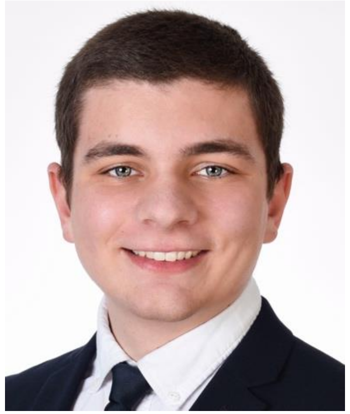
Hamish (TA)

Alex (AE)

Elif (AE)



Yi (TA)



Mathis (TA)

Mathis (TA)

Rishi (TA)

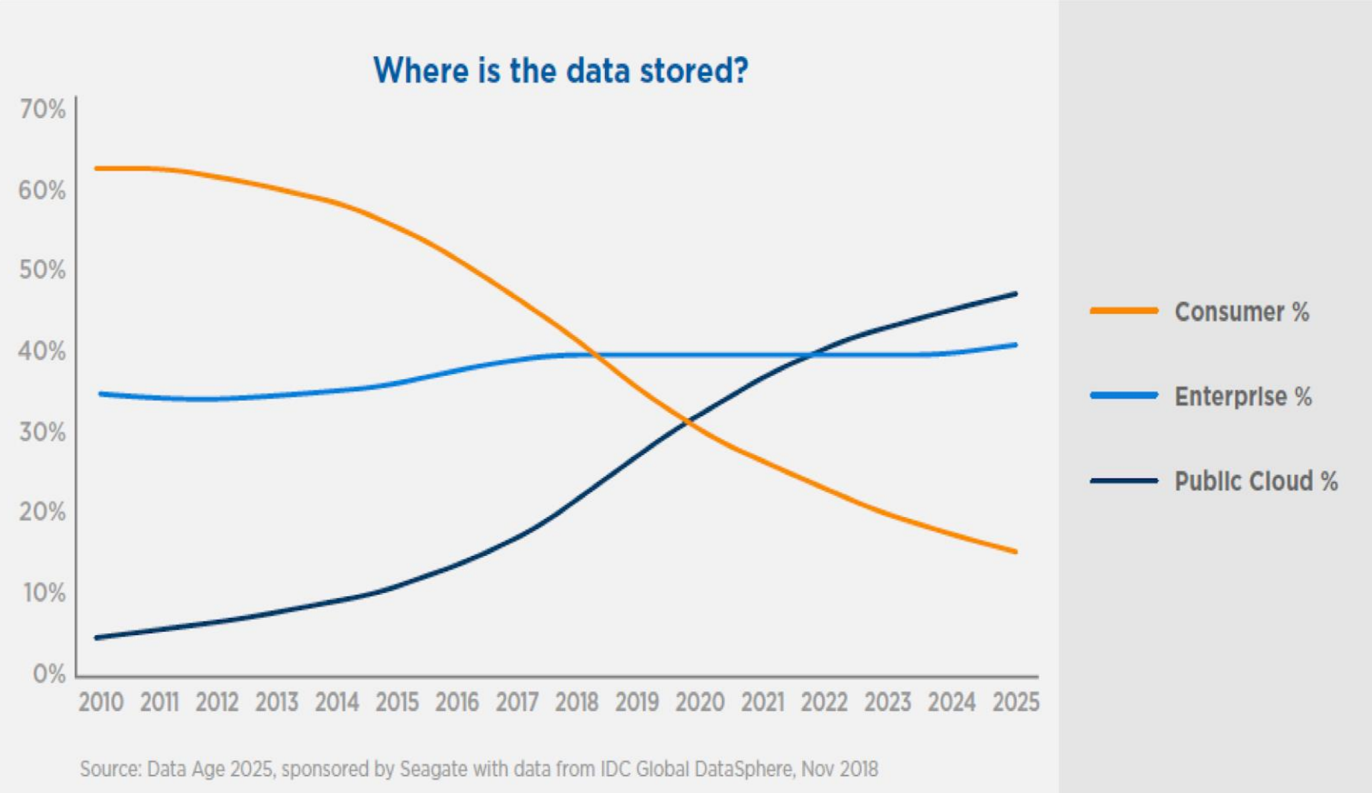
Jakob (AE)

Course logistics

- CS460 Moodle: all the material, updated every week
- Schedule
  - Lecture (Monday 2:15-4 pm) – CE12
  - Exercises (Monday 4:15-6 pm) – CE12 (week 1: project overview)
  - Individual Project (Tuesday 11-1pm: indicative time slot to work on the project)
- Grading scheme
  - Project (40%) (presentation at 16:15 today)
  - Midterm exam (not graded, highly recommended, covers weeks 1-5)
  - Final exam (60%)

Data moves to the Cloud

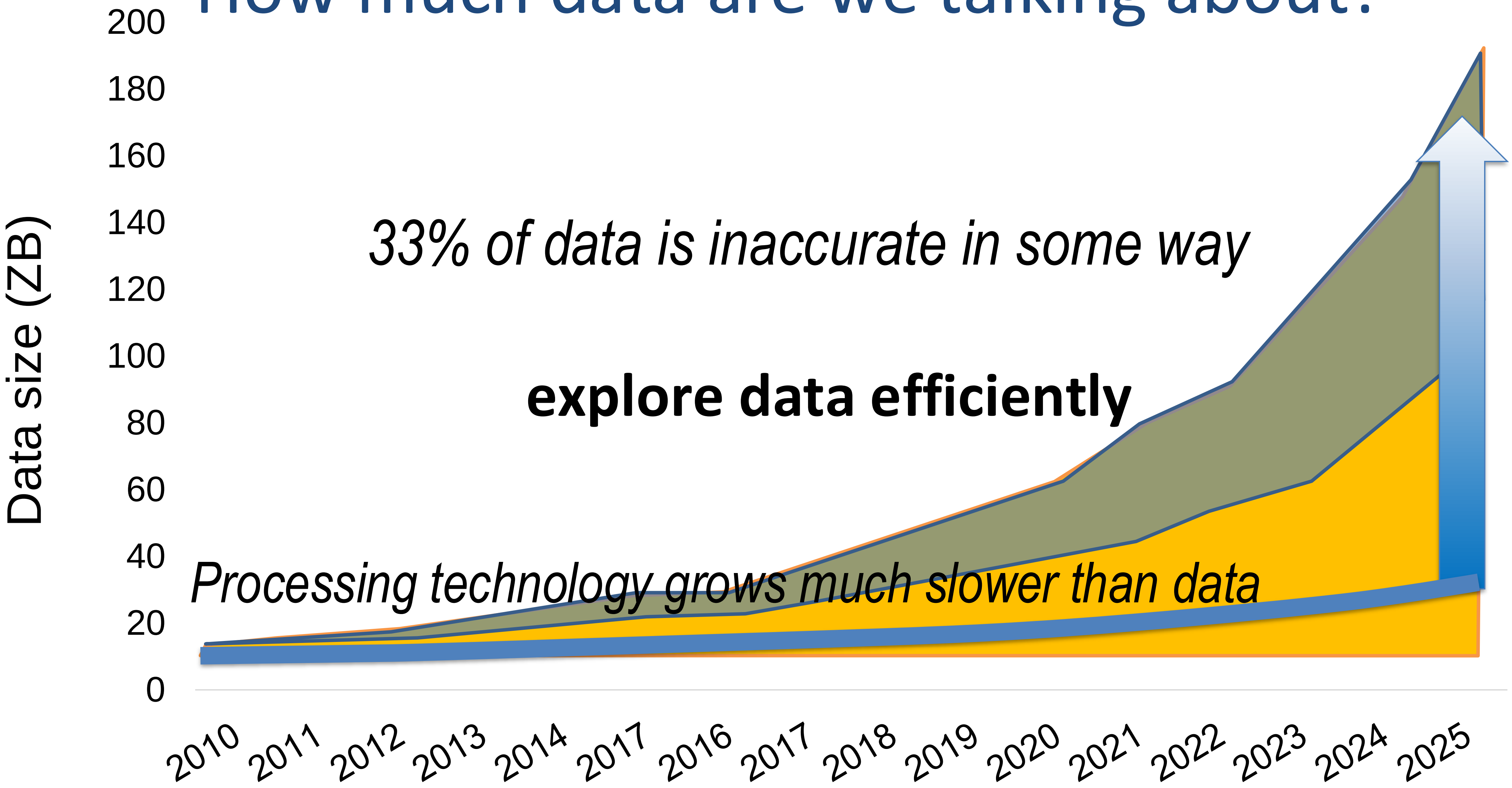
In 2021, cloud workloads represent 94% of all IT workload worldwide



FB spent 16Bn $ on datacenter in 2019
Google spent 13Bn $ datacenter in 2019

Hyperscale datacenters (fitting more IT in less space, **scale** hugely and quickly to increasing demand (**elasticity**, computing ability, memory, **networking** infrastructure, disaggregated **storage**) have been growing at a historic rate over the past 10 years

How much data are we talking about?



Data Age 2025, data from IDC Global DataSphere, Nov 2018

Scalability

- What if your system grows from 50,000 concurrent users to 10M
- Scalability: ability to cope with increasing load
- Load: number of requests/second, ratio of reads/writes in a database, number of simultaneously active users…

- Performance metrics
  - Latency/Response time: duration for a request to be handled
  - Average versus percentiles
    - The 95th: response time at which 95% of requests are faster than that threshold
  - Tail latency: refers to high latencies that clients see fairly infrequently

Today's topic

Gossip Protocols

Consistency protocols
CAP Theorem

Distributed/Decentralized systems

Data science software stack

Data Processing

Transaction Management

Graph Data
Pregel, GraphLab, X-Streem, Chaos

Structured Data
Spark SQL

Machine Learning

Query Execution

Batch Data
Map Reduce, Dryad, Spark

Streaming Data
Storm, Naiad, Flink, Spark Streaming Google Data Flow

Data Storage

Storage Hierarchies & Data Layout

Distributed File Systems
(GFS)

NoSQL DB
Dynamo
Big Table
Cassandra

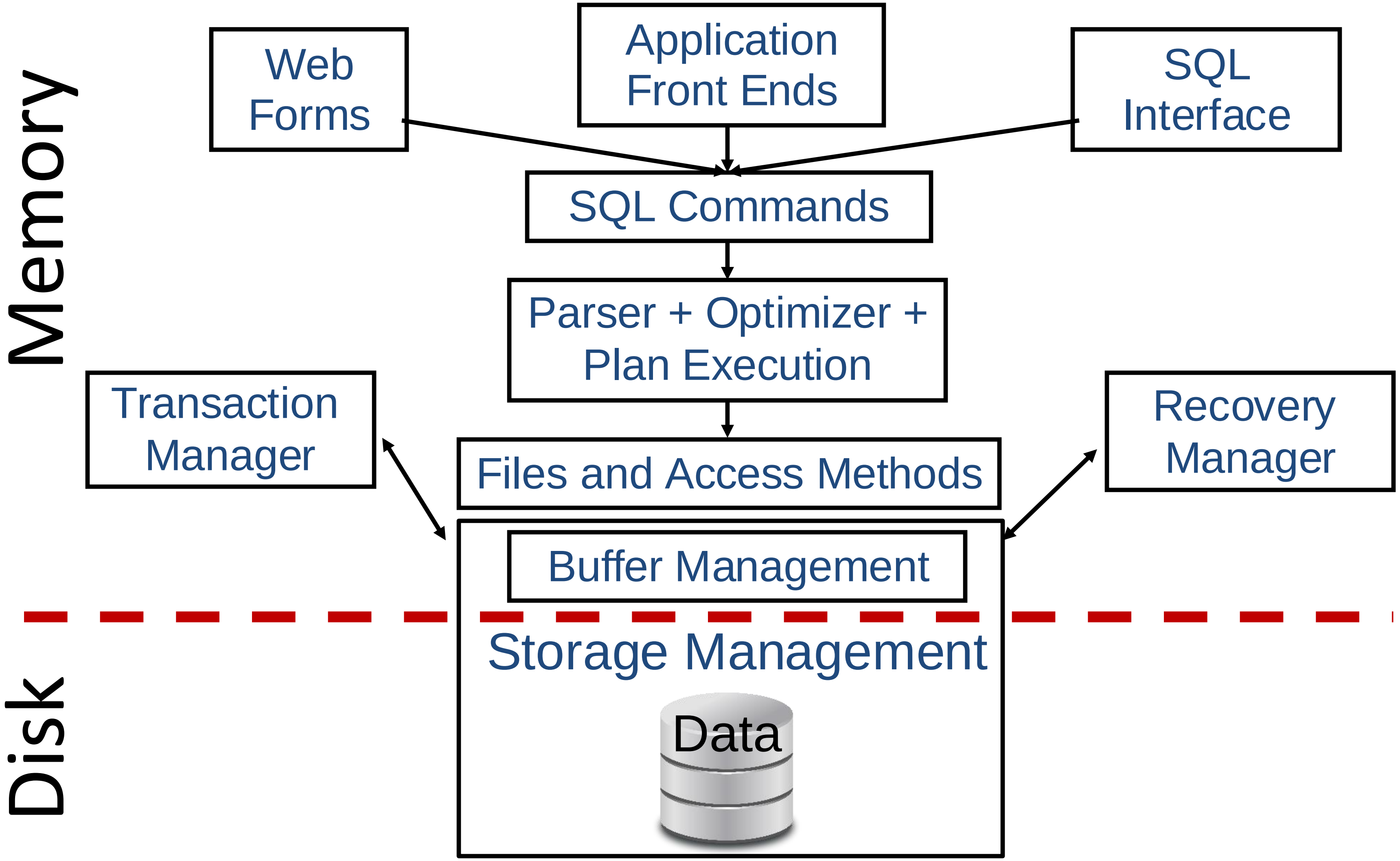
Distributed Messging systems
Kafka

Ressource Management & Optimization

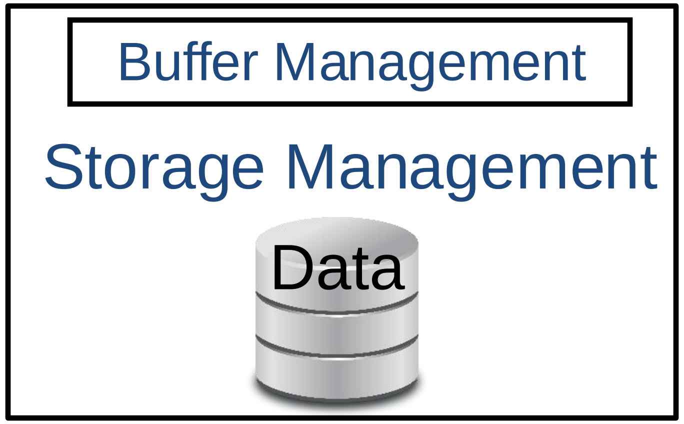
Query optimization

Scheduling (Mesos, YARN)

(Simplified) DBMS Architecture



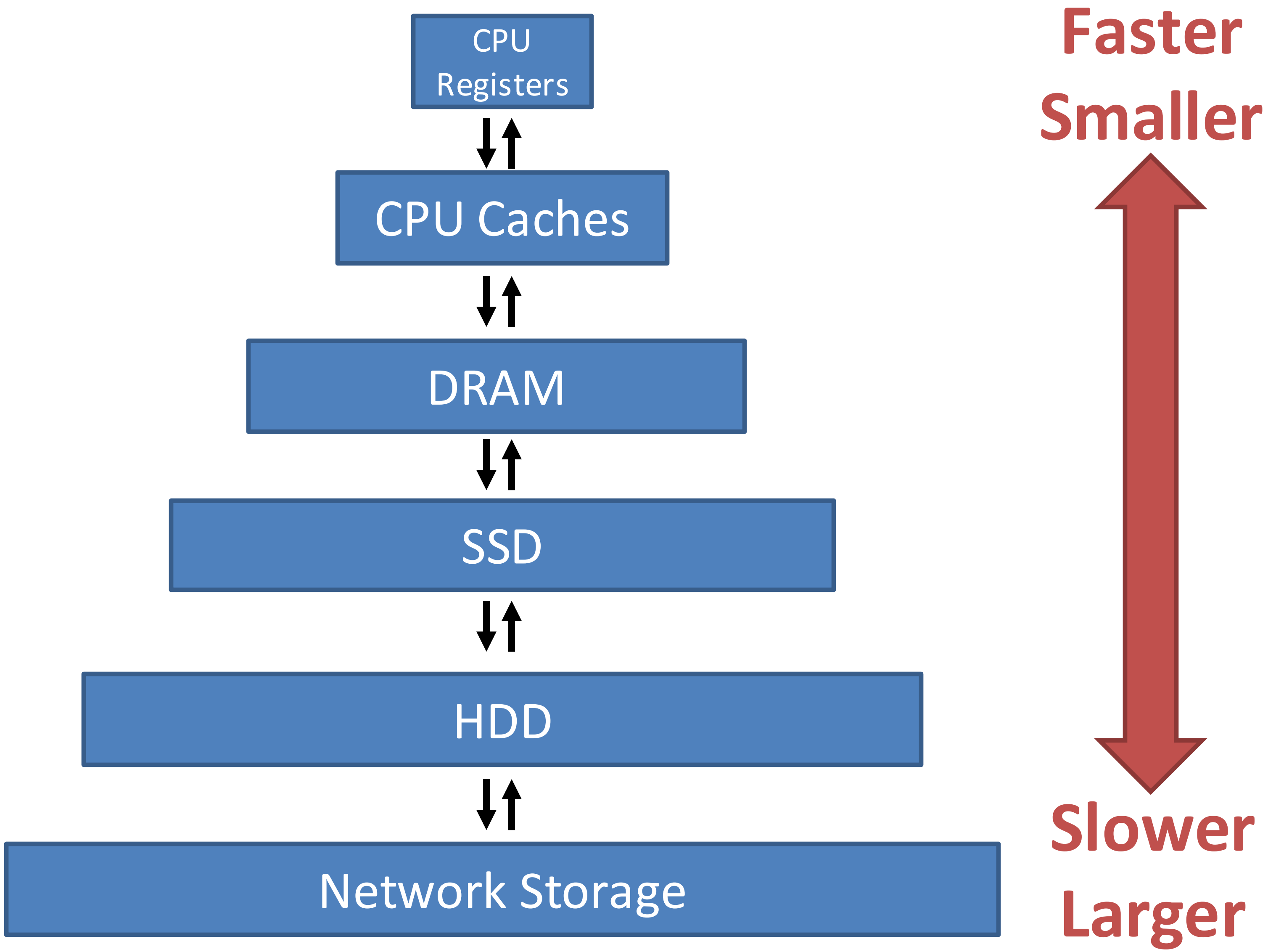
Today's topic



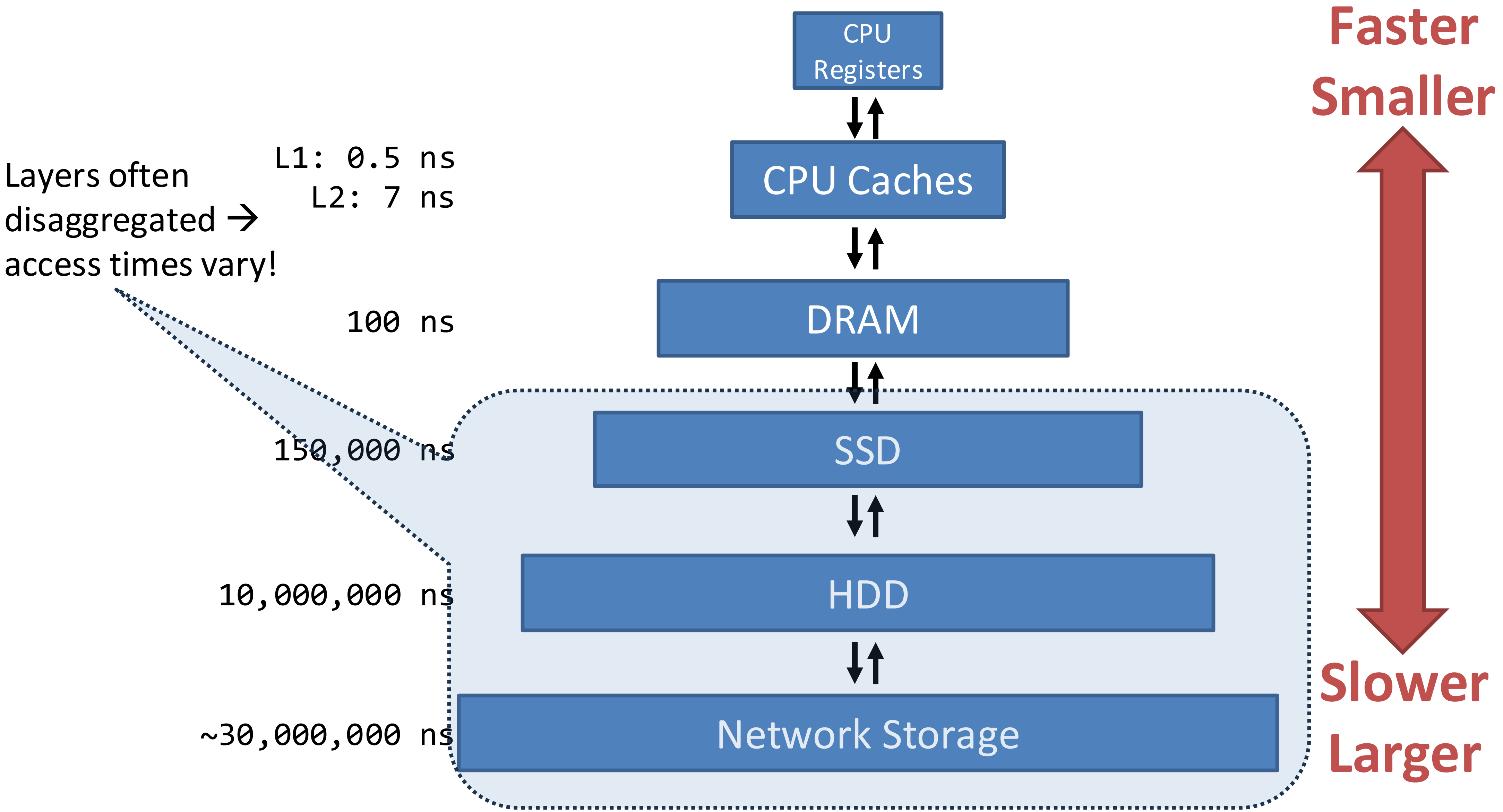
Storage Management: Outline

- **Storage Technologies**
- File Storage
- Buffer Management (refresher)
- Page Layout
  - NSM, aka row store
  - DSM, aka column store
  - PAX, hybrid

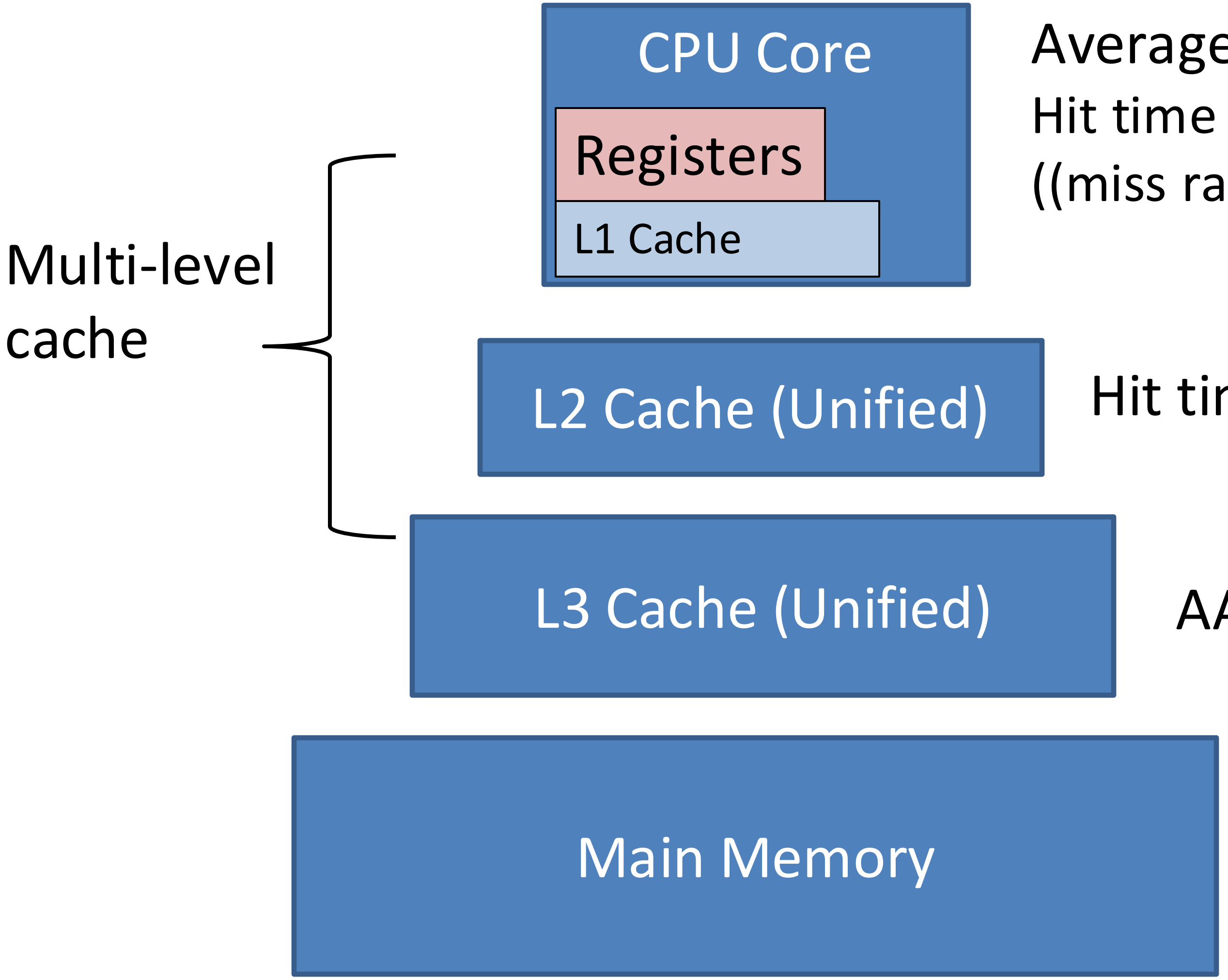
Storage Hierarchy



Storage layer access times



A surprisingly simple model for cache organization



Average access time (AAT)=
Hit time +
((miss rate) X (miss penalty))

$\text{Hit time}_{cache} < \text{Hit time}_{memory}$

⇓

$AAT_{cache} \ll AAT_{memory}$

Non-Volatile Memory vs Solid-State Drive

- **Goals**: data persists after power-cycle + reduce random/sequential access gap
  - No seek/rotational delays
- Like DRAM, low-latency loads and stores
- Like SSD, persistent writes and high density
- Byte-addressable

DRAM

NVM

SSD

- SSD technology uses non-volatile flash chips
  Package multiple flash chips into a single closure
- SSD controller
  Embedded processor that executes firmware-level software
  Bridges Flash chips to the SSD input/output interfaces
- Block-addressable

DRAM

NVM

SSD

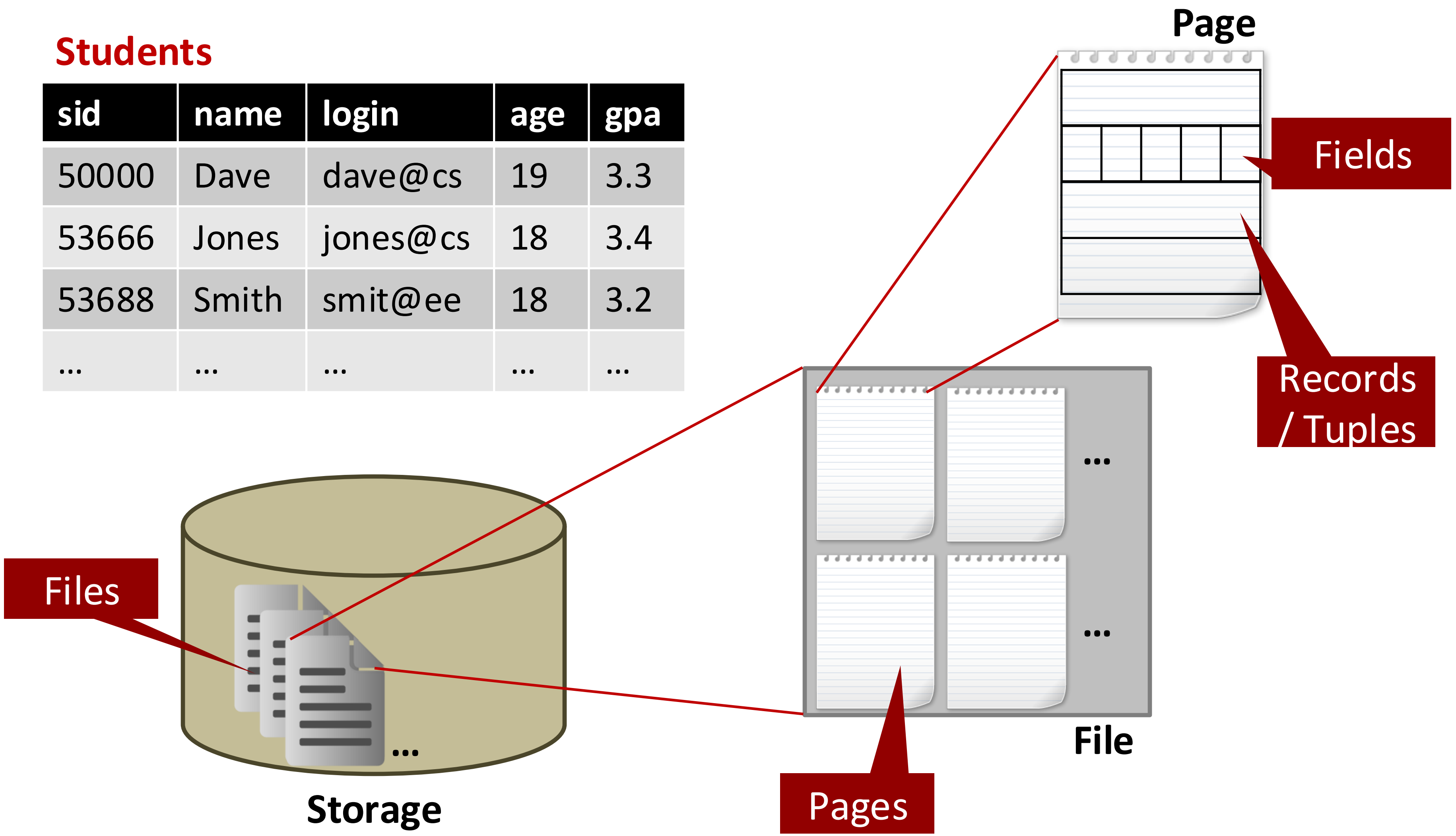
Storage Management: Outline

- Storage Technologies
- **File Storage**
- Buffer Management (refresher)
- Page Layout
  - NSM, aka row store
  - DSM, aka column store
  - PAX, hybrid

From tables/rows to files/pages

Students

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smit@ee	18	3.2
…	…	…	…	…



GOAL: DBMS must ***efficiently*** manage datasets larger than available memory

File Storage

The DBMS stores a database as one or more files on disk.

The **Storage Manager** is responsible for maintaining a database's files and organizes them as a collection of **pages**.

- Tracks data read/written to pages
- Tracks available space

Alternative File Organizations

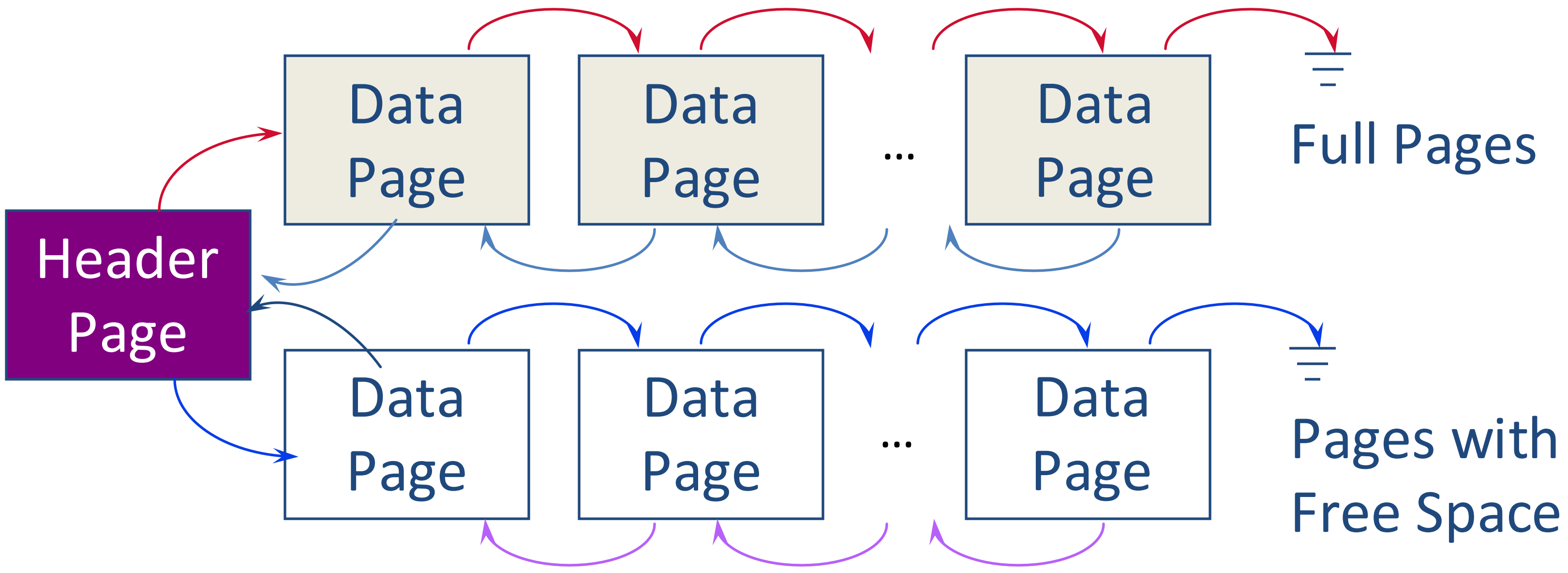
The **Storage Manager** is responsible for maintaining a database's files and organizes them as a collection of **pages**.
Many alternatives exist, *each good for some situations, and not so good in others*. Indicatively:

- Heap files: Best when typical access is a full file scan
- Sorted Files: Best for retrieval in an order, or for retrieving a 'range'
- Log-structured Files: Best for very fast insertions/deletions/updates

Heap (Unordered) Files

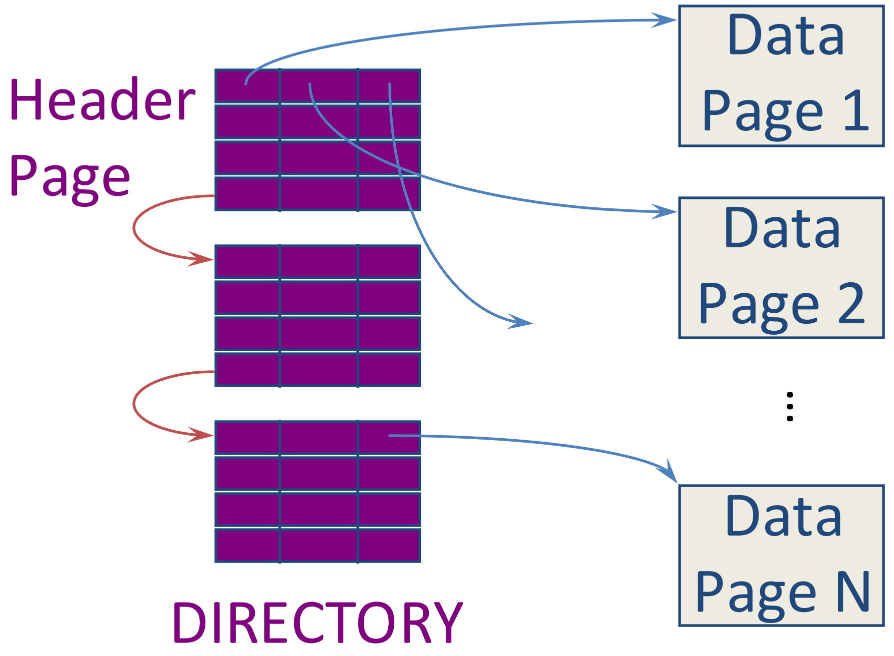
- Simplest file structure
  - contains records in no particular order
  - Need to be able to scan, search based on rid
- As file grows and shrinks, disk pages are allocated and de-allocated.
  - Need to manage free space

Heap File Implemented Using Lists



- <Heap file name, header page id> stored somewhere
- Each page contains 2 ‘pointers’ plus data.
- Manage free pages using free list
  - What if most pages have some space?

Heap File Using a Page Directory



- The directory is a collection of pages
  - linked list implementation is just one alternative.
- The entry for a page can include the number of free bytes on the page.
  - *Much smaller than linked list of all HF pages*!

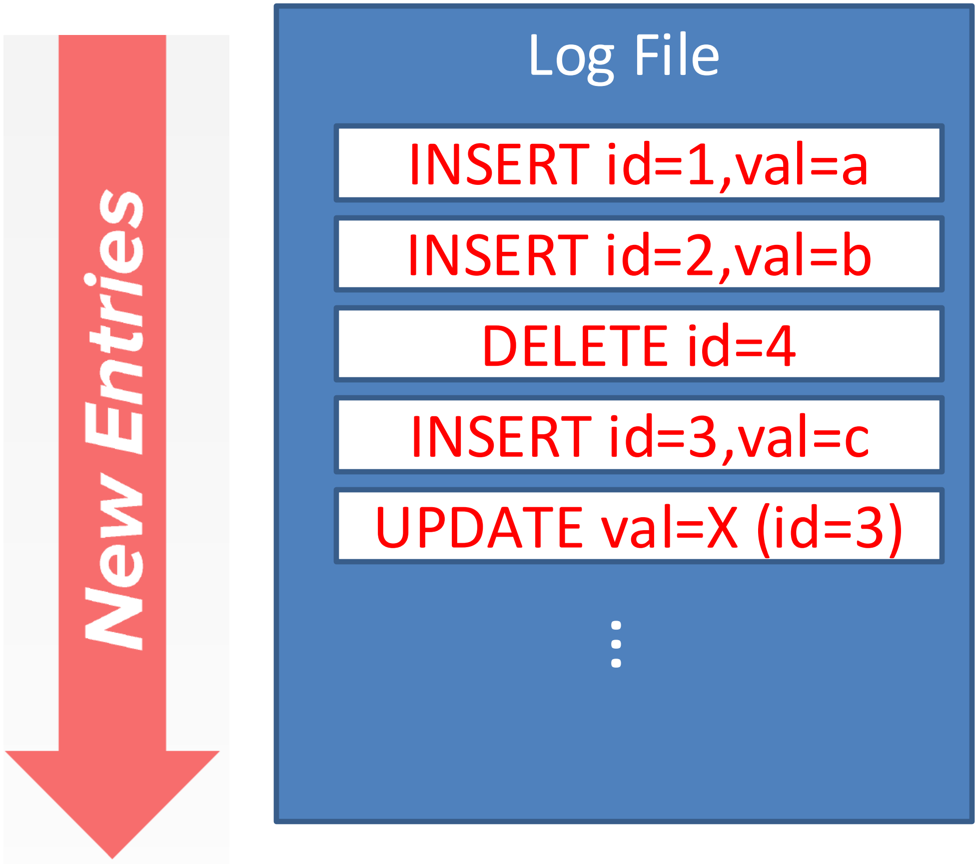
Log-structured files

Instead of storing tuples in pages, the DBMS only appends **log records**. Blocks are never modified.

For workloads with many small files, a traditional file system needs many small synchronous random writes, whereas a log-structured file system does few large asynchronous sequential transfers.

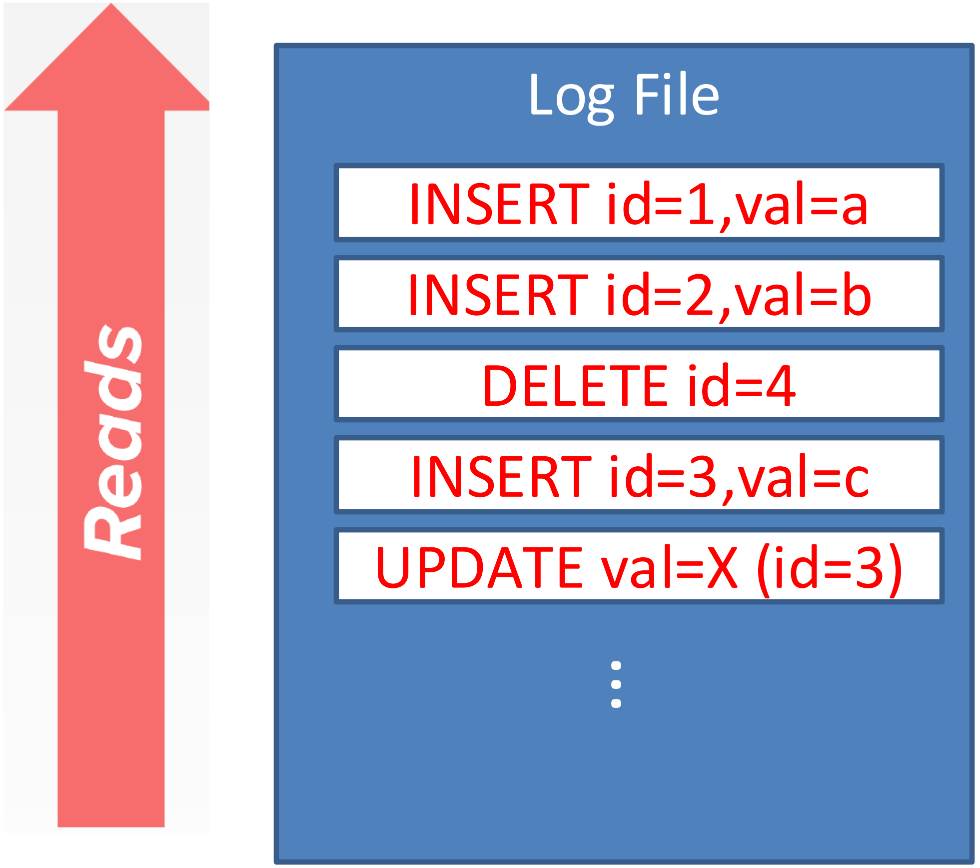
Writing to log-structured files

- Inserts: Store the entire tuple
- Deletes: Mark tuple as deleted
- Updates: Store delta of just the attributes that were modified



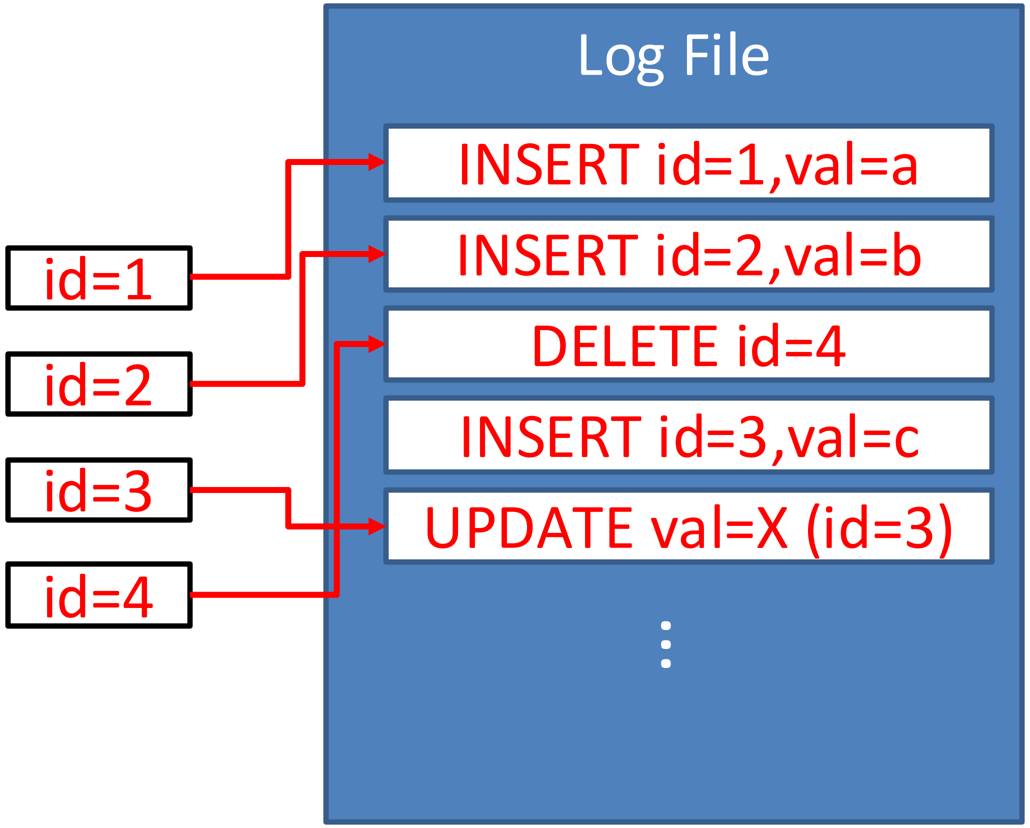
Reading from log-structured files

– DBMS scans log backwards, and "recreates" the tuple



Reading from log-structured files

- DBMS scans log backwards, and “recreates” the tuple
- Build indexes to allow **jumps** in the log
- Periodically compact the log



Net-net of log-structured files

- Advantages
  - High performance for inserts, deletes and updates
  - Ultra-fast recovery from failures
  - Good for SSD as writes are naturally leveled
- Disadvantages
  - Unpredictable performance in sequential reads
  - Need a lot of free space
  - Affects garbage collection (need for compaction)
  - Data can be lost if written but not checkpointed
- DBMS needs to address two issues
  - How to reconstruct tuples from logs efficiently
  - How to manage disk space with ever-growing logs

Storage Management: Outline

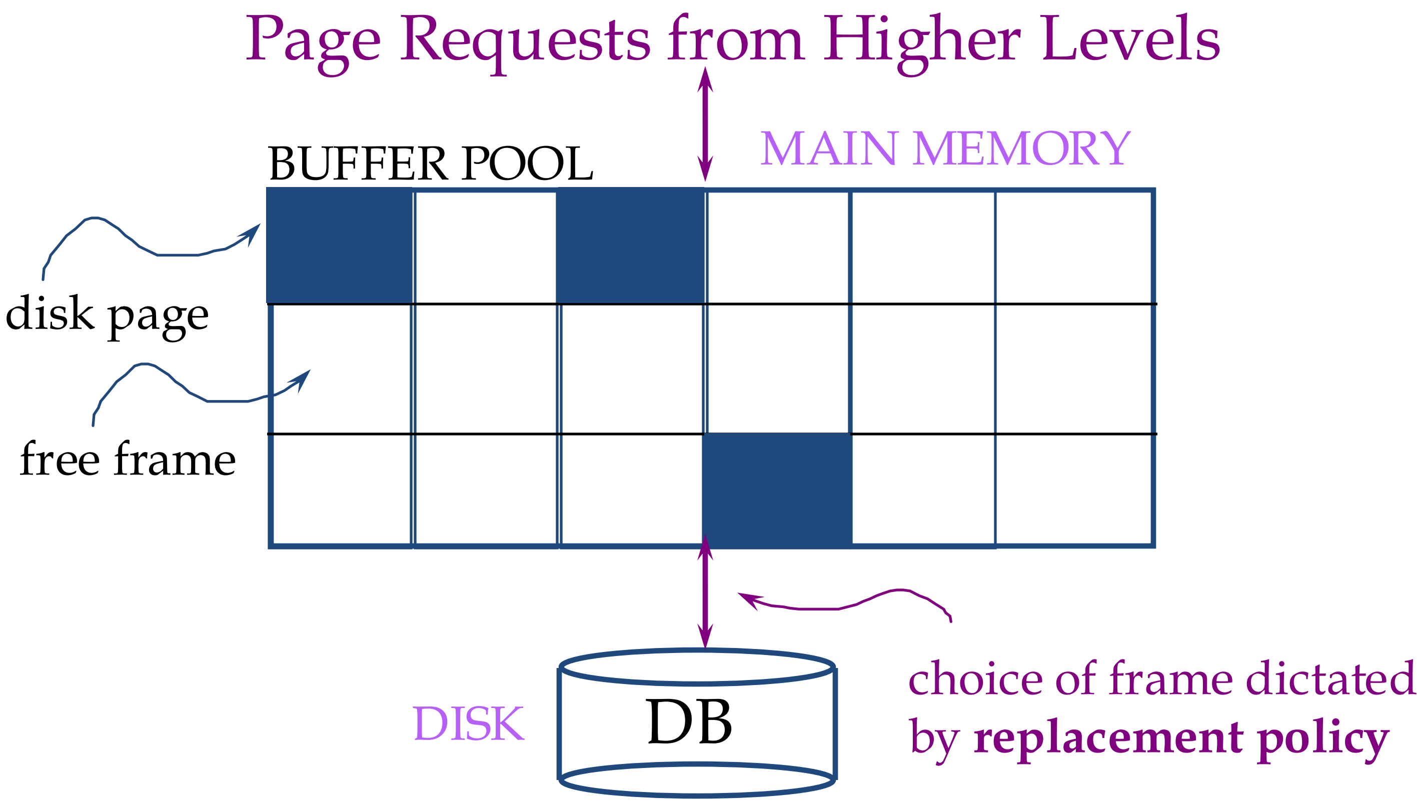
- Storage Technologies
- File Storage
- Buffer Management (refresher)
- Page Layout
  - NSM, aka row store
  - DSM, aka column store
  - PAX, hybrid

Can’t we just use the OS buffering?

- Layers of abstraction are good … but:
  - Unfortunately, OS often gets in the way of DBMS

- DBMS needs to do things “its own way”
  - Specialized prefetching
  - Control over buffer replacement policy
    - LRU not always best (sometimes worst!!)
  - Control over thread/process scheduling
    - “Convoy problem”
      - Arises when OS scheduling conflicts with DBMS locking
  - Control over flushing data to disk
    - WAL protocol requires flushing log entries to disk

Buffer Management in a DBMS



- *Data must be in RAM for DBMS to operate on it!*
- *Buffer manager hides the fact that not all data is in RAM (just like hardware cache policies hide the fact that not all data is in the caches)*

When a Page is Requested ...

- Buffer pool information table contains:
  <frame#, pageid, pin_count, dirty>

- If requested page is not in pool:
  - Choose a frame for *replacement*
    (only un-pinned pages are candidates)
  - If frame is "dirty", write it to disk
  - Read requested page into chosen frame
- *Pin* the page and return its address.

* *If requests can be predicted (e.g., sequential scans) pages can be pre-fetched several pages at a time!*

More on Buffer Management

- Requester of page must unpin it, and indicate whether page has been modified:
  - *dirty* bit is used for this.
- Page in pool may be requested many times,
  - a *pin count* is used. A page is a candidate for replacement iff *pin count* = 0 (*"unpinned"*)
- CC & recovery may entail additional I/O when a frame is chosen for replacement

Buffer Replacement Policy

- Frame is chosen for replacement by a *replacement policy:*
  - Least-recently-used (LRU), MRU, Clock, etc.

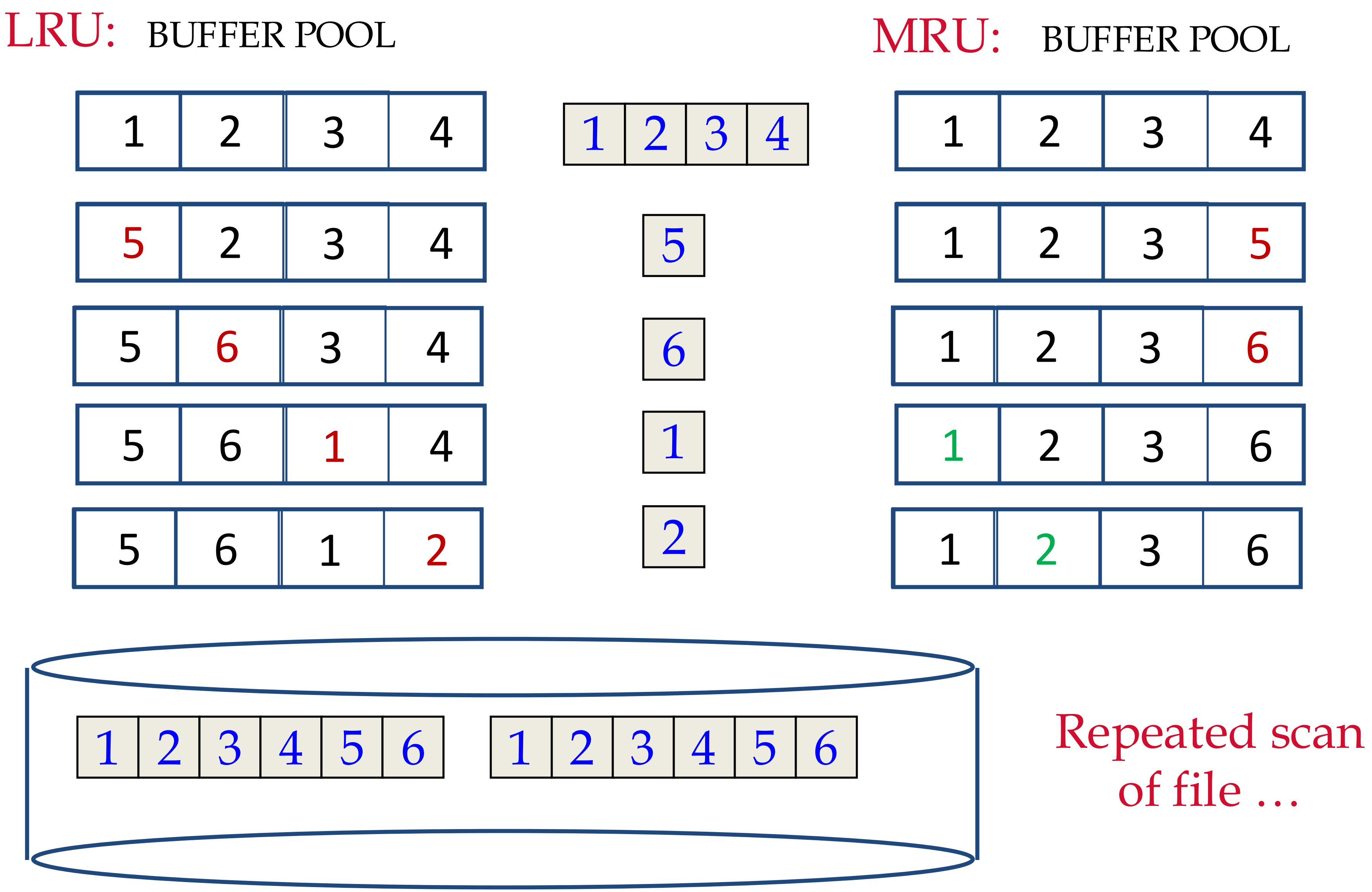
- Policy can have big impact on # of I/O's; depends on the *access pattern*.

LRU Replacement Policy

- ***Least Recently Used (LRU)***
  - for each page in buffer pool, keep track of time last *unpinned*
  - replace the frame which has the oldest (earliest) time
  - very common policy: intuitive and simple

- ***Problem: Sequential flooding***
  - LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O. *MRU* much better in this situation (but not in all situations, of course).

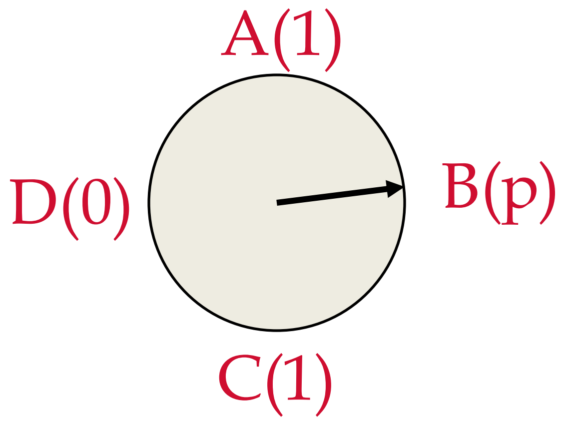
Sequential Flooding – Illustration



"Clock" Replacement Policy

- An approximation of LRU.
- Arrange frames into a cycle, store one "reference bit" per frame
- When pin count goes to 0, reference bit set on.
- When replacement necessary:

```
do {
    if (pincount == 0 && ref bit is off)
        choose current page for replacement;
    else if (pincount == 0 && ref bit is on)
        turn off ref bit;
    advance current frame;
} until a page is chosen for replacement;
```

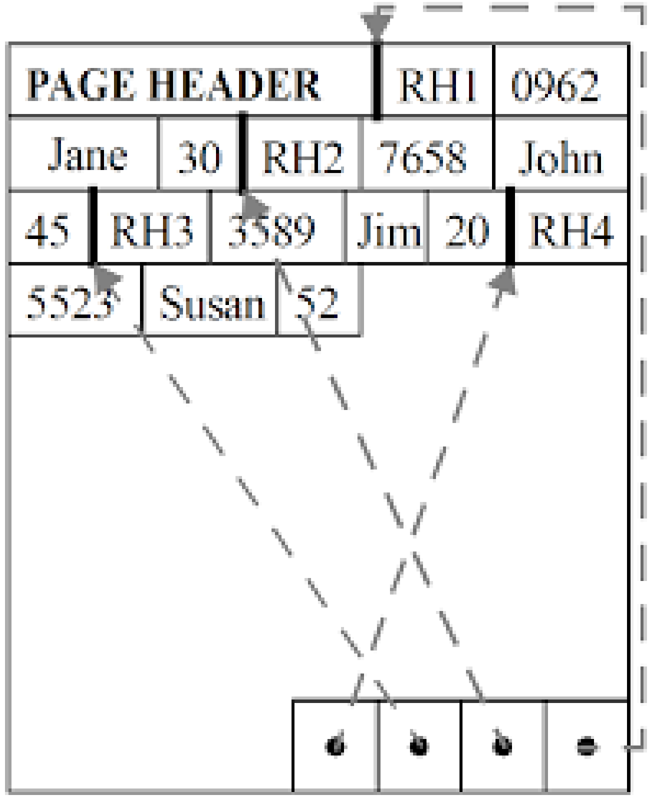


Storage Management: Outline

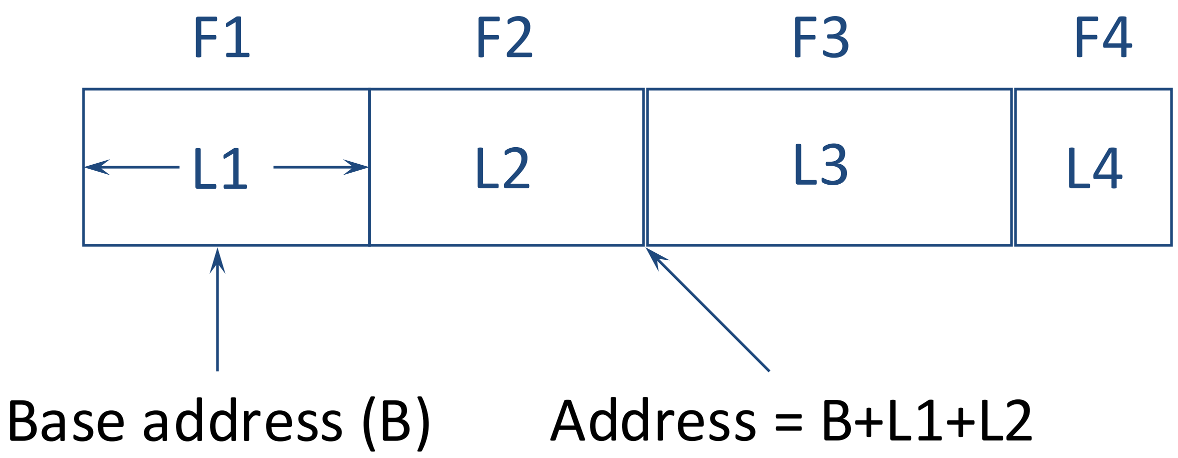
- Storage Technologies
- File Storage
- Buffer Management (refresher, slides on moodle)
- **Page Layout**
  - **NSM, aka row store**
  - DSM, aka column store
  - PAX, hybrid

The N-ary Storage Model

- Page = collection of slots
- Each slot stores one record
  - Record identifier: <page_id, slot_number>
  - Option 2: <uniq> -> <page_id, slot_number>
- Page format should support
  - Fast searching, inserting, deleting
- *Page format* depends on record format
  - Fixed-Length
  - Variable-Length

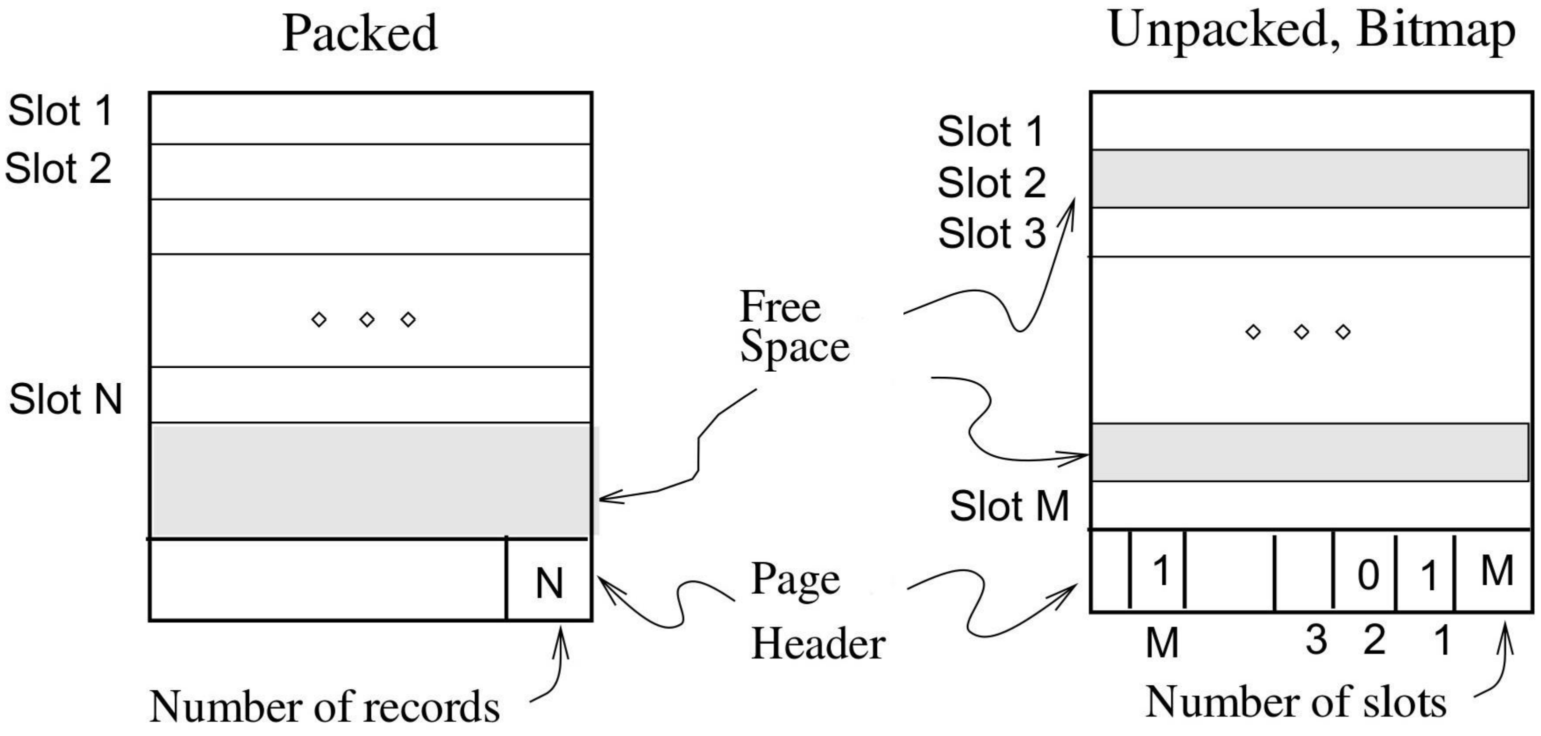


Record Formats: Fixed-Length



- Schema is stored in ***system catalog***
  - Number of fields is fixed for all records of a table
  - Domain is fixed for all records of a table
- Each field has fixed length
- Finding i^{th} field is done via arithmetic.

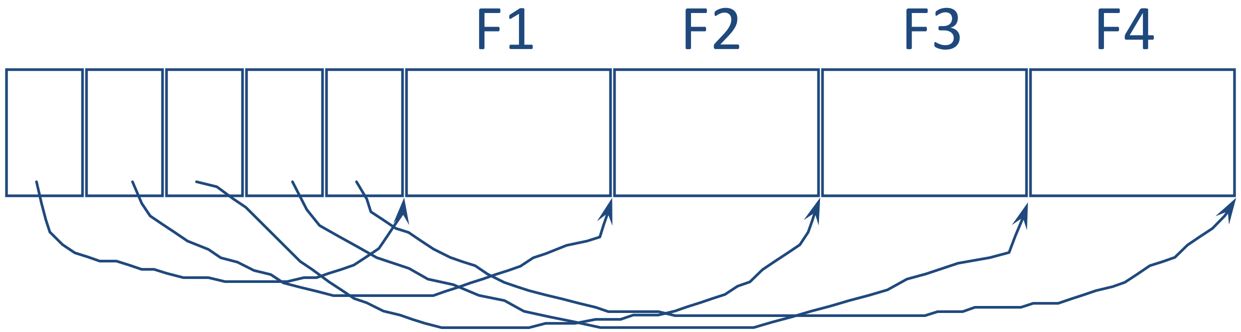
Page Format: Fixed-Length Records



- <u>*Record id*</u> *= <page id, slot #>*
- In the *packed* case, moving records for free space management changes rid; maybe unacceptable.

Record Formats: Variable-Length

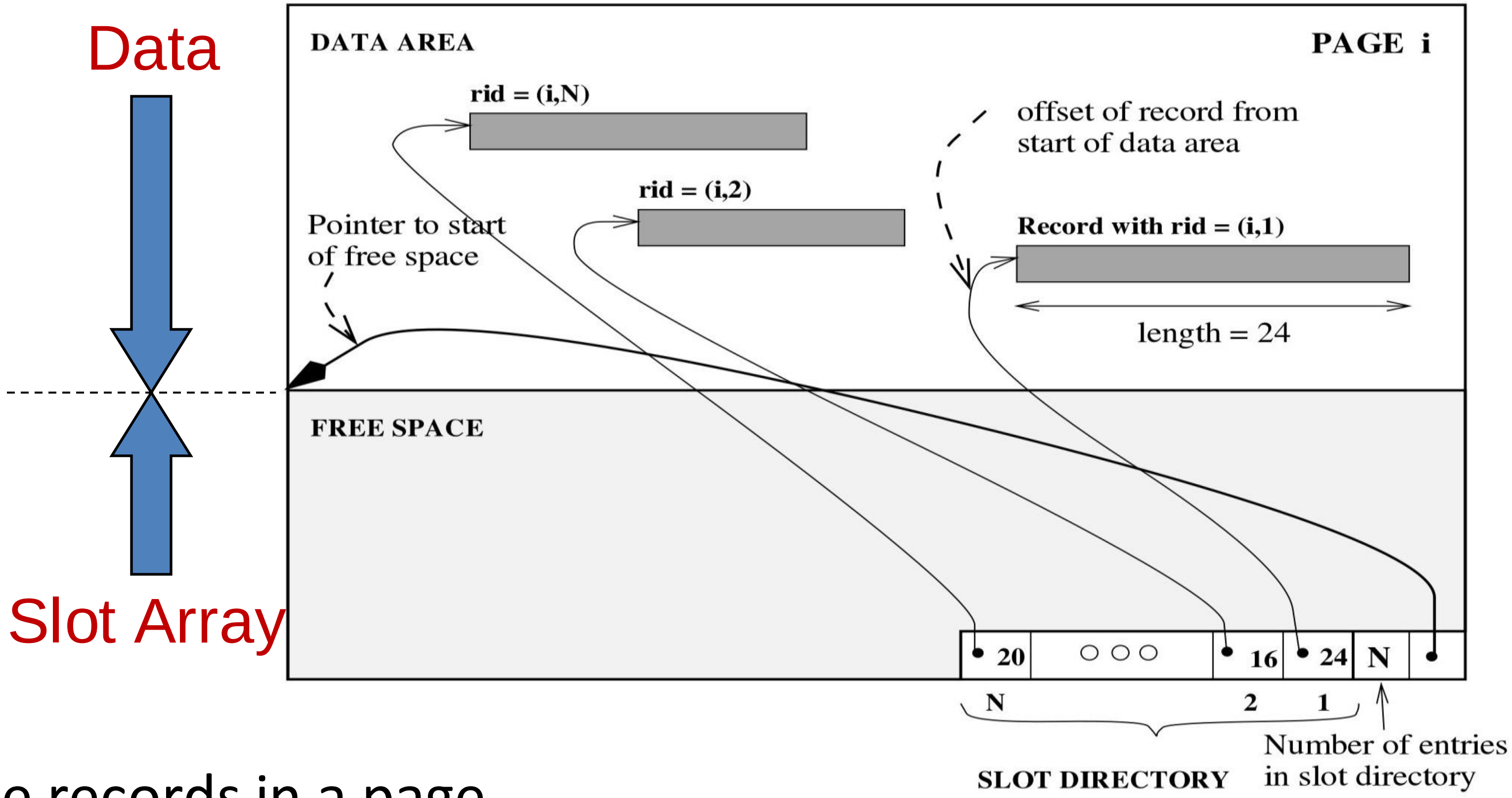
Fields Delimited by Special Symbols



Array of Field Offsets

- Array of field offsets is typically superior
  - Direct access to fields
  - Clean way of handling NULL values

Page Format: Variable-Length Records



- Need to move records in a page
  - Allocation/deletion must find/release free space
- Maintain slot directory with <record offset, record length> pairs
  - Records can move on page without changing rid
  - Useful for freely moving fixed-length records (ex: sorting)

Variable-Length Records: Issues

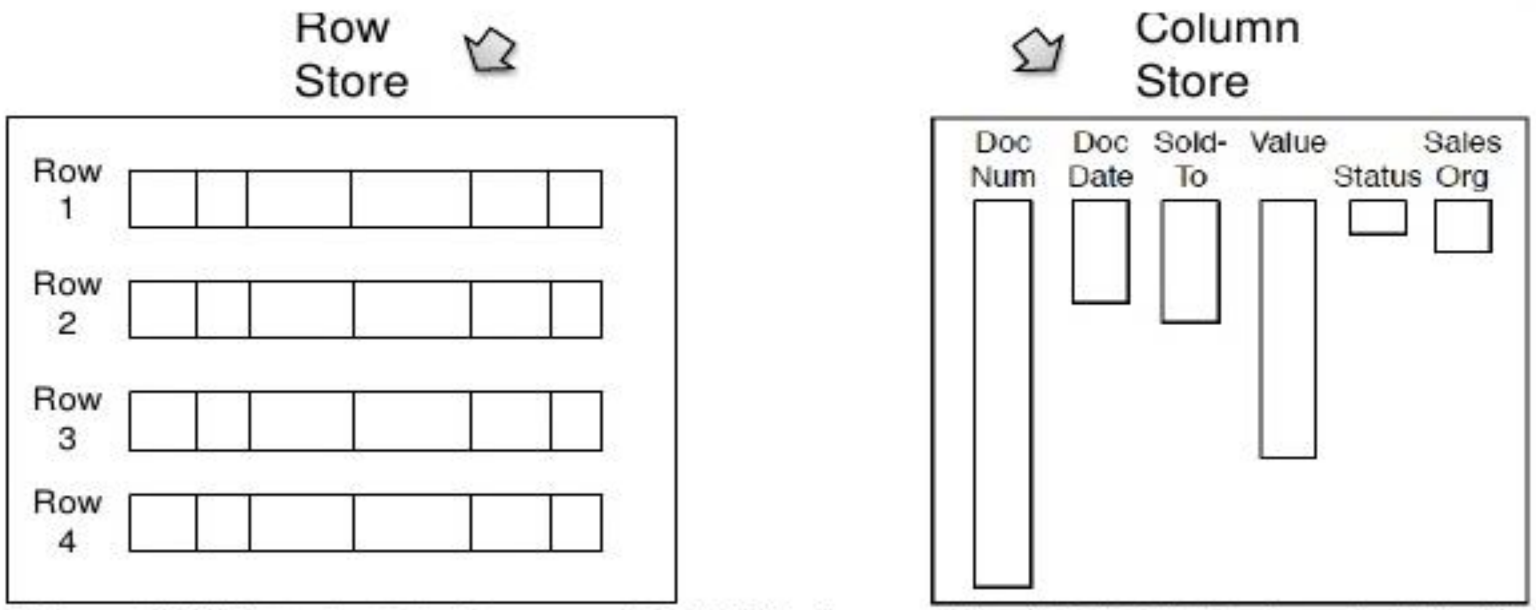
- If a field grows and no longer fits?
  - shift all subsequent fields
- If record no longer fits in page?
  - Move a record to another page after modification
- What if record size > page size?
  - Limit allowed record size

Storage Management: Outline

- Storage Technologies
- File Storage
- Buffer Management (refresher)
- **Page Layout**
  - NSM, aka row store
  - **DSM, aka column store**
  - PAX, hybrid

Decomposition Storage Model (DSM)

Document Number	Document Date	Sold-To Party	Order Value	Status	Sales Organization	...
95769214	2009-10-01	584	10.24	CLOSED	Germany Frankfurt	...
95769215	2009-10-01	1215	124.35	CLOSED	Germany Berlin	...
95779216	2009-10-21	584	47.11	OPEN	Germany Berlin	...
95779217	2009-10-21	454	21.20	OPEN	Germany Frankfurt	...



In-Memory/HANA Enterprise Data Management | SAP UA Conference | March 23rd 2012 | Dr. Alexander Zeier, MIT

DSM Page Format

R1

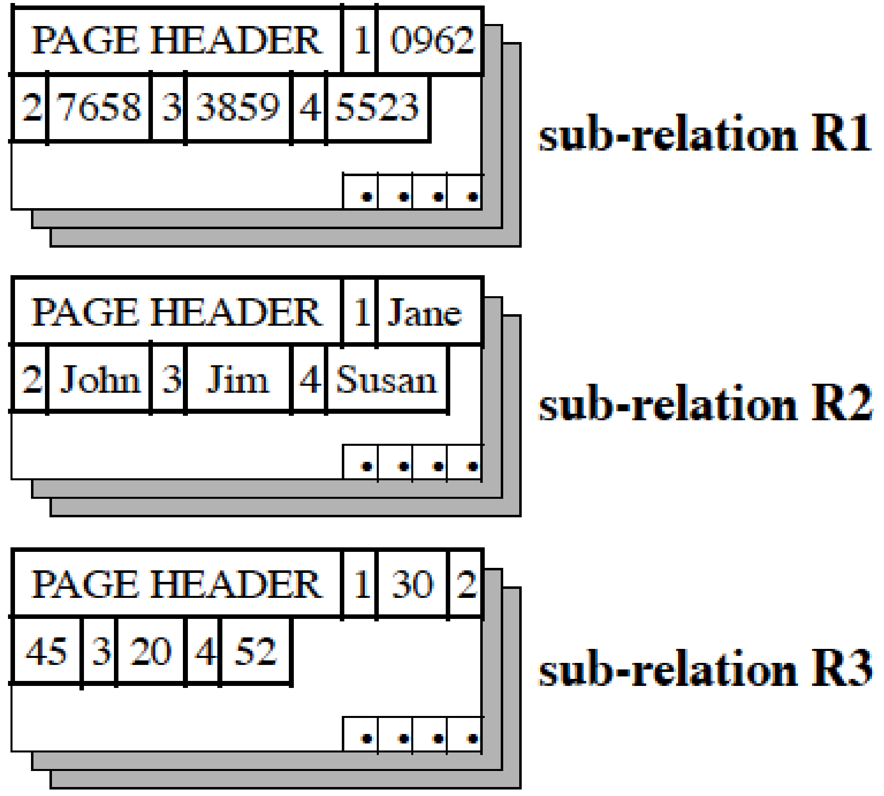
RID	SSN
1	0962
2	7658
3	3859
4	5523
5	9743
6	0618

R2

RID	Name
1	Jane
2	John
3	Jim
4	Susan
5	Leon
6	Dan

R3

RID	Age
1	30
2	45
3	20
4	52
5	43
6	37



Decompose a relational table to sub-tables per attribute

Column store (DSM): example

- Columns stored in pages
  - Denoted with different colors
- Each column can be accessed individually
  - Pages loaded only for the desired attributes

tbl1

Name	Age	Dept
John	22	HR
Jack	19	HR
Jane	37	IT
George	43	FIN
Wolf	51	IT
Maria	23	HR
Andy	56	FIN
Ross	22	SALES
Jack	63	FIN

Three different files: tbl1.name tbl1.age tbl1.dept

Column store (DSM) Properties

Pros

- Saves IO by bringing only the relevant attributes
- (Very) memory- compressing columns is typically easier

Cons

- Writes more expensive
- Need tuple stitching at some point (Tuple Reconstruction)
- Indexed selection with low selectivities
- Queries that require all or most of the attributes

Compression

- Lossless compression
- IO reduction implies less CPU wait time
  - Introduces small additional CPU load on otherwise idle CPU
- Run-length encoding (RLE): a lossless compression algorithm
  - sequences of redundant data are stored as a single data value

Dept
HR
HR
SALES
IT
IT

CDept
(2 x HR)
(1 x SALES)
(2 x IT)

Compression (2)

- Bit-vector encoding: compact and constant-time test
  - Useful when we have categorical data & Useful when a few distinct values
  - One bit vector for each distinct value
  - Vector length = # distinct elements

Dept
HR
HR
SALES
IT
IT

HR	SALES	IT
1	0	0
1	0	0
0	1	0
0	0	1
0	0	1

Compression (3)

- Dictionary encoding
  - Replace long values (e.g., strings) with integers

Dept
HR
IT
HR
SALES
HR
FINANCE
FINANCE
IT

Dictionary	
1	HR
2	IT
3	SALES
4	FINANCE

CDept
1
2
1
3
1
4
4
2

Compression (4)

- Frequency partitioning
  - Reorganize each column to reduce entropy at each page

Dept
HR
IT
FIN
FIN
HR
HR
FIN
HR
SALES

Column reorganization →

Dept	
1	HR
5	HR
6	HR
8	HR
2	IT
3	FIN
4	FIN
7	FIN
9	SALES

Dictionary-based compression → with **per-page** dictionaries

CDept	
1	1
5	1
6	1
8	1
2	1
3	2
4	2
7	2
9	1

Smaller dictionaries improve	- memory requirements - cache utilization - effectiveness of run-length encoding

Operators over compressed data

No need to decompress for most query operators

- Dictionary encoding => integer comparisons faster than string comparisons

```
SELECT name FROM tbl WHERE DEPT="HR"
vs
SELECT name FROM tbl WHERE CDEPT=1
```

  - Per-page dictionaries?

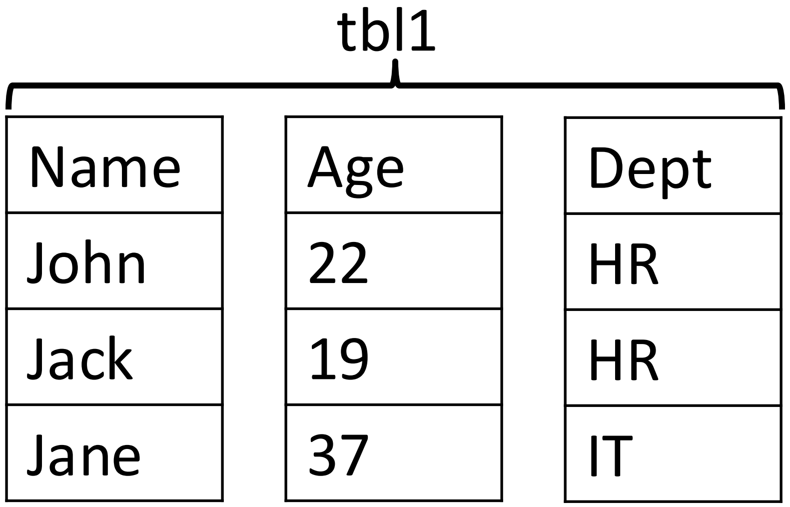
- Bit-vector encoding => find the 1's directly from the bit vectors

```
SELECT COUNT(*) FROM tbl WHERE CDEPT="HR"
```

- Run-length encoding => batch processing (aggregation)

DSM: Writes

- Row insertions/deletions
  - Affects all columns
  - Multiple I/Os
  - Complicated transactions
- Deletes/updates: Implicit
  - Mark record as deleted!



- Massive data loading: Write-optimized storage (WOS)

Write-optimized storage

In-memory buffer (fixed-size)

Name	Age	Dept

Filesystem storage: **3 different files**, possibly compressed!

Name	Age	Dept
John	22	HR
Jack	19	HR
Jane	37	IT
Jake	43	FIN
Jill	24	IT
James	56	FIN
Jessica	34	IT

Flush out

Batch-loading:

- <Jill, 24, IT>
- <James, 56, FIN>
- <Jessica, 34, IT>

Write rows in-memory, flush columns to disk

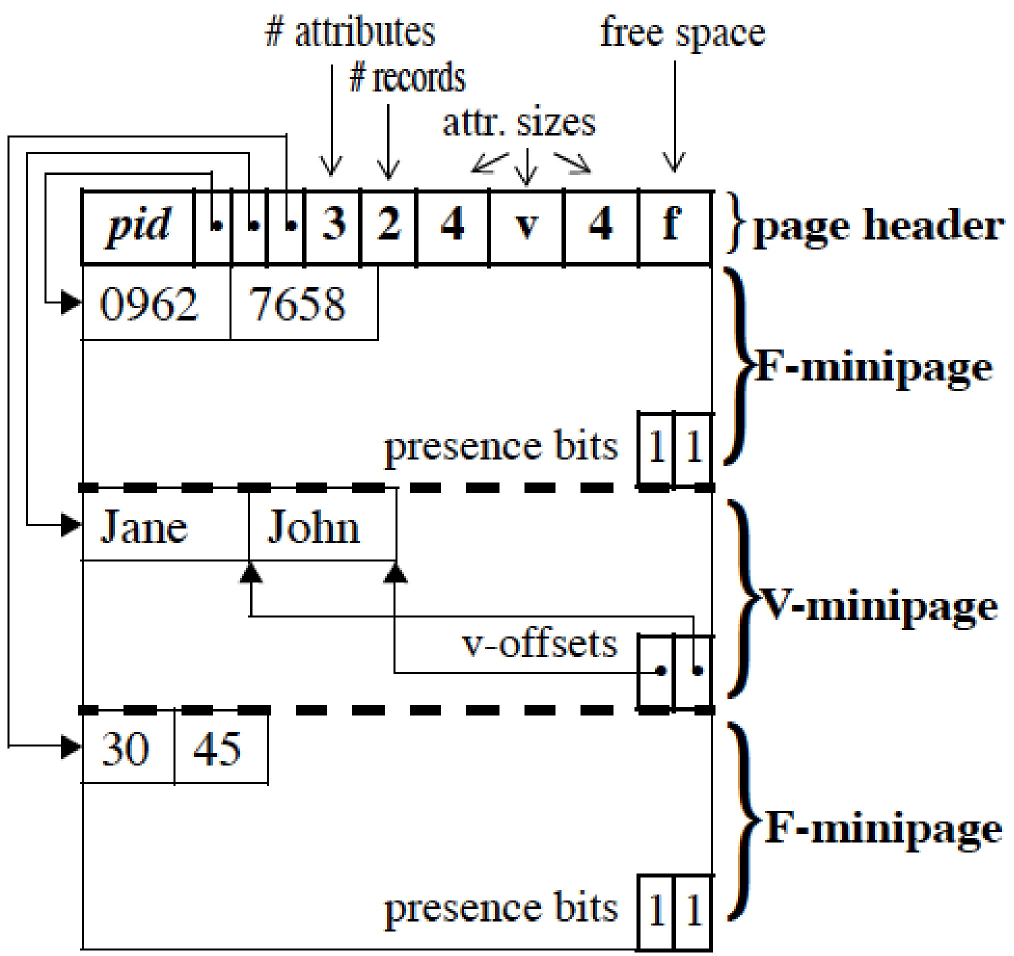
Storage Management: Outline

- Storage Technologies
- File Storage
- Buffer Management (refresher)
- **Page Layout**
  - NSM, aka row store
  - DSM, aka column store
  - **PAX, hybrid**

Partition Attributes Across (PAX)

Decompose a slotted-page internally in mini-pages per attribute

- ✓ Cache-friendly
- ✓ Compatible with slotted-pages
- ✓ Retain NSM I/O pattern
  - ✓ reduces column "stitching" delay
  - ✓ No per-column tuple ids
- ✓ Brings only relevant attributes to cache



PAX Americana

- DSM most suitable for analytical queries, but required major rewrites of existing DBMS, and penalized transactions **a lot**.

- PAX replaces NSM in-place
  - MonetDB/X100 (Vectorwise)
  - Oracle Exadata, Snowflake, Google Spanner, etc.
  - Data lake-oriented file formats
    - Parquet
    - Arrow
    - …

Conclusion

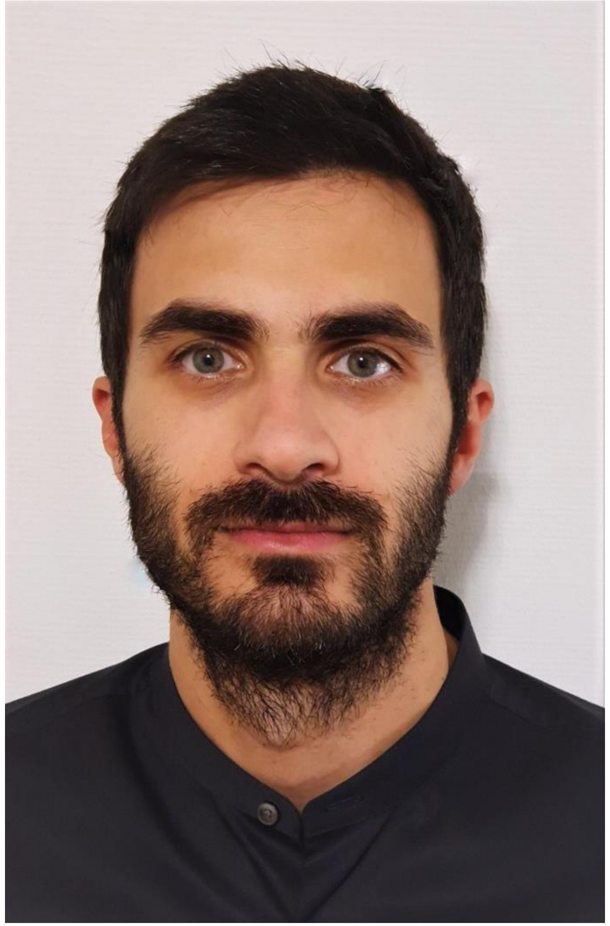
- One size does not fit all

Each storage technology favors a different storage layout
Different workloads require different storage layouts and data access methods

- To optimize use of resources and algorithms, we need to know the workload (unrealistic)

New way of building systems: JIT/code generation/virtualization

Next week



Dr. Angelos Anadiotis will lecture on Query Processing

Principal engineer at Oracle Zurich
Formerly professor at Ecole Polytechnique Paris

Reading material

- Row stores (material of CS300). Read one of:
  - COW Book. Chapters 7.3-7 & 8 (2nd ed) or Chapters 8 & 9.7-7 (3rd ed)
  - Database System Concepts, sixth edition. (Chapters 13.1-3, 13.5 + 14.1-9)
- D. Abadi et al.: The Design and Implementation of Modern column store Database Systems. Foundations and Trends in Databases, vol. 5, no. 3, **pp. 227-263 only**, 2013. Available online at: stratos.seas.harvard.edu/files/stratos/files/columnstoresfntdbs.pdf
- A. Ailamaki et al.: Weaving Relations for Cache Performance. VLDB 2001
- https://blog.twitter.com/engineering/en_us/a/2013/dremel-made-simple-with-parquet.html

Optional readings

- The remainder of: “The Design and Implementation of Modern column store Database Systems”
- I. Alagiannis, S. Idreos, A. Ailamaki: H2O: A hands-free adaptive store. SIGMOD’14. Available online at: http://dl.acm.org/citation.cfm?doid=2588555.2610502
- Joy Arulraj, Andrew Pavlo: How to Build a Non-Volatile Memory Database Management System. SIGMOD 2017, Tutorial