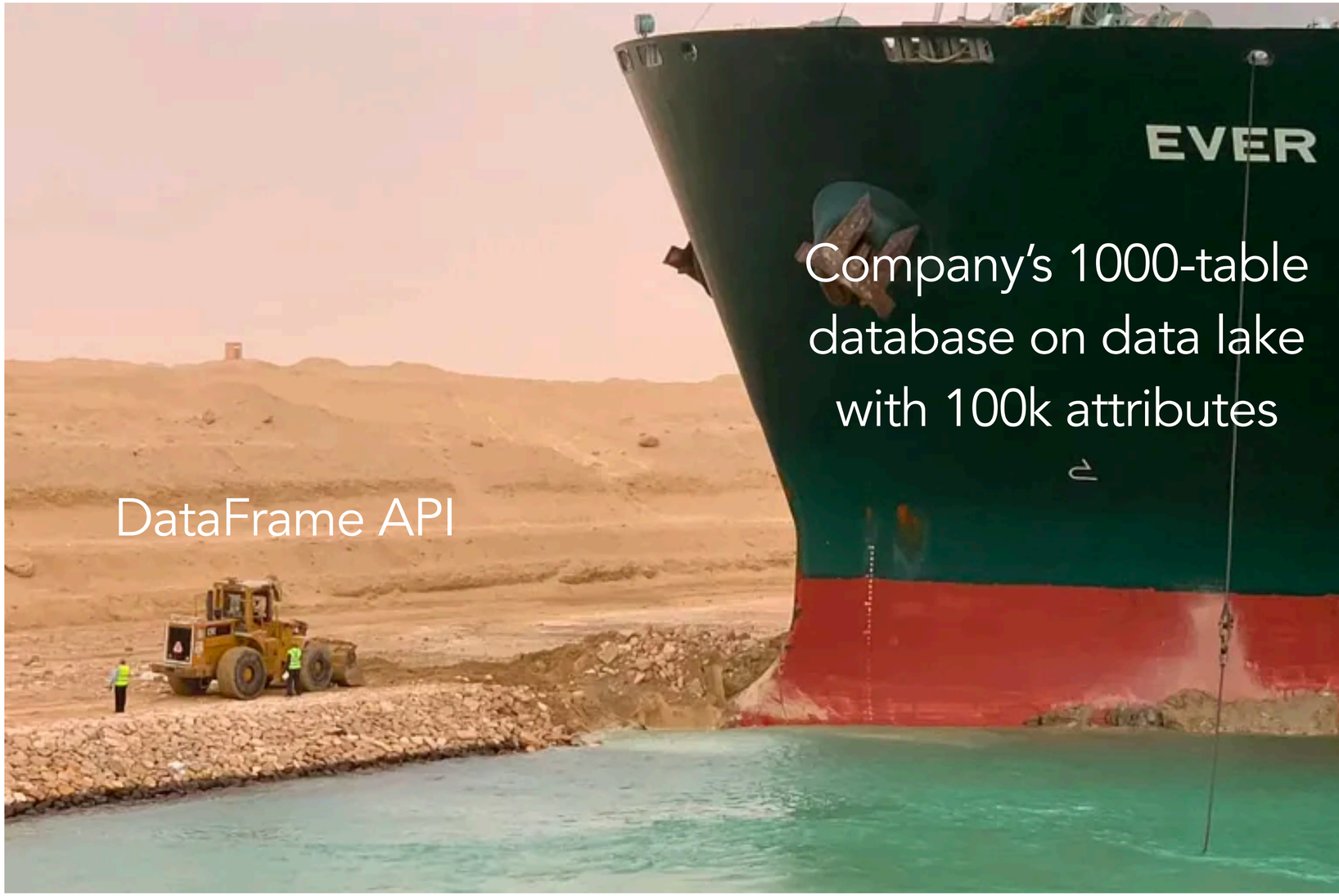
DSC 204a Scalable Data Systems

- Haojian Jin



Meme idea credit: https://datasystemsfun.tumblr.com/

Today's activities

1. Process.
2. PIA
3. Filesystems and Data Files

Where are we in the class?

Foundations of Data Systems

- Digital representation of Data → Computer Organization → Memory hierarchy → Process → **Storage**

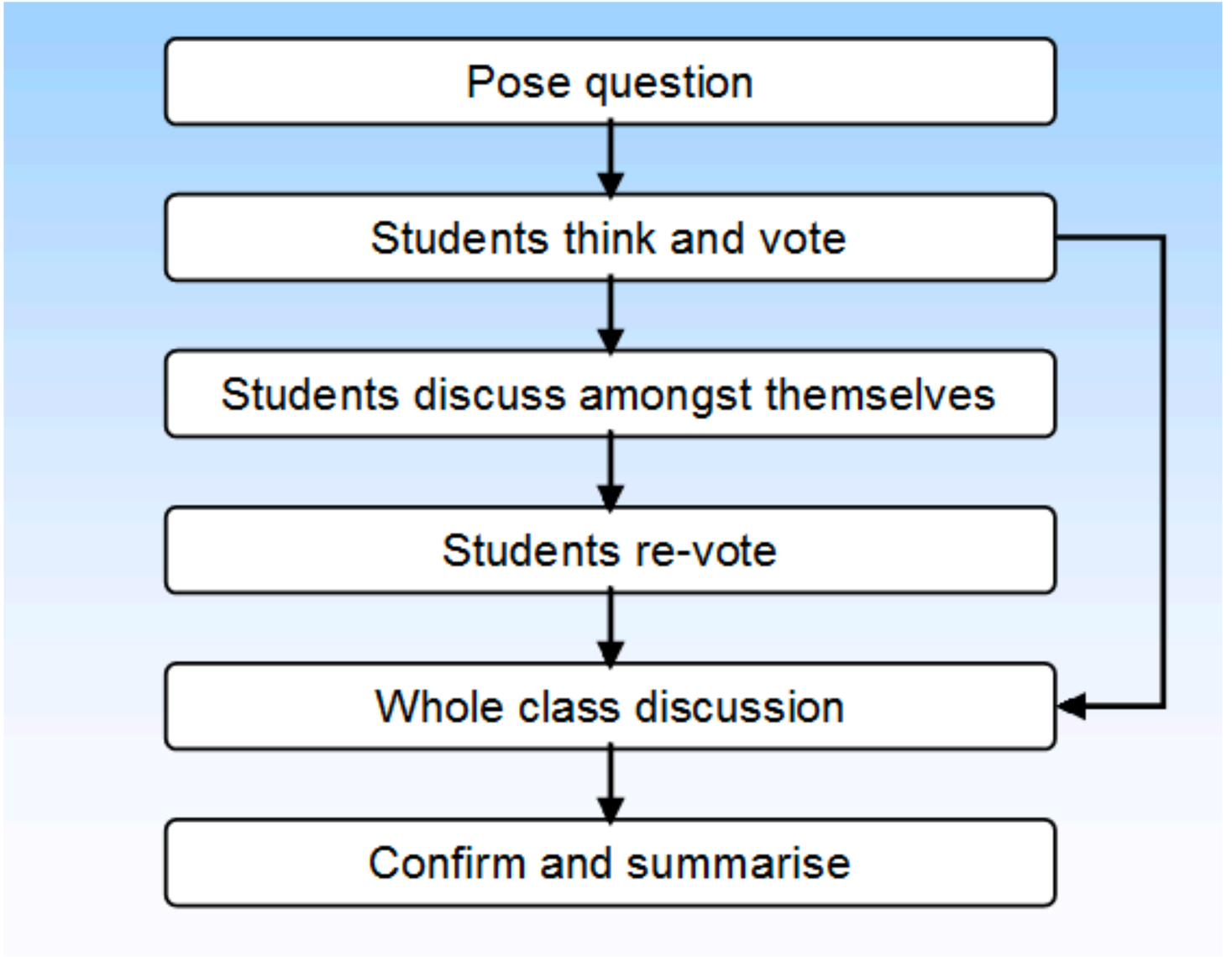
Scaling Distributed Systems

- Cloud → Distributed storage → Partition and replication (HDFS) → Distributed computation

Data Processing and Programming model

- Data encoding evolution → IO & Unix Pipes → Batch processing (MapReduce) → Stream processing (Spark)

Peer instruction activity



Peer Instruction Activity

bit.ly/dsc204aApr14

Peer Instruction Activity (About 1min per 1pt)

Q1. **[2pts]** Which of these levels of the memory hierarchy typically has the lowest latency to read data from?

A. Flash SSD
B. Magnetic hard disk
C. DRAM
D. All have similar latency
E. None of the above

C

Peer Instruction Activity (About 1min per 1pt)

Q2. **[2pts]** Which of these levels of the memory hierarchy typically has the lowest latency to write data from?

A. Flash SSD
B. Magnetic hard disk
C. DRAM
D. All have similar latency
E. None of the above

C

Peer Instruction Activity (About 1min per 1pt)

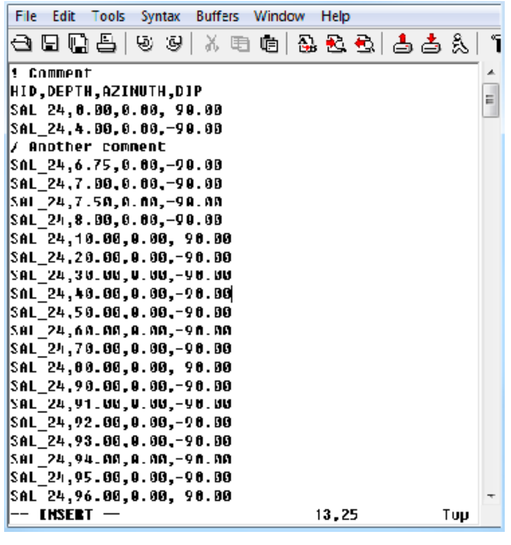
Q3. **[6pts]** Suppose you have a 750 x 250 matrix of 32-bit integers in Python memory. Its entries are known to be non-negative and < 10. You write it to disk as a CSV file with one row per line.

What is the rough maximum ratio of the size of your file on disk versus the size of the matrix as an object in DRAM?

The answer is one of the following. If you pick the right answer, you need not explain. If you pick the wrong answer and seek partial credits, please explain your answer succinctly and clearly.

A. 0.25 B. 0.5 C. 0.75
D. 1.0 E. 1.33 F. 1.5
G. 1.67 H. 2.0

Hints:



Peer Instruction Activity (About 1min per 1pt)

Q3. **[6pts]** Suppose you have a 750 x 250 matrix of 32-bit integers in Python memory. Its entries are known to be non-negative and < 10. You write it to disk as a CSV file with one row per line.

What is the rough maximum ratio of the size of your file on disk versus the size of the matrix as an object in DRAM?

The answer is one of the following. If you pick the right answer, you need not explain. If you pick the wrong answer and seek partial credits, please explain your answer succinctly and clearly.

A. 0.25	B. 0.5	C. 0.75
D. 1.0	E. 1.33	F. 1.5
G. 1.67	H. 2.0	

Final answer: B) 0.5
Total count of numbers = 750 * 250 = x, say (no need to calculate this!)
Size as matrix object in DRAM: x * 4B
A single digit integer is 1B as char
CSV means each number has "," after it, except for the last "\n"; so, 1B per number extra in the CSV file
So, size as file on disk is roughly x * 2B
Thus, the max ratio is x*2 / (x*4) = 0.5

Peer Instruction Activity (About 1min per 1pt)

Q4. **[3x6pts]** Here is a Gantt Chart for 3 processes of the given lengths that all arrive at time 0.

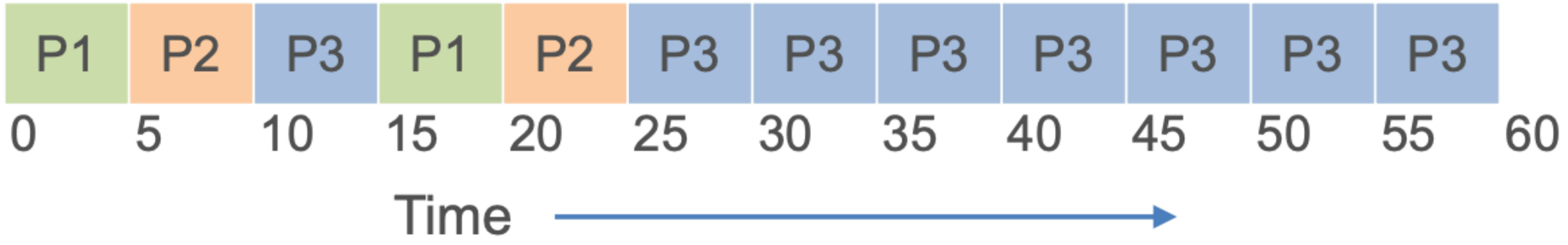
A) What is the rough *average response time*?

B) What is the rough *average turnaround time*?

A. 5

B. 35

P1, P2, and P3 are of lengths 10, 10, and 40 units, resp.



Peer Instruction Activity (About 1min per 1pt)

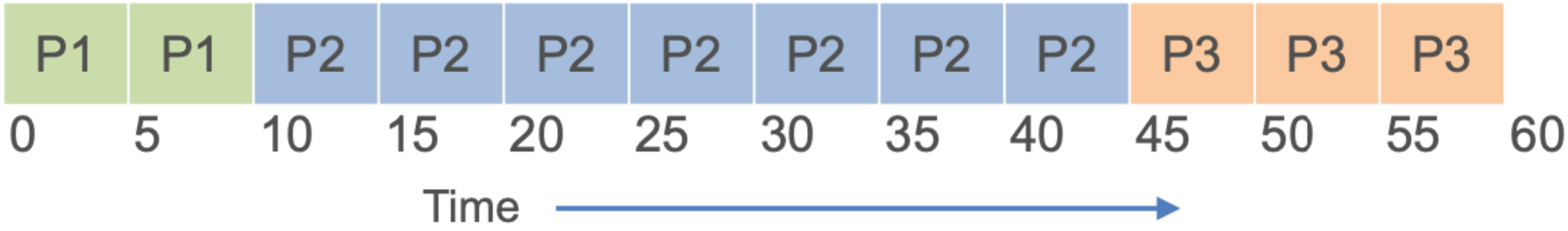
Q5. **(After class) [3x6pts]** Here is a Gantt Chart for 3 processes of the given lengths that arrive at the different times given.

A) What is the rough *average response time*?

B) What is the rough *average turnaround time*?

C) Which scheduling policy/policies discussed in class (FIFO, SJF,

P1, P2, and P3 are of lengths 10, 35, and 15 units, resp. and arrive at times 0, 10, and 20, resp.



Peer Instruction Activity (About 1min per 1pt)

Q6. **[3x2pts]** For each of the following answer True of False:

A) An OS typically has mechanisms to wrest control of hardware back from a user process.

B) SCTF is the fairest scheduling policy we discussed in class.

C) CPU caches are usually cheaper per MB than Flash SSD.

A. True
B. False
C. False

Peer Instruction Activity (About 1min per 1pt)

Q7.**(After class)** Suppose you are given that *n* processes of the same lengths (*k* units each) all arrive at roughly the same time. Answer these questions.

A. **[8pts]** Which scheduling policy/policies discussed in class (FIFO, SJF, SCTF, RR) will give the *lowest average turnaround time*? What is that lowest value?

B. **[8pts]** Which scheduling policy/policies discussed in class (FIFO, SJF, SCTF, RR) will give the *lowest average response time*? What is that lowest value?

C. **[4pts]** For what *k* will all 4 policies yield the same metrics?

Today

Process management: virtualization & Concurrency

Filesystems and Data Files

Main memory management

IO & Unix Pipes

Q: *What is a file?*


INVESTMENTS
SAVINGS

Abstractions: File and Directory

- **File:** A persistent sequence of bytes that stores a logically coherent digital object for an application
  - **File Format:** An application-specific standard that dictates how to interpret and process a file's bytes
  - 100s of file formats exist (e.g., TXT, DOC, GIF, MPEG); varying data models/types, domain-specific, etc.
  - **Metadata:** Summary or organizing info. about file content (aka *payload*) stored with file itself; format-dependent
- **Directory:** A cataloging structure with a list of references to files and/or (recursively) other directories
  - Typically treated as a special kind of file
  - Sub dir., Parent dir., Root dir.

Filesystem

- **Filesystem:** The part of OS that helps programs create, manage, and delete files on disk (sec. storage)
- Roughly split into *logical level* and *physical level*
  - Logical level exposes file and dir. abstractions and offers System Call APIs for file handling
  - Physical level works with disk firmware and moves bytes to/from disk to DRAM

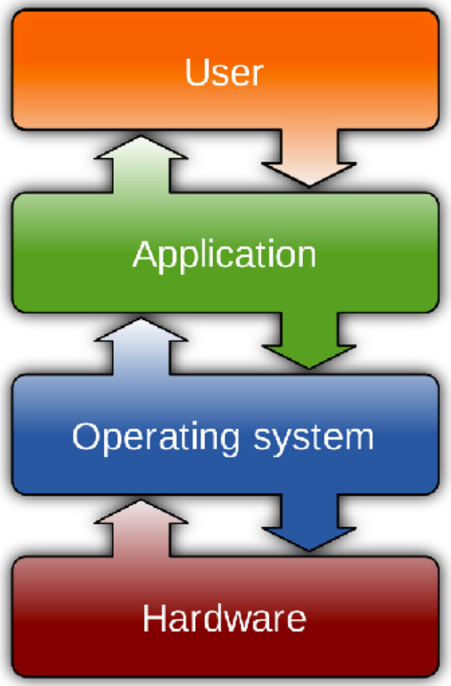
Filesystem

- Dozens of filesystems exist, e.g., ext2, ext3, NTFS, etc.
  - Differ on how they layer file and dir. abstractions as bytes, what metadata is stored, etc.
  - Differ on how data integrity/reliability is assured, support for editing/ resizing, compression/encryption, etc.
  - Some can work with ("**mounted**" by) multiple OSs

Virtualization of File on Disk

- OS abstracts a file on disk as a virtual object for processes
- **File Descriptor:** An OS-assigned +ve integer identifier/reference for a file's virtual object that a process can use
  - 0/1/2 reserved for STDIN/STDOUT/STDERR
  - **File Handle:** A PL's abstraction on top of a file descr. (fd)

System Call API for File Handling:



API of OS called “System Calls”

- **open**(): Create a file; assign fd; optionally overwrite
- **read**(): Copy file’s bytes on disk to in-mem. buffer; sized
- **write**(): Copy bytes from in-mem. buffer to file on disk
- **fsync**(): “Flush” (force write) “dirty” data to disk
- **close**(): Free up the fd and other OS state info on it
- **lseek**(): Position offset in file’s fd (for random R/W later)
- Dozens more (rename, mkdir, chmod, etc.)

Q: *What is a database? How is it different from just a bunch of files?*

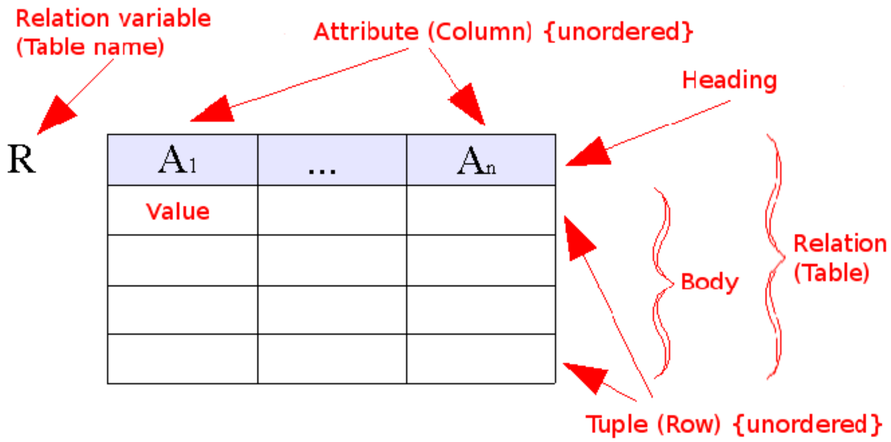
Files Vs Databases: Data Model

- **Database:** An *organized* collection of interrelated data
  - **Data Model:** An abstract model to define organization of data in a formal (mathematically precise) way
  - E.g., Relations, XML, Matrices, DataFrames
- Every database is just an *abstraction* on top of data files!
  - **Logical level:** Data model for higher-level reasoning
  - **Physical level:** How bytes are layered on top of files
  - All data systems (RDBMSs, Dask, Spark, TensorFlow, etc.) are application/platform software that use OS System Call API for handling data files

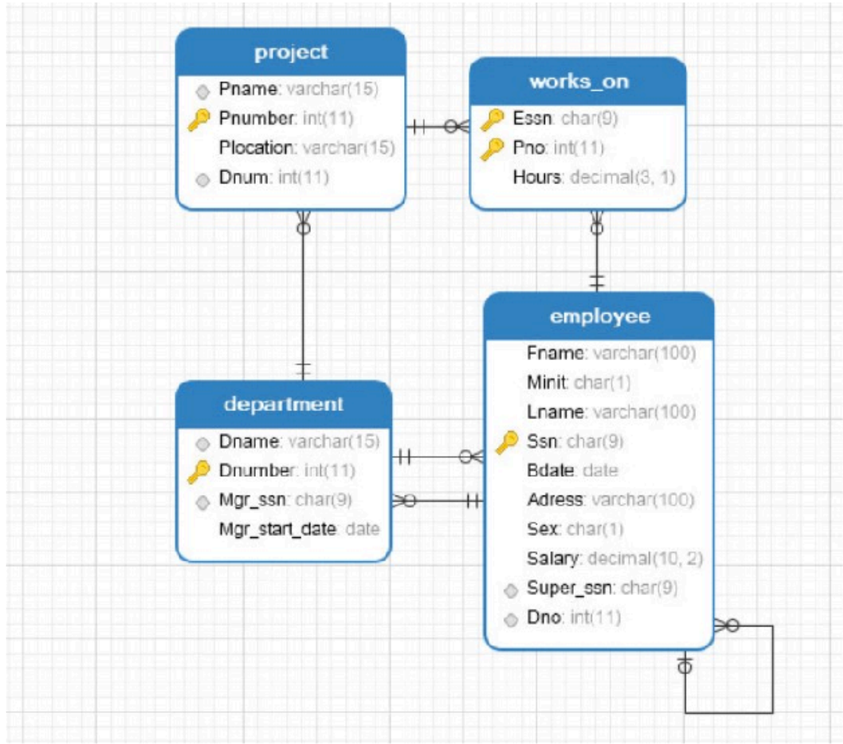
Data as File: Structured

- **Structured Data:** A form of data with regular substructure

Relation



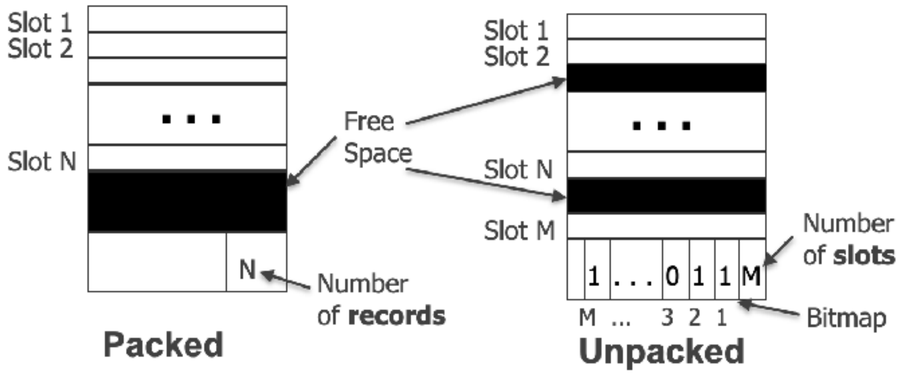
Relational Database



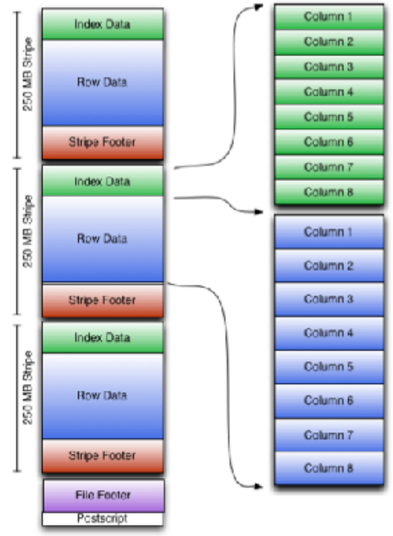
- Most RDBMSs and Spark serialize a relation as *binary* file(s), often compressed

Aside: Relational File Formats

- Different RDBMSs and Spark/HDFS-based tools serialize relation/tabular data in different binary formats, often compressed
  - One file per relation; *data layout* can be row vs columnar (e.g., ORC, Parquet) vs hybrid formats
  - RDBMS vendor-specific vs open Apache
  - Parquet becoming especially popular



Row-oriented



Columnar

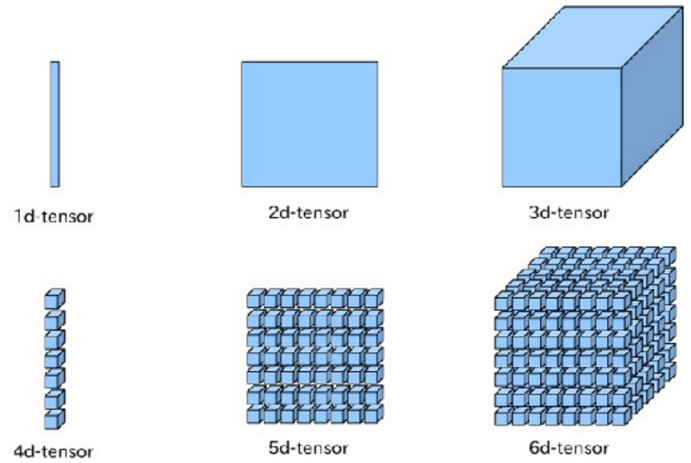
Data as File: Structured

- **Structured Data:** A form of data with regular substructure

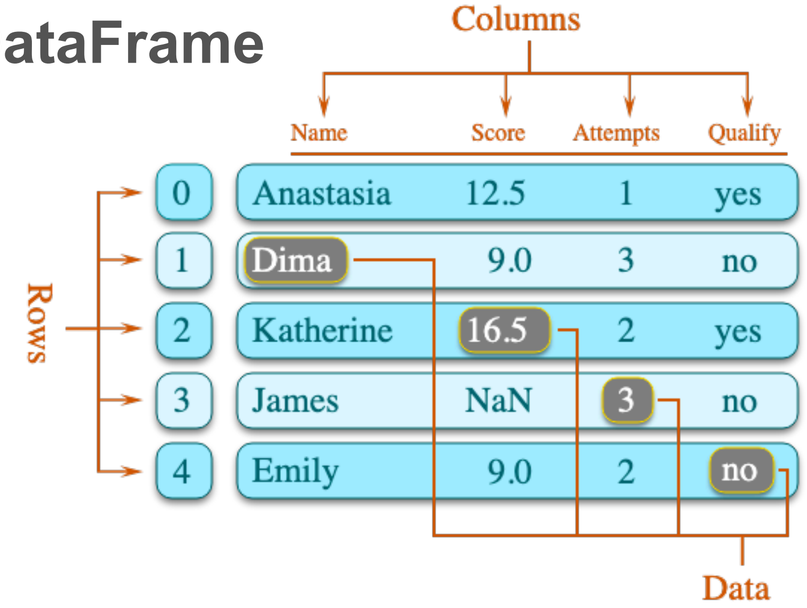
Matrix

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \end{array}$$

Tensor



DataFrame

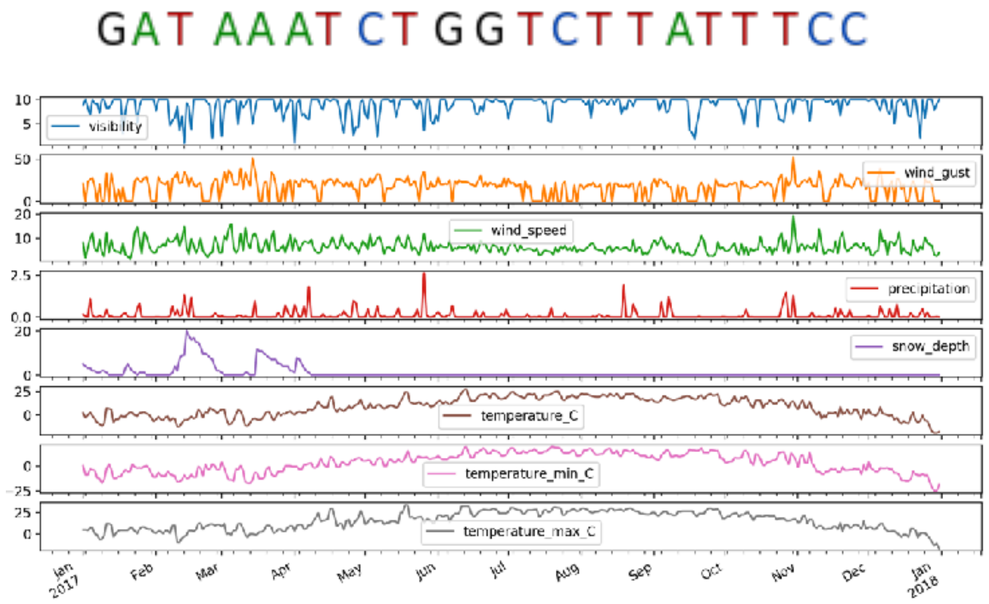


- Typically serialized as restricted ASCII text file (TSV, CSV, etc.)
- Matrix/tensor as binary too
- Can layer on Relations too!

Data as File: Structured

- **Structured Data:** A form of data with regular substructure

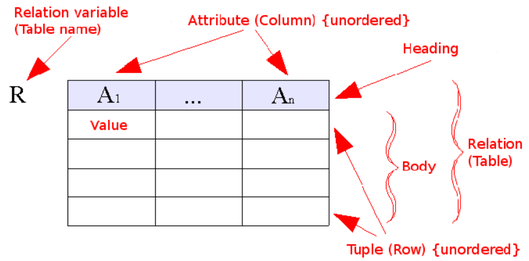
Sequence (Includes Time-series)

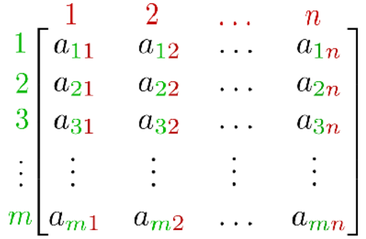


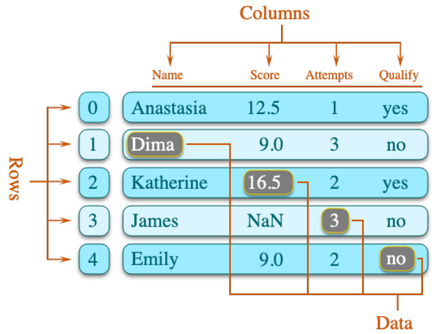
- Can layer on Relations, Matrices, or DataFrames, or be treated as first-class data model
- Inherits flexibility in file formats (text, binary, etc.)

Comparing Struct. Data Models

***Q:** What is the difference between Relation, Matrix, and DataFrame?*







- **Ordering:** Matrix and DataFrame have row/col numbers; Relation is orderless on both axes!
- **Schema Flexibility:** Matrix cells are numbers. Relation tuples conform to pre-defined schema. DataFrame has no pre-defined schema but all rows/ cols can have names; col cells can be mixed types!
- **Transpose:** Supported by Matrix & DataFrame, not Relation

If interested in reading more:
https://towardsdatascience.com/preventing-the-death-of-the-dataframe-8bca1c0f83c8

Data as File: Semistructured

- **Semistructured Data:** A form of data with less regular / more flexible substructure than structured data

Tree-Structured

```xml
<?xml version="1.0" encoding="UTF-8"?>
<customers>
    <customer>
        <customer_id>1</customer_id>
        <first_name>John</first_name>
        <last_name>Doe</last_name>
        <email>john.doe@example.com</email>
    </customer>
    <customer>
        <customer_id>2</customer_id>
        <first_name>Sam</first_name>
        <last_name>Smith</last_name>
        <email>sam.smith@example.com</email>
    </customer>
    <customer>
        <customer_id>3</customer_id>
        <first_name>Jane</first_name>
        <last_name>Doe</last_name>
        <email>jane.doe@example.com</email>
    </customer>
</customers>
```

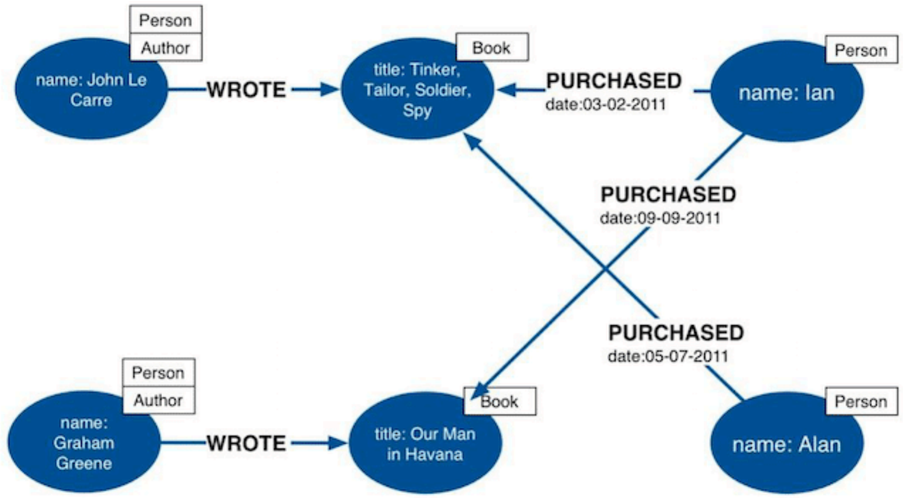
```
[
    {
        orderId: 1,
        date: '1/1/2014',
        orderItems: [
                {itemId: 1, qty: 3, price: 23.4},
                {itemId: 23, qty: 2, price: 3.3},
                {itemId: 7, qty: 5, price: 5.3}
            ]
    },
    {
        orderId: 2,
        date: '1/2/2014',
        orderItems: [
            {itemId: 31, qty: 7, price: 3.8},
            {itemId: 17, qty: 4, price: 9.2}
        ]
    },
    {
        orderId: 3,
        date: '1/5/2014',
        orderItems: [
            {itemId: 11, qty: 9, price: 13.3},
            {itemId: 27, qty: 2, price: 19.2},
            {itemId: 6, qty: 19, price: 3.6},
            {itemId: 7, qty: 22, price: 9.1}
        ]
    }
]
```

- Typically serialized as restricted ASCII text file (extensions XML, JSON, YML, etc.)
- Some data systems also offer binary file formats
- Can layer on Relations too

Data as File: Semistructured

- **Semistructured Data:** A form of data with less regular / more flexible substructure than structured data

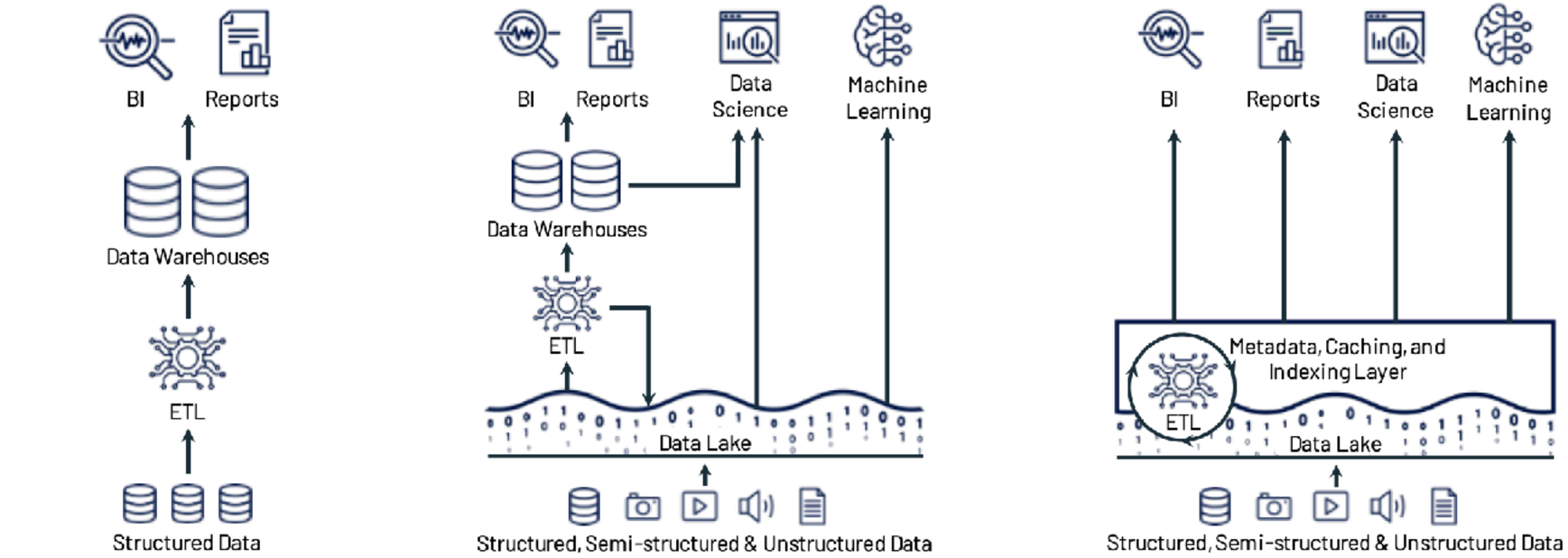
Graph-Structured



- Typically serialized with JSON or similar textual formats
- Some data systems also offer binary file formats
- Again, can layer on Relations too

Data Files on Data "Lakes"

- **Data "Lake":** *Loose coupling* of data file format for storage and data/query processing stack (vs RDBMS's tight coupling)
  - JSON for raw data; Parquet processed is common



(a) First-generation platforms. (b) Current two-tier architectures. (c) Lakehouse platforms.

If interested, check out this vision paper on the future of data lakes:
http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

Data Lake File Format Tradeoffs

- Pros and cons of Parquet vs text-based files (CSV, JSON, etc.):
  - **Less storage**: Parquet stores in **compressed** form; can be much smaller (even 10x); less I/O to read
  - **Column pruning**: Enables app to read only columns needed to DRAM; even less I/O now!
  - **Schema on file**: Rich metadata, stats inside format itself
  - **Complex types**: Can store them in a column
  - **Human-readability**: Cannot open with text apps directly
  - **Mutability**: Parquet is immutable/read-only; no in-place edits
  - **Decompression/Deserialization overhead**: Depends on application tool; can go either way
  - **Adoption in practice**: CSV/JSON support more pervasive but Parquet is catching up

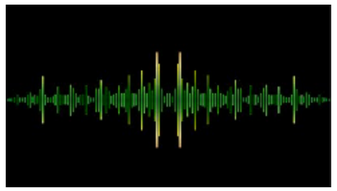
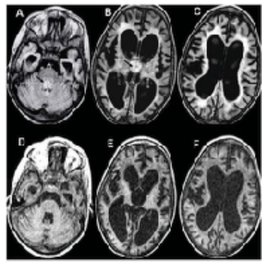
Data Lake File Format Tradeoffs

Dataset	Size on Amazon S3	Query Run Time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet Format	130 GB	6.78 seconds	2.51 GB	$0.01
Savings	87% less when using Parquet	34x faster	99% less data scanned	99.7% savings

https://databricks.com/glossary/what-is-parquet

Data as File: Other Common Formats

- **Machine Perception** data layer on tensors and/or time-series
- Myriad binary formats, typically with (lossy) compression, e.g., WAV for audio, MP4 for video, etc.

- **Text File** (aka plaintext): Human-readable ASCII characters
- **Docs/Multimodal File:** Myriad app-specific rich binary formats