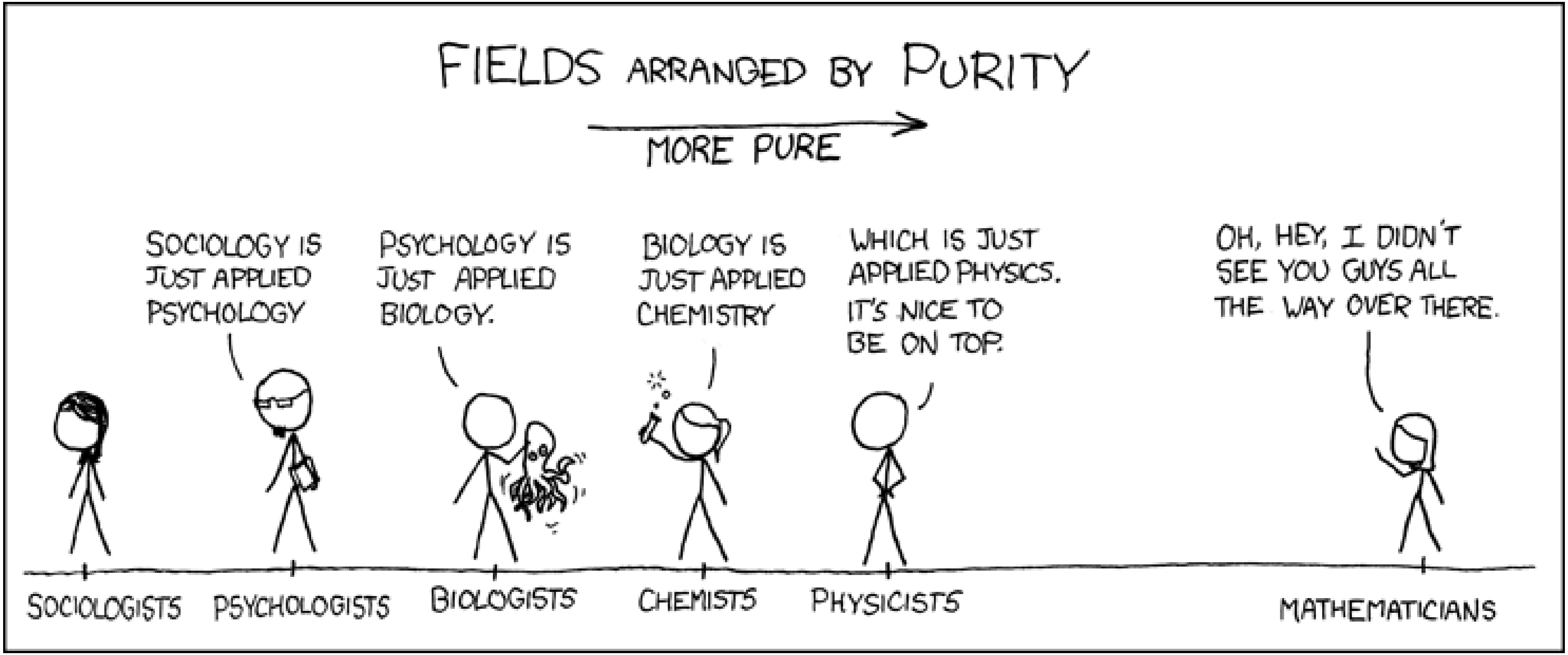
Introduction to Data Curation:

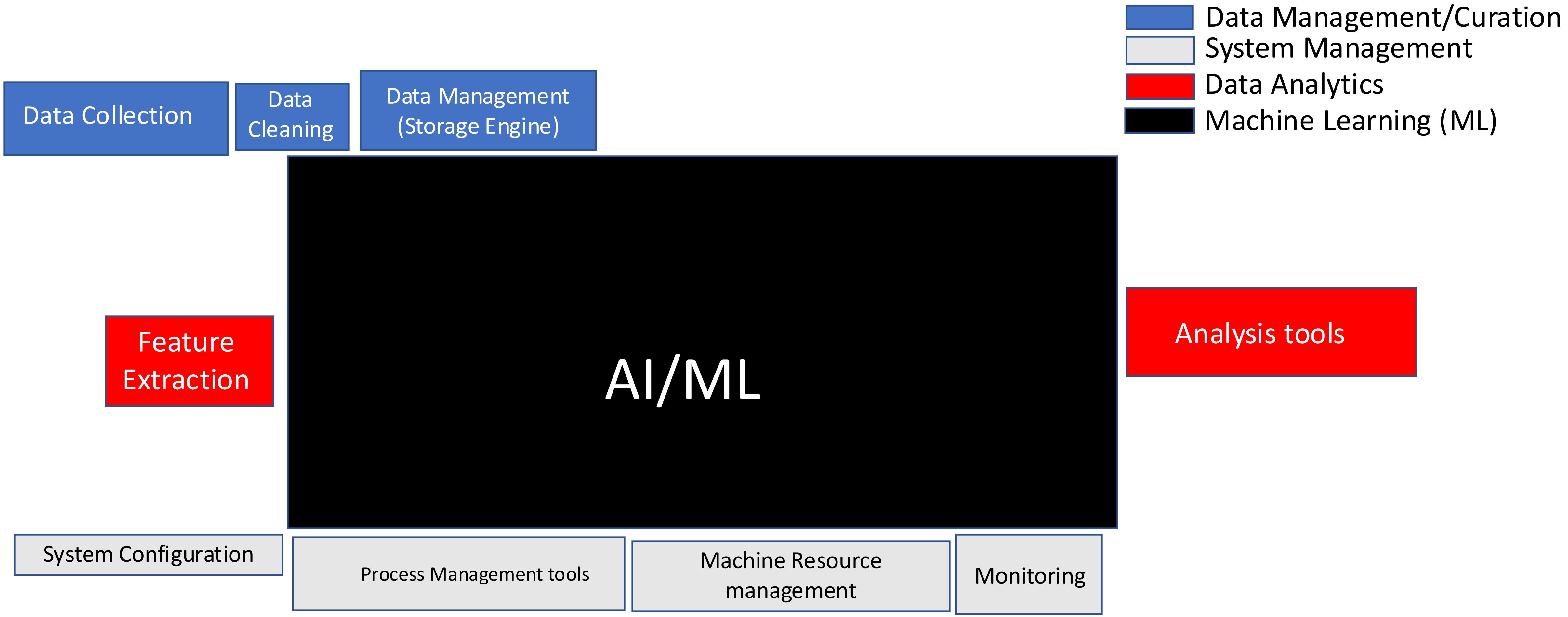
Databases, Data Management, Data Cleaning, ….

Tim Mattson



https://xkcd.com/435/

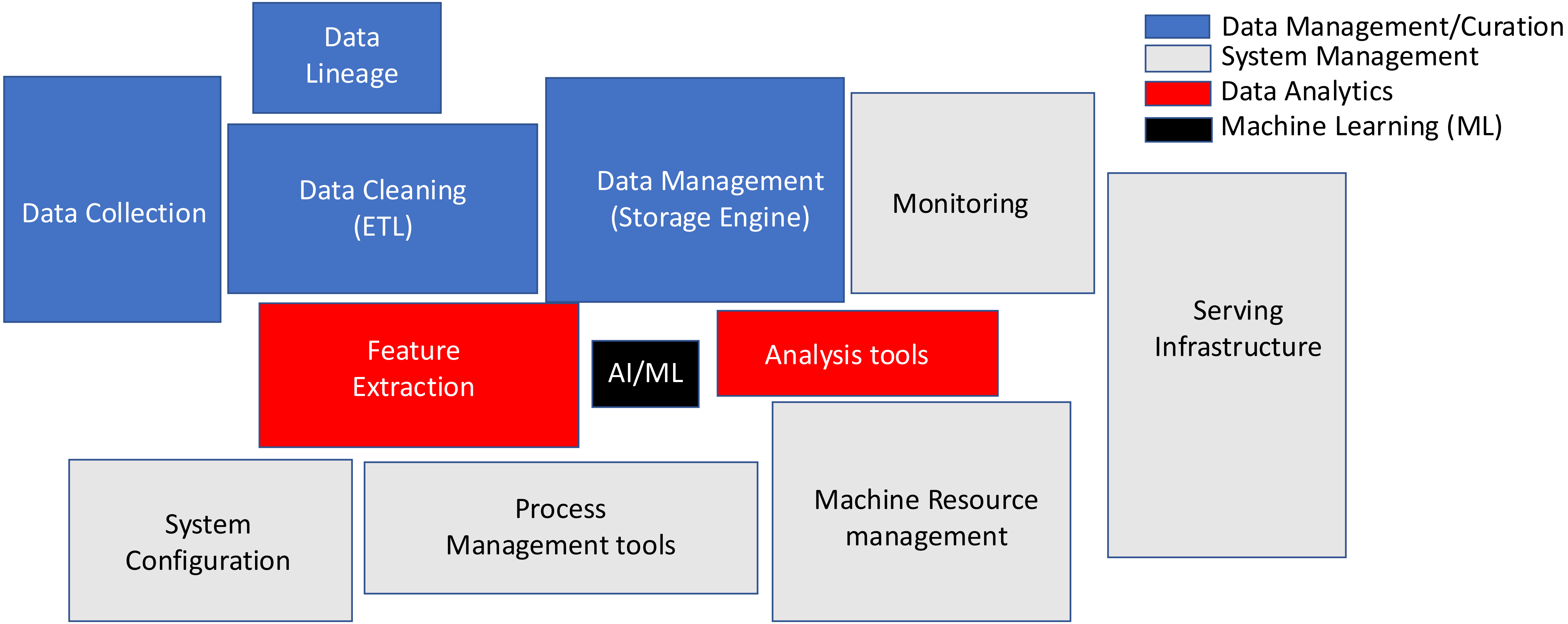
The hype suggests a world looks like this …



Area of rectangles approximates effort spent working in the indicated domain within a data sciences workflow

AI: Artificial Intelligence. ML: Machine Learning

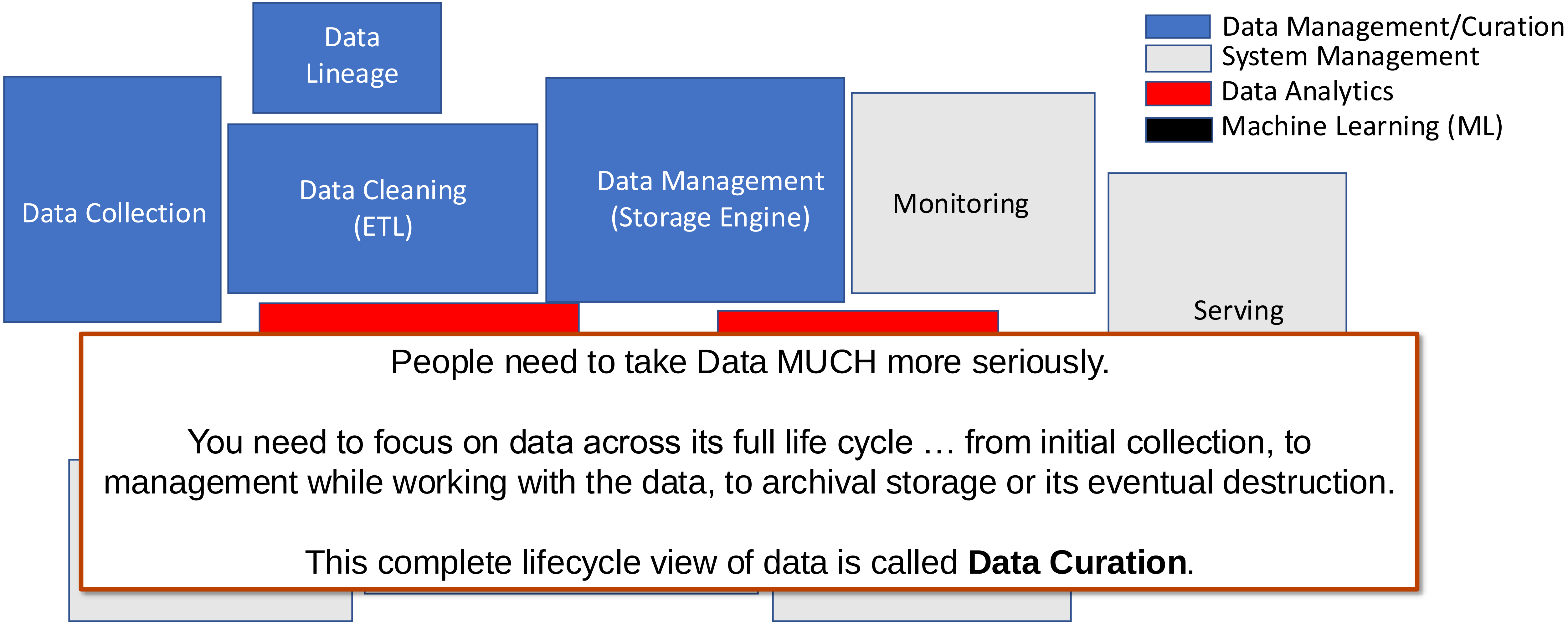
AI/ML developers see the world like this …



AI/ML code is 2 to 5 percent of the code written by a data scientist*. Most of the code is "glue code" to manage data and system components

*based on figure 1 from "Hidden Technical Debt in Machine Learning Systems", by D. Sculley et. al. from Google.

Users of AI/ML see the world like this …



People need to take Data MUCH more seriously.

You need to focus on data across its full life cycle … from initial collection, to management while working with the data, to archival storage or its eventual destruction.

This complete lifecycle view of data is called **Data Curation**.

AI/ML code is 2 to 5 percent of the code written by a data scientist*. Most of the code is “glue code” to manage data and system components

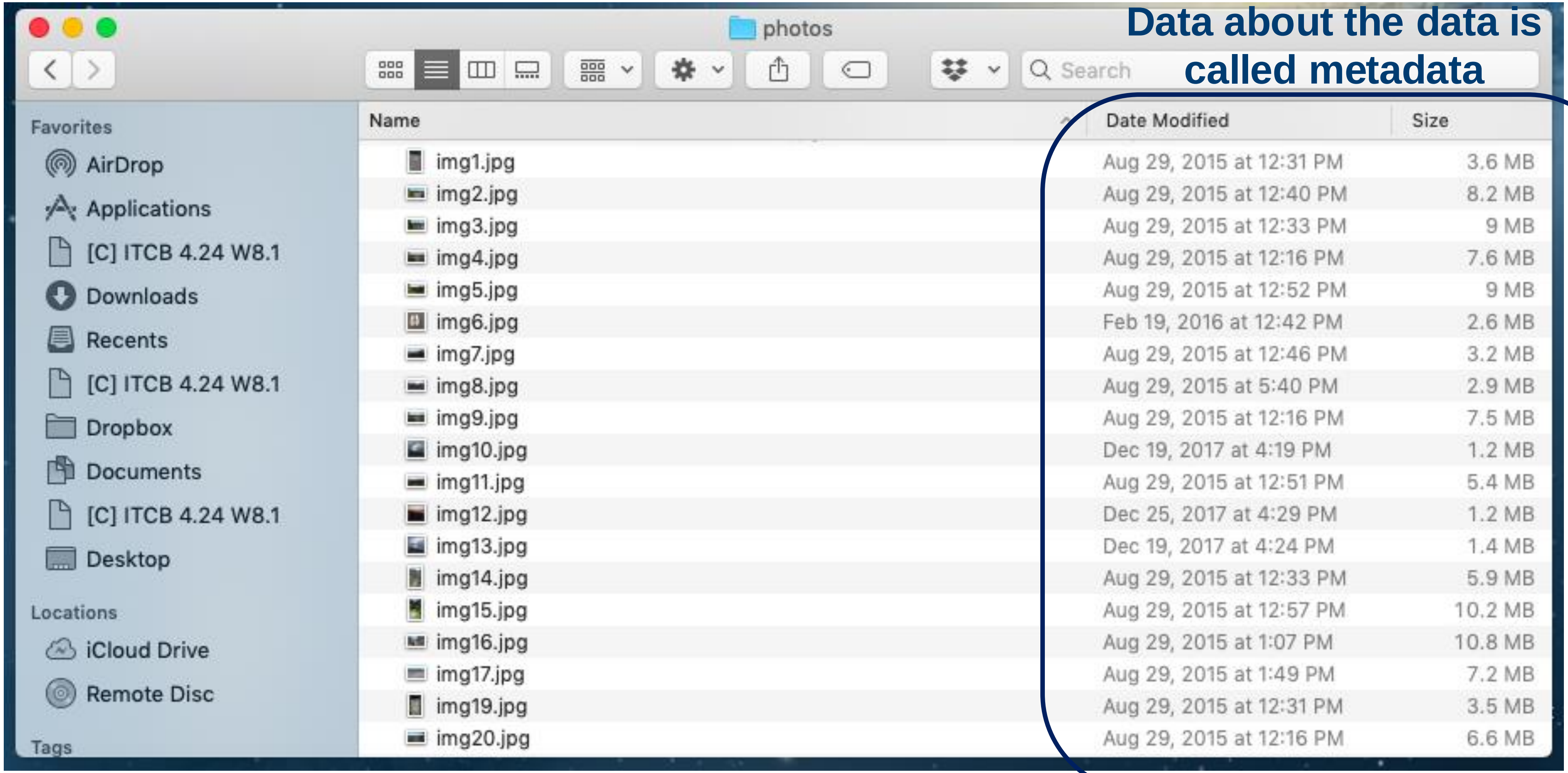
*based on figure 1 from “Hidden Technical Debt in Machine Learning Systems”, by D. Sculley et. al. from Google.

Outline

- Motivation: Why everyone needs a database management system?
- Database Technology: from ancient history to today
- Data Curation in the sciences
- My quest: One Algebra to rule them all

Raw Data from my camera

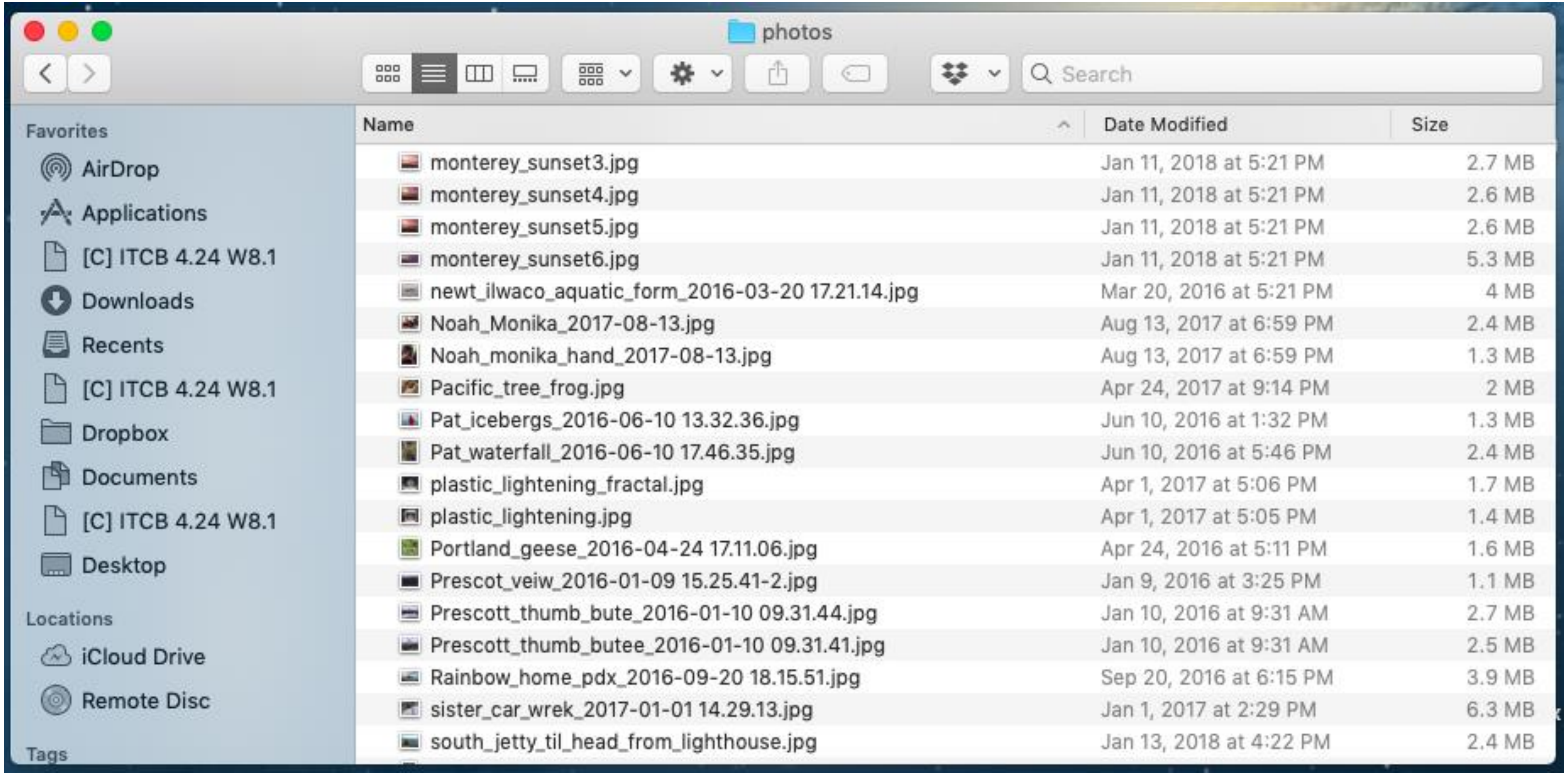
Unstructured, flat files



Data about the data is called metadata

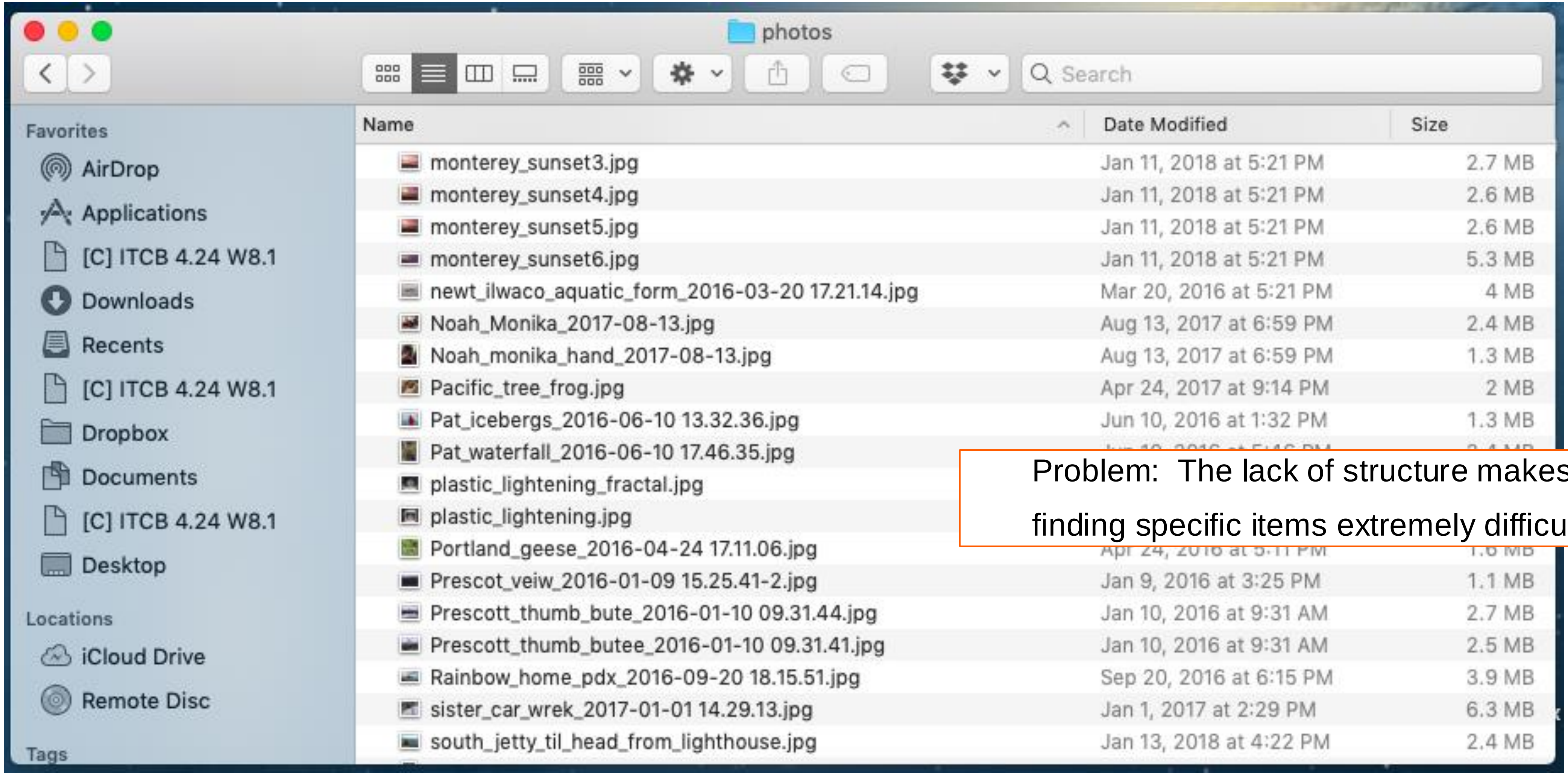
Raw Data from my camera

Unstructured, flat files. Attributes of the data captured in file names



Raw Data from my camera

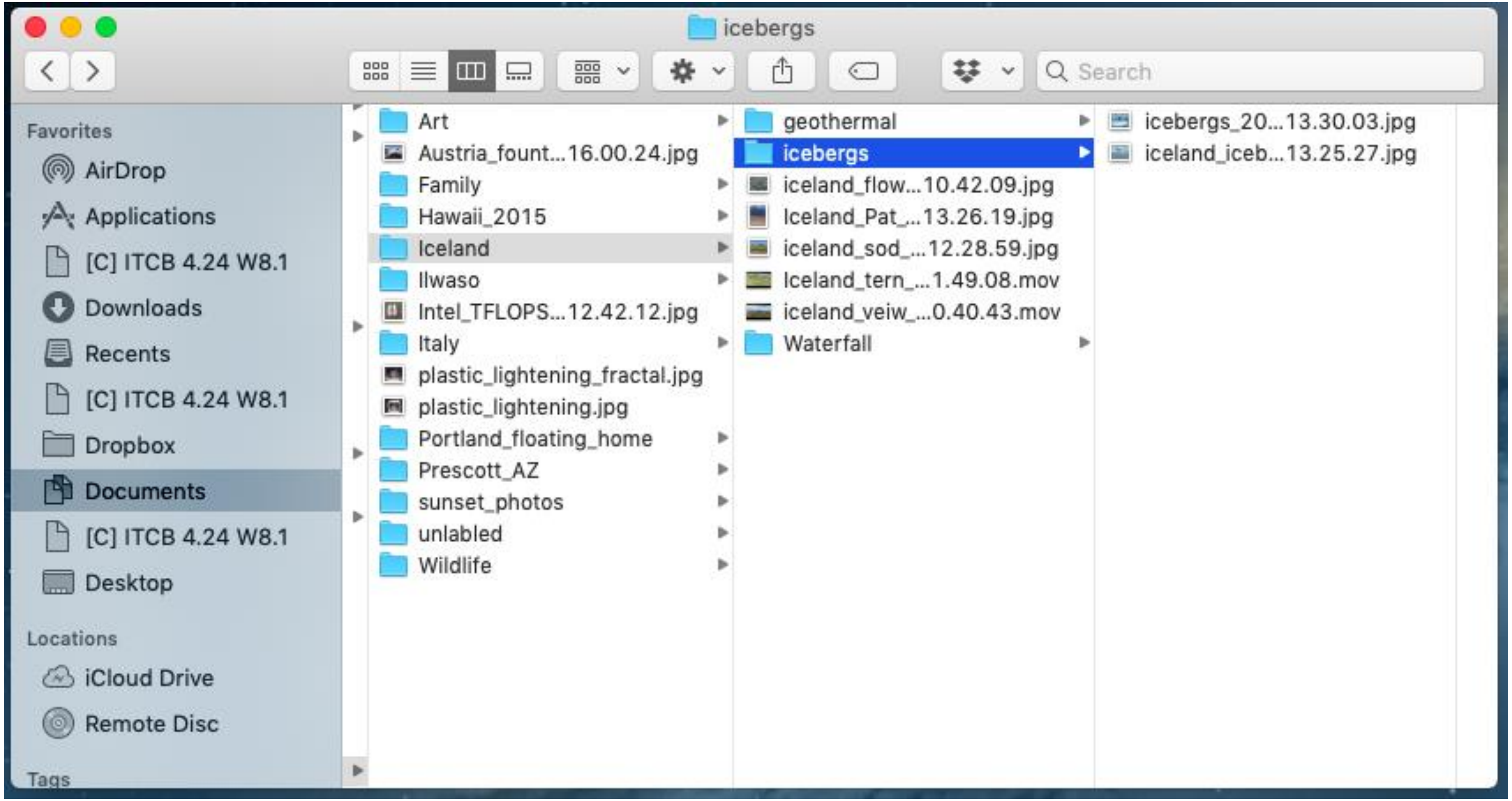
Unstructured, flat files. Attributes of the data captured in file names



Problem: The lack of structure makes finding specific items extremely difficult . . .

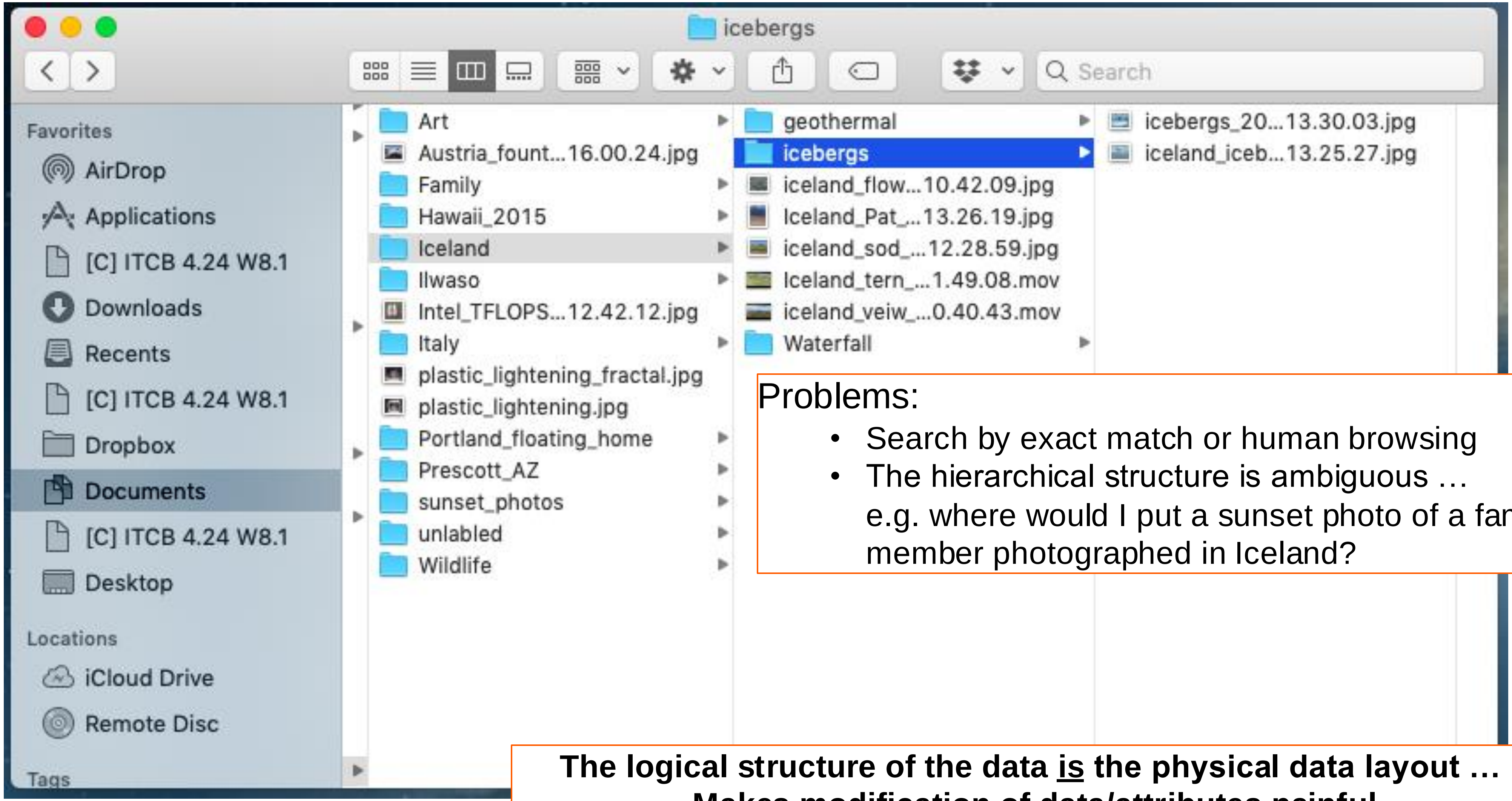
Raw Data from my camera

Hierarchical structure. Data Attributes in folder/file names



Raw Data from my camera

Hierarchical structure. Data Attributes in folder/file names



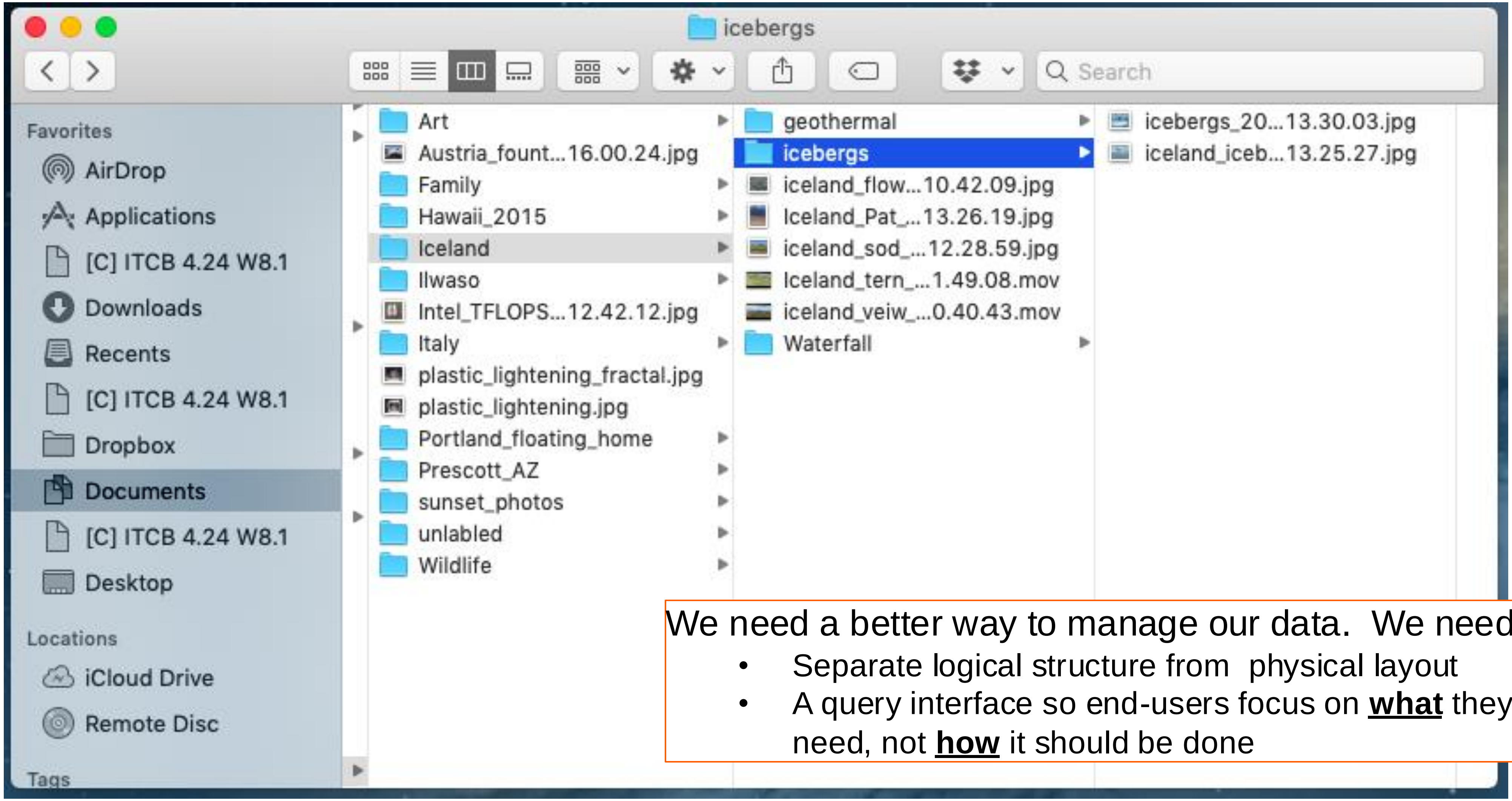
Problems:

- Search by exact match or human browsing
- The hierarchical structure is ambiguous …
  e.g. where would I put a sunset photo of a family member photographed in Iceland?

The logical structure of the data <u>is</u> the physical data layout …
Makes modification of data/attributes painful

Raw Data from my camera

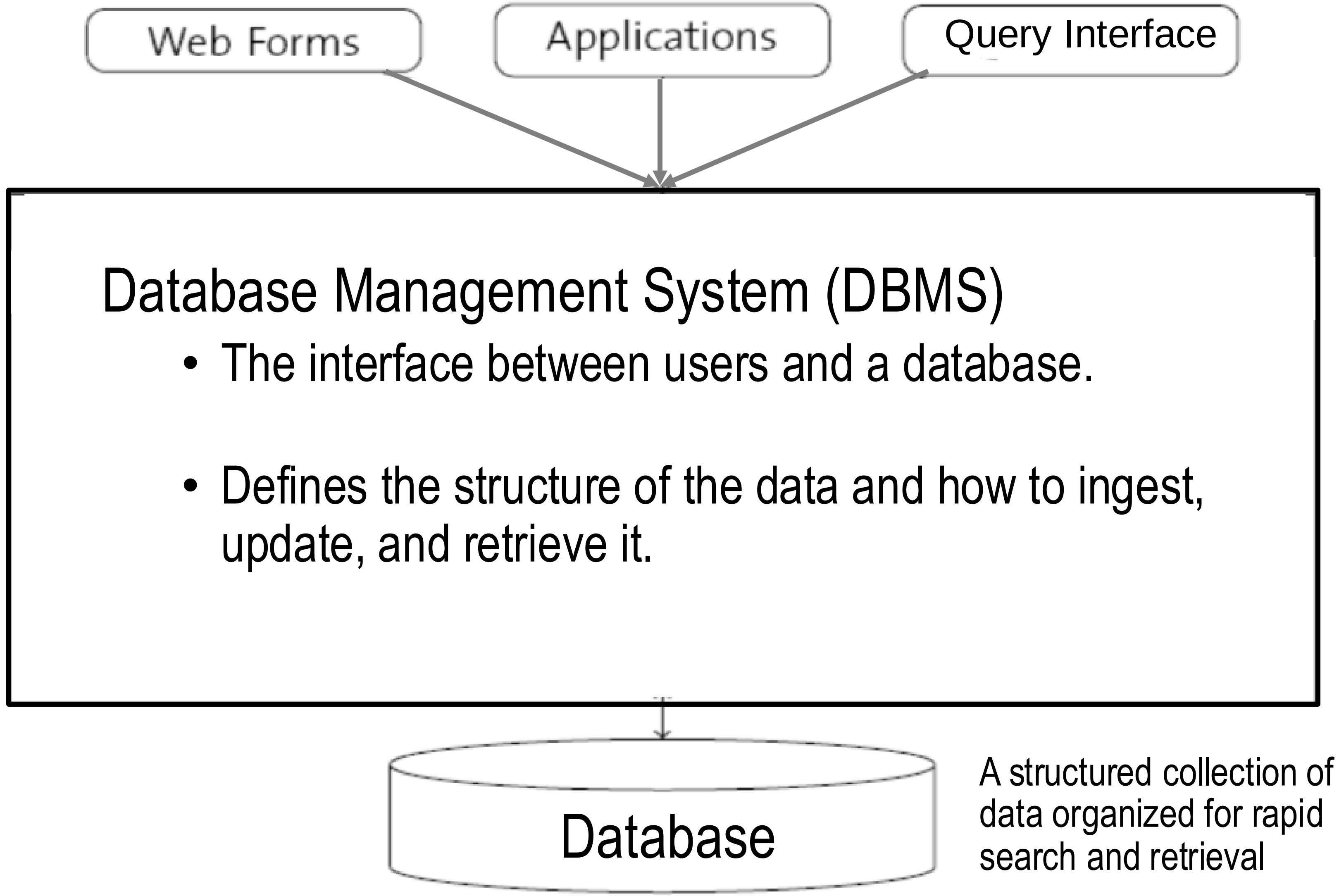
Hierarchical structure. Data Attributes in folder/file names



We need a better way to manage our data. We need:

- Separate logical structure from physical layout
- A query interface so end-users focus on **what** they need, not **how** it should be done

Databases and Database Management Systems

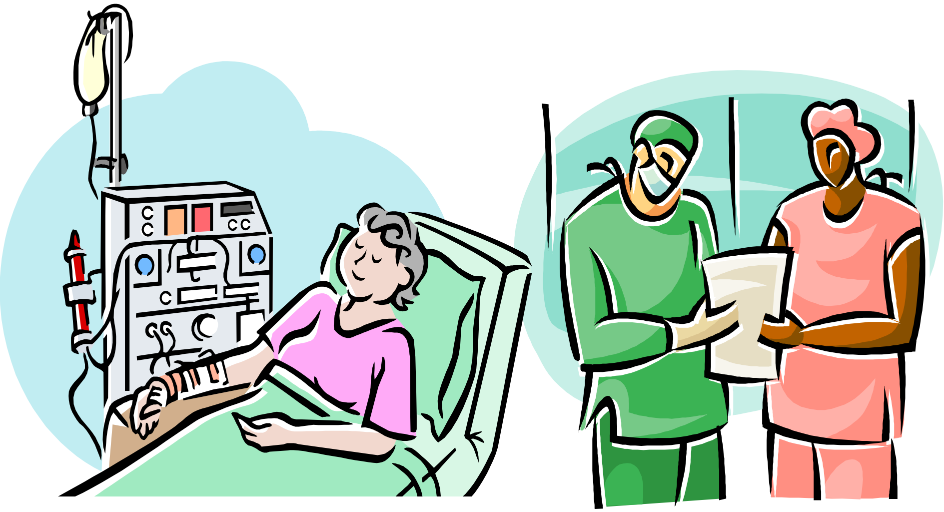


The DBMS separates how you work with Data from how the Data is stored.

Why a DBMS is so important: Big Data in the Real World

- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

- EKG traces
- Blood oxygen
- Blood pressure
- EEG traces



- Demographic
- Caregiver notes
- Medical charts
- Lab test results
- Xray, MRI, etc.

MIMIC: Multiparameter Intelligent Monitoring in Intensive Care, https://www.physionet.org/content/mimicdb/1.0.0/

Why a DBMS is so important: Big Data in the Real World

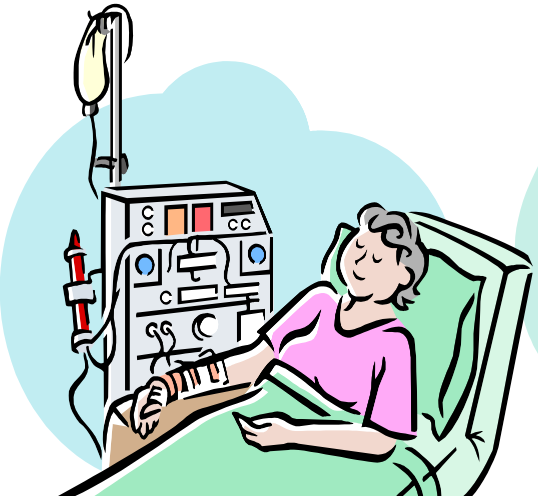
- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

Arrays
- EKG traces

Time Series
- Blood oxygen

Time Series
- Blood pressure

Arrays
- EEG traces



- Demographic

tables

- Caregiver notes

documents

- Medical charts

tables

- Lab test results

tables

- Xray, MRI, etc.

#images

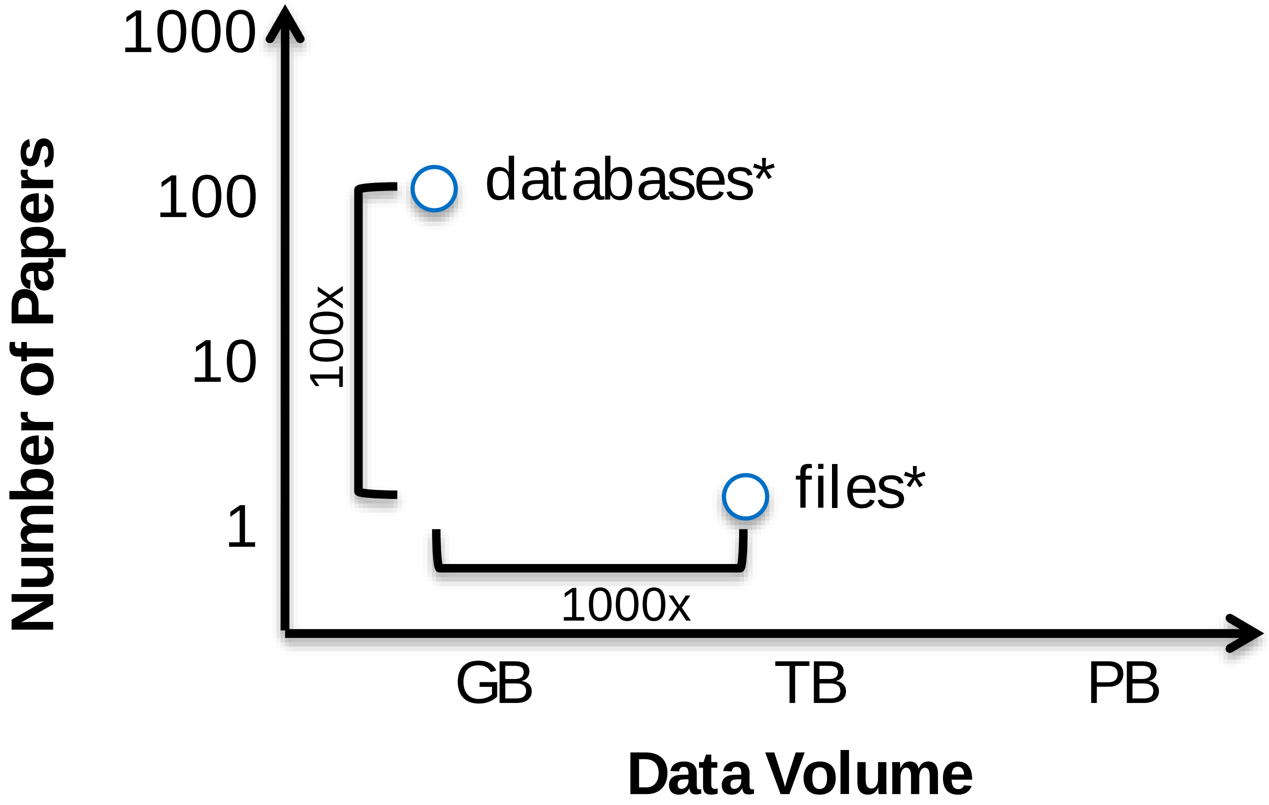
Time series and tabular data: Structured data in a Database.

Other data? Flat files

MIMIC: Multiparameter Intelligent Monitoring in Intensive Care, https://www.physionet.org/content/mimicdb/1.0.0/

Why a DBMS is so important: Big Data in the Real World

Analysis of published MIMICII papers, 2015



Storing data as flat files is roughly equivalent to deleting the data

We must bring the power of databases to all data

Source: Vijay Gadepally of MIT Lincoln labs

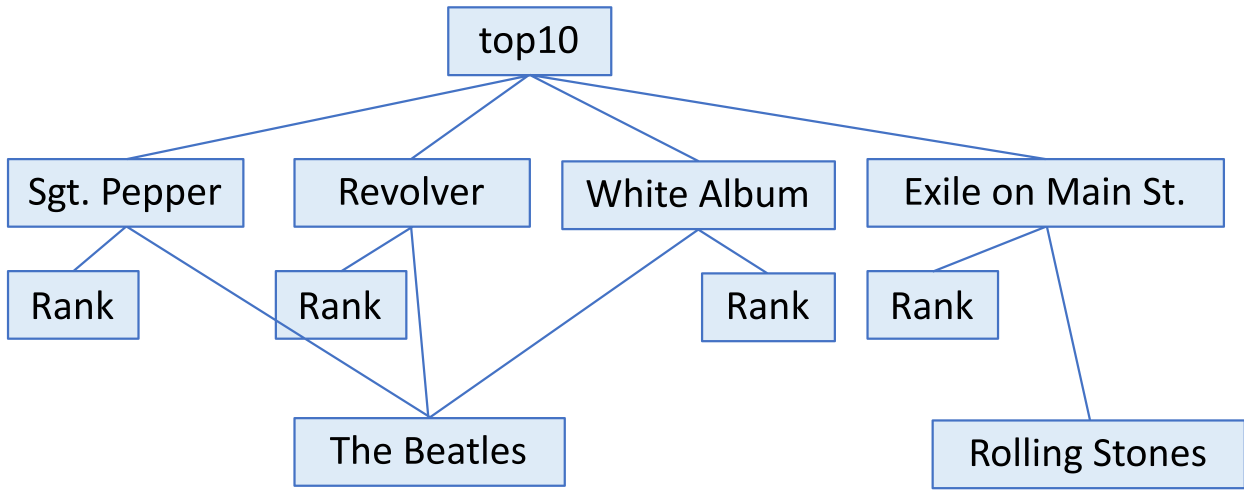
Outline

- Motivation: Why everyone needs a database management system?
- Database Technology: from ancient history to today
- Data Curation in the sciences
- My quest: One Algebra to rule them all

DBMS ancient history (1960s and 70s)

- In the 60's
  - Data in **Flat files** tied to a program (usually written in COBOL)

- Late 60s and throughout the 70s …. Hierarchical and Network models



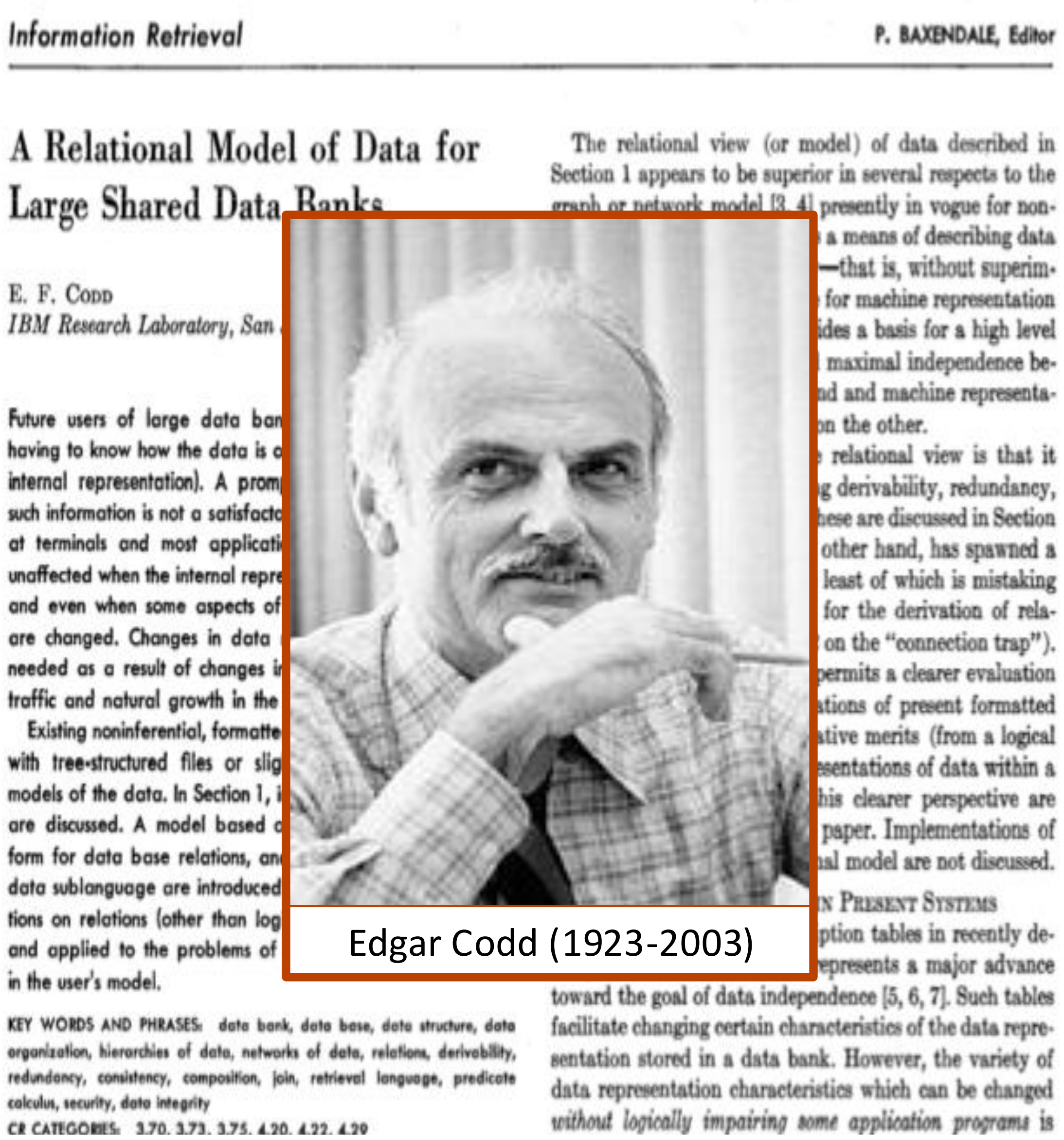
- These approaches were expensive, difficult to adapt to changing data, and lacked a standard query interface for users.

The Relational Model of Databases

- In 1970 Edgar Codd (IBM) published one of the most important papers in the history of computer science.

- It defined a formal algebra* for building databases … the **relational model**.
  - Object: A relation.
    - A set of tuples that share a set of attributes.
    - The set of attributes is defined by a schema
    - A relation is typically represented as a table.
  - A set of operators that act on relations. This set includes:

- Select σ
- Join $\bowtie$
- Rename ρ
- Project π



Information Retrieval

P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San

Future users of large data ban
having to know how the data is o
internal representation). A prom
such information is not a satisfacto
at terminals and most applicati
unaffected when the internal repre
and even when some aspects of
are changed. Changes in data
needed as a result of changes i
traffic and natural growth in the
Existing noninferential, formatte
with tree-structured files or slig
models of the data. In Section 1, i
are discussed. A model based o
form for data base relations, an
data sublanguage are introduced
tions on relations (other than log
and applied to the problems of
in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-
a means of describing data
—that is, without superim-
for machine representation
ides a basis for a high level
maximal independence be-
nd and machine representa-
on the other.
relational view is that it
g derivability, redundancy,
hese are discussed in Section
other hand, has spawned a
least of which is mistaking
for the derivation of rela-
on the "connection trap").
permits a clearer evaluation
ations of present formatted
ative merits (from a logical
esentations of data within a
his clearer perspective are
paper. Implementations of
nal model are not discussed.

n Present Systems
ption tables in recently de-
epresents a major advance
toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impairing some application programs is

Edgar Codd (1923-2003)

Communications of the ACM, vol 13, no. 6 p. 337, 1970

* Note: An "algebra" is a set of objects, operators that act on those objects, and rules for how those operators interact with each other

Database Queries and the Relational Model

- Users interact with the relational database by issuing queries.
- Codd proposed elegant and mathematically rich procedural queries:

$$\pi_{e.name}\left(\sigma_{e.salary > m.salary}\left(\rho_e(employee) \bowtie_{e.manager = m.name} \rho_m(employee)\right)\right)$$

- When applied to this relation:

Name	Salary	Manager
Smith	45,000	Harker
Jones	40,000	Smith
Baker	50,000	Smith
Nelson	55,000	Baker

- The output is **Baker** and **Nelson** …. employees who earn more than their managers

*This example comes from "Early History of SQL" by D. Chamberlin, IEEE Annals of the history of computing, 2012

Queries and the Structured Query Language

- Codd's notation was too obtuse for the general user

$$\pi_{e.name}\left(\sigma_{e.salary > m.salary}\left(\rho_e(employee) \bowtie_{e.manager = m.name} \rho_m(employee)\right)\right)$$

- In 1974, Codd's colleagues at IBM (Ray Boyce and Don Chamberlin) created a Query Language for Codd's relational Algebra called the Structured Query Language (SQL … Pronounced as Sequel)
- The SQL Query equivalent to Codd's notation above reads:

```
select e.name
from employee e, employee m
where e.manager = m.name and e.salary > m.salary
```

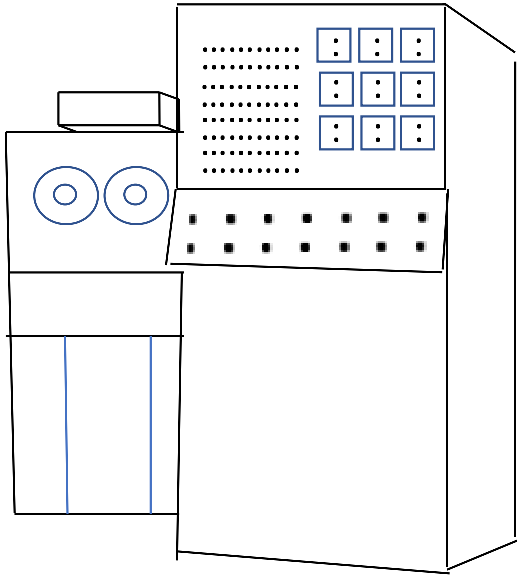
- Codd's notation is procedural. SQL is Declarative … That difference is an important reason for why SQL is the most successful DSL of all time.

DBMS History: A platform perspective

60’s to 70’s

Flat-files to network models

Custom + emerging vendors



Mainframe

The early days of database technology … centrally maintained, “big-iron” mainframe computers.

RDBMS: A deeper dive into Relations

- Example: the following relations come from the Rolling Stones top 500 albums database

Rolling Stone top 10 Albums (**top10**)

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The White Album")

Album Data (**albData**)

Album	Artist	Genre	Year
Pet Sounds	The Beach Boys	Rock	1966
Abbey Road	The Beatles	Rock	1969
Rubber Soul	The Beatles	Rock, Pop	1965
The Beatles ("The White Album")	The Beatles	Rock	1968
Sgt. Pepper's Lonely Hearts Club Band	The Beatles	Rock	1967
Revolver	The Beatles	Rock	1966
Highway 61 Revisited	Bob Dylan	Rock	1965
Blonde on Blonde	Bob Dylan	Rock, Blues	1966
London Calling	The Clash	Rock	1979
Are You Experienced	The Jimi Hendrix Experience	Rock, Blues	1967
What's Going On	Marvin Gaye	Funk / Soul	1971
Kind of Blue	Miles Davis	Jazz	1959
The Sun Sessions	Elvis Presley	Rock	1976
Exile on Main St.	The Rolling Stones	Rock	1972
The Velvet Underground & Nico	The Velvet Underground	Rock	1967

Each Row is a **record**
Each column is an **attribute**

https://www.rollingstone.com/music/music-lists/500-greatest-albums-of-all-time-156826/the-stone-roses-the-stone-roses-155889/

RDBMS: The concept of a Schema

- Schema: defines how the data is organized … (1) the column labels/types, (2) the primary key (uniquely identifies a record), and (3) the foreign keys (columns that map onto primary keys of other relations)

Primary keys

Rolling Stone top 10 Albums (**top10**)

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The White Album")

Album Data (**albData**)

Album	Artist	Genre	Year
Pet Sounds	The Beach Boys	Rock	1966
Abbey Road	The Beatles	Rock	1969
Rubber Soul	The Beatles	Rock, Pop	1965
The Beatles ("The White Album")	The Beatles	Rock	1968
Sgt. Pepper's Lonely Hearts Club Band	The Beatles	Rock	1967
Revolver	The Beatles	Rock	1966
Highway 61 Revisited	Bob Dylan	Rock	1965
Blonde on Blonde	Bob Dylan	Rock, Blues	1966
London Calling	The Clash	Rock	1979
Are You Experienced	The Jimi Hendrix Experience	Rock, Blues	1967
What's Going On	Marvin Gaye	Funk / Soul	1971
Kind of Blue	Miles Davis	Jazz	1959
The Sun Sessions	Elvis Presley	Rock	1976
Exile on Main St.	The Rolling Stones	Rock	1972
The Velvet Underground & Nico	The Velvet Underground	Rock	1967

A foreign key relationship

A complex database might need hundreds of tables.
Defining consistent schema for all those tables is extremely difficult

Relational Database example with SQL

Input two tables

(**top10** and **albData**)

```
SELECT top10.Number, top10.Album, albData.Artist, albData.year
From top10
INNER JOIN albData
ON top10.Album = albData.Album;
```

Rolling Stone top 10 Albums (**top10**)

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The White Album")

Album Data (**albData**)

Album	Artist	Genre	Year
Pet Sounds	The Beach Boys	Rock	1966
Abbey Road	The Beatles	Rock	1969
Rubber Soul	The Beatles	Rock, Pop	1965
The Beatles ("The White Album")	The Beatles	Rock	1968
Sgt. Pepper's Lonely Hearts Club Band	The Beatles	Rock	1967
Revolver	The Beatles	Rock	1966
Highway 61 Revisited	Bob Dylan	Rock	1965
Blonde on Blonde	Bob Dylan	Rock, Blues	1966
London Calling	The Clash	Rock	1979
Are You Experienced	The Jimi Hendrix Experience	Rock, Blues	1967
What's Going On	Marvin Gaye	Funk / Soul	1971
Kind of Blue	Miles Davis	Jazz	1959
The Sun Sessions	Elvis Presley	Rock	1976
Exile on Main St.	The Rolling Stones	Rock	1972
The Velvet Underground & Nico	The Velvet Underground	Rock	1967

Relational Database example with SQL

Input two tables

(**top10** and **albData**)

```
SELECT top10.Number, top10.Album, albData.Artist, albData.year
From top10
INNER JOIN albData
ON top10.Album = albData.Album;
```

Output a new table

Rolling Stone top 10 Albums (**top10**)

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The Whit

Album Data (**albData**)

Album	Artist	Genre	Year
Pet Sounds	The Beach Boys	Rock	1966
Abbey Road	The Beatles	Rock	1969
Rubber Soul	The Beatles	Rock, Pop	1965
The Beatles ("The White Album")	The Beatles	Rock	1968
		Rock	1967
		Rock	1966
		Rock	1965
		Rock, Blues	1966
		Rock	1979
	perience	Rock, Blues	1967
		Funk / Soul	1971
		Jazz	1959
		Rock	1976
		Rock	1972
	und	Rock	1967

Number	Album	Artist	Year
1	Sgt. Pepper's Lonely Hearts Club Band	The Beatles	1967
2	Pet Sounds	The Beach Boys	1966
3	Revolver	The Beatles	1966
4	Highway 61 Revisited	Bob Dylan	1965
5	Rubber Soul	The Beatles	1965
6	What's Going On	Marvin Gaye	1971
7	Exile on Main St.	The Rolling Stones	1972
8	London Calling	The Clash	1979
9	Blonde on Blonde	Bob Dylan	1966
10	The Beatles ("The White Album")	The Beatles	1968

Online Transaction Processing (OLTP)

- When you use an ATM (bancomat), you expect that your transaction will be:

Atomic: the transaction either completes or it doesn't happen.

Consistent: The data is always in a valid state.

Isolated: Transactions cannot interfere with each other.

Durable: Once committed, the effect of a transaction is always there … regardless of system crashes.

A Database Management Systems that supports these conditions is said to be **<u>ACID compliant</u>**

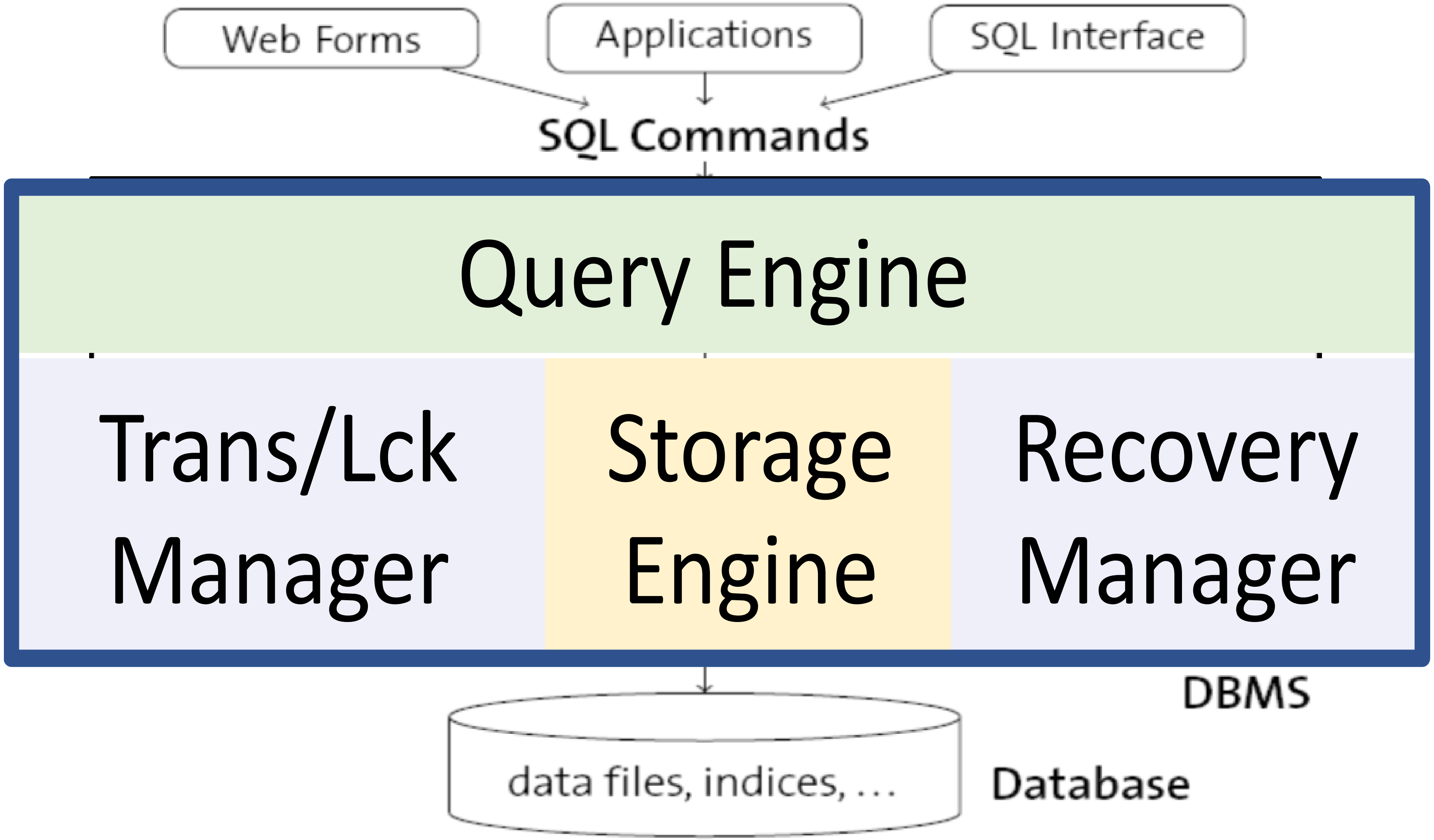
ACID compliance is essential for a multiuser, online transaction Processing System (e.g. an ATM)

ATM: Automated Teller Machine. … or in Italy a bancomat

Structure of a Modern Relational DBMS

Transactions emphasize mixed reads and writes.

Low latency writes favor a row-oriented data store (i.e. storage by rows in the table).



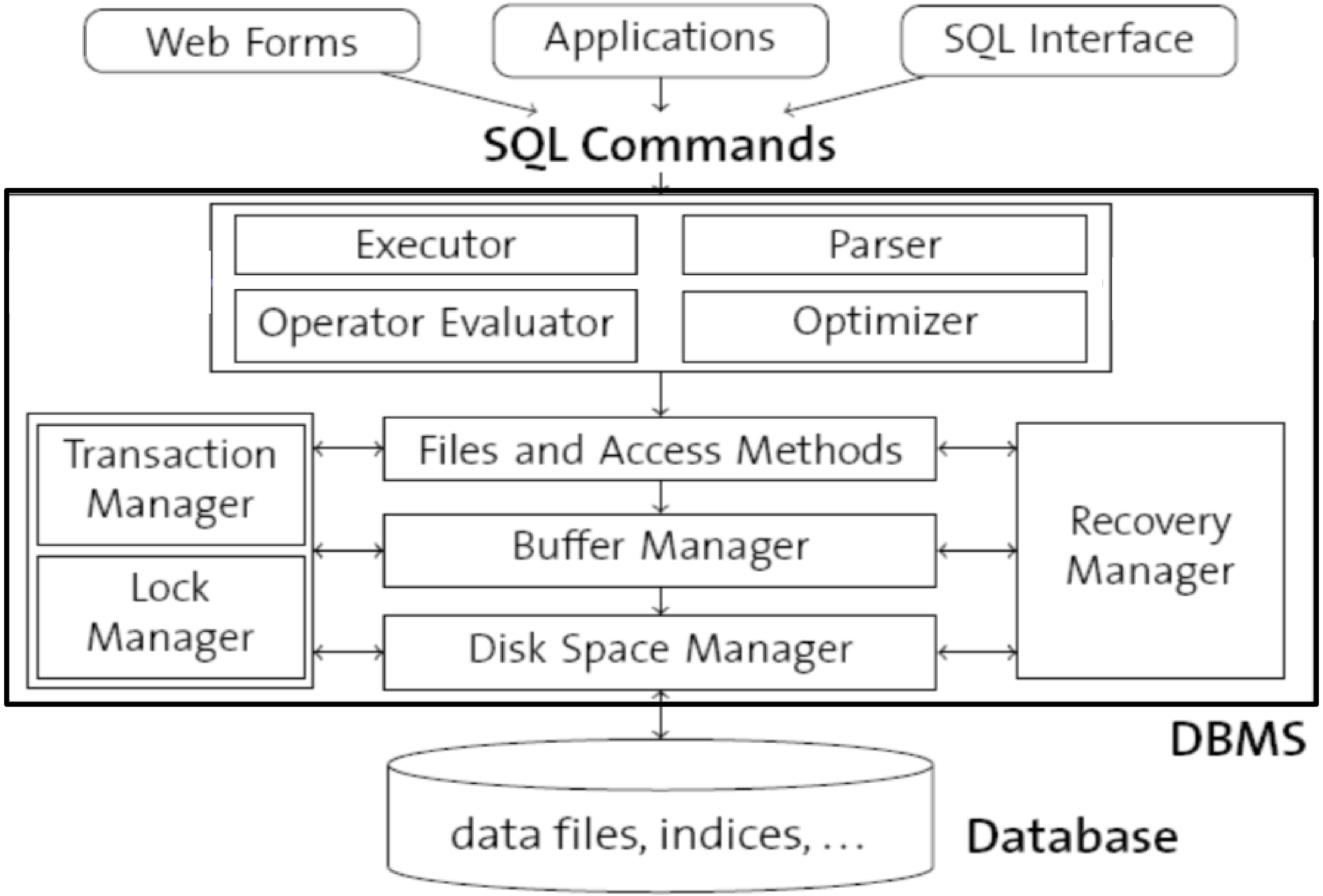
Trans: Transaction Lck: Lock

Source: Nesime Tatbul, information systems, ETH, 2012
And Ramakrishnam/Gehrke: “Database managementSystems”, McGraw-Hill 2003

Structure of a Modern Relational DBMS

Transactions emphasize mixed reads and writes.

Low latency writes favor a row-oriented data store (i.e. storage by rows in the table).



Trans: Transaction   Lck: Lock

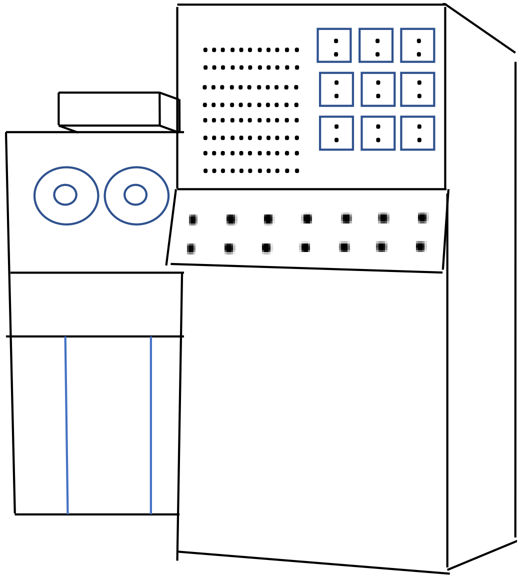
Source: Nesime Tatbul, information systems, ETH, 2012
And Ramakrishnam/Gehrke: “Database managementSystems”, McGraw-Hill 20

DBMS History: A platform perspective

60’s to 70’s

Flat-files to network models

Custom + emerging vendors

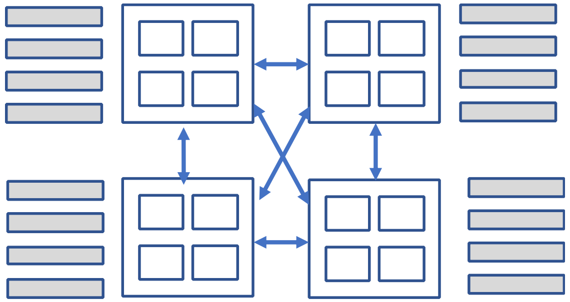


Mainframe

80’s to 90’s

Relational models

RDBMS vendors (Oracle and friends)



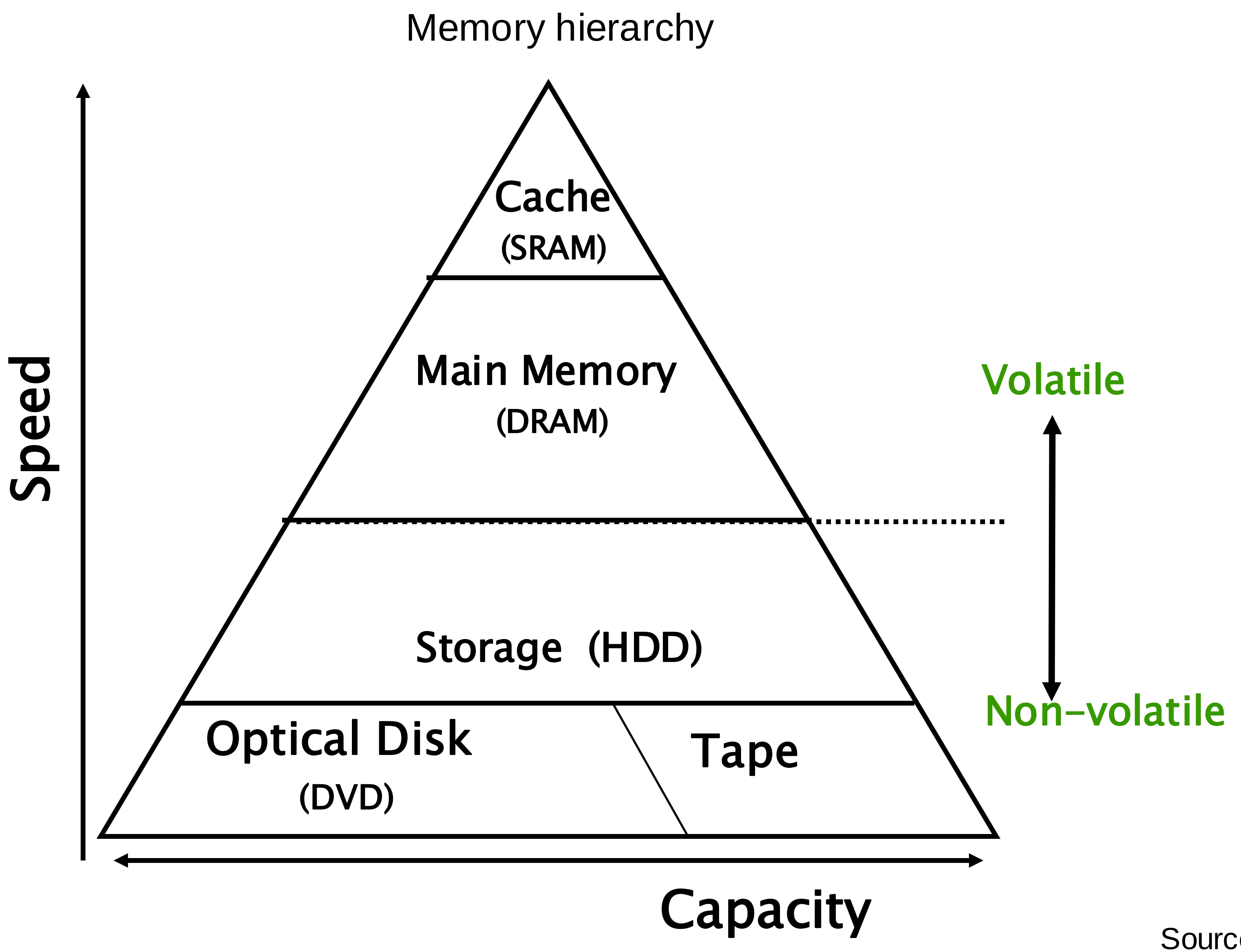
SMP Server

Databases to support ACID compliance while handling larger problems depended on multithreading and large address spaces.

Scale-up: increasing processors in a single physical address space

RDBMS: Relational Database Management System SMP: Symmetric Multiprocessor

RDBS: Storing the data



- Traditional databases live "on disk".
- Consider memory latencies:

CPU: <nanosecond
Memory: ~80 ns
3D Xpoint: ~300ns
Optane SSD: ~10 usec
NAND SSD: ~80 usec
Disk: ~6 milliseconds

- Disk access is slow compared to CPU speeds.
- Disk access is the primary performance bottleneck in traditional RDBMS
- How should we layout data on disk?

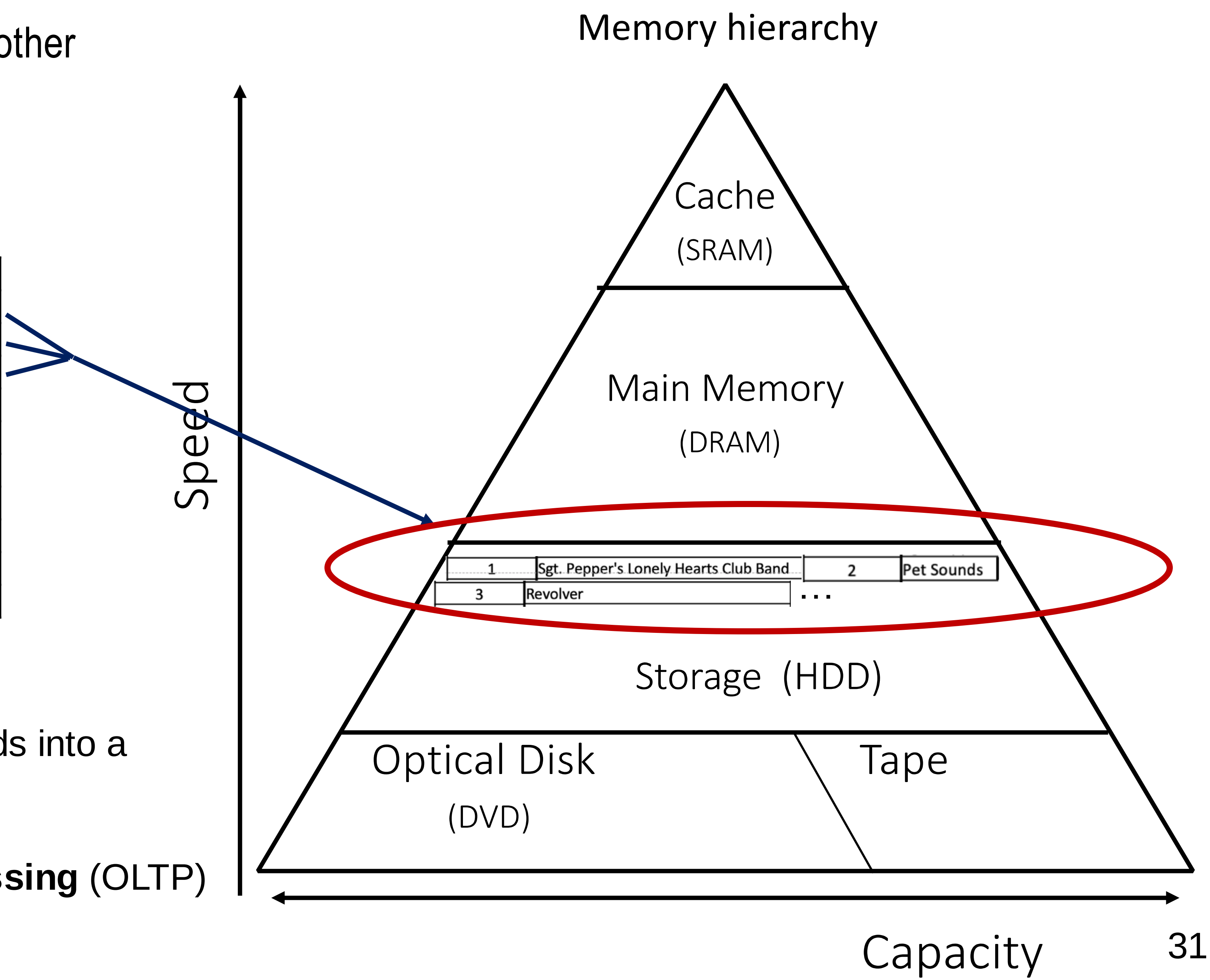
Source: Michael Mesnier, Intel labs

RDBMS: Storing Relations … Row Store

- Organize by row … one record after another

Table: Rolling Stone top 10 Albums

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The White Album")



A ***row store*** is optimized for writing records into a database.

Preferred for **Online Transaction Processing** (OLTP)

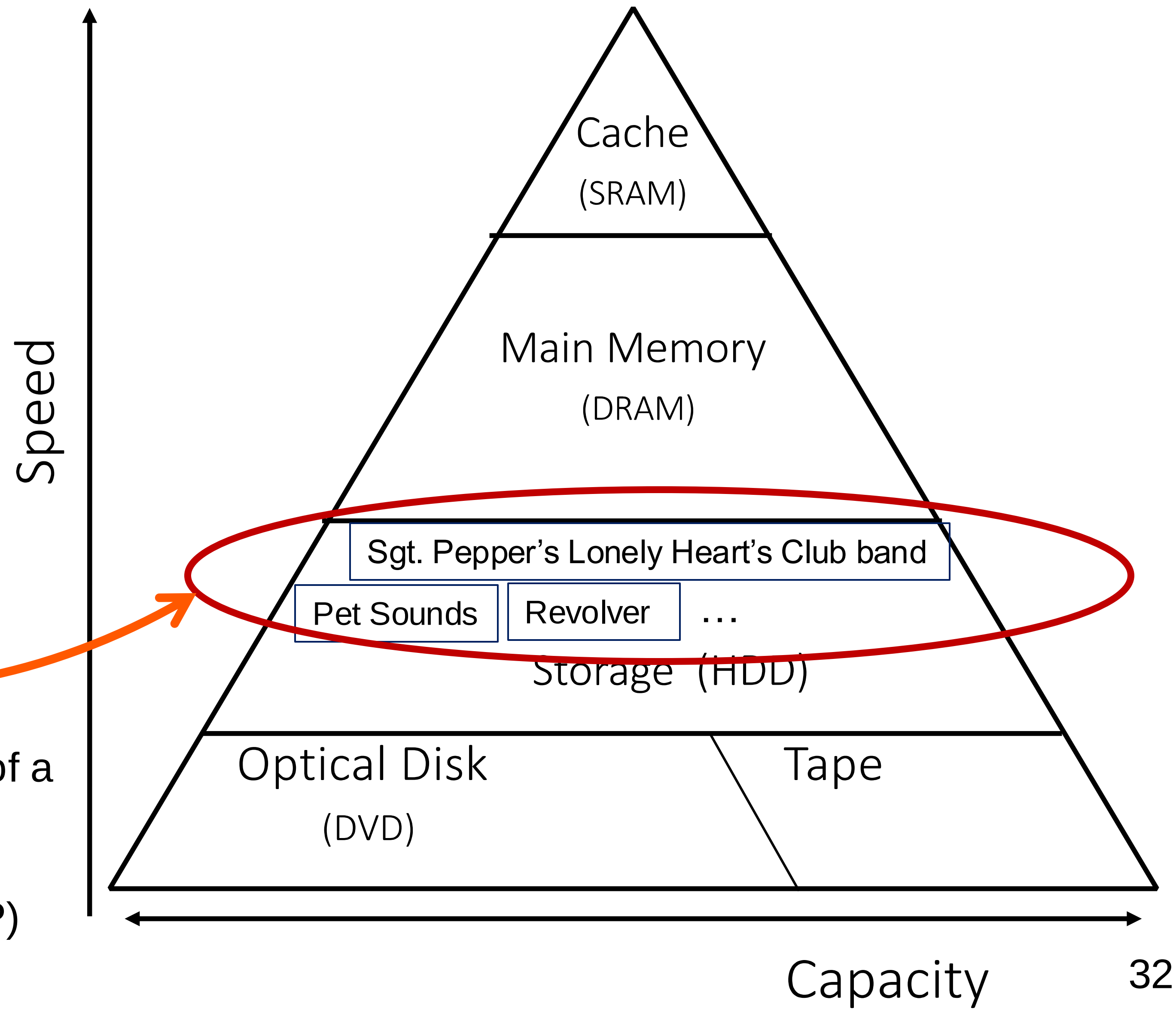
RDBMS: Storing Relations … Column Store

- Organize by columns (attributes) … one attribute after another

Table: Rolling Stone top 10 Albums

Number	Album
1	Sgt. Pepper's Lonely Hearts Club Band
2	Pet Sounds
3	Revolver
4	Highway 61 Revisited
5	Rubber Soul
6	What's Going On
7	Exile on Main St.
8	London Calling
9	Blonde on Blonde
10	The Beatles ("The White Album")

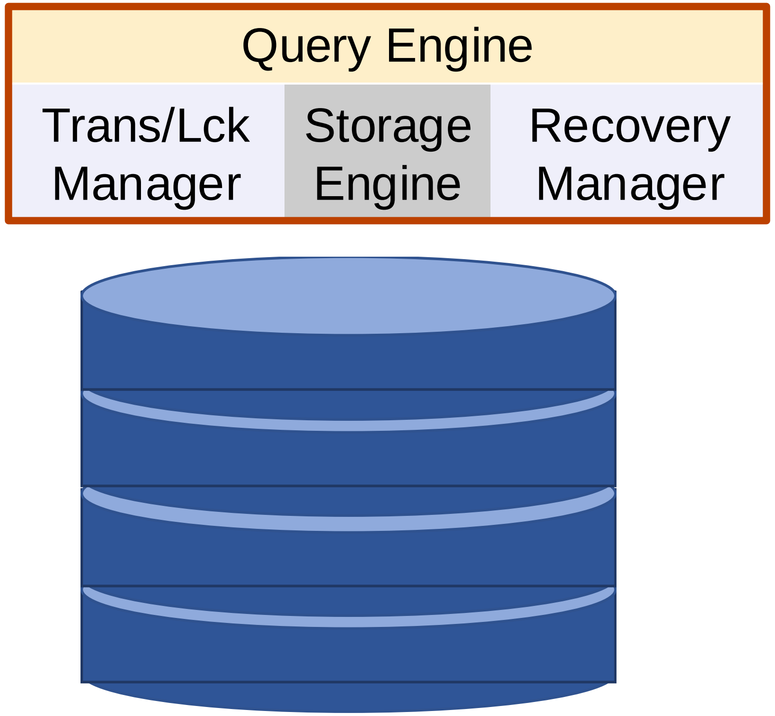
Memory hierarchy



A *column store* is optimized for reading a set of attributes as is often done in computing properties of a set of records.

Preferred for **Online Analytical Processing** (OLAP)

Databases in the Internet Age

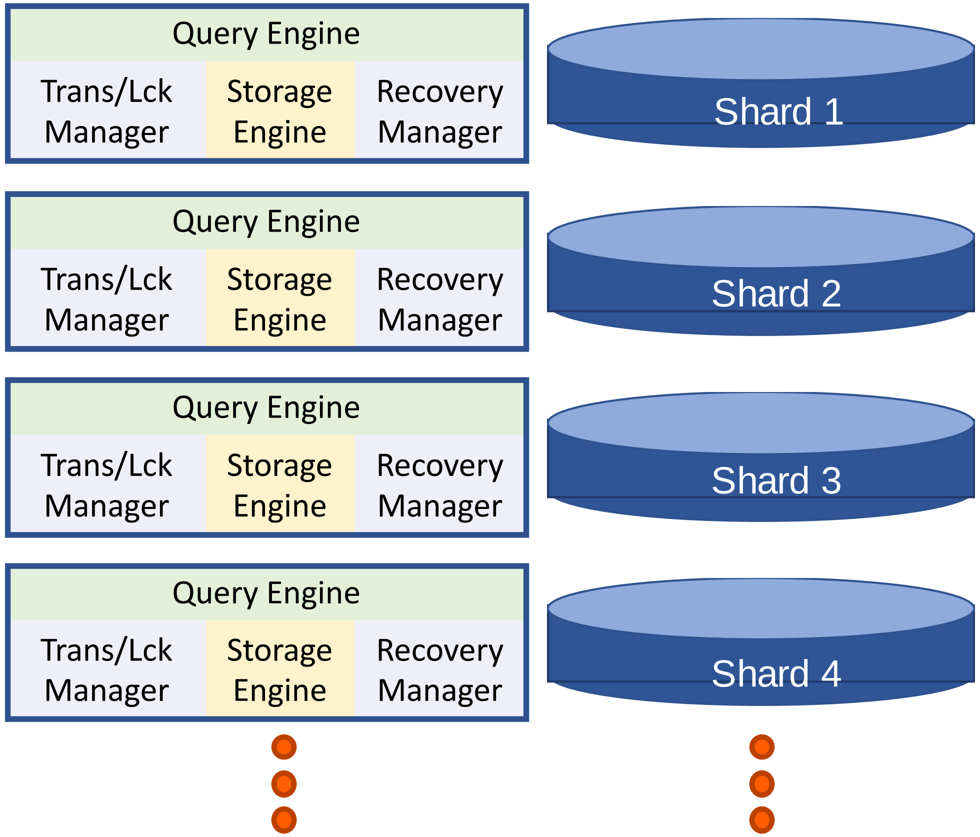


RDBMS Challenges

- Must define Schemas upfront
- Scale-up → expensive multiprocessor servers
- Scalability limited: You can only connect so-many processors to a single shared memory.
- ACID compliance is REALLY HARD to scale.

Internet-scale problems → Extreme scalability and unstructured data → moving beyond RDMS/Scale-up

Databases in the Internet Age: the birth of NoSQL



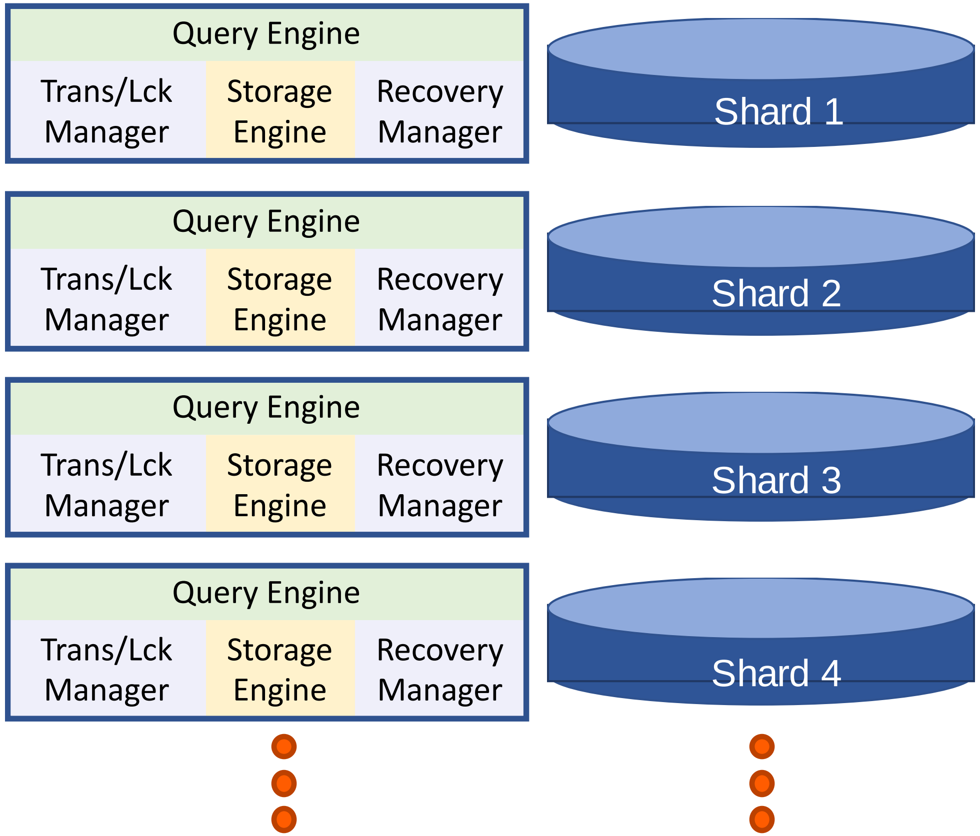
NoSQL

- Break up the data into chunks called Shards
- Distribute shards among multiple DBMS instances
- Scale-out solutions: use “shared nothing” clusters
- Eventual consistency … violates ACID, but more scalable
- Uses a simple schema free-algebra .. Key-value store

Scale-out: Lots of servers connected by a network (no shared address space). Scalability limited by $$$ and electric bill.

Classic example of NoSQL: Google Table, production use in 2006

Databases in the Internet Age: the birth of NoSQL



<u>Example of a NoSQL key-value store</u>

Data as a set of tuples: <key,value>

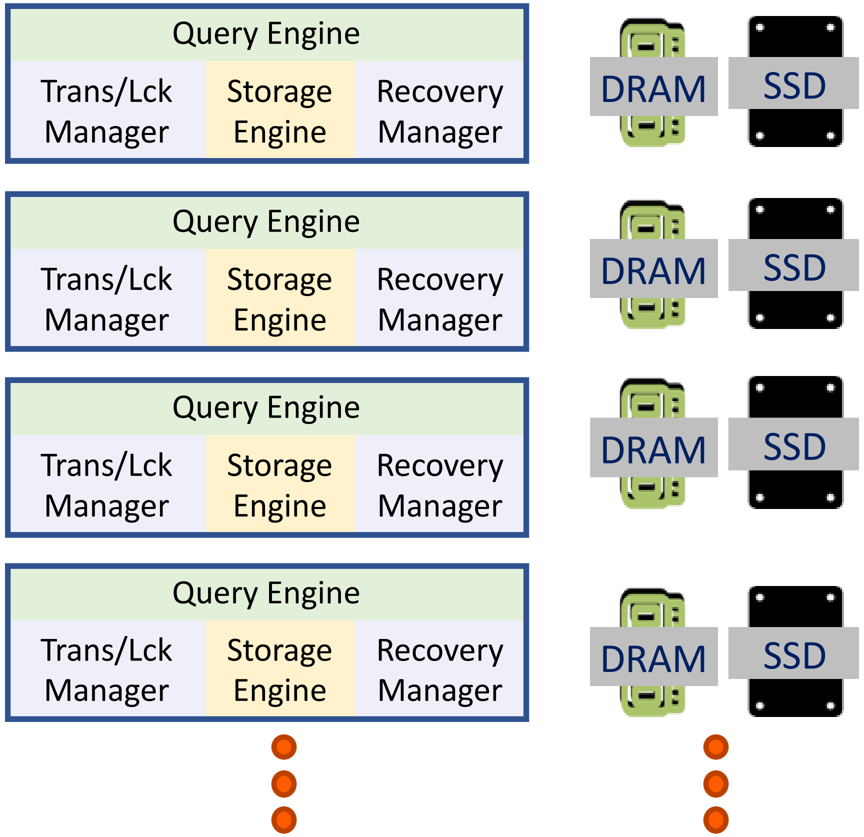
<“Pet Sounds”, Rank=2> <“Rubber Soul”, Artist=“The Beatles”>

<“Rubber Soul”, Rank=5> <“Blonde on Blonde”, Artist=“Bob Dylan”>

Scale-out: Lots of servers connected by a network (no shared address space).
Scalability limited by $$$ and electric bill.

Classic example of NoSQL: Google Table, production use in 2006

Databases in the Internet Age: the birth of NoSQL



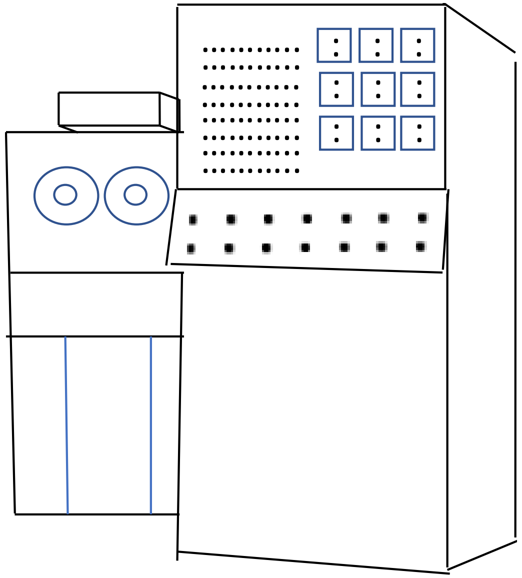
- NoSQL originally used disk-based solutions such as **Hadoop**.
- Later moved to **in-memory storage engines** such as **Spark** with SSDs restricted to backing up persistent data.

NoSQL - arbitrary scalability, reliability by replication, easy ingestion of new data, flexibility as data changes

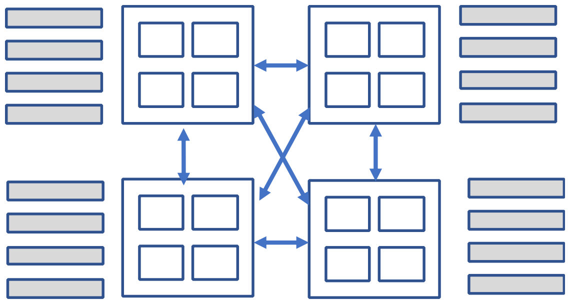
SSD: solid state drive DRAM: Dynamic random-access memory

DBMS History: A platform perspective

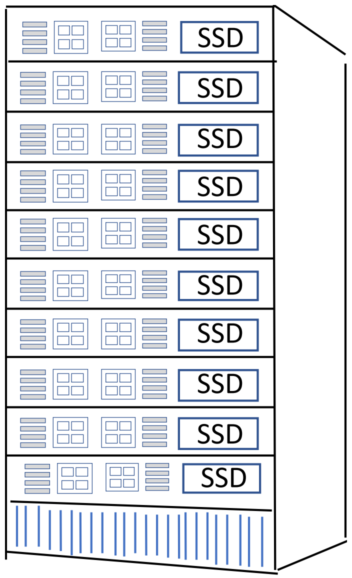
60’s to 70’s	80’s to 90’s	00’s to 10’s
Flat-files to network models	Relational models	NoSQL
Custom + emerging vendors	RDBMS vendors (Oracle and friends)	Legacy RDBMS + a swarm of NoSQL vendors



Mainframe



SMP Server



Hyperconverged Infrastructure (HCI)
dual-processor Xeon Servers

RDBMS: Relational Database Management System SMP: Symmetric Multiprocessor

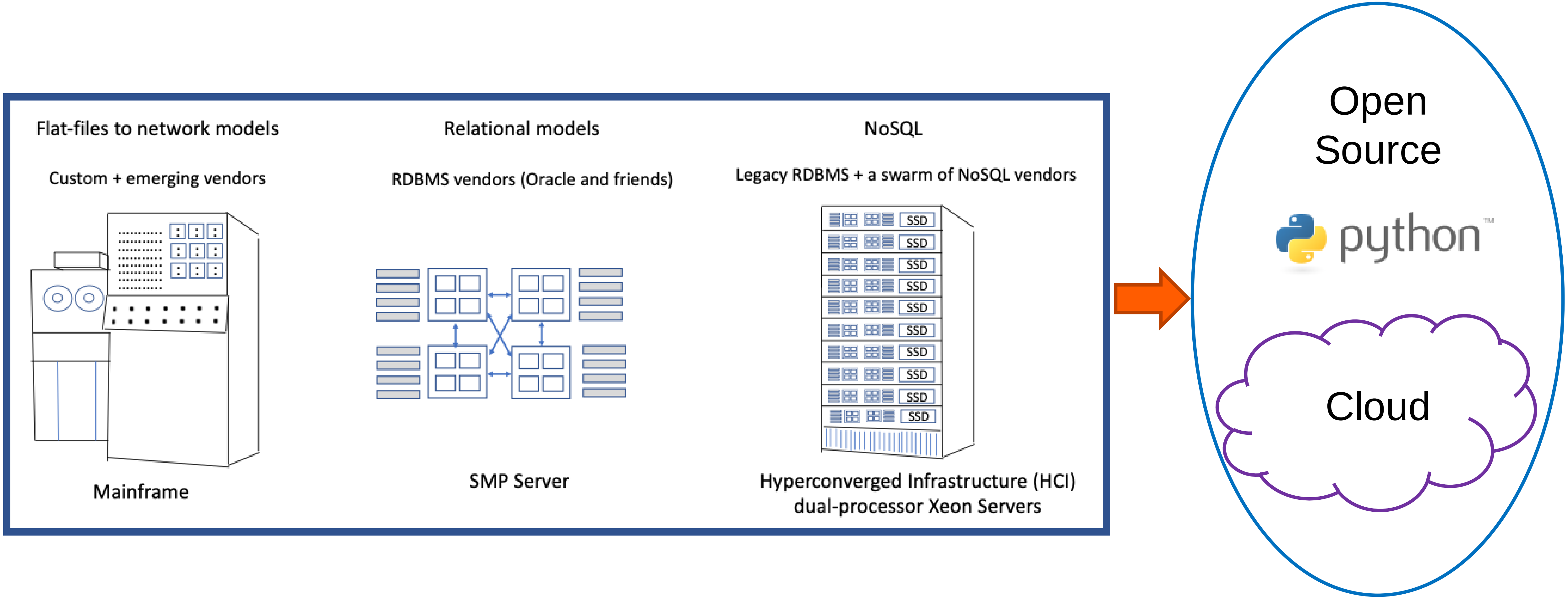
Four classes of systems in use today

DBMS Classes	When they emerged	Feature-set/notes
RDBMS/SQL	1980s	Online Transaction Processing (OLTP) with full ACID guarantees for transactions. Universal declarative query language (SQL)
OLAP data warehouses	2000s	Extract Transform Load (ETL) data from multiple sources and gather into a "single" system for Online Analytic Processing (OLAP). The birth of column stores data for high performance analytics
NoSQL (later … not only SQL)	Mid-2000s	DBMS for the internet age. Pioneered (1) eventual consistency and relaxed ACID, (2) shard data for distributed systems, (3) in memory storage, and (4) replication
NewSQL	2010s	All the benefits of NoSQL but ACID is brought back with lock-free consistency models for OLTP applications. Pioneered **Hybrid Transaction-Analytics Processing** (HTAP)

Source: What's Really New with NewSQL, Pavlo and Aslett, SIGMOD Record, June 2016

ACID: **A**tomicity, **C**onsistency, **I**ndependence, **D**ependability

DBMS Platforms: Where are we going?



DBMS technology is moving into the cloud using FaaS (Function as a Service) to make the physical computers “invisible”.

Outline

- Motivation: Why everyone needs a database management system?
- Database Technology: from ancient history to today
- Data Curation in the sciences
- My quest: One Algebra to rule them all

How Data Scientists deal with data today …

- A DBMS separates how data is stored from how it is processed. If data is arbitrarily complex, heterogeneous, and needs protections such as ACID, you must use a DataBase Management System such as Postgress.

- In the sciences, our data is potentially huge but the structure is much simpler than the more general data dealt with in the database world.

- Data tends to be used for analytics so online-transaction-processing is not a major concern.

- Data Scientists use a number of systems for their data management workflows … some of the more common systems you hear about include:

  - Arrow
  - Parquet
  - JSON
  - HDF5
  - TileDB
  - CSV
  - XML
  - YAML
  - Binsparse
  - pandas

- There are also common execution models for working with these systems:
  - Hadoop: a map-reduce model organized around operating on data on disks
  - SPARC: Like Hadoop (map-reduce + streaming) but data stored in memory … MUCH faster than Hadoop. Based on Scala virtual machine.
  - Ray: An in-memory replacement for SPARC with more flexible processing (actors model) and without a performance-killing virtual machine.

Map-Reduce is a variation on SPMD and the Bulk Synchronous Pattern restricted to reductions for the communication phase

How Data Scientists deal with data today …

- A DBMS separates how data is stored from how it is processed. If data is arbitrarily complex, heterogeneous, and needs protections such as ACID, you must use a DataBase Management System such as Postgress.

- In the sciences, our data is potentially huge but the structure is much simpler than the more general data dealt with in the database world.

- Data ten… ics so o… ng is not a major concern.

- Data Sci… system… nt workflows … some of the more common systems you hear about include:

An in-memory column store creating a interchange format for machine learning applications.

A sophisticated storage engine for array data … optimized for sparse data but effective for dense arrays as well.

A new standards effort to create a binary file format for sparse data. Tightly aligned with the GraphBLAS ecosystem.

A column-oreited file-format for complex data. Includes compression.

– Arrow	– TileDB	– YAML
– Parquet	– CSV	– Binsparse
– JSON	– XML	– pandas
– HDF5		

A hierarchal file-store for high speed, parallel I/O.

Comma-Separated values. A human readable file-storage format for row-oriented data. Used by Pandas, Excel and other data processing tools

The dominant data management tool for python-based data science. Builds data frames from CSV files

- … on execution mode… ese syster…
  - Hadoop: a map-reduce model organized aroun… s
  - SPARC: Like Hadoop (map-reduce + streamin… y … MUCH … ala virtual machine.
  - Ray: An in-memory replacement for SPARC wi… actors model) and without a performance-killing virtual machine.

Map-Reduce is a variation on SPMD and the Bulk Synchronous Pattern restricted to reductions for the communication phase

Managing Scientific Data

How Data Scientists deal with data today …

- A DBMS separates how data is stored from how it is processed. If data is arbitrarily complex, heterogeneous, and needs protections such as ACID, you must use a DataBase Management System such as Postgress.

- In the sciences, our data is potentially huge but the structure is much simpler than the more general data dealt with in the database world.

- Data tends to be used for analytics so online-transaction-processing is not a major concern.

- Data Scientists use a number of systems for their data management workflows … some of the more common systems you hear about include:
  - Arrow
  - Parquet
  - JSON
  - HDF5
  - TileDB
  - CSV
  - XML
  - YAML
  - Binsparse
  - pandas

It would require an entire course to survey all of these

- There are also common execution models for working with these systems:
  - Hadoop: a map-reduce model organized around operating on data on disks
  - SPARC: Like Hadoop (map-reduce + streaming) but data stored in memory … MUCH faster than Hadoop. Based on Scala virtual machine.
  - Ray: An in-memory replacement for SPARC with more flexible processing (actors model) and without a performance-killing virtual machine.

Map-Reduce is a variation on SPMD and the Bulk Synchronous Pattern restricted to reductions for the communication phase

Managing Scientific Data

How Data Scientists deal with data today …

- A DBMS separates how data is stored from how it is processed. If data is arbitrarily complex, heterogeneous, and needs protections such as ACID, you must use a DataBase Management System such as Postgress.
- In the sciences, our data is potentially huge but the structure is much simpler than the more general data dealt with in the database world.
- Data tends to be used for analytics so online-transaction-processing is not a major concern.

ata Scientists use a number of systems for their data management workflows … some of the more common
ystems you hear about include:

- Arrow
- Parquet
- JSON
- HDF5
- TileDB
- CSV
- XML
- YAML
- Binsparse
- pandas

here are also common execution models for working with these systems:

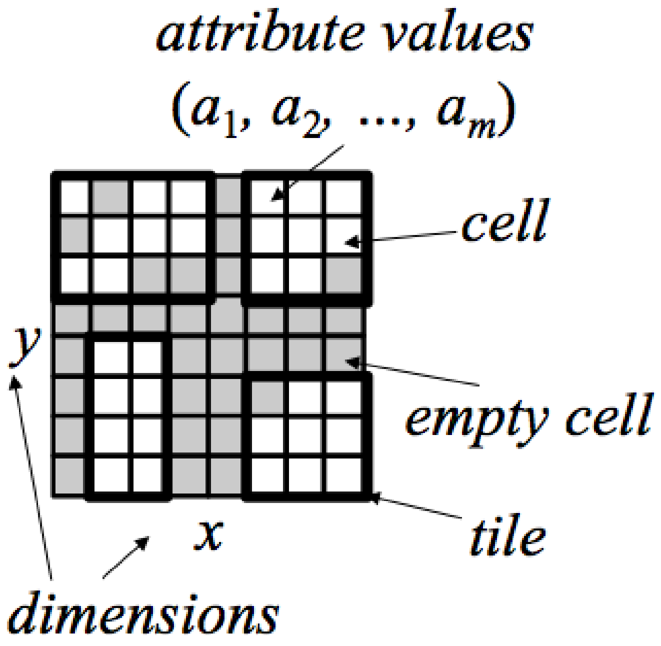
- Hadoop: a map-reduce model organized around operating on data on disks
- SPARC: Like Hadoop (map-reduce + streaming) but data stored in memory … MUCH faster than Hadoop. Based on Scala virtual machine.
- Ray: An in-memory replacement for SPARC with more flexible processing (actors model) and without a performance-killing virtual machine.

Map-Reduce is a variation on SPMD and the Bulk Synchronous Pattern restricted to reductions for the communication phase

It would require an entire course to survey all of these … so we'll just pick one to discuss to give you a feel for why it is so important to use a data storage system suited to your problem.

TileDB: an array data storage manager

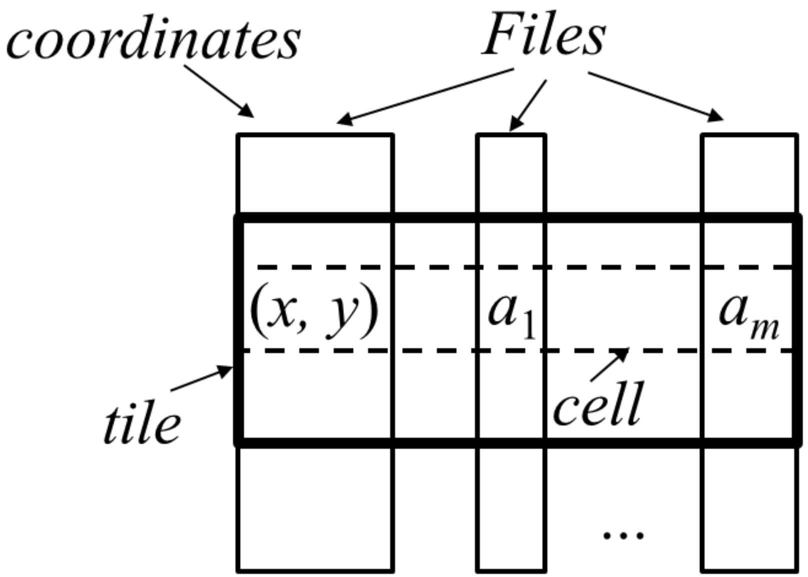
Logical representation



TileDB stores only the non-empty elements of sparse arrays

Sorts cells and packs them into groups of fixed capacity, called tiles

Physical representation



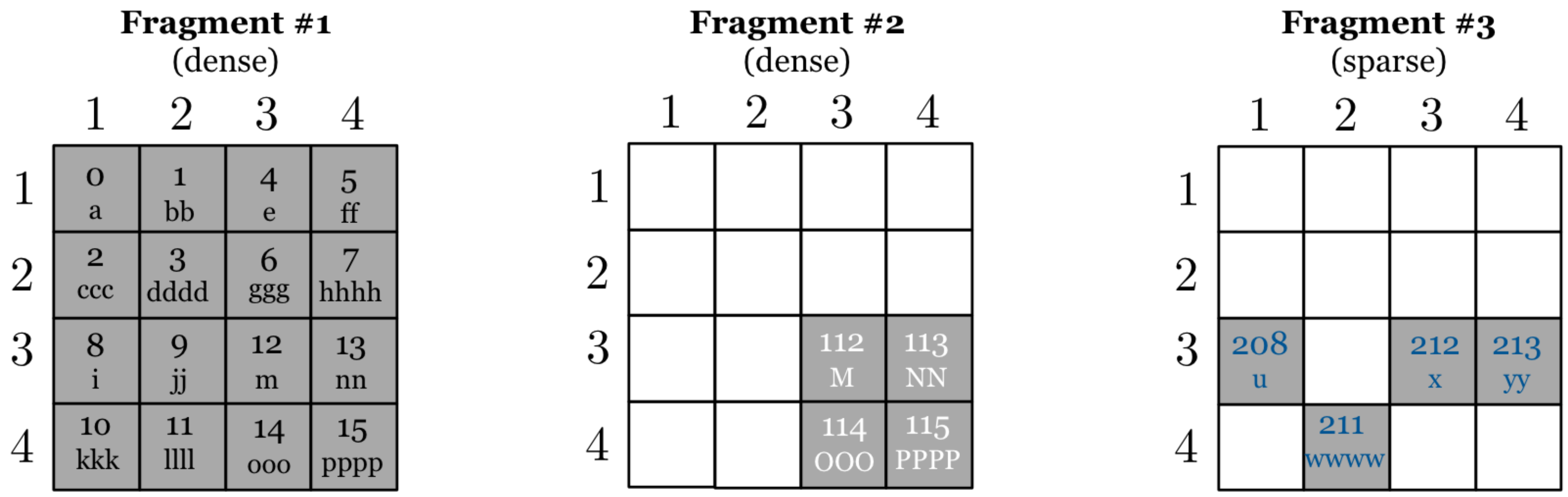
- The tile is the atomic unit of compression
- Their fixed capacity leads to balanced computations

TilleD The Universal Data Engine: A spin-off from Intel and MIT. https://tiledb.com

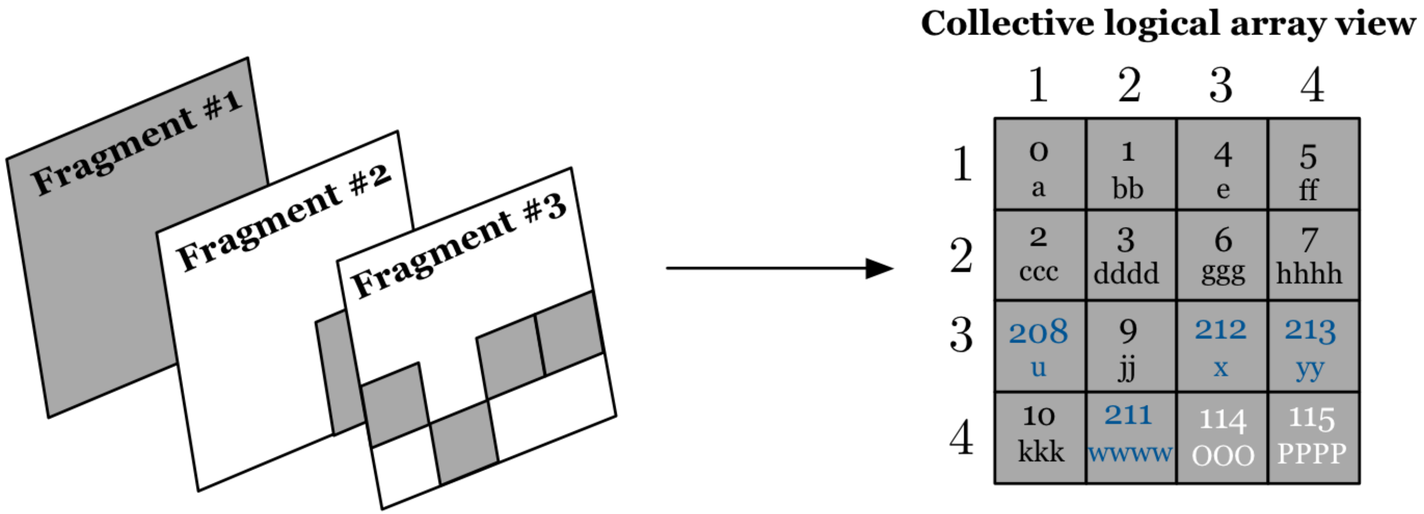
The TileDB Array Data Storage Manager, Stavros Papadopoulos, Kushal Datta, Samuel Madden, Tim Mattson, VLDB 2017

TileDB: Updates at high speed with fragments

Writes are buffered in fragments



Batches up fragments for later consolidation in the background



Provides a consistent view for reads

Source: The TileDB Array Data Storage Manager, Stavros Papadopoulos, Kushal Datta, Samuel Madden, Tim Mattson, VLDB 2017

TileDB Performance

- Extensively benchmarked versus HDF5, SciDB and Vertica[%]

- Benchmarks heavily optimized for each storage engine with help from SciDB and HDF5 teams.

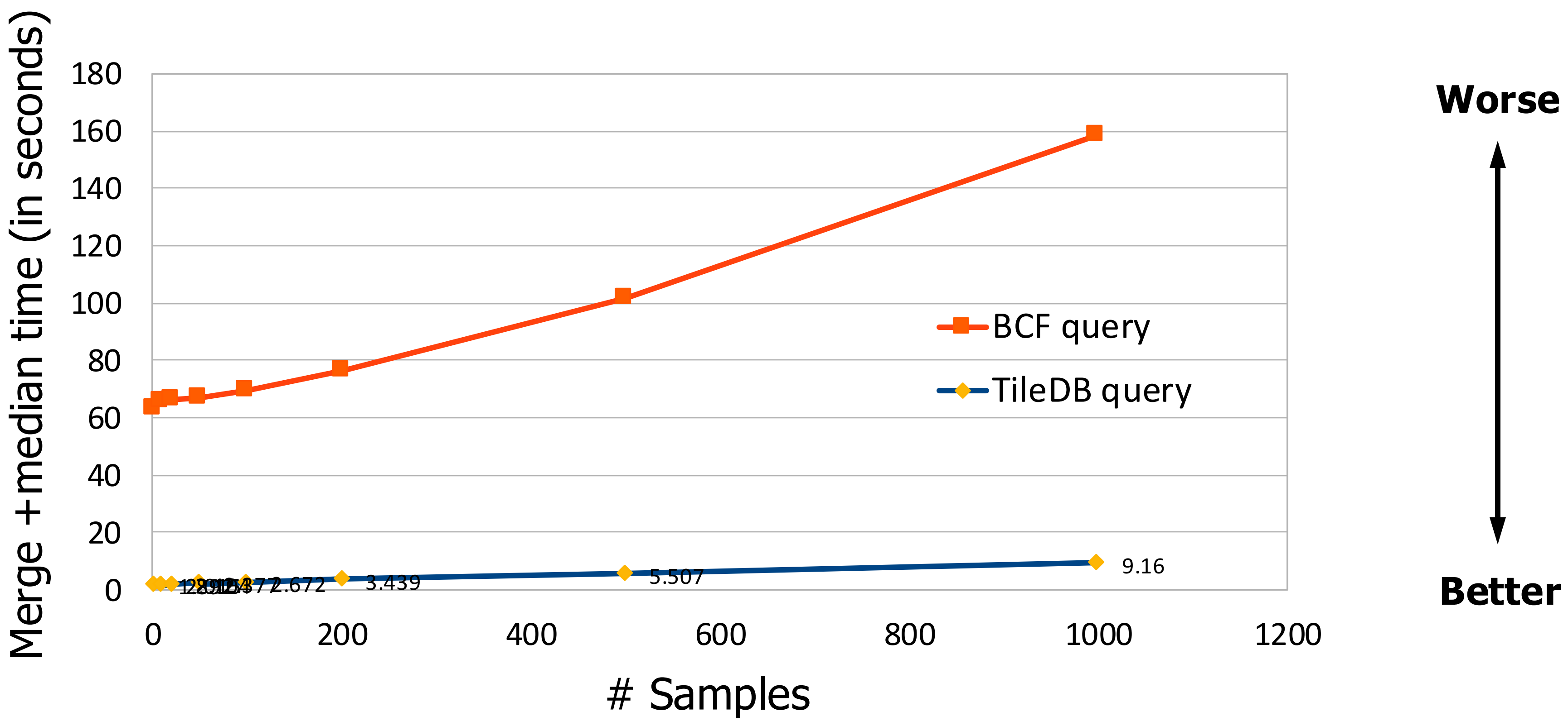
- **Takeaways:**
  - 2x-4x faster than HDF5 on dense reads and sequential writes
  - Orders of magnitude better than HDF5 on random writes
  - Orders of magnitude better than SciDB in all settings
  - Up to 40x faster than Vertica on dense arrays, 2x faster on sparse

Intel® Xeon™ platform with a 2.3 GHz 36-core CPU and 128 GB of RAM, running CentOS6. We utilized a 4 TB, 7200 rpm Western Digital HDD. SciDB v15.12, Vertica v7.02.0202, and HDF5 v.10.0.

[%]Source: Papadopoulos, Datta, Madden, Mattson, VLDB 2017

GenomicsDB: A Data Store optimized for Genomics built on top of TileDB

GenomicsDB combine gVCF operation + median (5K random positions)

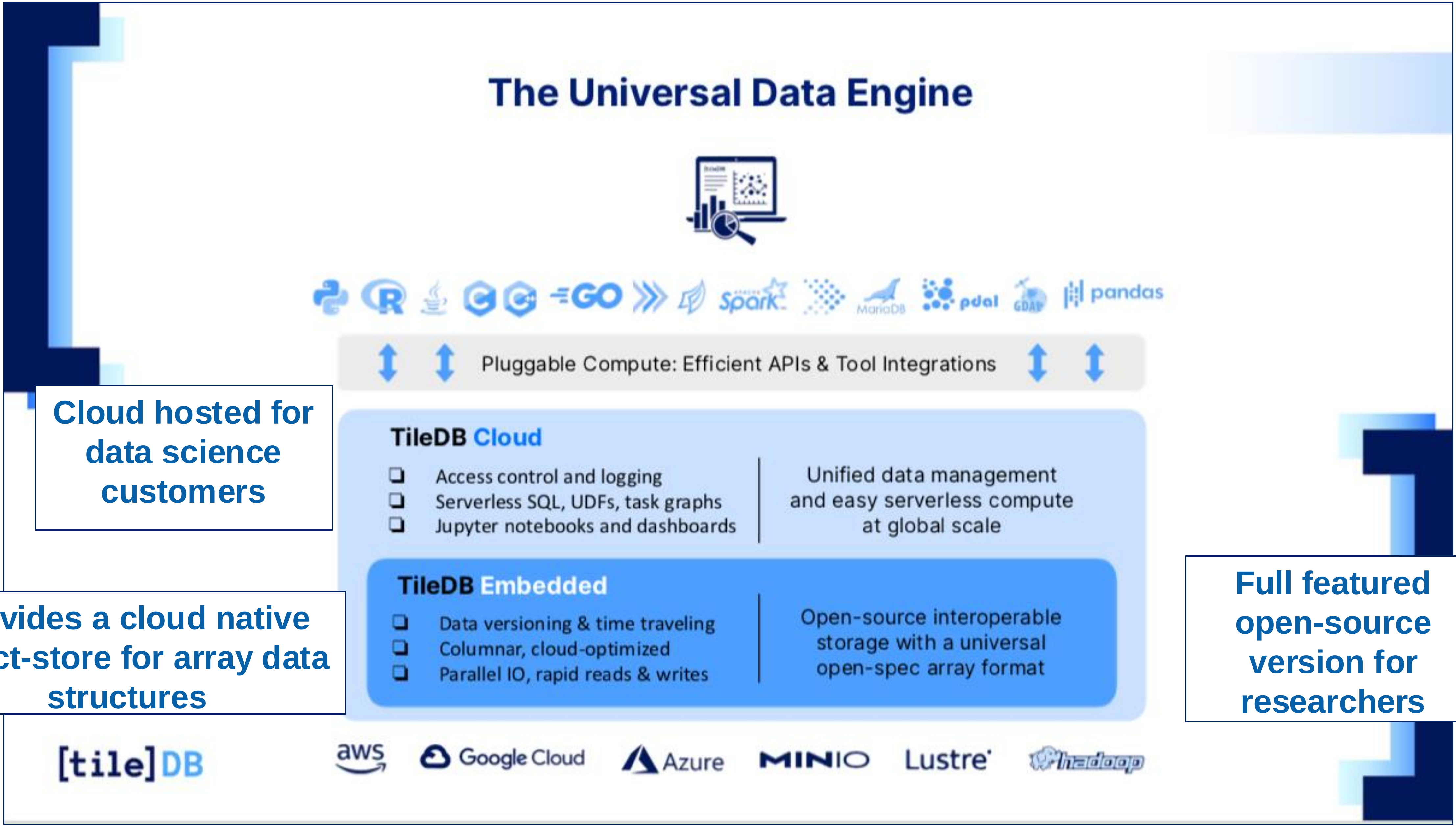


BCF refers to the Broad processing pipeline highly optimized by Intel.

gVCF is the genomic Variant Call Format used in the Broad GATK platform for genomics

Intel® Xeon® E5 2697 v2 CPU, 12 cores, dual socket, 128 GB RAM, CentOS6.6, Western Digital 4 TB WD4000F9YZ-0 as a ZFS RAID0 pool.

Open-source and commercial versions of TileDB



Cloud hosted for data science customers

Provides a cloud native object-store for array data structures

Full featured open-source version for researchers

Outline

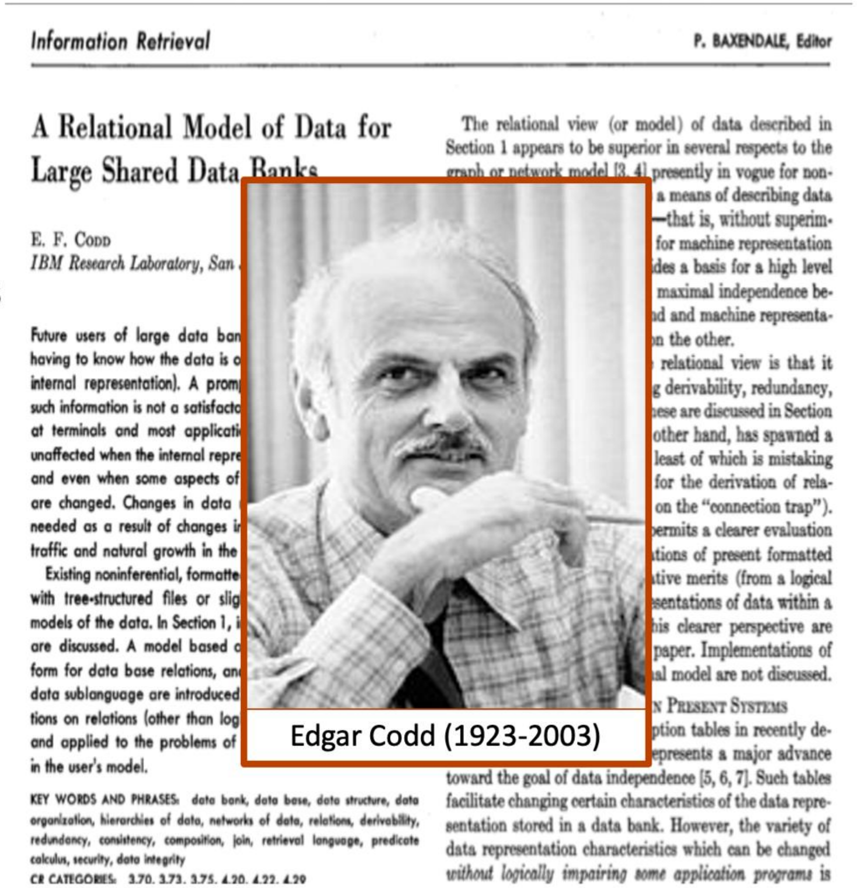
- Motivation: Why everyone needs a database management system?
- Database Technology: from ancient history to today
- Data Curation in the sciences
- My quest: One Algebra to rule them all

The importance of Algebras

- Remember how Codd's relational algebra revolutionized database management systems?

The Relational Model of Databases

- In 1970 Edgar Codd (IBM) published one of the most important papers in the history of computer science.
- It defined a formal algebra* for building databases … the **relational model**.
  - Object: A relation.
    - A set of tuples that share a set of attributes.
    - The set of attributes is defined by a schema
    - A relation is typically represented as a table.
  - A set of operators that act on relations. This set includes:
    - Select σ
    - Rename ρ
    - Join $\bowtie$
    - Project π



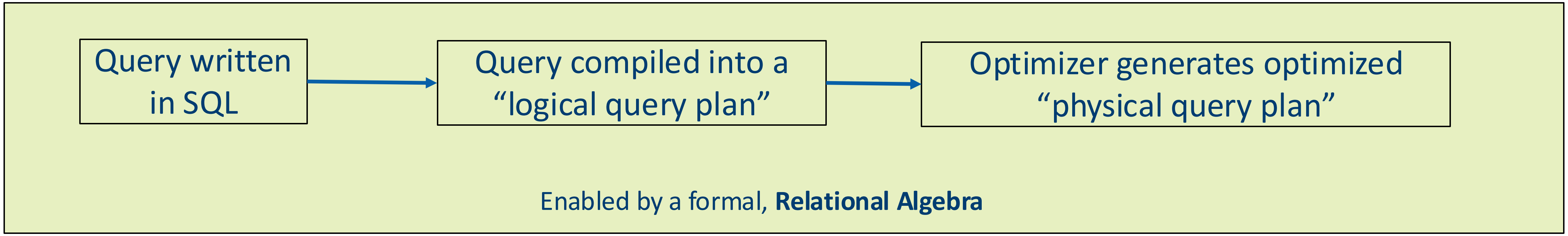
Communications of the ACM, vol 13, no. 6 p. 337, 1970

* Note: An "algebra" is a set of objects, operators that act on those objects, and rules for how those operators interact with each other

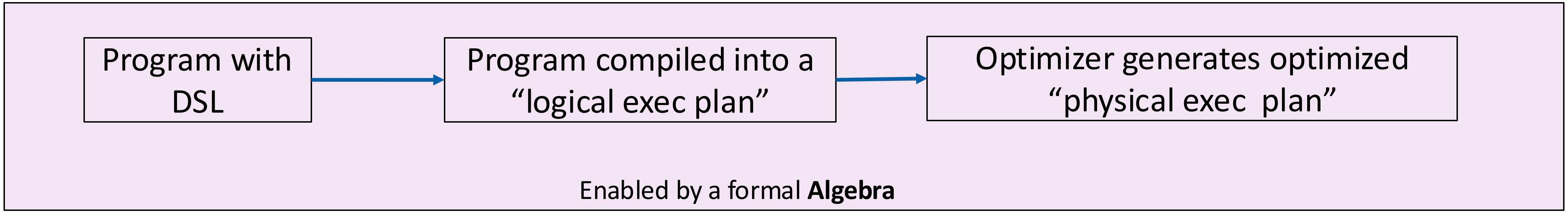
19

Productivity, Performance and Portability in one programming framework

- By the 1980s, database researchers at IBM and UC Berkeley exploited the declarative nature of SQL to build systems that delivered on the "3 Ps" …



- Can we replicate this strategy for programming heterogeneous systems?



SQL: Structured Query Language

The lesson from Edgar Codd so long ago was the power of an algebra to unify disparate approaches to a problem.

Relational algebras are great at data management, but they suck at computation. It would be stupid to build a PDE solver around a relational algebra.

So if we want ”one algebra to rule them all”, what should be our algebra?

PDE: Partial Differential Equation

Linear Algebra: One Algebra to rule them all

- Computational physics is basically applied linear algebra
  - We create differential equations from the physics, discretize domains to replace derivatives with differences, and solve resulting algebraic equations.
  - Since the differential operators are replaced by modest sized stencils, the arrays in physics problems are sparse (with a small number of exceptions such as in ab initio quantum chemistry).

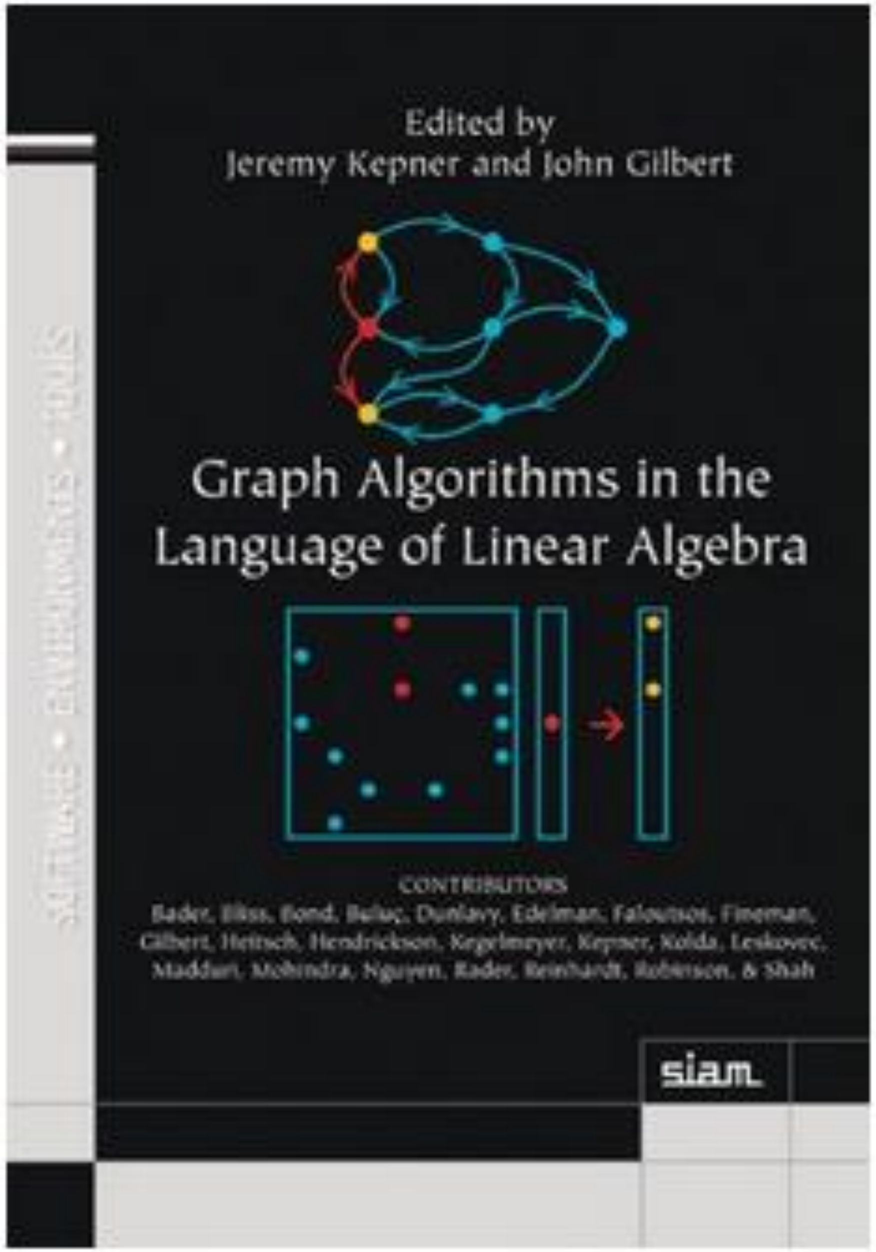
- Graphs are linear algebra, databases map onto linear algebra, science and engineering is linear algebra … if you go deep enough, in almost any field, you end up doing linear algebra.

- All we need is a good library for Sparse Linear Algebra.

Sparse Linear Algebra

If it can do graph algorithms, it can do anything!

- Graph algorithms can be represented in terms of Linear Algebra.
- This is important for Graphs, but it is also used for a wide range of applications … from engineering codes to databases.
- We need the data structures and a fundamental set of building blocks from which we can construct algorithms … We need the GraphBLAS



GraphBLAS is a specification (graphblas.org)

Mathematical Foundations of the GraphBLAS

Jeremy Kepner (MIT Lincoln Laboratory Supercomputing Center), Peter Aaltonen (Indiana University), David Bader (Georgia Institute of Technology), Aydın Buluç (Lawrence Berkeley National Laboratory), Franz Franchetti (Carnegie Mellon University), John Gilbert (University of California, Santa Barbara), Dylan Hutchison (University of Washington), Manoj Kumar (IBM), Andrew Lumsdaine (Indiana University), Henning Meyerhenke (Karlsruhe Institute of Technology), Scott McMillan (CMU Software Engineering Institute), Jose Moreira (IBM), John D. Owens (University of California, Davis), Carl Yang (University of California, Davis), Marcin Zalewski (Indiana University), Timothy Mattson (Intel)

IEEE HPEC 2016

Design of the GraphBLAS API for C

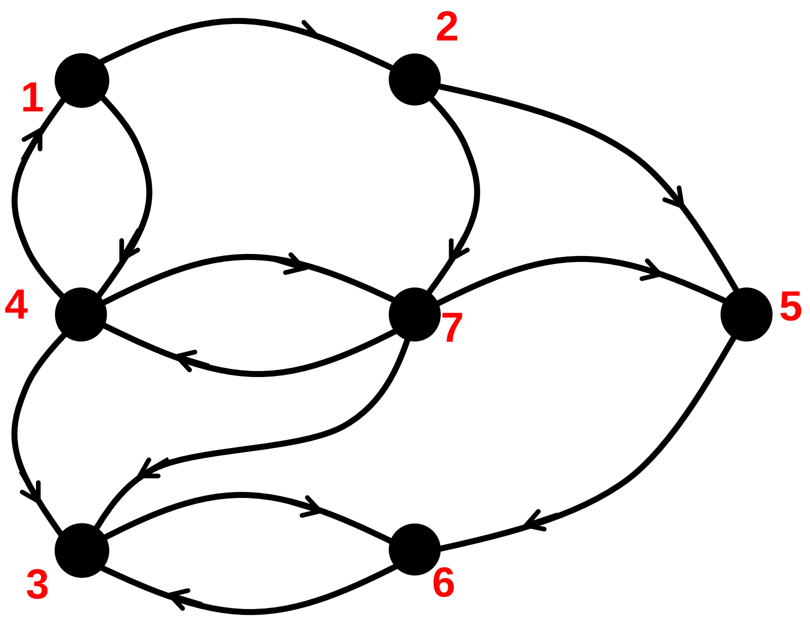
Aydın Buluç[†], Tim Mattson[‡], Scott McMillan[§], José Moreira[¶], Carl Yang[*,†]

[†]*Computational Research Division, Lawrence Berkeley National Laboratory*
[‡]*Intel Corporation*
[§]*Software Engineering Institute, Carnegie Mellon University*
[¶]*IBM Corporation*
[*]*Electrical and Computer Engineering Department, University of California, Davis, USA*

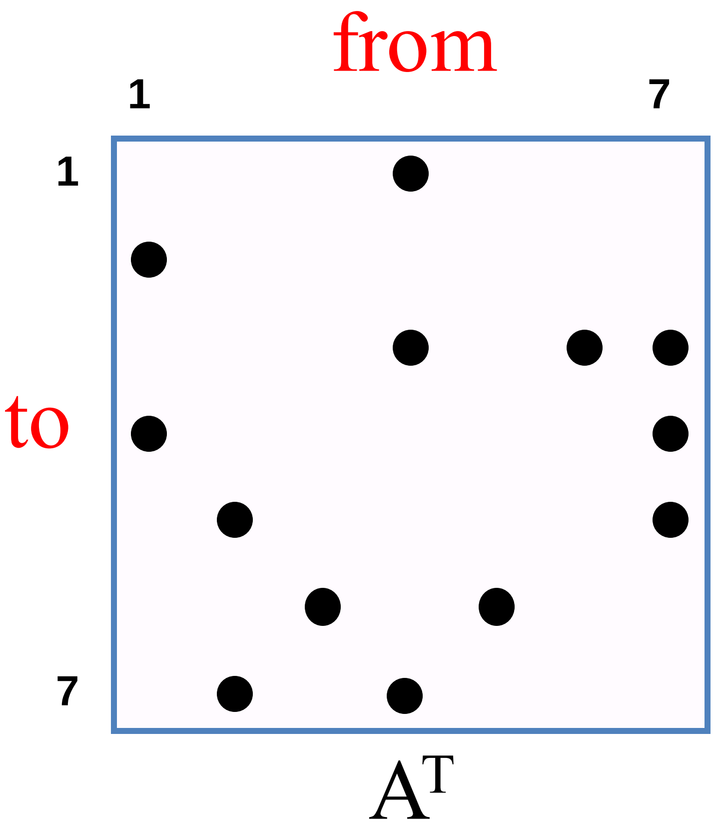
IEEE HPEC 2017

The official GraphBLAS C spec can be found at: www.graphblas.org

Graphs in the Language of Linear Algebra

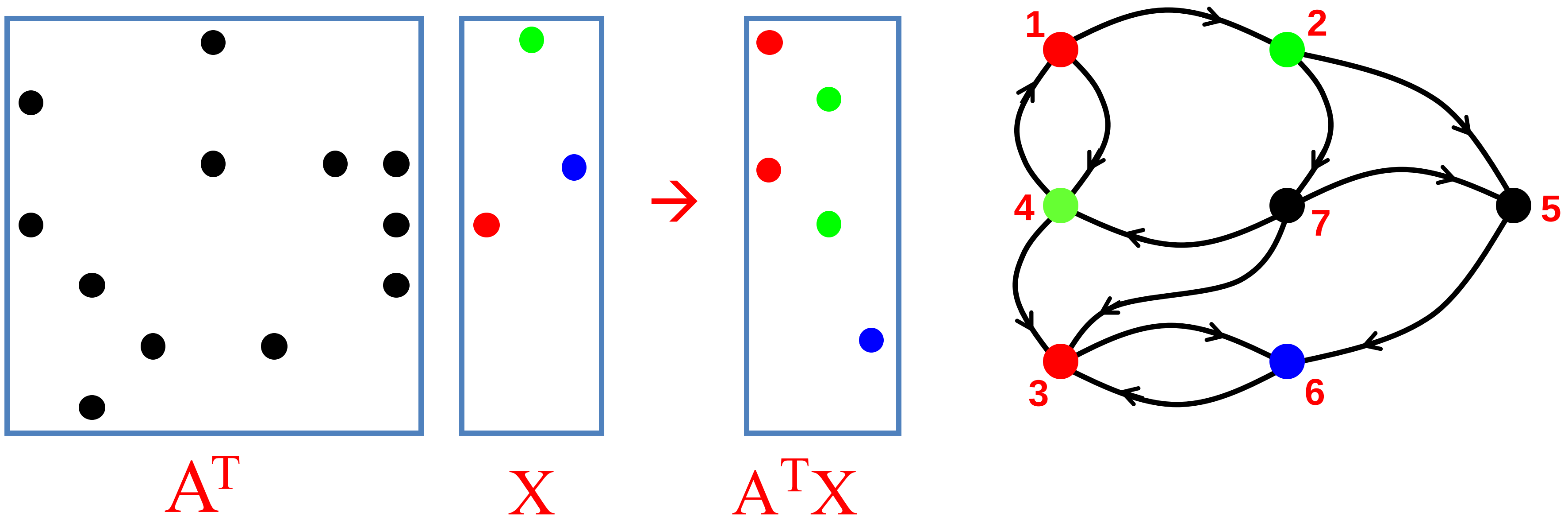


These two diagrams are equivalent representations of a graph.



A = the adjacency matrix … Elements denote edges between vertices

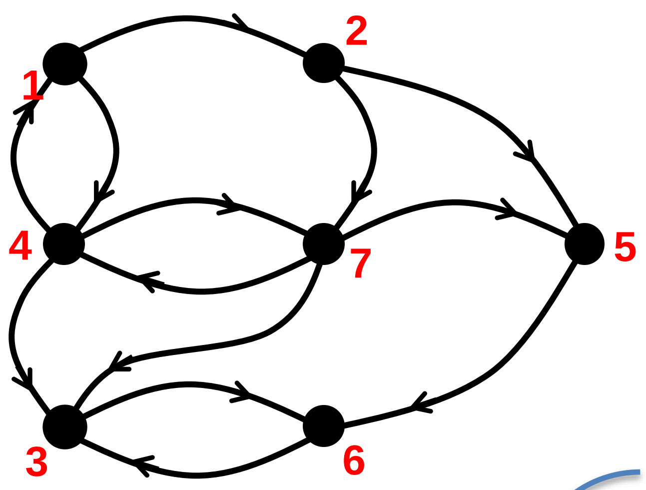
Multiple-source breadth-first search



- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges

Multiplication of sparse matrices captures breadth first search and serves as the foundation of all algorithms based on BFS

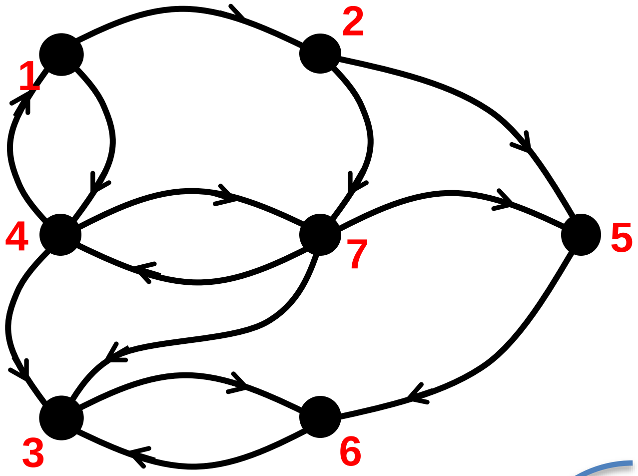
Working with paths



Consider the adjacency matrix with unit cost for "one hop" paths between vertices.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Working with paths



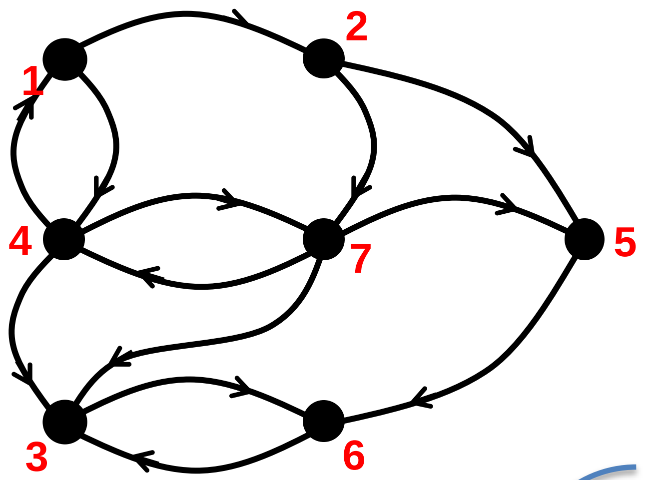
A^2 finds all the "two hop" paths in the graph.

$$A \text{ min.+} A = A^2 = \begin{pmatrix} 2 & 0 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 2 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 2 & 0 & 2 & 0 & 0 & 2 & 2 \end{pmatrix}$$

Same pattern through the matrices as familiar matrix multiply but:

- replace +/* with min/+
- Replace "zero" with identity of min (∞)

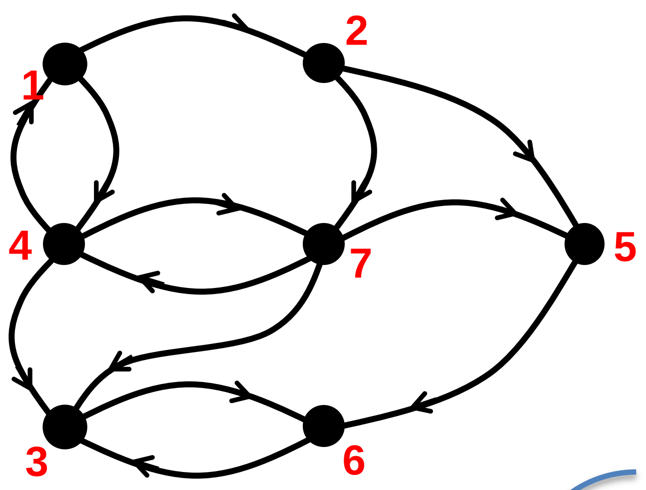
Working with paths



A^3 finds all the "three hop" paths in the graph.

$$A \text{ min.+} A^2 = A^3 = \begin{pmatrix} 0 & 3 & 3 & 3 & 3 & 3 & 0 \\ 3 & 0 & 3 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 3 & 0 & 3 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 3 & 3 & 3 & 0 \end{pmatrix}$$

Working with paths



Continue until the shortest path matrix no longer changes

In this case, Beyond A^4 the shortest paths don't change. We are done.

$$\text{Shortest paths} = \begin{pmatrix} 2 & 1 & 2 & 1 & 2 & 3 & 2 \\ 3 & 4 & 2 & 2 & 1 & 2 & 1 \\ 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 2 & 1 \\ 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 2 & 3 & 1 & 1 & 1 & 2 & 2 \end{pmatrix}$$

Generalizing Linear Algebra with Algebraic Semirings

- A semiring generalizes the operations of traditional linear algebra by replacing (+,*) with binary operations (Op1, Op2)
  - Op1 and Op2 have identity elements sometimes called 0 and 1
  - Op1 and Op2 are associative.
  - Op1 is commutative,   Op2 distributes over op1 from both left and right
  - The Op1 identify is an Op2 annihilator.

Generalizing Linear Algebra with Algebraic Semirings

- A semiring generalizes the operations of traditional linear algebra by replacing (+,*) with binary operations (Op1, Op2)
  - Op1 and Op2 have identity elements sometimes called 0 and 1
  - Op1 and Op2 are associative.
  - Op1 is commutative, Op2 distributes over op1 from both left and right
  - The Op1 identify is an Op2 annihilator.

(R, +, *, 0, 1) Real Field	Standard operations in linear algebra

Notation:	(R,	+,	*,	0,	1)
	Scalar type	Op1	Op2	Identity Op1	Identity Op2

Generalizing Linear Algebra with Algebraic Semirings

- A semiring generalizes the operations of traditional linear algebra by replacing (+,*) with binary operations (Op1, Op2)
  - Op1 and Op2 have identity elements sometimes called 0 and 1
  - Op1 and Op2 are associative.
  - Op1 is commutative,   Op2 distributes over op1 from both left and right
  - The Op1 identify is an Op2 annihilator.

(R, +, *, 0, 1) Real Field	Standard operations in linear algebra
({0,1}, |, &, 0, 1) Boolean Semiring	Graph traversal algorithms
(R U {∞},min, +, ∞, 0) Tropical semiring	Shortest path algorithms
(R U {∞}, min, *, ∞, 1)	Selecting a subgraph or contracting nodes to form a quotient graph.

The GraphBLAS Operations

Operation Name	Mathematical Notation
mxm	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A} \oplus . \otimes \mathbf{B}$
mxv	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot \mathbf{A} \oplus . \otimes \mathbf{u}$
vxm	$\mathbf{w}^T\langle\mathbf{m}^T, z\rangle = \mathbf{w}^T \odot \mathbf{u}^T \oplus . \otimes \mathbf{A}$
eWiseMult	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A} \otimes \mathbf{B}$
	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot \mathbf{u} \otimes \mathbf{v}$
eWiseAdd	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A} \oplus \mathbf{B}$
	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot \mathbf{u} \oplus \mathbf{v}$
reduce (row)	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot [\oplus_j \mathbf{A}(:, j)]$
reduce (scalar)	$s = s \odot [\oplus_{i,j} \mathbf{A}(i, j)]$
	$s = s \odot [\oplus_i \mathbf{u}(i)]$
apply	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot f_u(\mathbf{A})$
	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot f_u(\mathbf{u})$
transpose	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A}^T$
extract	$\mathbf{C}\langle\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A}(\boldsymbol{i}, \boldsymbol{j})$
	$\mathbf{w}\langle\mathbf{m}, z\rangle = \mathbf{w} \odot \mathbf{u}(\boldsymbol{i})$
assign	$\mathbf{C}\langle\mathbf{M}, z\rangle(\boldsymbol{i}, \boldsymbol{j}) = \mathbf{C}(\boldsymbol{i}, \boldsymbol{j}) \odot \mathbf{A}$
	$\mathbf{w}\langle\mathbf{m}, z\rangle(\boldsymbol{i}) = \mathbf{w}(\boldsymbol{i}) \odot \mathbf{u}$

⟨M,m⟩ are write masks (Matrix/vector). ⟨z⟩ selects replace or combine for elements outside the mask.
$\odot$ is an accumulation operator.

Sparse arrays do science simulations (HPC people have been using them for years).

Sparse arrays do Graphs.

Sparse arrays do ML

… but they can also be used in databases.

LARA DB

A Minimalist Kernel for Linear and Relational Algebra

Shana Hutchison, Bill Howe, Dan Suciu

BeyondMR @SIGMOD, 19 May 2017

UNIVERSITY of WASHINGTON

LARA

$\oplus \bowtie \otimes$

Objects:

Associative Tables

Total functions from keys to values with finite support

Attributes

Keys | Values

Default Values

k_1	k_2	v_1	v_2
		[0]	[‘’]
a	37	7	‘dan’
a	20	0	‘’
b	25	0	‘dylan'
b	20	2	‘bill’

Support

Operators:

UDFs: $\otimes$, $\oplus$, f
Think “Semiring”

Join

$\bowtie_{\otimes}$

“horizontal concat”

Union

$⧖_{\oplus}$

“vertical concat”

Extension

ext_f

“flatmap”

Join and Union adapted from:
M. Spight and V. Tropashko.
First steps in relational lattice. 2006.

Ext is a restricted form of monadic bind

One algebra to rule them all

- This is very much “work in progress”.
- We know how to do engineering/scientific computing with sparse arrays.
- We have a sophisticated storage engine for sparse arrays.
- We know we can build a full featured database with sparse arrays
- We know in principle that we can indeed create “one algebra to rule them all”. There’s just a bit of engineering work needed to pull everything together.

Conclusion

- Long ago, in scientific computing we selected problems that did not involve much data … the input/output behavior of our supercomputers was so awful we avoided I/O as much as possible.

- That is no longer the case … currently, much of scientific computing involves data.

- Hence, a computational scientist is also a data scientist.

- In this lecture, we covered the core concepts to get you started in your journey into the depths of data science. We covered.
  - What is a database and why we need to make our data useful through database technology.
  - The importance of using data-storage engines instead of “flat files”.
  - Key trends in database technology
  - A very brief survey of key data science tools in use today