CS4221
Database Applications Design and Tuning

Yao LU

2024 Semester 2

National University of Singapore
School of Computing

Course instructor

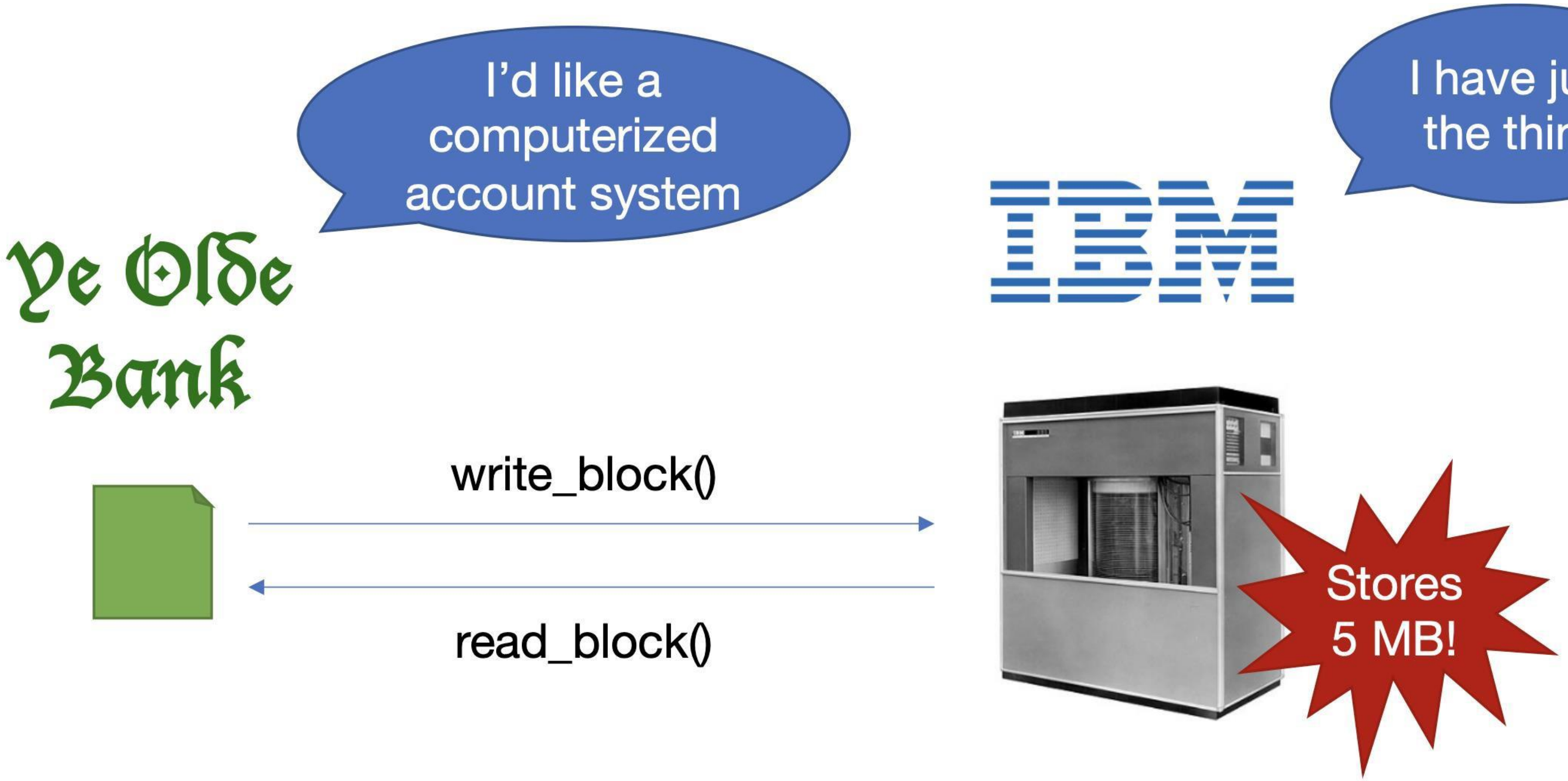
Yao LU, Assistant Professor in CS

- PhD in CS, University of Washington, 2018
- Data Systems Group, Microsoft Research Redmond, 2018-2023
- Experiences in AI, databases, cloud systems, ML systems

In memory of Stephane Bressan

1960 - 2000: Early data management

Each application did its own data management directly against storage (e.g., book-selling website).



Problems with App Storage Management

- How should we lay out and navigate data?
- How do we keep the application reliable?
- What if we want to share data across apps?

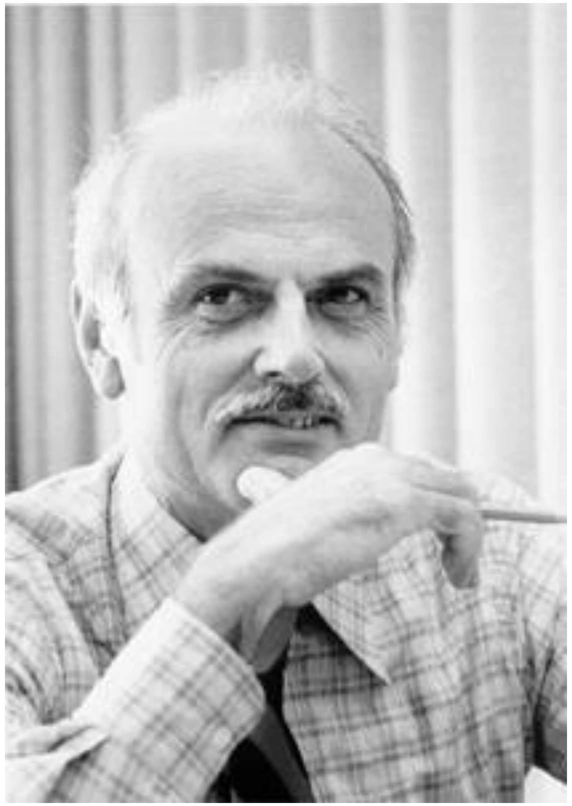
Every app is solving the *same* problems.

1970s - Relational data model

Turing Award 1981

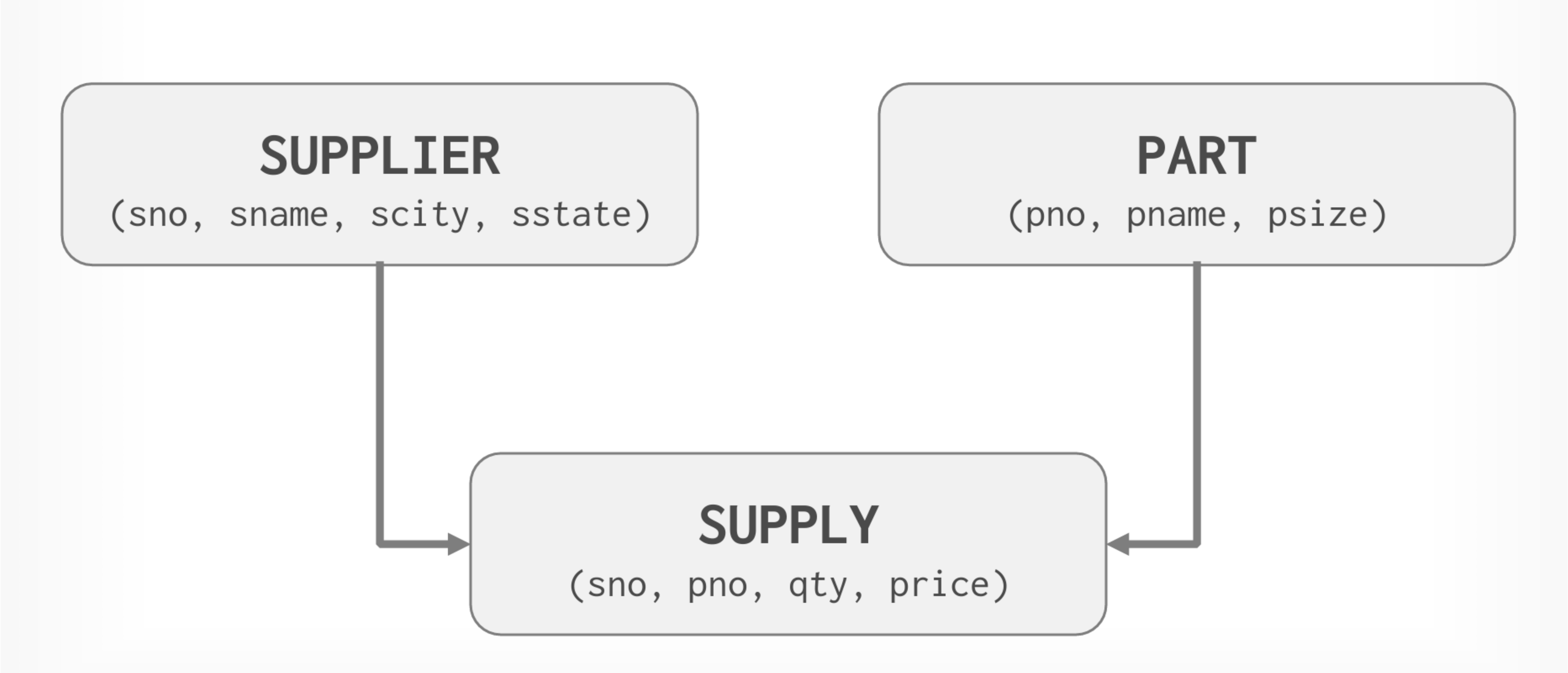
- Ted Codd was a mathematician working at IBM Research. He saw developers spending their time rewriting IMS programs every time the database's schema or layout changed.

- Database abstraction to avoid this maintenance:
    - Store database in simple data structures.
    - Access data through set-at-a-time high-level language.
    - Physical storage left up to implementation.



Codd

Relational Data Model - *schema*



Relational Data Model - *instance*

SUPPLIER

sno	sname	scity	sstate
1001	Dirty Rick	New York	NY
1002	Squirrels	Boston	MA

PART

pno	pname	psize
999	Batteries	Large

SUPPLY

sno	pno	qty	price
1001	999	10	$100
1002	999	14	$99

Database applications design and tuning

- The design question

How many tables? What tables? How many columns in each table? What columns? What constraints?

- The tuning question

In addition, what indexes? What queries? What triggers? What stored procedures? What views?

1990s – DATA CUBES

DBMSs would maintain multi-dimensional arrays as pre-computed aggregations to speed up queries.

→ Periodically refreshed materialized views.

→ Administrator had to specify cubes ahead of time.

Data cubes were often introduced in existing operational DBMSs originally designed to operate on row-oriented data.

Microsoft SQL Server

ORACLE

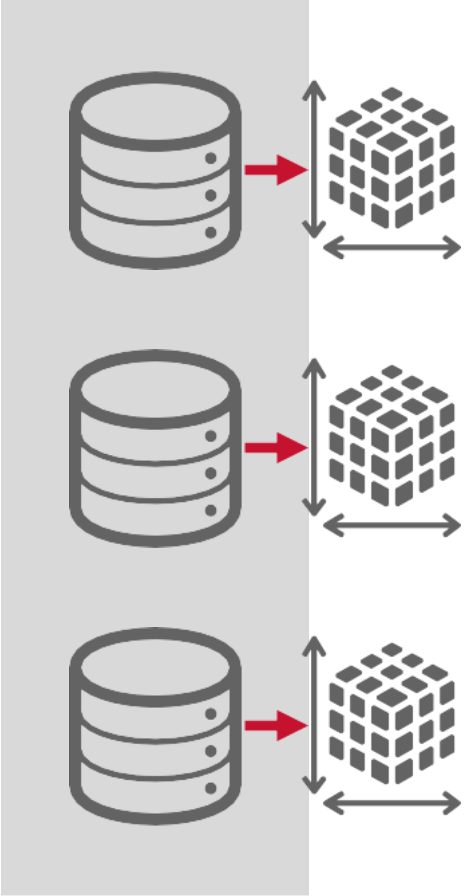
TERADATA

IBM DB2

SYBASE

Informix

1990s – DATA CUBES



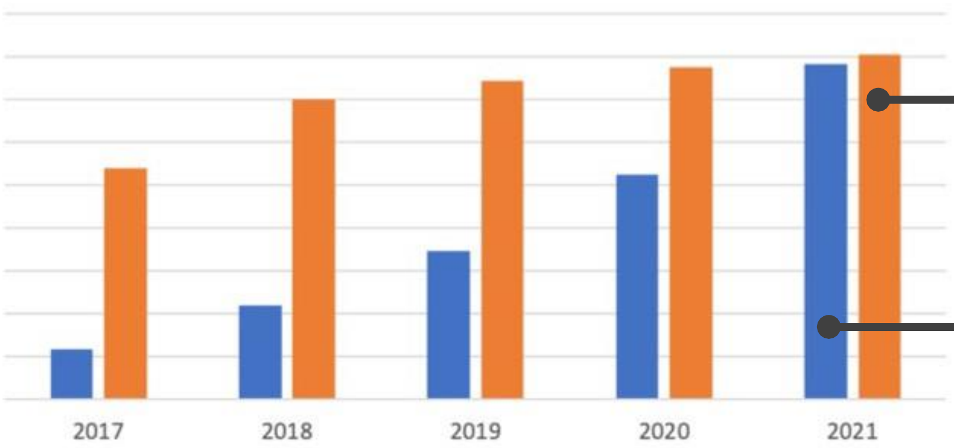
OLTP Databases

```
SELECT product, region, cdate,
  SUM(amount) AS total_sales
FROM sales GROUP BY
CUBE (product, region, cdate);
```

2000 – 2010s: Rise of cloud computing

According to Gartner Report [1]

$39.2 billion, 49% of all DBMS revenue from cloud in 2021



Cloud vs. On-premises Revenue

On-prem database
$40.8B in 2021

Cloud database
$39.2B in 2021

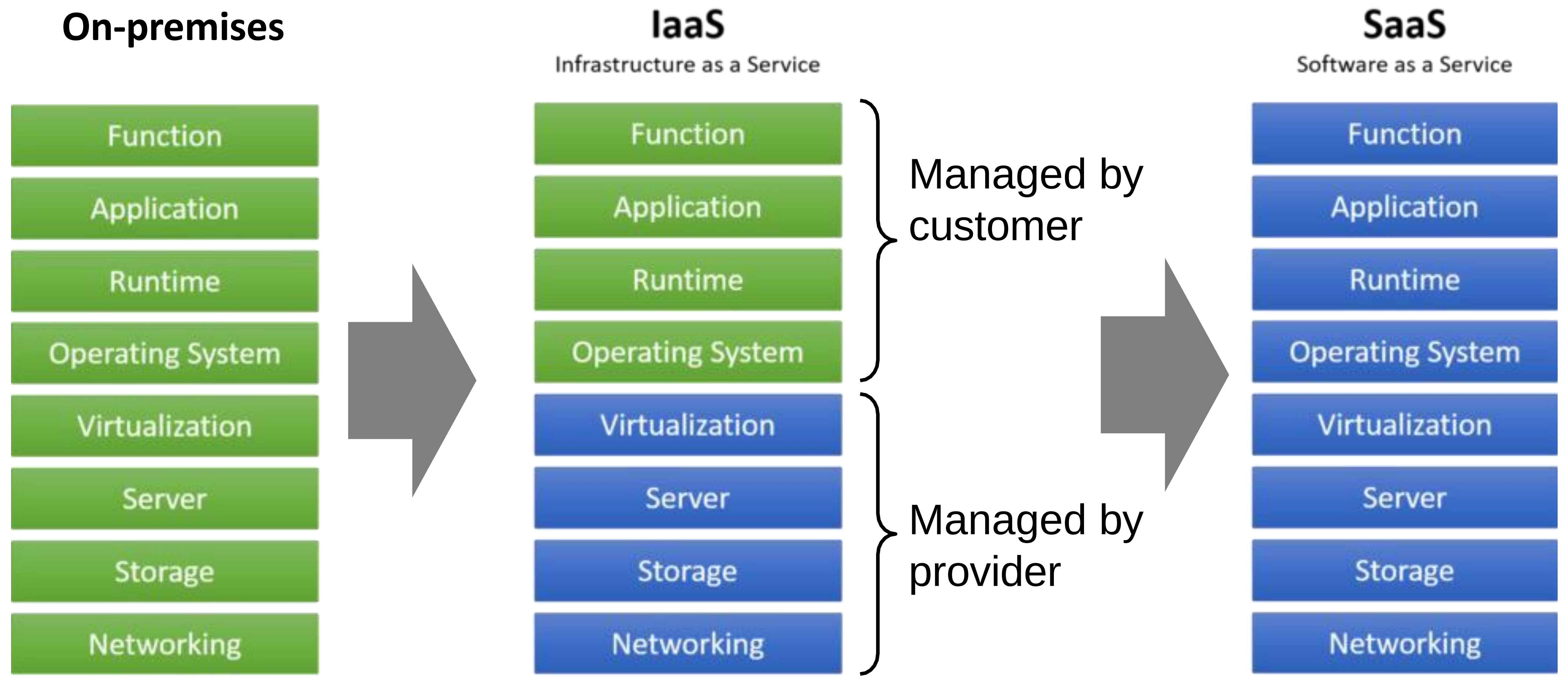
Low Cost

Elasticity

Availability

[1] DBMS Market Transformation 2021: The Big Picture, https://blogs.gartner.com/merv-adrian/2022/04/16/dbms-market-transformation-2021-the-big-picture/

2000-2010s: Rise of cloud computing



Self-manage Hardware

Self-deploy database

DB as a Service (DBaaS)

Databases moving to the cloud

Transactional DB

- Amazon Aurora
- POLARDB
- ORACLE DATABASE CLOUD SERVICE
- Orleans
- Microsoft SQL Server
- GaussDB
- TiDB
- Cockroach DB
- Cloud Spanner
- yugabyteDB
- Azure Cosmos DB

Analytical DB

- Spark SQL
- snowflake
- mongoDB
- HIVE
- VERTICA
- presto
- Amazon Athena
- amazon REDSHIFT
- Google BigQuery
- trino

New challenges in cloud databases

New Requirements

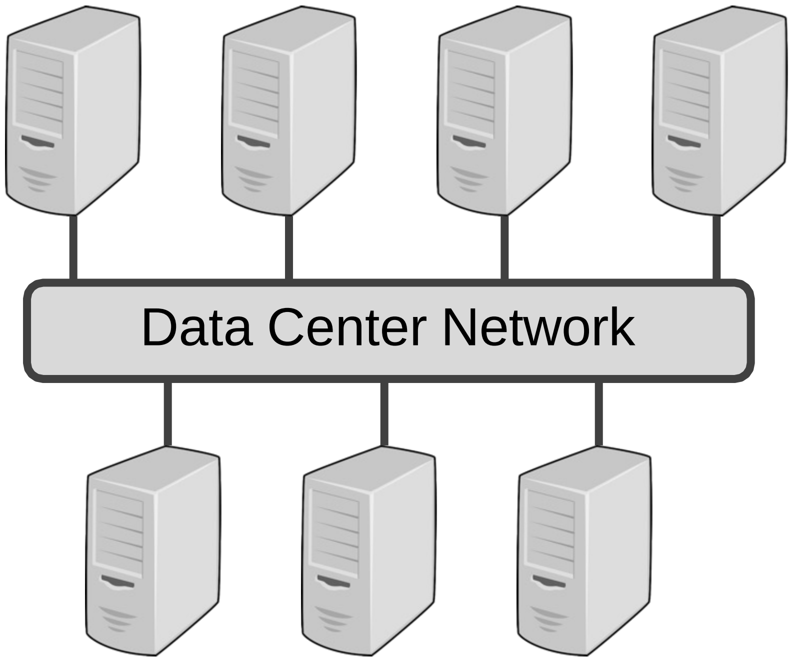
- Geo-distribution
- High availability
- Low cost
- Elasticity
- Autoscaling

→ **Higher design complexity**

Solution: Modularity in distributed system design

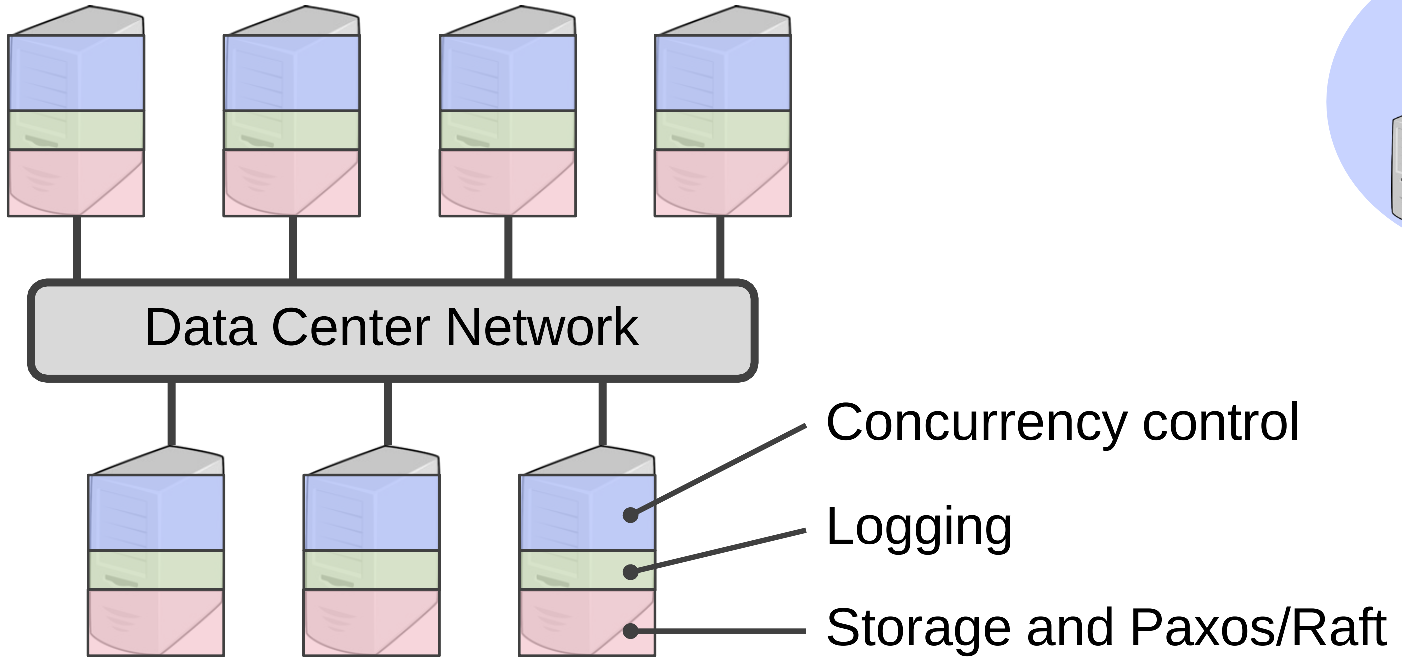
Modular distributed system design

Conventional distributed system architecture

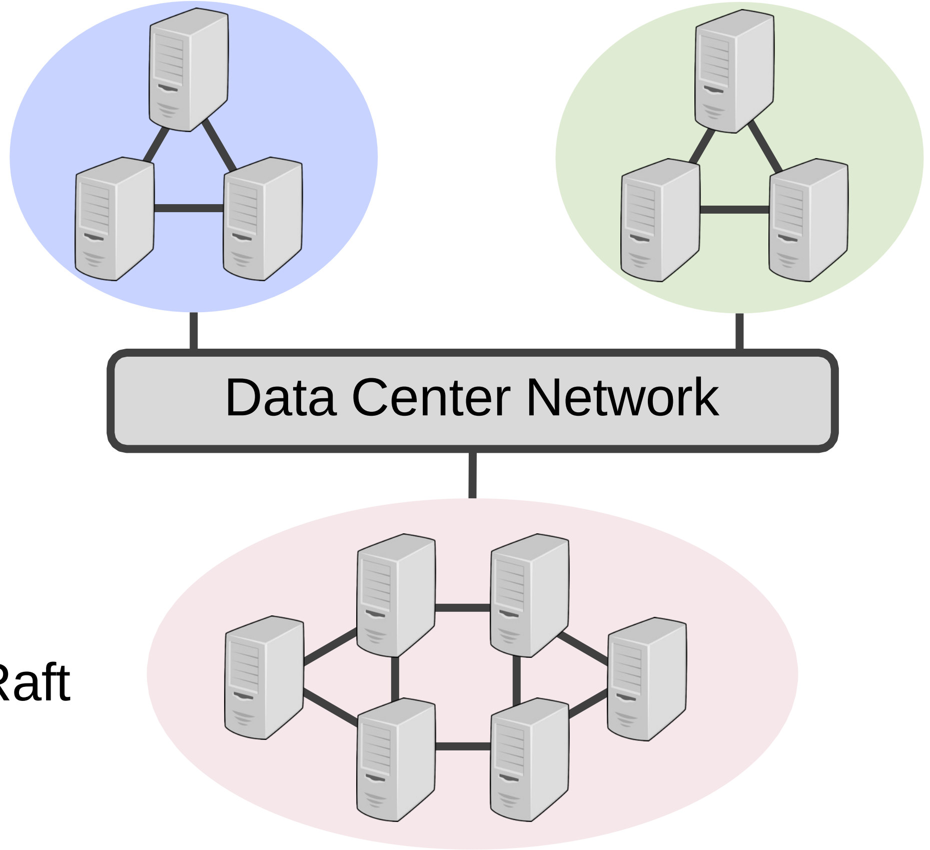


Modular distributed system design

Conventional distributed system architecture



Disaggregated distributed system architecture



Each service is deployed as a separate distributed cluster

Disaggregated distributed systems

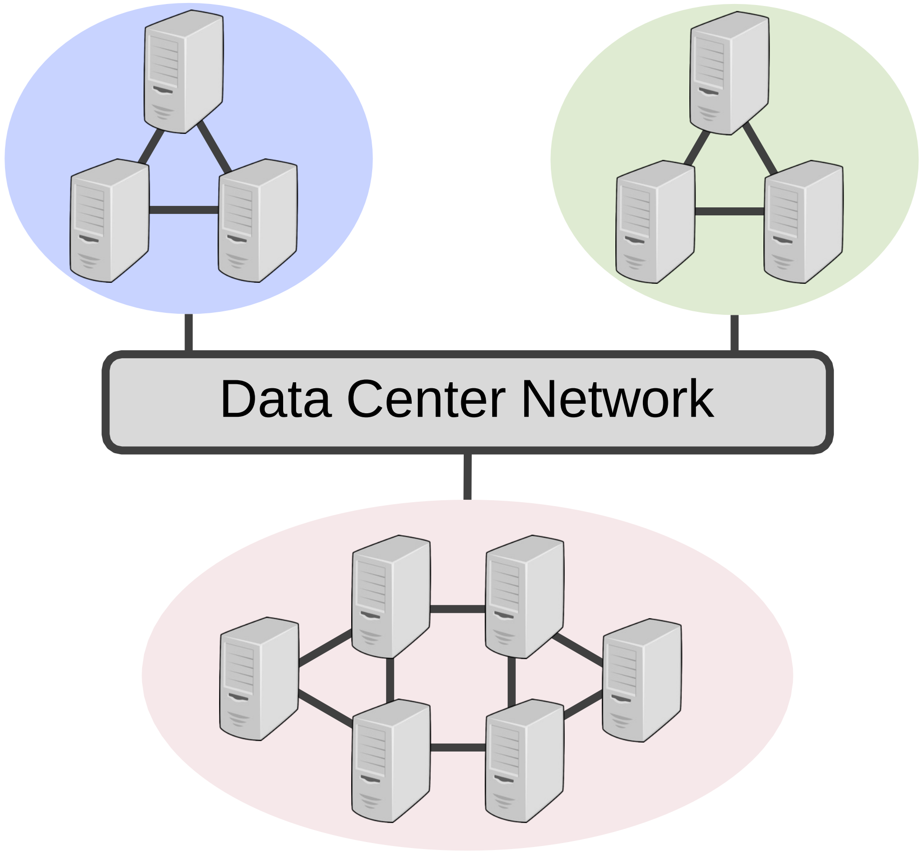
Advantages

- **Scalability**: Services can scale independently
- **Performance and cost**: Services can be custom optimized (e.g., low cost storage service)
- **Separation of concerns**: Services can be independently developed

Disadvantage

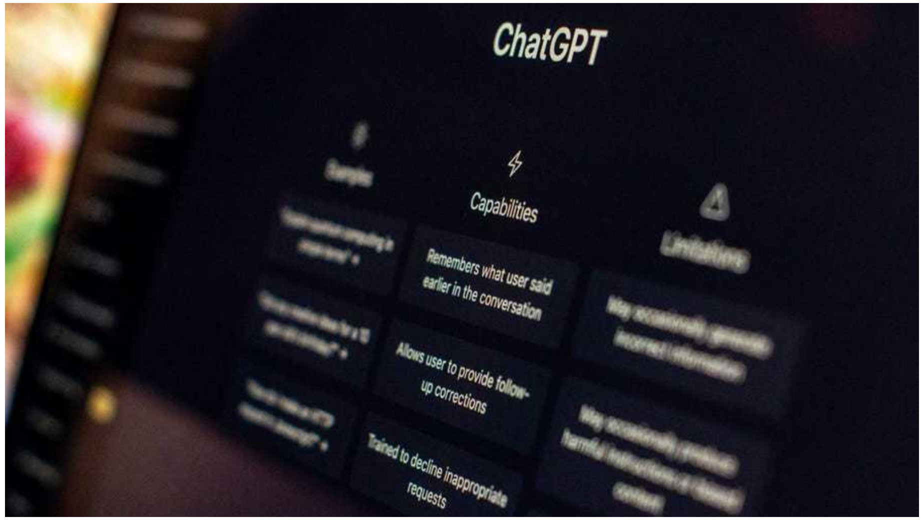
- Network can throttle performance

Disaggregated distributed system architecture



2020 – Now: Databases for Large Generative Models

Large language models and ChatGPT



Multi-modal models

2020 – Now: Large generative models

Scale of Embeddings - example: OpenAI

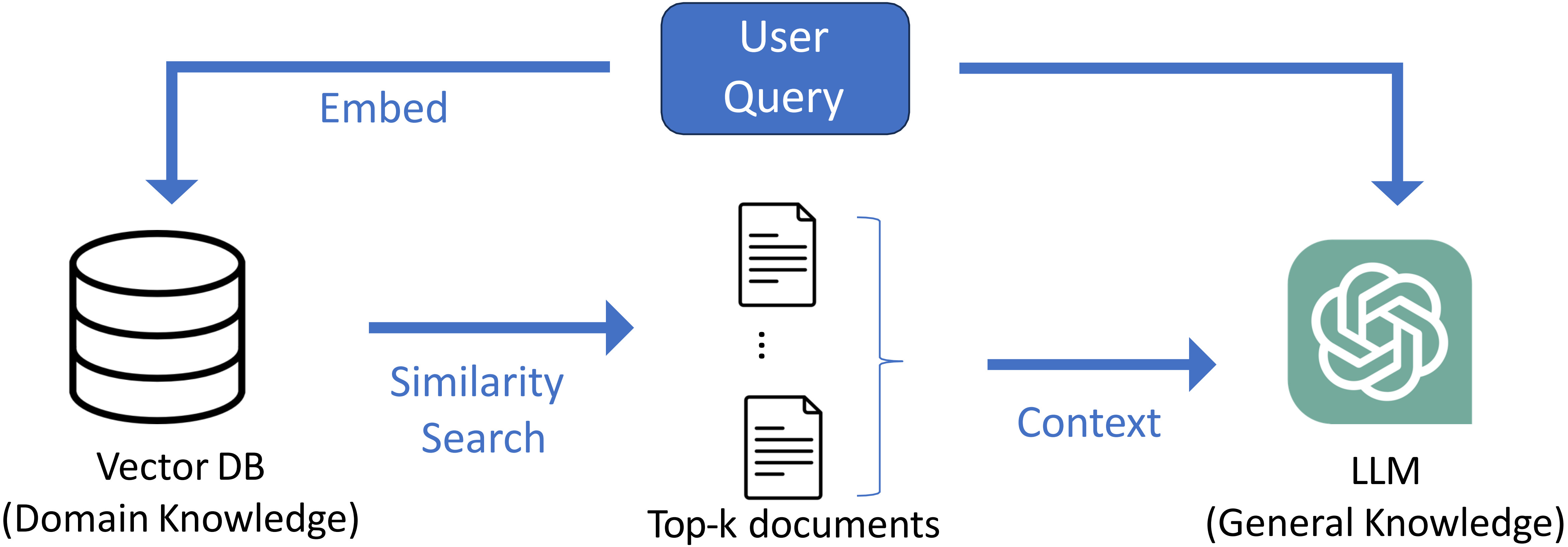
- text-embedding-3-small: 1536 dims
  - 1536 * 4 bytes = 6 KB
  - 6 KB * 1B = 6 TB
  - 6 KB * 1T = 6 PB

- text-similarity-davinci-001: 12288 dims
  - 12288 * 4 bytes = 49 KB
  - 49 KB * 1B = 49 TB
  - 49 KB * 1T = 49 PB

OpenAI

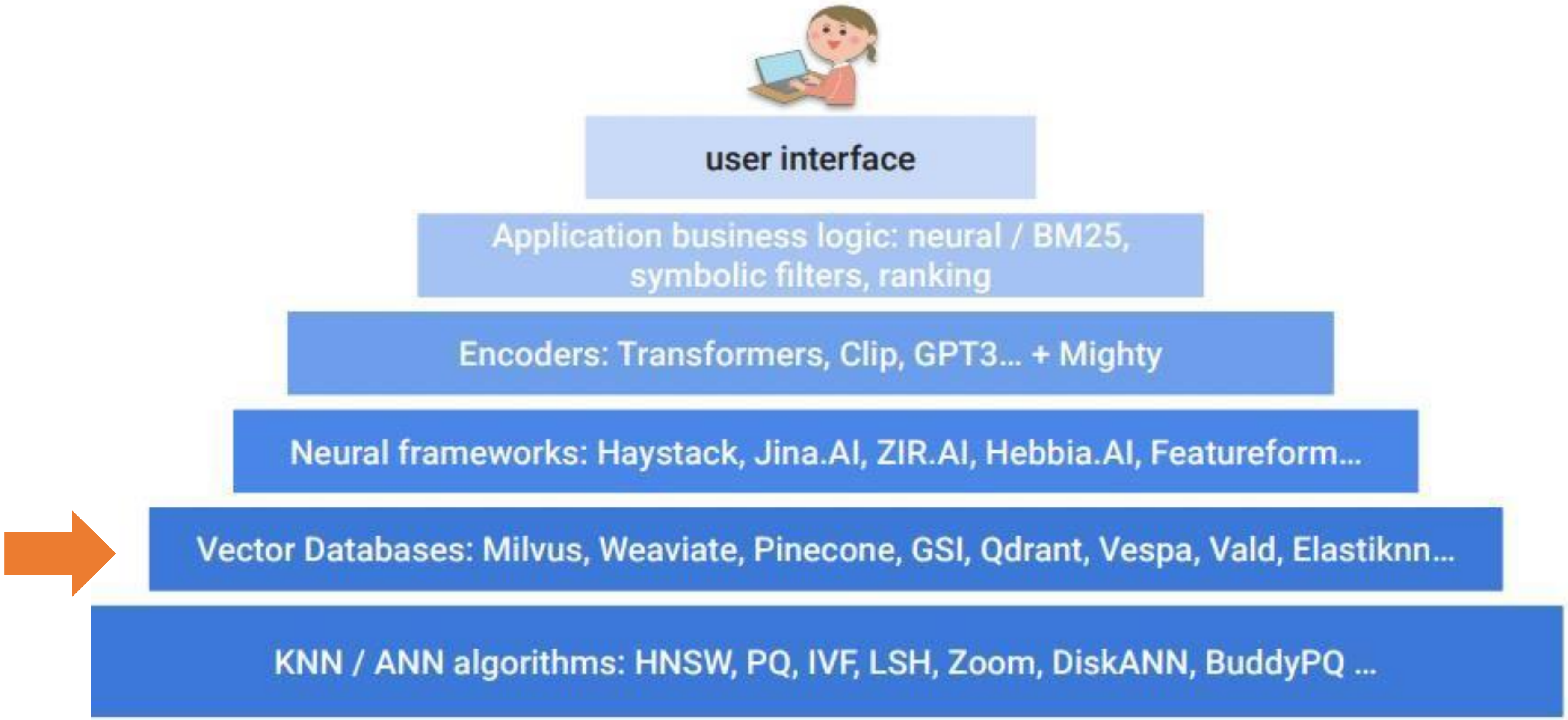
Significant memory requirement for processing billion/trillion scale vector datasets

Vector search in LLMs (Retrieval Augmented Generation)



Vector DB is in the critical path of LLM applications – we need them to be performant!

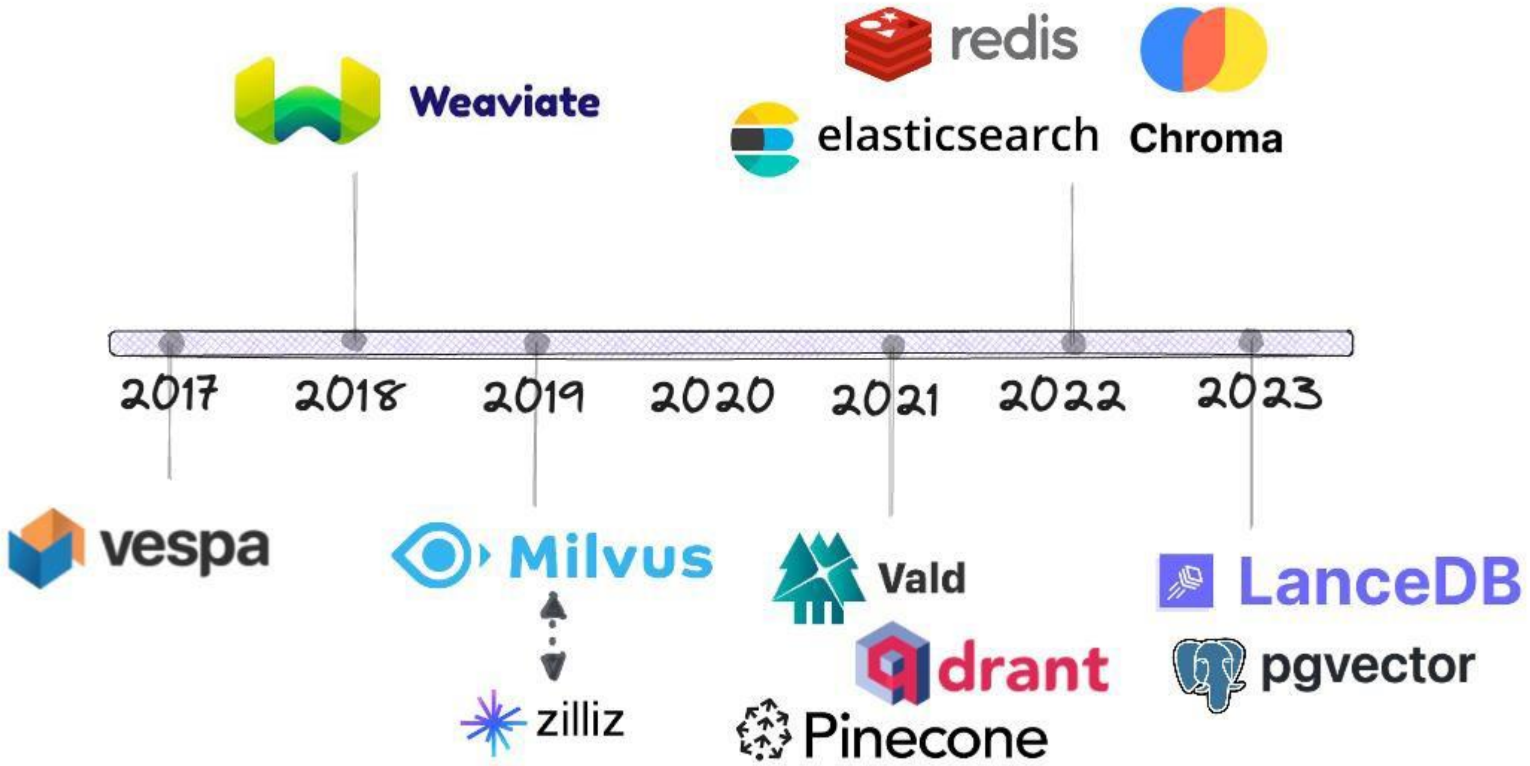
Vector search pyramid



Vector databases

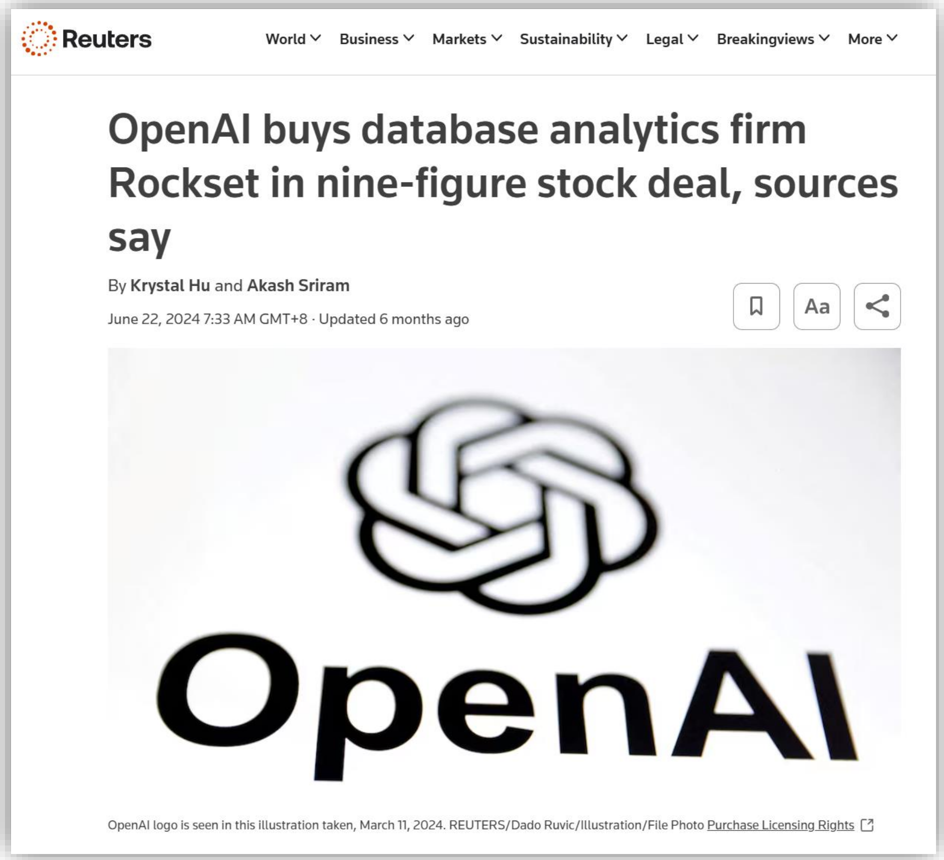
- Fast similarity searches and retrieval for high-dimensional vectors

- Consistency guarantees, multi- tenancy, cloud-native, CRUD, logging and recovery, serverless, etc



2020 – Now: Large generative models

- Training large gen models needs a huge dataset
  - Data cleaning and curation
  - Multi-modal data
  - Annotations for post-training


Reuters
World Business Markets Sustainability Legal Breakingviews More

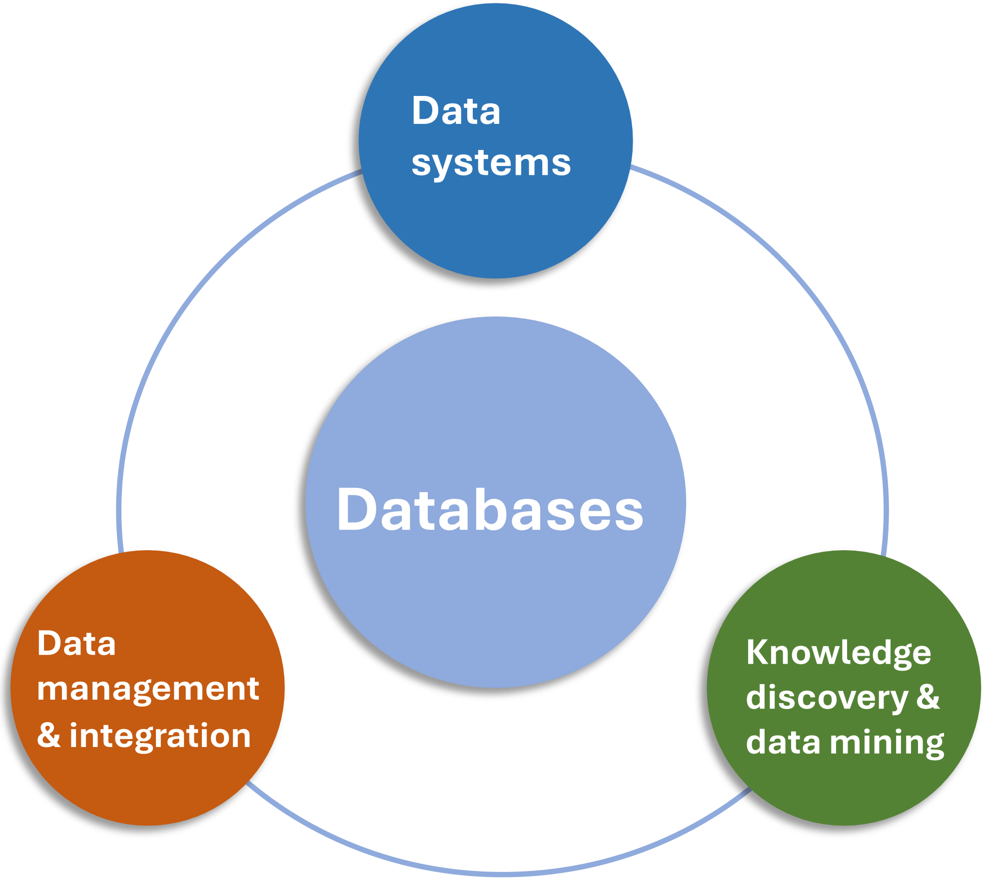
OpenAI buys database analytics firm Rockset in nine-figure stock deal, sources say

By Krystal Hu and Akash Sriram

June 22, 2024 7:33 AM GMT+8 · Updated 6 months ago

OpenAI logo is seen in this illustration taken, March 11, 2024. REUTERS/Dado Ruvic/Illustration/File Photo Purchase Licensing Rights

Databases as an evolving research field



Established conferences:
SIGMOD, VLDB, ICDE, KDD

Emerging fields:
Data-centric AI, AI for systems, security and privacy, data govornance

Databases as a startup arena

Yingjun Wu
PhD from NUS

DuckDB

RisingWave

databricks

Pinecone

DBOS

neo4j

influxdb

snowflake

Course schedule (subject to change)

Date	Lecture schedule	Tutorial schedule	HW/Proj schedule
Jan 17	Introduction	/	
Jan 24	Relational Databases I. Concepts	/	
Jan 31	Relational Databases II. Tuning Strategies	Lab 1: Relational DB design	HW1 out
Feb 07	Modern Databases I. Key-Value and Vector Databases	Lab 2: Vector DB design	
Feb 14	Modern Databases II. Streaming and Time Series Databases	Lab 3: Time series DB design	HW1 due
Feb 21	Modern Databases III. Document Databases	Lab 4: Relational DB tuning	HW2 out
Feb 28	Recess week	/	
Mar 07	Cloud Databases I: MapReduce and Spark	Project presentation: Group I	
Mar 14	Cloud Databases II: Data Lakes and Warehouses	Project presentation: Group II	HW2 due
Mar 21	Query Optimization	Project presentation: Group III	Final project out
Mar 28	Well-Being Day	/	
Apr 04	Data Integration	TBD	
Apr 11	Data Curation for Machine Learning	TBD	
Apr 18	Final project presentations	/	Time/location TBD

Course schedule (subject to change)

Date	Lecture schedule
Jan 17	Introduction
Jan 24	Relational Databases I. Concepts
Jan 31	Relational Databases II. Tuning Strategies
Feb 07	Modern Databases I. Key-Value and Vector Databases
Feb 14	Modern Databases II. Streaming and Time Series Databases
Feb 21	Modern Databases III. Document Databases
Feb 28	Recess week
Mar 07	Cloud Databases I: MapReduce and Spark
Mar 14	Cloud Databases II: Data Lakes and Warehouses
Mar 21	Query Optimization
Mar 28	Well-Being Day
Apr 04	Data Integration
Apr 11	Data Curation for Machine Learning
Apr 18	Final project presentations

Traditional relational DB → Individual DBs categorized by data model

Cloud databases & optimizations

Data integration & curation

Ultimate goal: data lake for a comprehensive application from scratch

Data model

A notation for describing data or information consists of:

- Structure of the data
- Operations on the data
- Constraints on the data

Data model

- Relational → Traditional DBMS
- Key/Value
- Graph
- Document (Semi-structured)
- Column-family

No SQL

- Array/Matrix → Machine Learning
- Hierarchical
- Network

Obsolete

The relational model

- Structure
  - Based on tables (relations)
  - Looks like an array of structs in C, but this is just one possible implementation
  - In database systems, tables are not stored as main-memory structures
  - and must take into account the need to access relations on disk

title	*year*	*length*	*genre*
Oldboy	2003	120	mystery
Ponyo	2008	103	anime
Frozen	2013	102	anime

The relational model

- Operations
  - Relational algebra
  - E.g., all the rows where genre is “anime”
- Constraints
  - E.g., Genre must be one of a fixed list of values, no two movies can have the same title

title	*year*	*length*	*genre*
Oldboy	2003	120	mystery
Ponyo	2008	103	anime
Frozen	2013	102	anime

The semi-structured model

- Structure
  - Resembles trees or graphs, rather than tables or arrays
  - Represent data by hierarchically nested tagged elements
- Operations
  - Involve following path from element to sub-elements
- Constraints
  - Involve types of values associated with tags
  - E.g., <Length> tag values are integers, each <Movie> element must have a <Length>

```
<Movies>
  <Movie title="Oldboy">
    <Year>2003</Year>
    <Length>120</Length>
    <Genre>mystery</Genre>
  </Movie>
  <Movie title="Ponyo">
    <Year>2008</Year>
    …
</Movies>
```

The key-value model

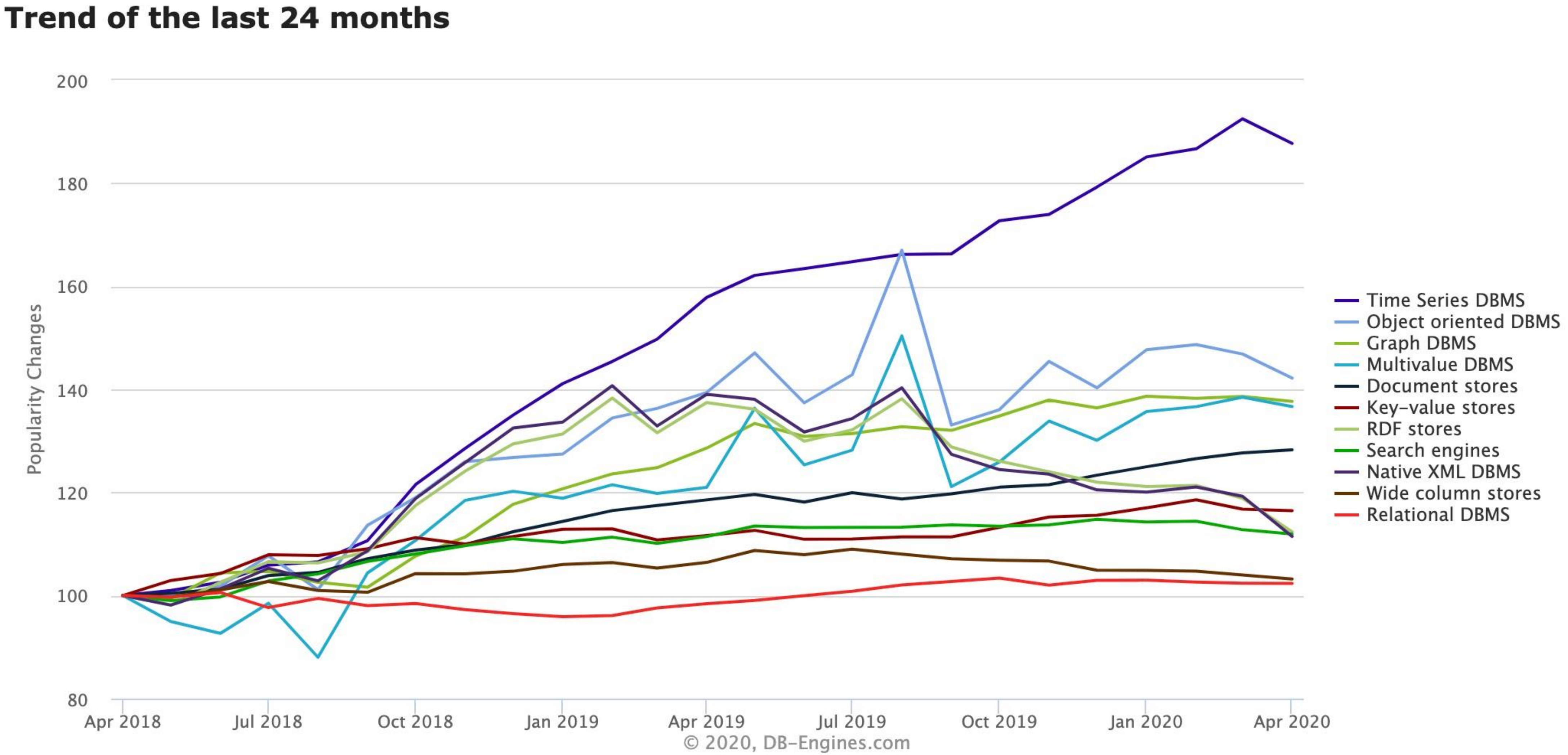
- Structure
  - (key, value) pairs
  - Key is a string or integer
  - Value can be any blob of data
- Operations
  - get (key), put(key, value)
  - Operations on values not supported
- Constraints
  - E.g., key is unique, value is not NULL

key	*value*
1000	(oldboy, 2003)
1001	(ponyo, 2008)
1002	(frozen, 2013)

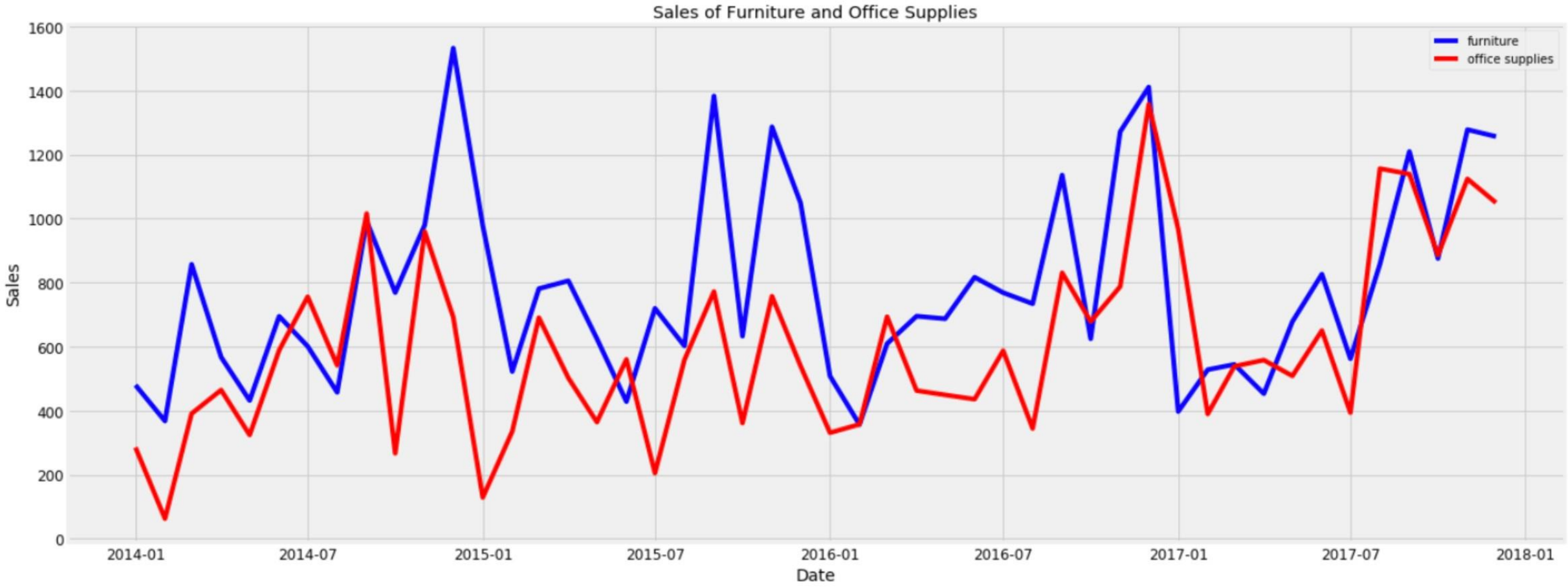
Comparison of modeling approaches

- Relational model
  - Simple and limited, but reasonably versatile
  - Limited, but useful operations
  - Efficient access to large data
  - A few lines of SQL can do the work of 1000’s of lines of C code
  - Preferred in DBMS’s
- Semi-structured model
  - More flexible, but slower to query
- Key-value model
  - Even more flexible, but cannot query

Popularity changes



Time series data



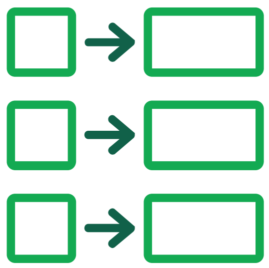
Retail

Popular time series DBs

☐ include secondary database models   33 systems in ranking, April 2020

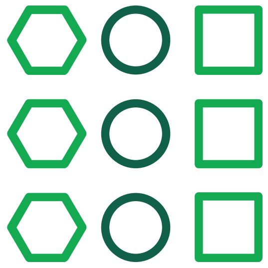
Rank Apr 2020	Rank Mar 2020	Rank Apr 2019	DBMS	Database Model	Score Apr 2020	Score Mar 2020	Score Apr 2019
1.	1.	1.	InfluxDB	Time Series	21.62	-0.81	+4.40
2.	2.	2.	Kdb+	Time Series, Multi-model	5.27	-0.08	-0.57
3.	3.	↑ 4.	Prometheus	Time Series	4.25	+0.09	+1.34
4.	4.	↓ 3.	Graphite	Time Series	3.43	-0.01	+0.30
5.	5.	5.	RRDtool	Time Series	2.61	-0.10	-0.09
6.	6.	6.	OpenTSDB	Time Series	2.00	+0.02	-0.37
7.	↑ 8.	7.	Druid	Multi-model	1.92	+0.07	+0.28
8.	↓ 7.	8.	TimescaleDB	Time Series, Multi-model	1.87	-0.01	+0.92
9.	9.	↑ 11.	FaunaDB	Multi-model	0.87	-0.07	+0.50
10.	10.	↓ 9.	KairosDB	Time Series	0.55	+0.00	-0.08
11.	11.	↑ 13.	GridDB	Time Series, Multi-model	0.44	-0.02	+0.12
12.	12.		Alibaba Cloud TSDB	Time Series	0.40	+0.01	
13.	13.	↓ 10.	eXtremeDB	Multi-model	0.37	-0.02	-0.03
14.	14.	↓ 12.	Amazon Timestream	Time Series	0.34	0.00	+0.00
15.	15.	↑ 26.	DolphinDB	Time Series	0.31	0.00	+0.31
16.	16.	↓ 15.	IBM Db2 Event Store	Multi-model	0.30	+0.01	+0.05

Non-relational data modellings

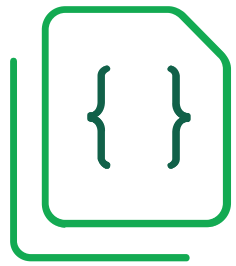


Key/Value

Graph



Column



Document

Document database

- Structure
  - Polymorphic data models
  - Each document contains markup that identifies fields and values
- Strengths
  - Obvious relationships using embedded arrays and documents
  - No complex mapping

```
{
  "_id":
ObjectId("6ef8d4b32c9f12b6d4a")
,
  "user_id": "John Watson",
  "age": 45,
  "address":
     {
       “Country: “England”
       “City”: “London”,
       “Street”: “221B Baker
     St.”
     },
  “Medical license”: “Active”
}
```

Popular document DBs

☐ include secondary database models — 58 systems in ranking, January 2025

Rank			DBMS	Database Model	Score		
Jan 2025	Dec 2024	Jan 2024			Jan 2025	Dec 2024	Jan 2024
1.	1.	1.	MongoDB	Document, Multi-model	402.50	+2.12	-14.98
2.	2.	↑ 3.	Databricks	Multi-model	87.85	+0.16	+7.31
3.	3.	↓ 2.	Amazon DynamoDB	Multi-model	73.00	+0.27	-7.94
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model	22.96	-0.10	-10.51
5.	5.	5.	Couchbase	Multi-model	16.01	+0.18	-5.18
6.	6.	6.	Firebase Realtime Database	Document	13.11	+0.05	-3.52
7.	7.	7.	CouchDB	Document, Multi-model	7.78	+0.08	-5.34
8.	8.	↑ 9.	Realm	Document	6.99	-0.13	-1.12
9.	9.	↓ 8.	Google Cloud Firestore	Document	6.73	+0.15	-4.39
10.	10.	↑ 11.	Aerospike	Multi-model	5.05	-0.29	-1.71
11.	11.	↓ 10.	MarkLogic	Multi-model	3.95	-0.37	-3.83
12.	12.	↑ 13.	Google Cloud Datastore	Document	3.80	-0.16	-1.19
13.	13.	↓ 12.	Virtuoso	Multi-model	3.32	-0.26	-1.76
14.	14.	↑ 17.	Oracle NoSQL	Multi-model	3.24	+0.03	-0.47
15.	15.	↓ 14.	ArangoDB	Multi-model	2.92	-0.04	-1.43

So far, you are (mostly) dealing with OLTP

- OLTP: OnLine Transactional Processing
  - Often used to store and manage relevant data to the day-to-day operations of a system or company.
    - e.g., ATM transactions, online hotel bookings
  - INSERT, UPDATE, DELETE commands
  - Handles real-time transactions (response times often in milliseconds)
  - ACID properties are often important

- This is where relational databases shine!

Another important topic: OLAP

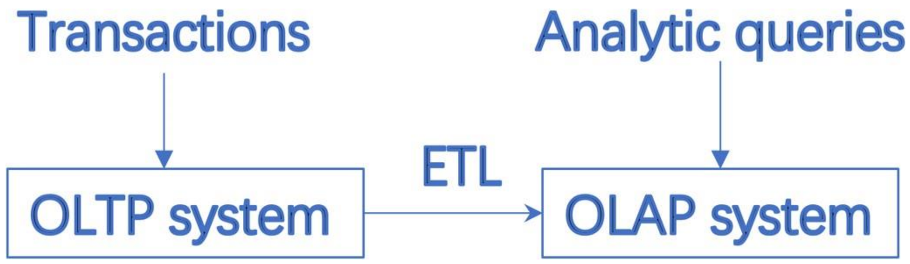
- OLAP: OnLine Analytical Processing
  - Also known as decision support or business intelligence (BI), but now BI has grown to include more (e.g., AI)
  - A specialization of relational databases that prioritizes the reading and summarizing large volumes (TB, PB) of relational data to understand high- level trends and patterns
    - e.g., the total sales figures of each type of Honda car over time for each county
  - “Read-only” queries

- Contrast this to OLTP
  - “Read-write” queries
  - Usually touch a small amount of data
    - e.g., append a new car sale into the sales table

Another important topic: OLAP

- Usually, OLAP is performed on a separate data warehouse away from the critical path of OLTP transactions (a live/transactional database).

- This data warehouse is periodically updated with data from various sources (e.g., once an hour or once a day)
  - This is through a process of ETL (Extract, Transform, Load)
  - Extract useful business that needs to be summarized, transform it (e.g., canonicalize values, clean it up), load it in the data warehouse
  - By doing it periodically, this data warehouse can become stale



OLAP in data warehouses

snowflake

amazon REDSHIFT

Google BigQuery

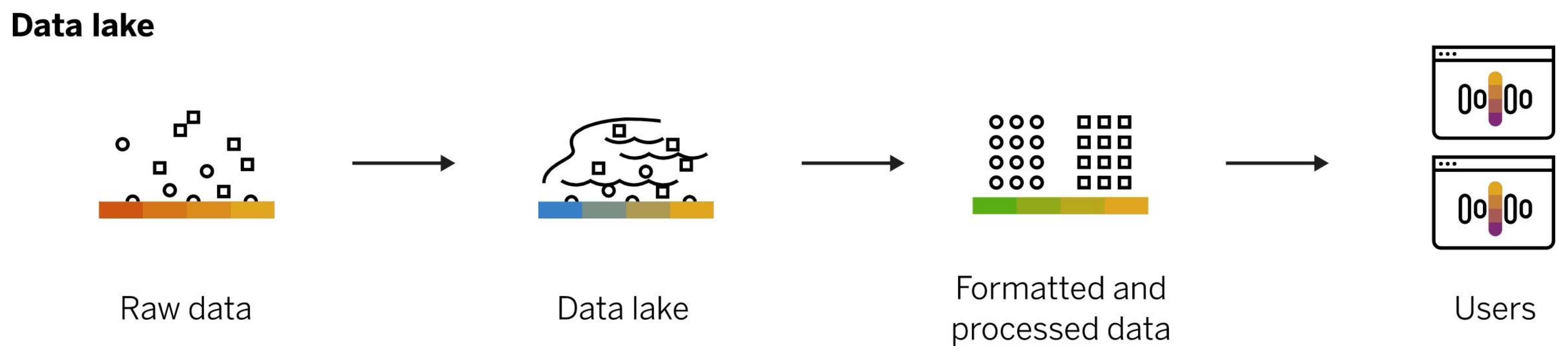
IBM DB2

Data warehouse vs. data lake

- Data warehouse:
  - Structured data (schema-on-write)
  - Expensive for large data volumes
  - Managers and business analysts

Data warehouse vs. data lake

- Data lake:
  - Raw data, can be unstructured (schema-on-read)
  - Low-cost storage, but no transactions, data quality checks
  - Data scientists and engineers



Data lake + data warehouse = ?

- Observation #1: People want to execute more than just SQL on data.
- Observation #2: Decoupling data storage from DBMS reduces ingest/egress barriers.
- Observation #3: Most data is unstructured / semi-structured.

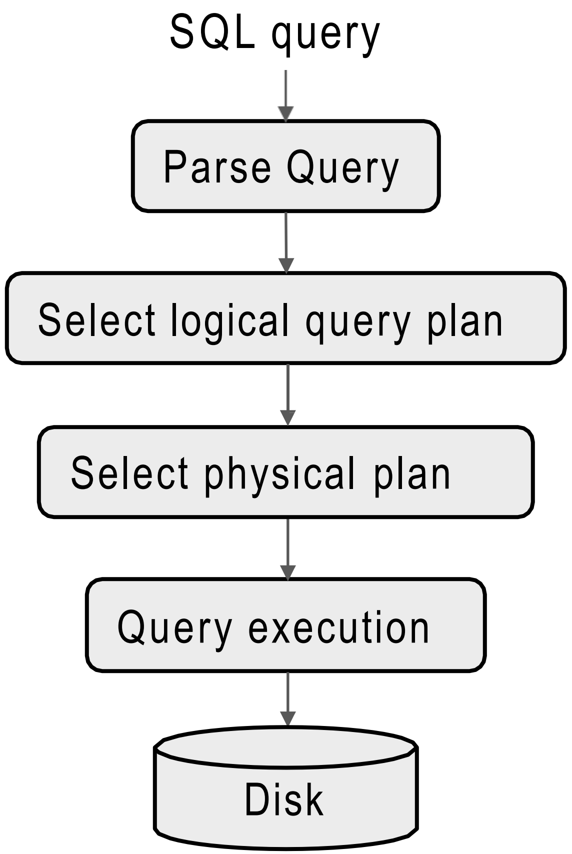
Data lake + data warehouse = lakehouse

- Middleware for data lakes that adds support for better schema control / versioning with transactional CRUD operations.
  - → Store changes in row-oriented log-structured files with indexes.
  - → Periodically compact recently added data into read-only columnar files.

- We will not be covering this aspect of these systems in this course.

LAKEHOUSE: A NEW GENERATION OF OPEN PLATFORMS THAT UNIFY DATA WAREHOUSING AND ADVANCED ANALYTICS
CIDR 2021

Putting it all together: data systems architecture



Translate to RA expression and find logically equivalent but more efficient plans

Cost-based query optimization: estimate cost and select physical plan with the smallest cost

Query execution (e.g., run join algorithms against tuples on disk)

Query optimization: select physical plan

- A logical query plan is turned into a physical query plan
  - Algorithm for each operator
  - Order of execution
  - How to access relations

$$\pi_{starName}$$

$$\sigma_{year\ =\ 2008\ AND\ studioName\ =\ \text{‘Ghibli’}}$$

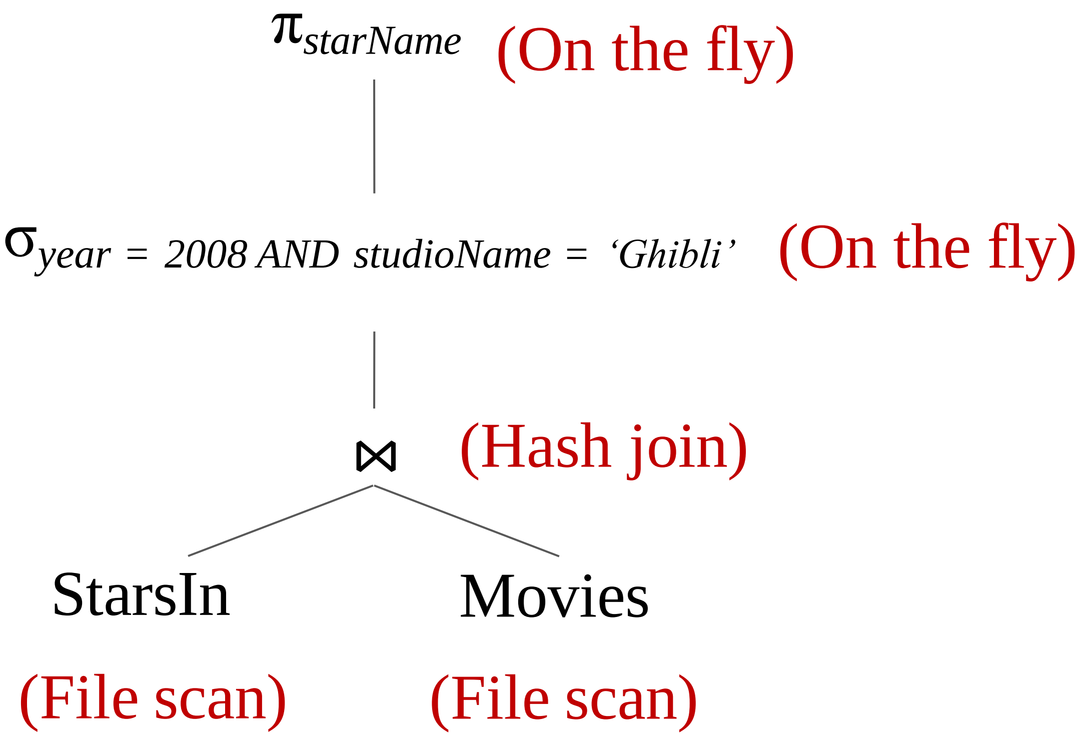
$$\bowtie$$

StarsIn Movies

Query optimization: select physical plan

- A logical query plan is turned into a physical query plan
  - Algorithm for each operator
  - Order of execution
  - How to access relations

Physical query plan 1



Query optimization: select physical plan

- A logical query plan is turned into a physical query plan
  - Algorithm for each operator
  - Order of execution
  - How to access relations

$\pi_{starName}$ (On the fly)

$\sigma_{year\ =\ 2008\ AND\ studioName\ =\ 'Ghibli'}$ (On the fly)

$\bowtie$ (Nested loop join)

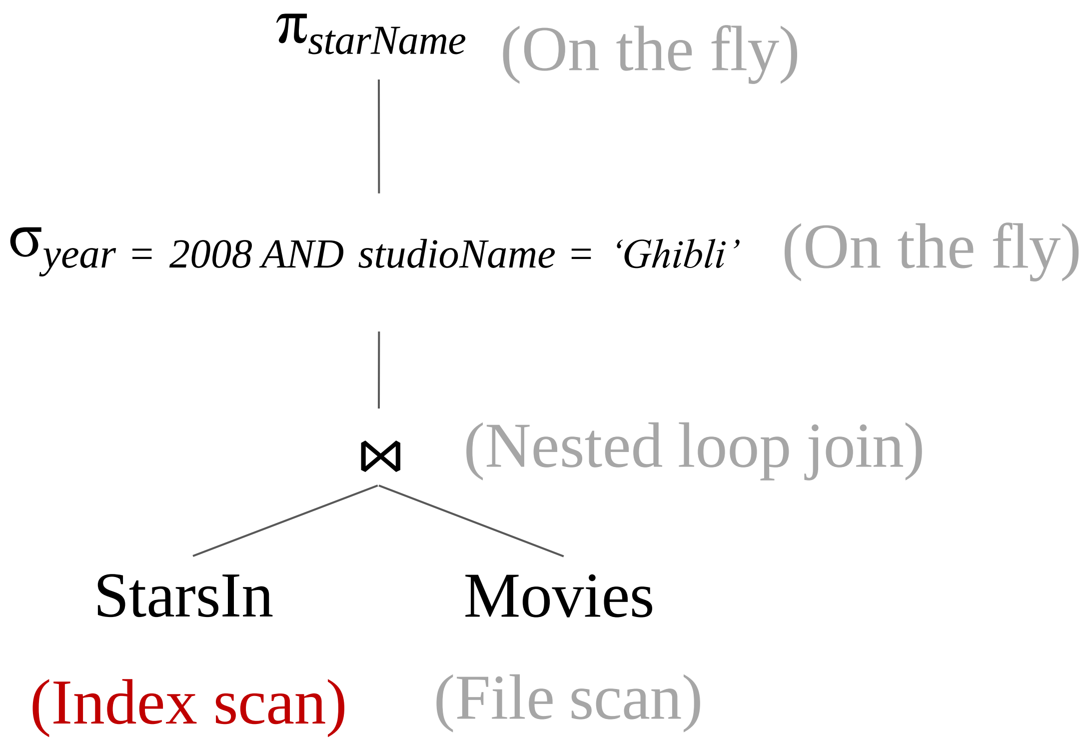
StarsIn (File scan) | Movies (File scan)

Physical query plan 2

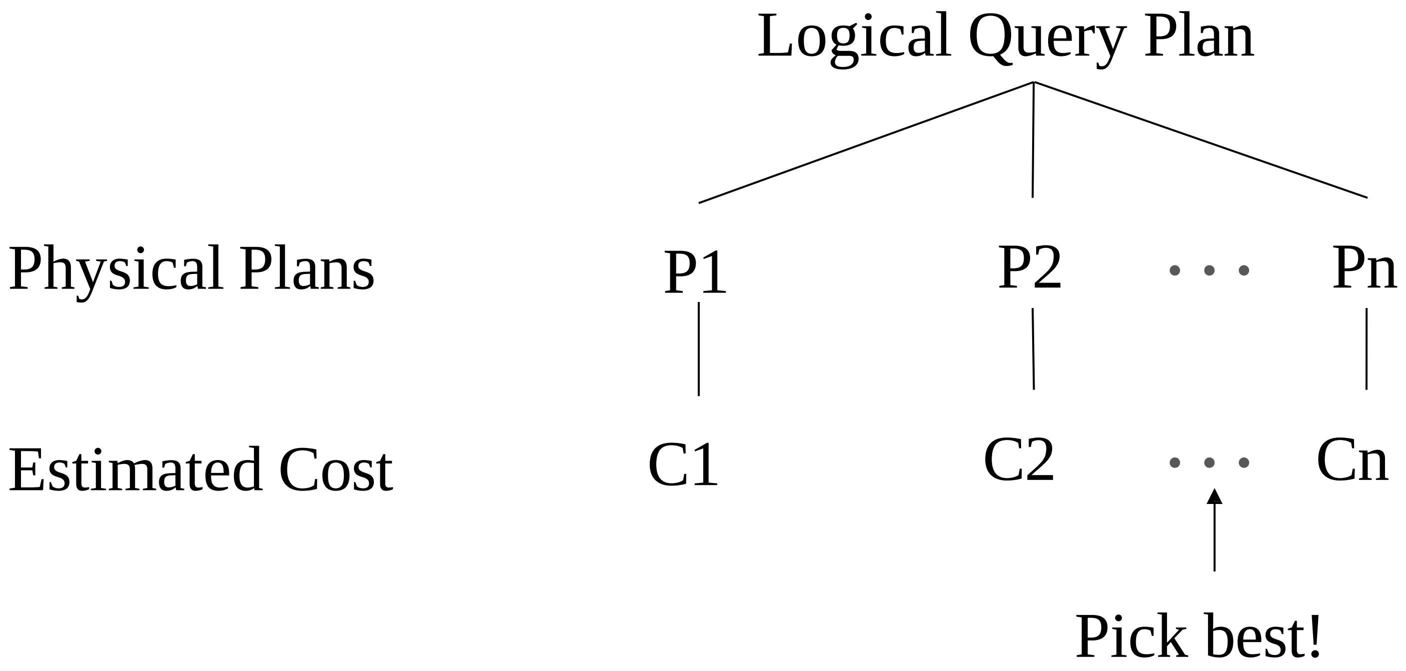
Query optimization: select physical plan

- A logical query plan is turned into a physical query plan
  - Algorithm for each operator
  - Order of execution
  - How to access relations

Physical query plan 3



Query optimization: select physical plan



In general, there can be many possible physical plans

Query execution

$\pi_{starName}$ (On the fly)

$\sigma_{year\ =\ 2008\ AND\ studioName\ =\ 'Ghibli'}$ (On the fly)

$\bowtie$ (Nested loop join)

StarsIn (File scan)

Movies (File scan)

Machine Code (e.g., C)

The best physical plan is translated to actual machine code

Query optimization: methodology

- Output: A good physical query plan

- Basic cost-based query optimization algorithm
  - Enumerate candidate query plans (logical and physical)
  - Compute estimated cost of each plan (e.g., number of I/Os)
    - Without executing the plan!
  - Choose plan with lowest cost

Query optimization: methodology

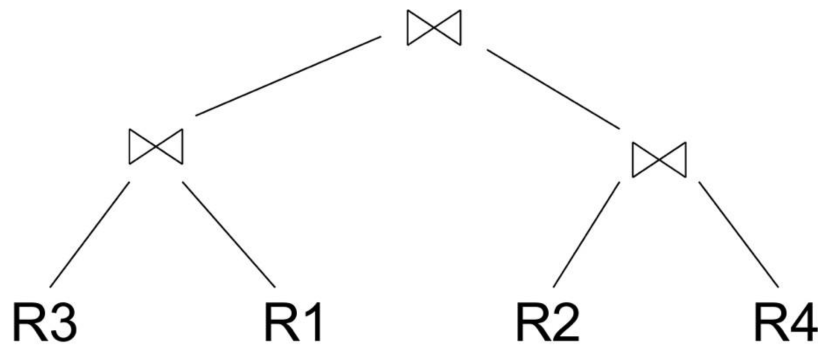
- Cost estimation
  - Estimate size of results
  - Also consider whether output is sorted/intermediate results written to disk etc.

- Search space
  - Algebraic laws, restricted types of join trees

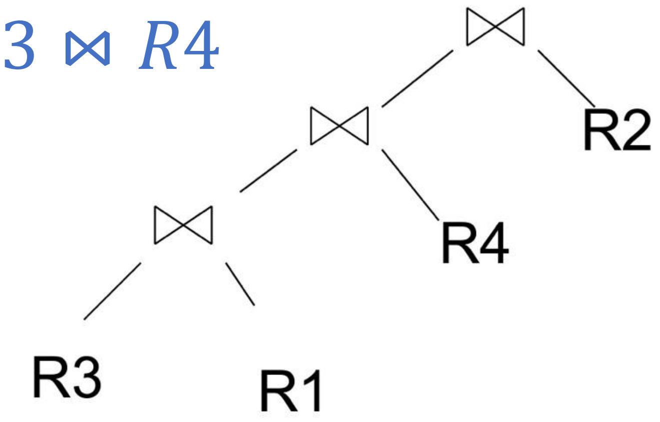
- Search algorithm
  - Example: Selinger algorithm

Search space

Query: $R1 \bowtie R2 \bowtie R3 \bowtie R4$



Bushy plan



Left-deep plan

- Logical plan space:
  - Several possible structures of the trees
  - Each tree can have n! permutations of relations on leaves
- Physical plan space:
  - Different implementation (e.g., join algorithm) and scanning of intermediate operators for each logical plan

Heuristic for pruning plan space

- Apply predicates as early as possible
  - Avoid plans with cartesian products
  - $(R(A, B) \bowtie T(C, D)) \bowtie \mathrm{S}(B, C)$

- Consider only left-deep join trees
  - Studied extensively in traditional query optimization literature
  - Works well with existing join algorithms such as nested-loop and hash join
    - e.g., might not need to write tuples to disk if enough memory

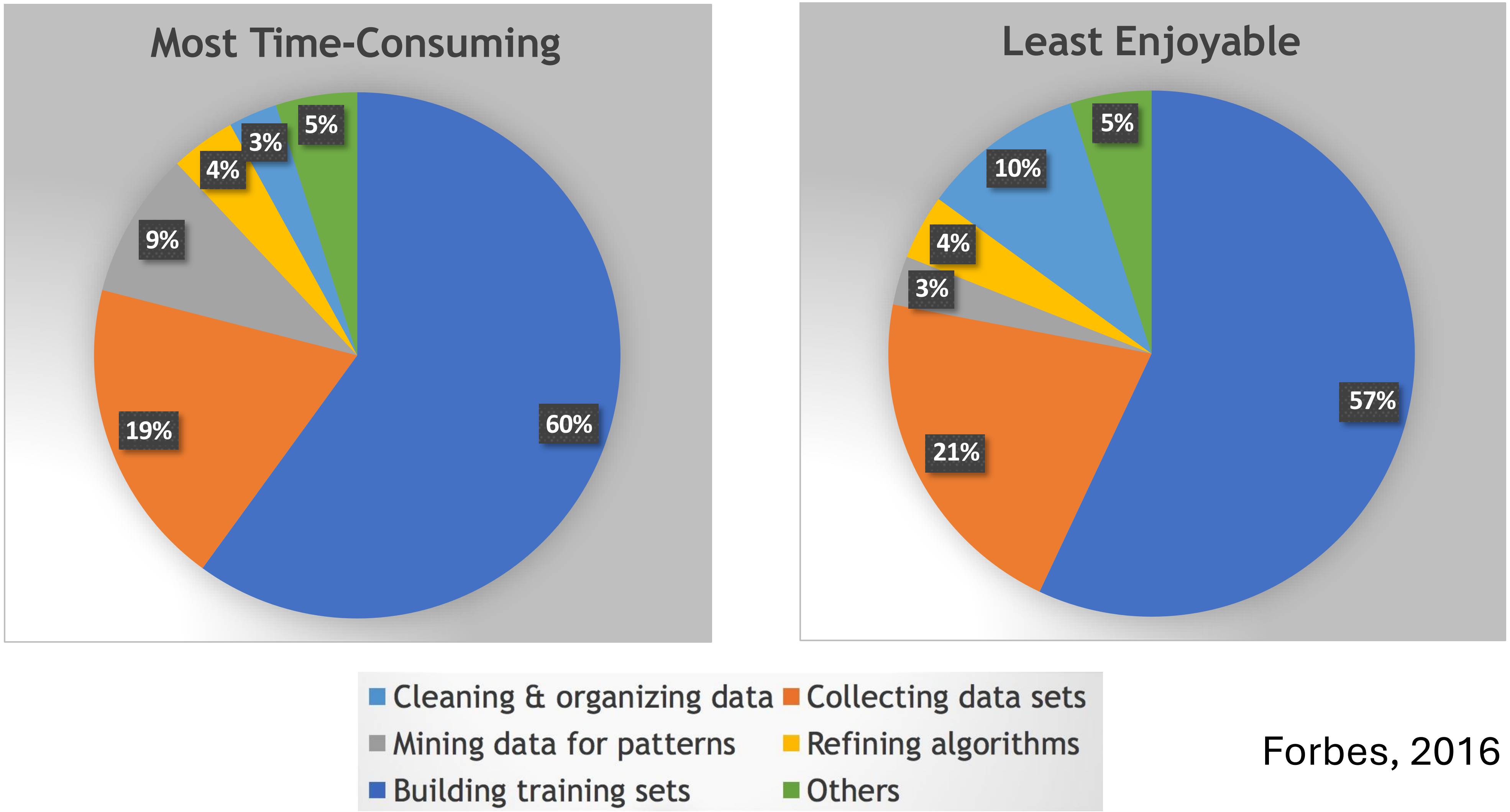
Search algorithm

- Selinger Algorithm: dynamic programming based
  - Based on System R (aka Selinger) style optimizer [1979]
  - Consider different logical and physical plans at the same time
  - Limited to joins: join reordering algorithm
  - Cost of a plan is I/O + CPU

- Exploits “principle of optimality”
  - Optimal for “whole” made up from optimal for “parts”

- Consider the search space of left-deep join trees
  - Reduces search space but still n! permutations

Cleaning data: most time-consuming, least enjoyable



Forbes, 2016

Common data problems

Incomplete

Country	UN R/P 10%[4]	UN R/P 20%[5]	World Bank Gini (%)[6]	WB Gini (year)	CIA R/P 10%[7]	Year	CIA Gini (%)[8]	CIA Gini (year)	GPI Gini (%)[9]
Seychelles			65.8	2007					
Comoros			64.3	2004					
Namibia	106.6	56.1	63.9	2004	129.0	2003	59.7	2010	
South Africa	33.1	17.9	63.1	2009	31.9	2000	65.0	2005	
Botswana	43.0	20.4	61.0	1994			63	1993	
Haiti	54.4	26.6	59.2	2001	68.1	2001	59.2	2001	
Angola			58.6	2000					62.0
Honduras	59.4	17.2	57.0	2009	35.2	2003	57.7	2007	

Common data problems

Inconsistent

Financial

Employee	Salary
John	1000

Employee $\rightarrow$ Salary

Human Resources

Employee	Salary
John	2000
Mary	3000

Employee $\rightarrow$ Salary

Target Database

Employee	Salary
John	**1000**
John	**2000**
Mary	3000

Employee $\rightarrow$ Salary

Mapping

Financial(e,s) $\subseteq$ Global(e,s)

HumanRes(e,s) $\subseteq$ Global(e,s)

Common data problems

Inaccurate

Sheepdog or mop?

Poodle or fried chicken?

Fox or dog?

Constraints and minimality

Functional dependencies

$$\text{c1: DBAName} \rightarrow \text{Zip}$$
$$\text{c2: Zip} \rightarrow \text{City, State}$$
$$\text{c3: City, State, Address} \rightarrow \text{Zip}$$

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Cicago**	IL	60608

Action: Fewer erroneous than correct cells; perform minimum number of changes to satisfy all constraints

External information

Matching dependencies

$$\text{m1: } \text{Zip} = \text{Ext\_Zip} \rightarrow \text{City} = \text{Ext\_City}$$
$$\text{m2: } \text{Zip} = \text{Ext\_Zip} \rightarrow \text{State} = \text{Ext\_State}$$
$$\text{m3: } \text{City} = \text{Ext\_City} \wedge \text{State} = \text{Ext\_State} \wedge \wedge \text{Address} = \text{Ext\_Address} \rightarrow \text{Zip} = \text{Ext\_Zip}$$

External list of addresses

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

External dictionaries may have limited coverage or not exist altogether

Quantitative statistics

Reason about co-occurrence of values across cells in a tuple

Estimate the distribution governing each attribute

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t4	**John Veliotis Sr.**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

Again, fails to repair the wrong zip code

Combine everything

Constraints and minimality

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t4	Johnnyo's	Johnnyo's	3465 S Morgan ST	Cicago	IL	60608

External data

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t4	Johnnyo's	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

Quantitative statistics

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609
t4	**John Veliotis Sr.**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

Different solutions suggest different repairs

Data is the bottleneck for ML!

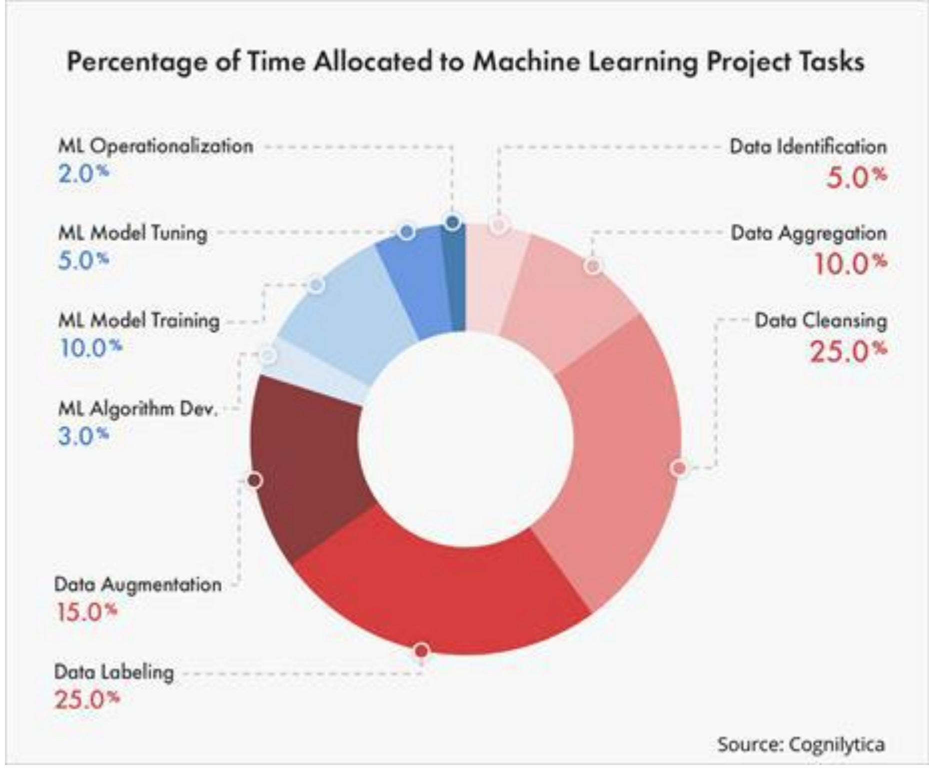
ML ≈ Model + Data

Model is gradually commoditized

- Out-of-the-box invocation of ML libraries gives decent results
- Transformers for “all” tasks

Data is the bottleneck

OpenAI has hired an army of contractors
to do what’s called “data labeling”



Sources:
https://www.semafor.com/article/01/27/2023/openai-has-hired-an-army-of-contractors-to-make-basic-coding-obsolete
https://www.datanami.com/2023/01/20/openai-outsourced-data-labeling-to-kenyan-workers-earning-less-than-2-per-hour-time-report/

Final project: putting together everything

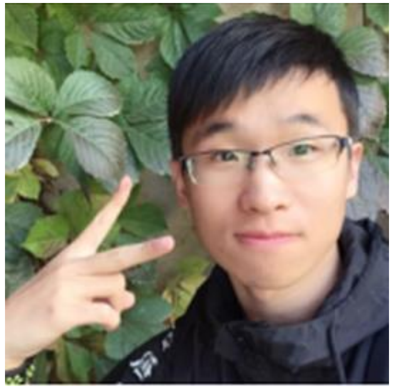
- Given the workload and input data, find best strategies
  - Which DB for what data?
  - How to aggregate results?
  - How to tune and optimize for better performance?

- Benchmark on a standard server

Course TAs

Runze Cai

PhD student @ Synteraction Lab



Lingze Zeng

PhD student @ DB System Lab

Haichen Huang

PhD student @ HPC-AI Lab

Assignments and grading

- **Coding/Written assignments (40%)**
  - 2 individual HWs
- **Tutorials (10%)**
  - 4 labs
  - Please bring your laptop
- **Mid projects (20%)**
  - Group of 3 people
  - Pick a project in Modern Database I, II, III
  - Presentation, writeup & QA in tutorial sessions
- **Final projects (30%)**
  - Group of 3 people
  - Release late March
  - Presentation, writeup, QA & standard benchmarks

Communications

- Office hour: By appointment

- Instructor email: luyao@comp.nus.edu.sg
- TA email: lingze@comp.nus.edu.sg
  runze.cai@u.nus.edu
  hai2000@comp.nus.edu.sg

- Canvas
  - Only for notifications, gradebooks and homework submissions
  - Course content on my webpage

Disclaimers

- Very short time for revamping this course. Only a few similar offerings around the world.
- Industry & open-source world evolving ultra fast.
- The materials and outline will likely adjust throughout the semester.
- There will be bugs in the content.

Credits

- Andy Pavlo, Carnegie Mellon University
- Kexin Rong, Georgia Institution of Technology
- Xiangyao Yu, University of Wisconsin-Madison