

Data Lakehouse Demystified

massively scalable, highly performant,
low maintenance and cost-effective analytics

Tyler Rich Dudley

<http://rjdudley.com/>

@rj_dudley

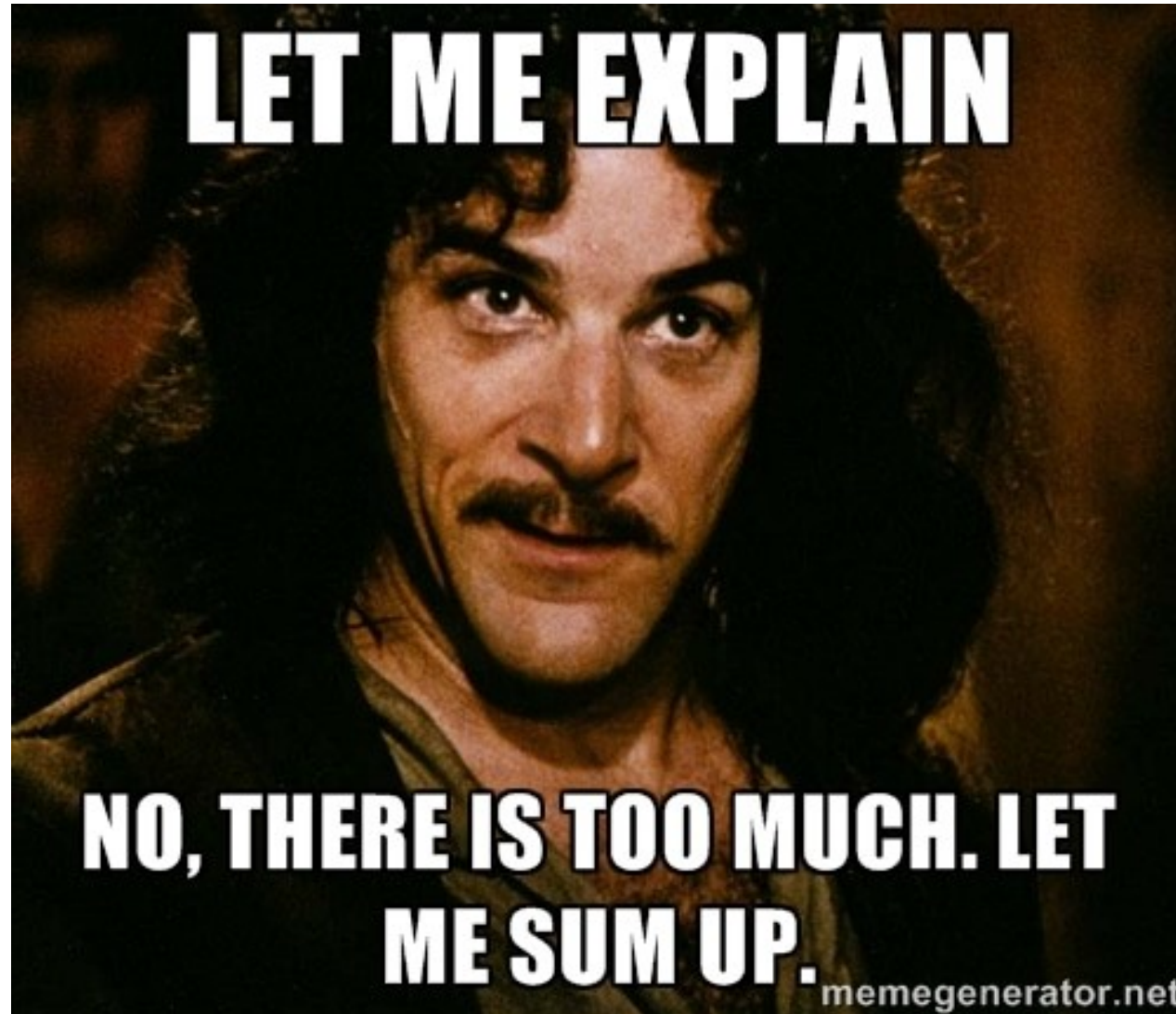


The unique ability of the lakehouse to manage data in an **open environment**, blend all **varieties of data** from all parts of the enterprise, and **combine** the data **science focus** of the data lake with the **end user analytics** of the data warehouse will unlock incredible value for organizations.

Bill Inmon, "The Father of the Data Warehouse"



A long way to go and a short time to get there



Partial Timeline



SQL

+

- All-in-one (storage, query, indexing)
- Row based storage
- Enforced schema
- ACID Transactions
- Very optimizable



ORACLE



-

- All-in-one
- Contention and locking
- All data in one file
- Proprietary
- Single server or small cluster



Data Warehouse

+

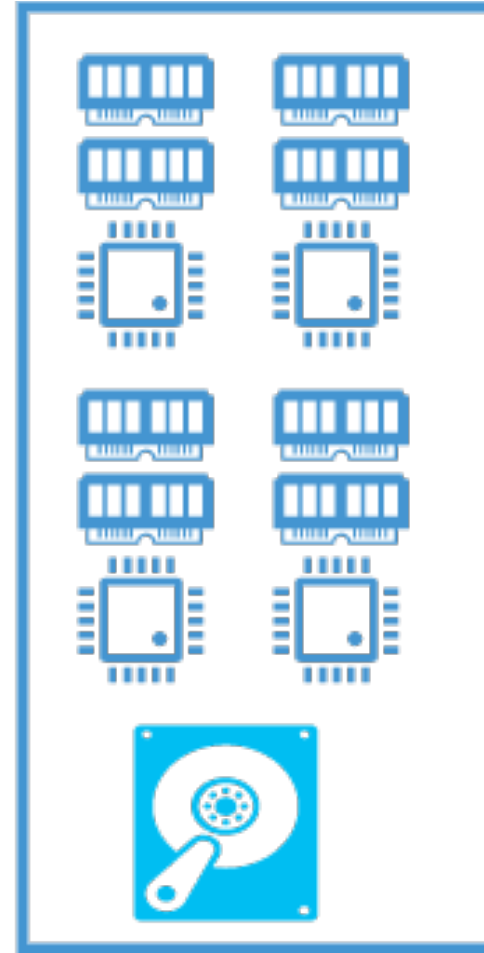
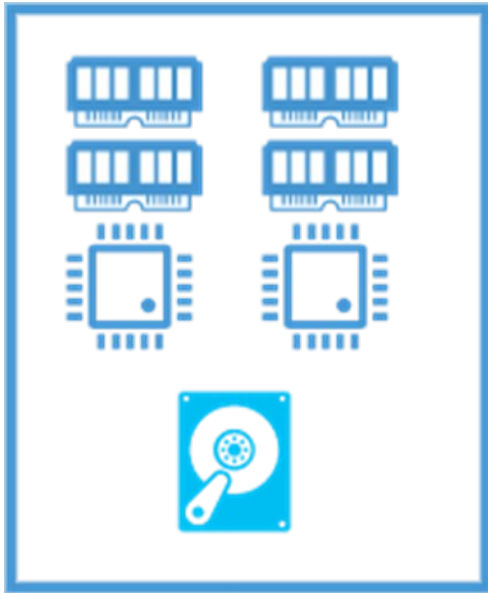
- Ideal for BI
- Highly governed with predictable data structures

-

- Rigid structures
- Difficult to add history or modify schema
- Scalability



Scaling...Back in the Day



NoSQL

+

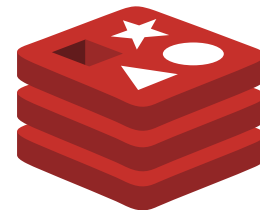
- All-in-one
- Not tabular
- Some highly governed with predictable data structures
- Some can distribute compute and storage

-

- Varying schema enforcement support
- All-in-one
- Proprietary
- Small clusters
- Unfamiliar query languages



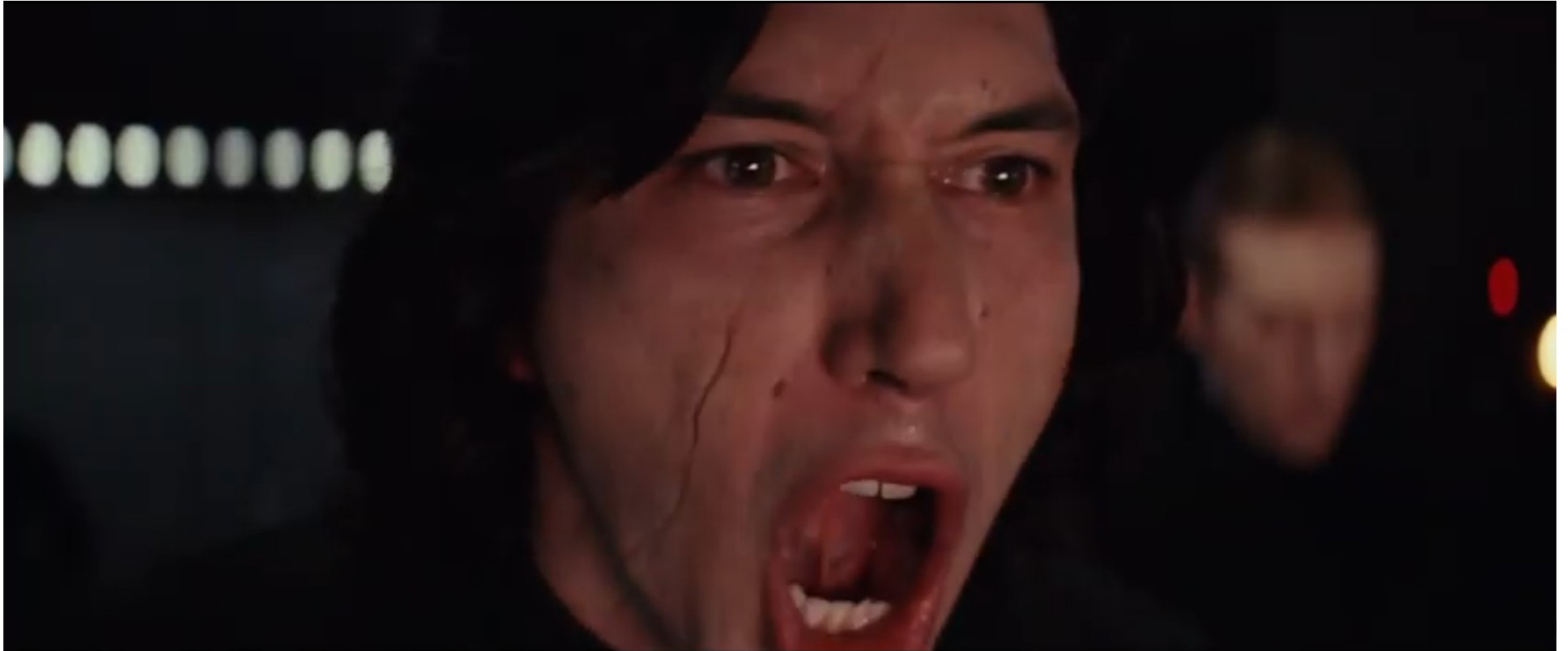
mongoDB



neo4j



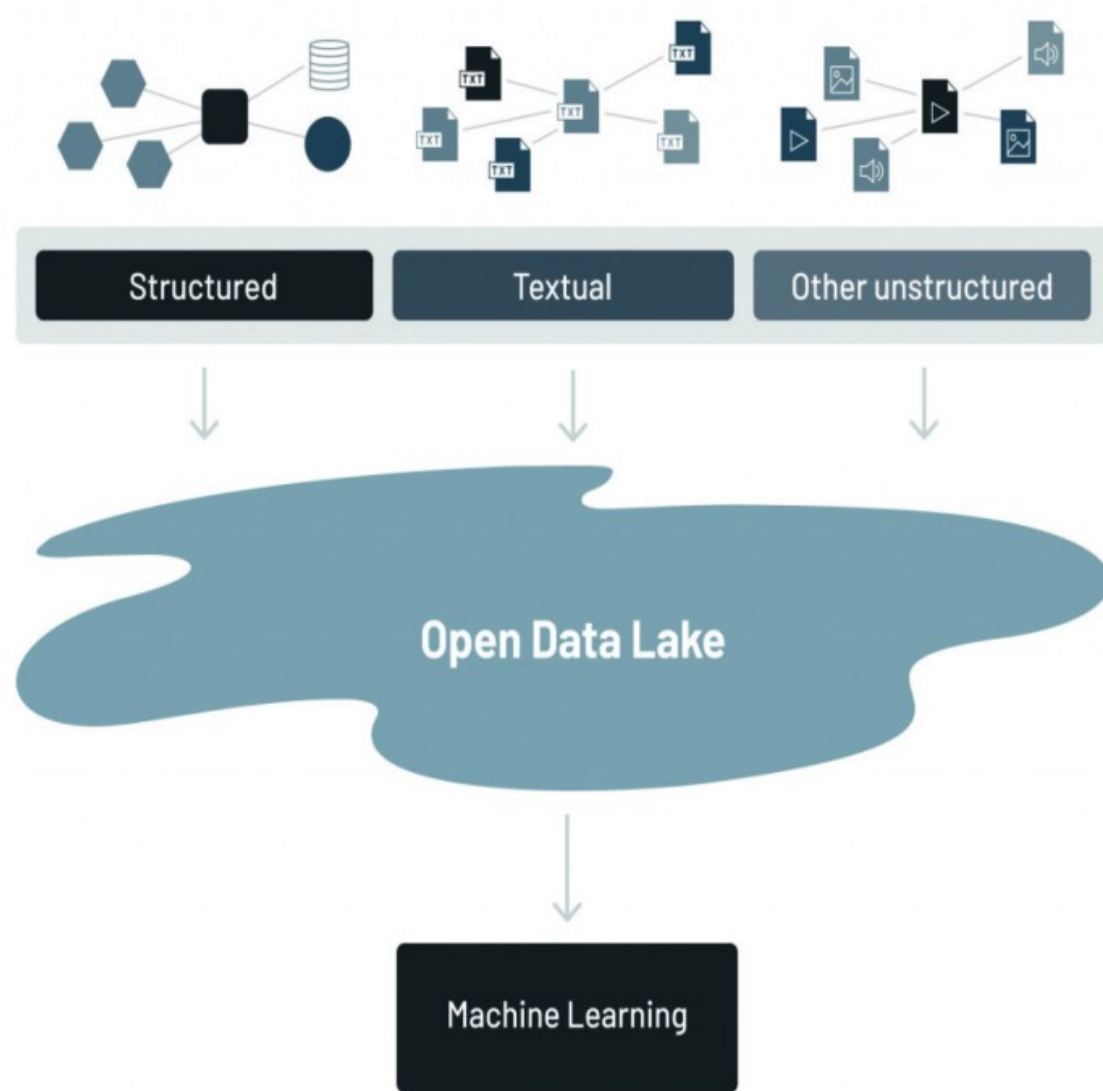
NoSQL Was Better, But We Needed



MORE

Data Lake Unmasked





Data Lakes

Solved-ish

- Volume, velocity, variety, variability
- Scalability
- Swappable parts
- Open standards
- Parallel execution
- Enabled data science and ML

Failed

- Versioning
- Schema enforcement
- Data quality
- Consistent integrated stack
- Guaranteed reads
- BI
- Simple, familiar queries
- Silos



Why is Hadoop like that?



The Dad
@thedad

God: what are they doing down there?

Angel: they are making milk from almonds

God: what?! I gave them, like, 8 animals to get milk from

A: they dont like that milk

God: [mockingly] tHey DonT LiKe THat miLk *flips a table*



Hadoop Ecosystem

- File System = HDFS
- Storage Engine = HBase, Ozone, Phoenix, Hypertable
- Scheduling/Management = YARN, Airflow
- Query Language = Pig, Hive, Mahout
- Processing = MapReduce, Spark

Missing

- Indexing, governance, quality



SSDD



Matt Turck • Following
Managing Director at FirstMark
[Visit my website](#)

1w • 🌐



2010: "the problem (for analytics) is that our data is messy, siloed and all over the place"

2016: "the problem (for BI and ML) is that our data is messy, siloed and all over the place"

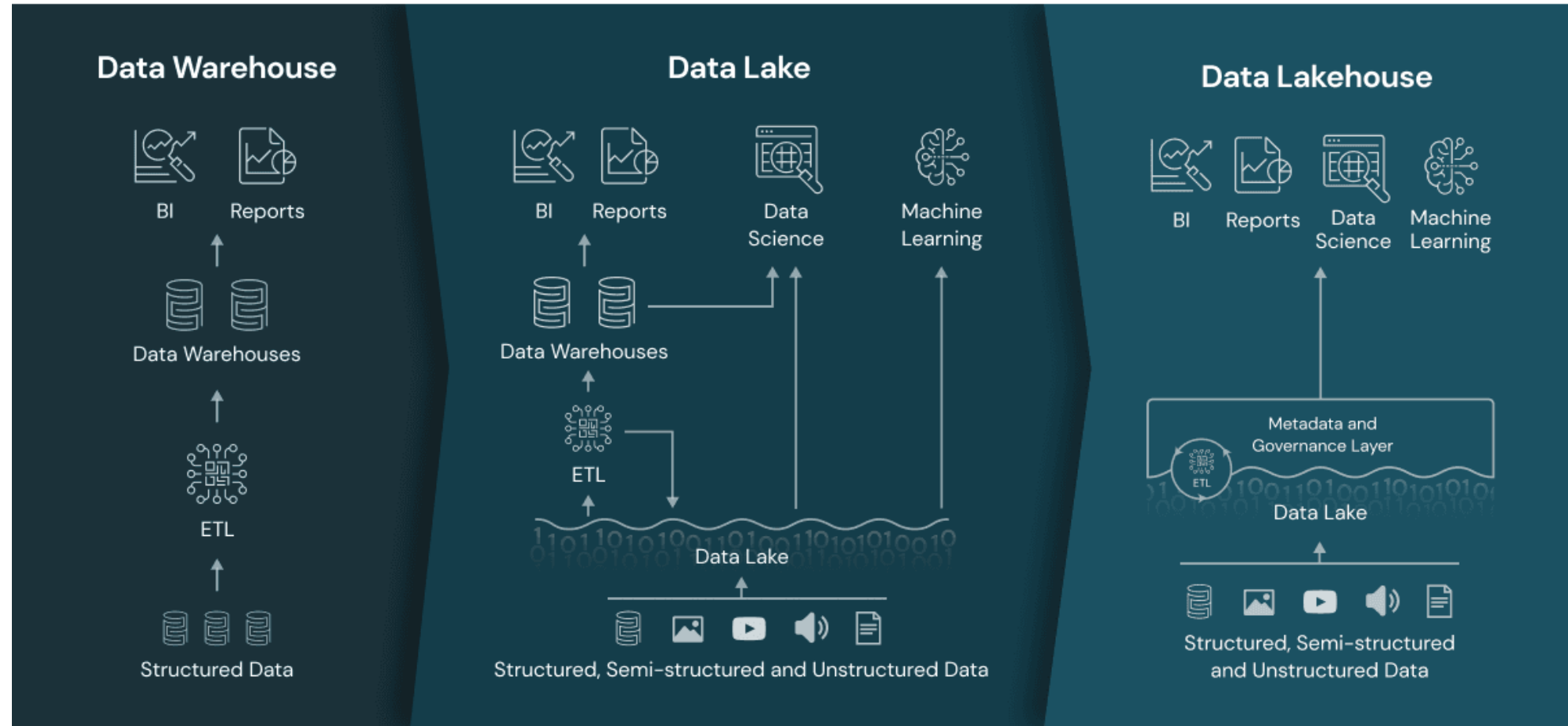
2024: "the problem (for AI) is that our data is messy, siloed and all over the place"

👍🗨️❤️ 758

94 comments • 55 reposts



Behold: Lakehouse!



Data Lakehouse Characteristics

- All data in one place, from raw to reference to dimensional
- Supports BI, ML, AI
- Lineage, metadata and security governance
- Open standards
- Slightly opinionated in the technology stack
- Some parts swappable
- Rich partner ecosystem



Data Catalog (Metamodel)

- Metadata
 - Data models
 - Transformation rules
 - Lineage
 - Security
-
- Unity Catalog, Purview, Glue+Hive



Layers

Data (Medallion)

- Bronze
- Silver
- Gold

Technology

- Storage
- Compute
- Consumption

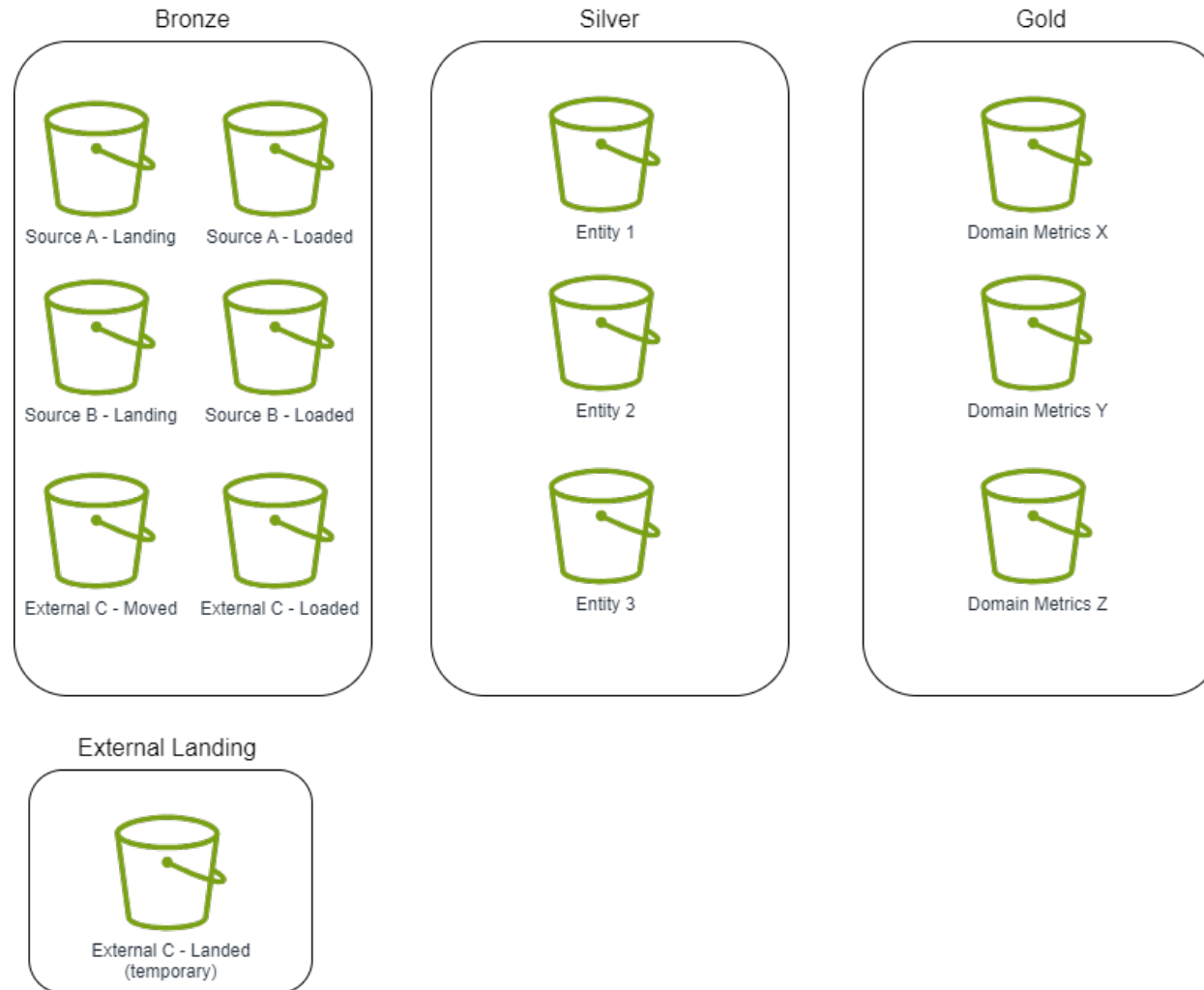


Medallion Architecture

- Bronze
 - The raw stuff
 - QBO, Sage, Freshbooks all have different invoice formats
- Silver
 - Standardized, may not make sense if only one input like an ERP
 - Usually entity and reference tables
- Gold
 - The AI/BI data
 - Data Warehouse
 - Facts, dimensions, references (sometimes straight copy from Silver)
- Sometimes Landing
 - When external systems pushing data to you
 - Remove from landing when processed



Physical medallion storage



Medallion Security

- Access only through the platform UI, not directly against storage layer
- Bronze
 - Likely full of PII, PHI, etc.
 - Most restrictive access: data engineers only
 - No access from outside your organization
- Silver
 - Anonymized as much as possible
 - Less restrictive access: data engineers, data scientists
 - No access from outside your organization
- Gold
 - Primarily anonymized facts and dimensions
 - Least restrictive: data engineers, data scientists, data analysts, business users, AI
 - Access via API or similar mechanism
- Landing
 - Limited open to outside organizations directly to storage
 - Restricted to data engineers only



Storage Layer

- Blob Storage
 - S3, Azure Data Lake G2, GCP Object
- Usually many buckets/containers
- Landed/Raw Bronze = CSV, JSON, XML, PDF, JPG, PNG, MP4, etc.
- Transformed Bronze, Silver and Gold
 - Delta Lake, Parquet, Avro, Iceberg, Hudi



Delta Lake and Iceberg

- Delta Lake = “Parquet on ACID”
- ACID Compliance
- History / Time Travel / Versioning
- Partitioning and indexing
- Liquid clustering
- Distributed
- Schema evolution



Storage Layer Comparison

Storage layer attributes — data lake vs. data warehouse vs. data lakehouse

Data Lake	Data Warehouse	Data Lakehouse
Open format	Closed, proprietary format	Open format
Low quality, "data swamp"	High-quality, reliable data	High-quality, reliable data
File-level access control	Fine-grained governance (tables row/columnar level)	Fine-grained governance (tables row/columnar level)
All data types	Structured only	All data types
Requires manually specifying how to lay out data	Automatically lays out data to query efficiently	Automatically lays out data to query efficiently

"Why the Data Lakehouse Is Your Next Data Warehouse"



Storage Layer Security

- Only Landing should have any direct public access
- Highest restrictive access: data engineers only
- Should have multi-AZ replication at a minimum



Upskill

- Basic
 - Data security
 - Medallion architecture
 - Differences between data file types, especially Avro and Parquet
 - How does a columnstore work?
- Advanced
 - Partitioning strategies when you have massive datasets
 - Time travel and data retention policies



Compute

- Usually Apache Spark
- Also Trino, Flink, Presto, Hive, Impala, Amazon Athena
- Elastic, scales to zero
- Compatible with many open source storage formats
- Python, Scala, R, Java, SQL



Upskill

- Basic
 - Python and PySpark
 - Spark dataframes
- Advanced
 - Spark execution plans
 - Spark performance tuning



Consumption

- Model Context Protocol (MCP)
- Agent2Agent (A2A)
- Delta Sharing
- "Genie" (text-to-sql) API
- ODBC/JDBC

Rapidly developing space!







Upskill



- Compare/contrast Delta Sharing and MCP
- Understand the options your chosen vendor has
- Stay aware of releases





Data quality

 + Machine Learning = 
Data

 + Artificial Intelligence = 
Data

 + Generative AI = 
Data

 + Agentic AI = 
Data



Upskill

- Common data quality issues
- Data expectations
- Data quality monitoring



Governance

- Security governance
- Metadata governance
- Key feature of data lakehouse
- Although storage and compute are distributed, all treated as a single silo by a unified governance layer
- Access rules are in effect if you're in the UI, or consuming a report



5 key steps to building a successful data lakehouse

1. Start with the data lake that already manages most of the enterprise data
2. Bring quality and governance to your data lake
3. Optimize data for fast query performance
4. Provide native support for machine learning
5. Prevent lock-in by using open data formats and APIs

"Building the Data Lakehouse"



My Additional Suggestions

- Lakehouses are complicated with a lot of moving parts
- First make it work, then make it work better
- Don't let perfect be the enemy of good



Platforms

- Databricks (runs on AWS, Azure, GCP)
- Snowflake
- Microsoft Fabric
- Cloudera Data Platform
- Dremio
- AWS = DIY
- Every vendor on this planet will tell you their product will solve all the problems you know about (and most you don't) if you just give them enough money.
- You have to know enough to sort out the BS



Upskills

- “Well Architected” frameworks, most companies have them
- Cost management
 - This isn’t the first thing you optimize for but you need to know there are ways to optimize it



Lakehouse Bad Ideas

- Any kind of operational back-end
- Normalized, relational database with key enforcement



References

- <https://aws.amazon.com/blogs/big-data/build-a-lake-house-architecture-on-aws/>
- <https://www.databricks.com/resources/ebook/building-the-data-lakehouse>
- <https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html>

