

# **Большие данные**

**613x-010402D       $x=\{1,2,3\}$**

**осень 2024**

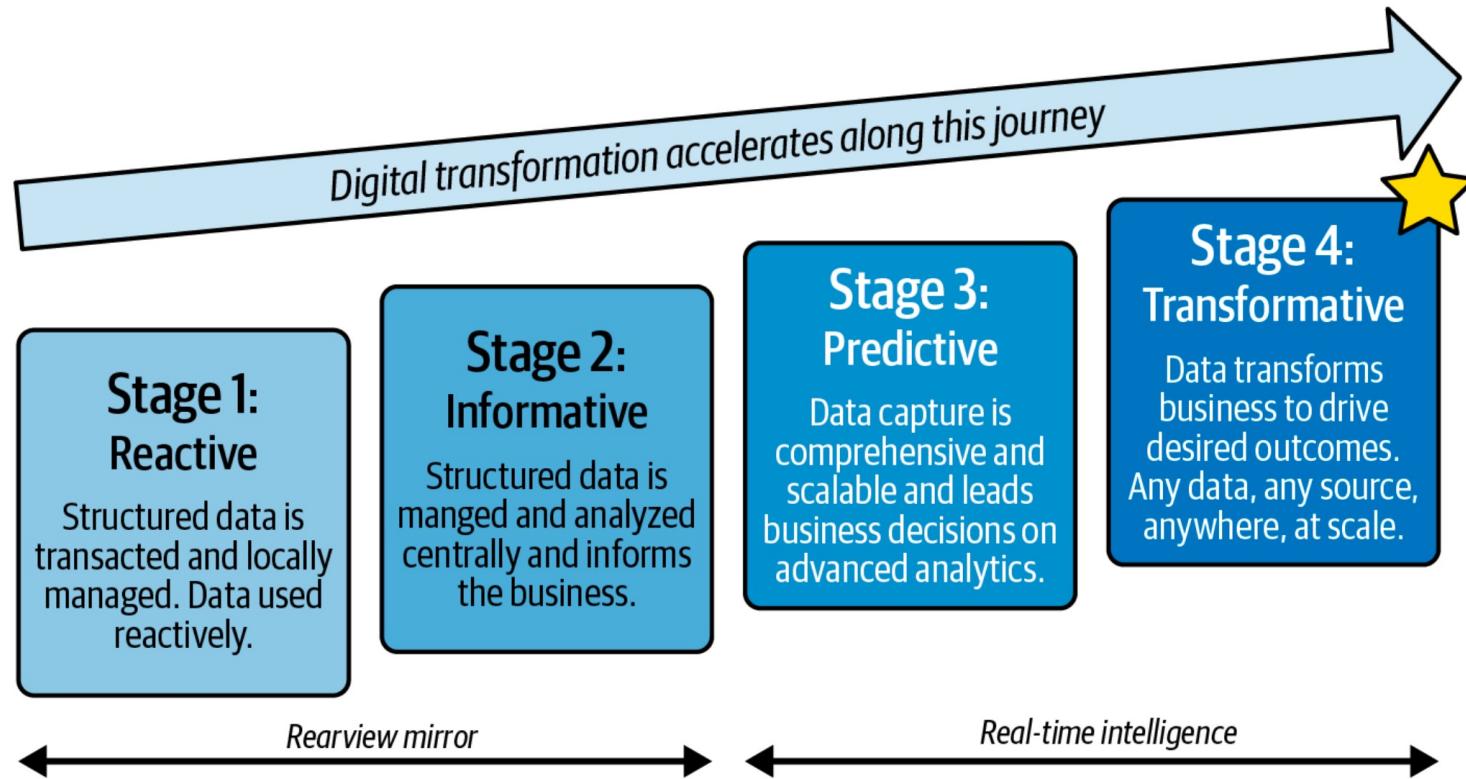
## **Лекция 2: Архитектуры данных**

Сергей Борисович Попов  
[sepo@ssau.ru](mailto:sepo@ssau.ru)

### **Материалы лекций:**

[https://1drv.ms/f/s!ApFj4iOLPNegvEPfMZL\\_5jjkXsqfQ](https://1drv.ms/f/s!ApFj4iOLPNegvEPfMZL_5jjkXsqfQ)

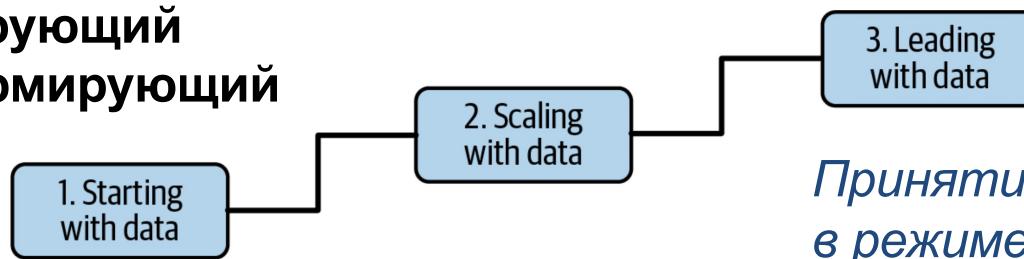
# Этапы зрелости корпоративных данных



## *Реагирование на ситуацию в прошлом:*

## Этап 1: Реагирующий

## **Этап 2: Информирующий**

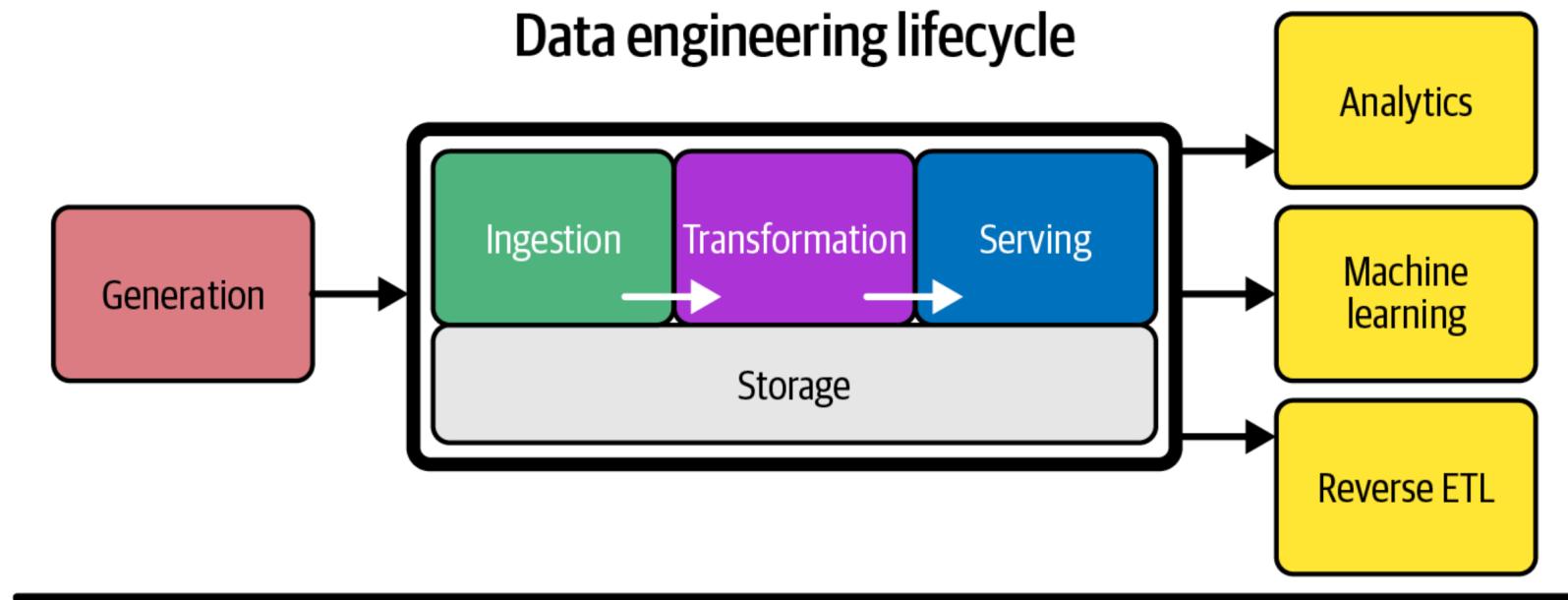


## *Принятие решений в режиме реального времени:*

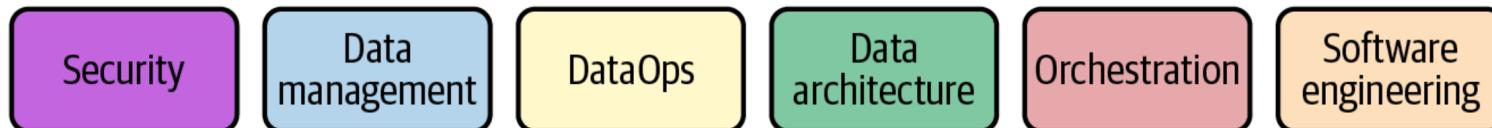
## **Этап 3: Предсказательный**

## **Этап 4: Преобразующий**

# Жизненный цикл инженерии данных



## Undercurrents:



- Генерация
- Хранение
- Приём/Сбор
- Преобразование
- Предоставление

- Аналитика
- Машинное обучение
- Обратное ETL

## Фоновые процессы:

- Безопасность
- Управление данными
- DataOps
- Архитектура данных
- Оркестровка
- Программная инженерия

# Элементы архитектуры данных

- **Хранилище данных (Data storage)**

Во всех архитектурах данных необходимо указать, как хранятся данные, включая физический носитель данных (например, жесткие диски или облачное хранилище) и структуры данных, используемые для организации данных.

- **Обработка данных (Data processing)**

Архитектура данных должна определять, как обрабатываются данные, включая любые преобразования или вычисления, которые выполняются с данными перед их сохранением или анализом.

- **Доступ к данным (Data access)**

Архитектуры данных должны обеспечивать механизмы доступа к данным, включая пользовательские интерфейсы и интерфейсы прикладных программ (API), которые позволяют запрашивать и анализировать данные.

- **Безопасность и конфиденциальность данных (Data security and privacy)**

Архитектуры данных должны включать механизмы обеспечения безопасности и конфиденциальности данных, такие как контроль доступа, шифрование и маскирование данных.

- **Управление данными (Data governance)**

Архитектура данных должна обеспечивать основу для управления данными, включая стандарты качества, отслеживание происхождения и политику хранения.

# Хронология развития БД и хранилищ данных

## 1960–1970



1959 – организация CODASYL создала язык программирования COBOL  
1969 – опубликована спецификация языка для сетевой модели данных DBTG или модели данных CODASYL (CODASYL Data Model).

1963 – Integrated Data Store (IDS) сетевая база данных компьютерного подразделения General Electric

1966 – первая промышленная СУБД – иерархическая система IMS (Information Management System) от компании IBM была разработана для космической программы Аполлон

## 1970–1980

1970 – Эдгар Франк «Тед» Кодд «A Relational Model of Data for Large Shared Data Banks» («Реляционная модель данных для больших общих банков данных»).

язык программирования SEQUEL (Structured English Query Language), потом SQL

1975 – проект трехуровневой архитектуры СУБД ANSI-SPARC  
пользовательское – логическое – физическое представления данных

1976 – ER-модель данных (Entity-Relationship model)

1979 – Oracle Database на базе SQL РСУБД (реляционная СУБД)

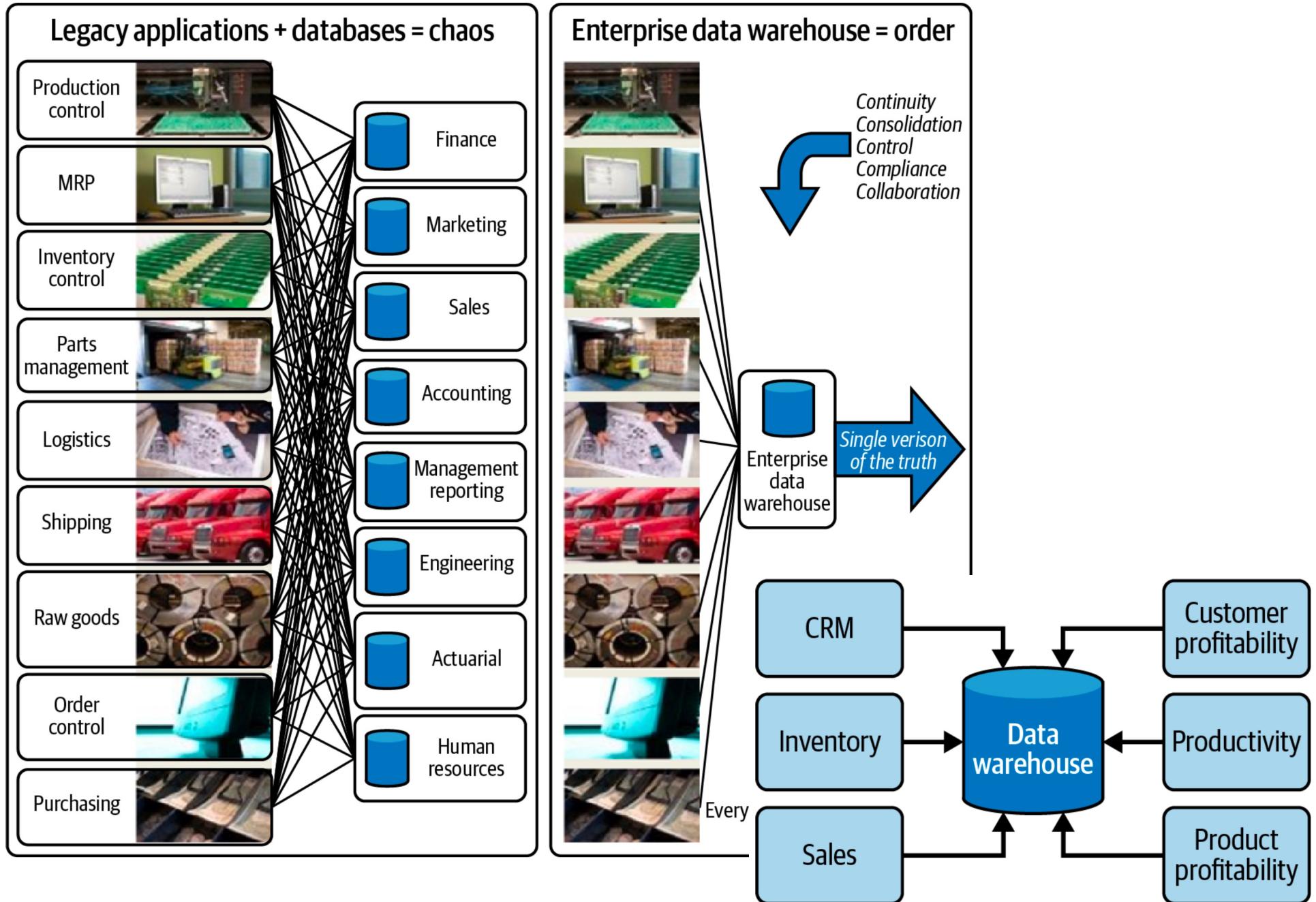
1980 – СУБД IBM dBASE

# Хронология развития БД и хранилищ данных

<b>1980–1990</b>	1986 – ANSI SQL 1987 – ICO SQL-87  DB2 (СУБД IBM Database 2)
<b>1990–2000</b>	1991 – Билл Инмон «Building the Data Warehouse» – концепция DWH (централизованное хранилище всех данных компании, разработка которого начинается с создания нормализованной модели данных)  Ральф Кимбалл – денормализация структуры хранилища и упрощение анализа для бизнес-пользователей  Подход «снизу вверх»: <ul style="list-style-type: none"><li>• сначала отдельные витрины для конкретных бизнес-задач, согласованные между собой на уровне мерностей,</li><li>• затем концептуальное хранилище данных.</li></ul> OLAP (OnLine Analytical Processing) – технология интерактивной аналитической обработки данных в реальном времени  1990–1991 – Microsoft SQL Server 1994 – СУБД Postgres95 (open source) 1996 – объектно-реляционная СУБД PostgreSQL.

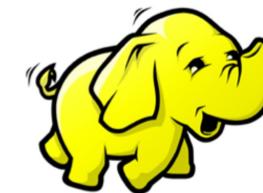


# Хранилище данных – Data Warehouse

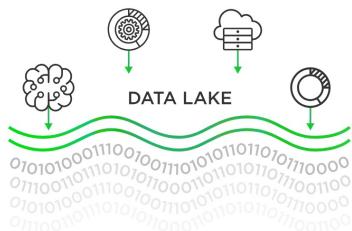


# Хронология развития БД и хранилищ данных

NoSQL	2000–2010	2003 – <b>GFS</b> : The Google file system 19th ACM Symposium on Operating Systems Principles
	2004	– <b>MapReduce</b> : Simplified Data Processing on Large Clusters (OSDI'04) 6th Symposium on Operating System Design and Implementation
	2006	– <b>Bigtable</b> : A Distributed Storage System for Structured Data (OSDI'06) 7th Symposium on Operating Systems Design and Implementation
	2006	– <b>Hadoop</b> : HDFS и MapReduce (Doug Cuttig)
	2008	– <a href="http://hadoop.apache.org">http://hadoop.apache.org</a>



2010–2020		NoSQL data bases: <b>Cassandra &amp; Hbase</b> SQL-processing: <b>Hive</b> High level Processing: <b>Pig</b> Hadoop compute engine: <b>Spark</b>
-----------	--	---



PostgreSQL на базе МРР: СУБД **Greenplum**  
Stream processing: **Apache Airflow, Kafka, Storm**  
Analytic system: **Google BigQuery, Amazon Redshift, Snowflake**



## Экосистема Hadoop

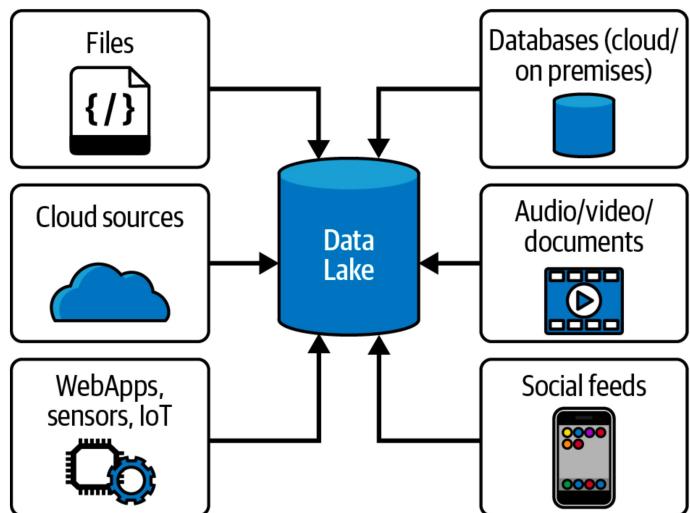
- Изначально Hadoop был известен в основном из-за двух ключевых компонент:
  - HDFS – Hadoop Distributed File System
  - MapReduce – Фреймворк распределённой обработки данных
- Сейчас проект Apache™ Hadoop® содержит четыре базовых модуля:
  - **Hadoop Common**: The common utilities that support the other Hadoop modules.
  - **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
  - **Hadoop YARN**: A framework for job scheduling and cluster resource management.
  - **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

# Экосистема Hadoop

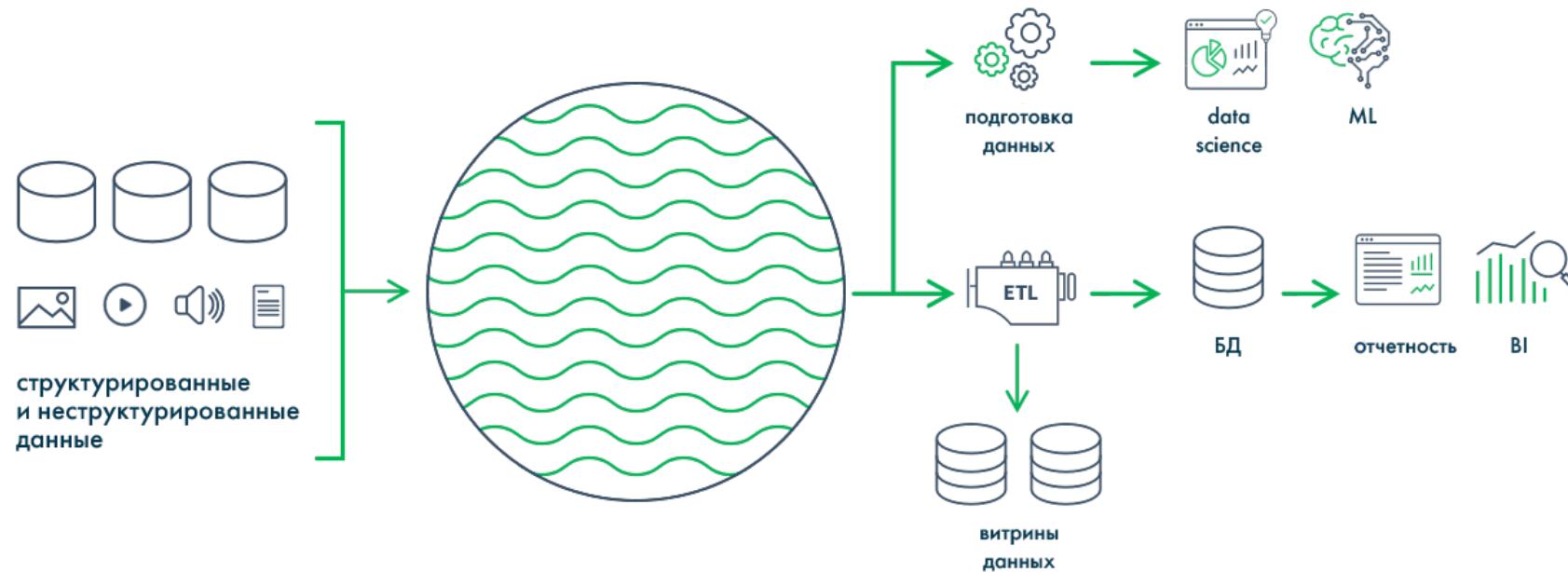
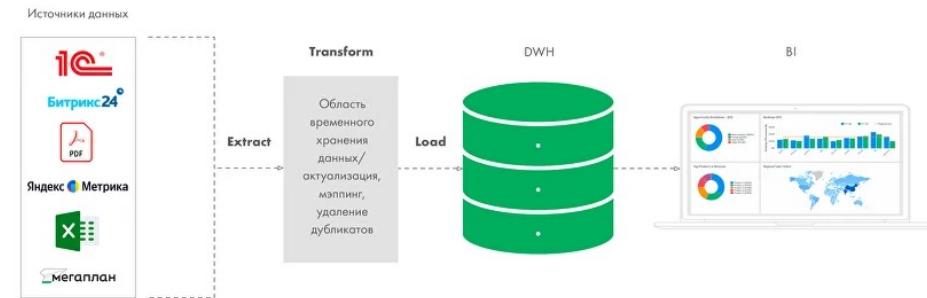


- Другие Apache-проекты, связанные с Hadoop:
  - **Ambari™**: web-средства мониторинга/управления Hadoop-кластерами, включая поддержку для Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop.
  - **Avro™**: система сериализации данных (передача схемы вместе с данными, работа с динамически типизированными объектами).
  - **Cassandra™**: A scalable multi-master database with no single points of failure.
  - **Chukwa™**: A data collection system for managing large distributed systems.
  - **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
  - **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
  - **Mahout™**: A Scalable machine learning and data mining library.
  - **Ozone™**: A scalable, redundant, and distributed object store for Hadoop.
  - **Pig™**: A high-level data-flow language and execution framework for parallel computation.
  - **Spark™**: A fast and general compute engine for Hadoop data.
  - **Submarine**: A unified AI platform which allows engineers and data scientists to run Machine Learning and Deep Learning workload in distributed cluster.
  - **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN.
  - **ZooKeeper™**: A high-performance coordination service for distributed applications.

# Озеро данных – Data Lake



Процессы в управлении хранилищами данных :  
ETL — Extract, Transform, Load  
ELT — Extract, Load, Transform

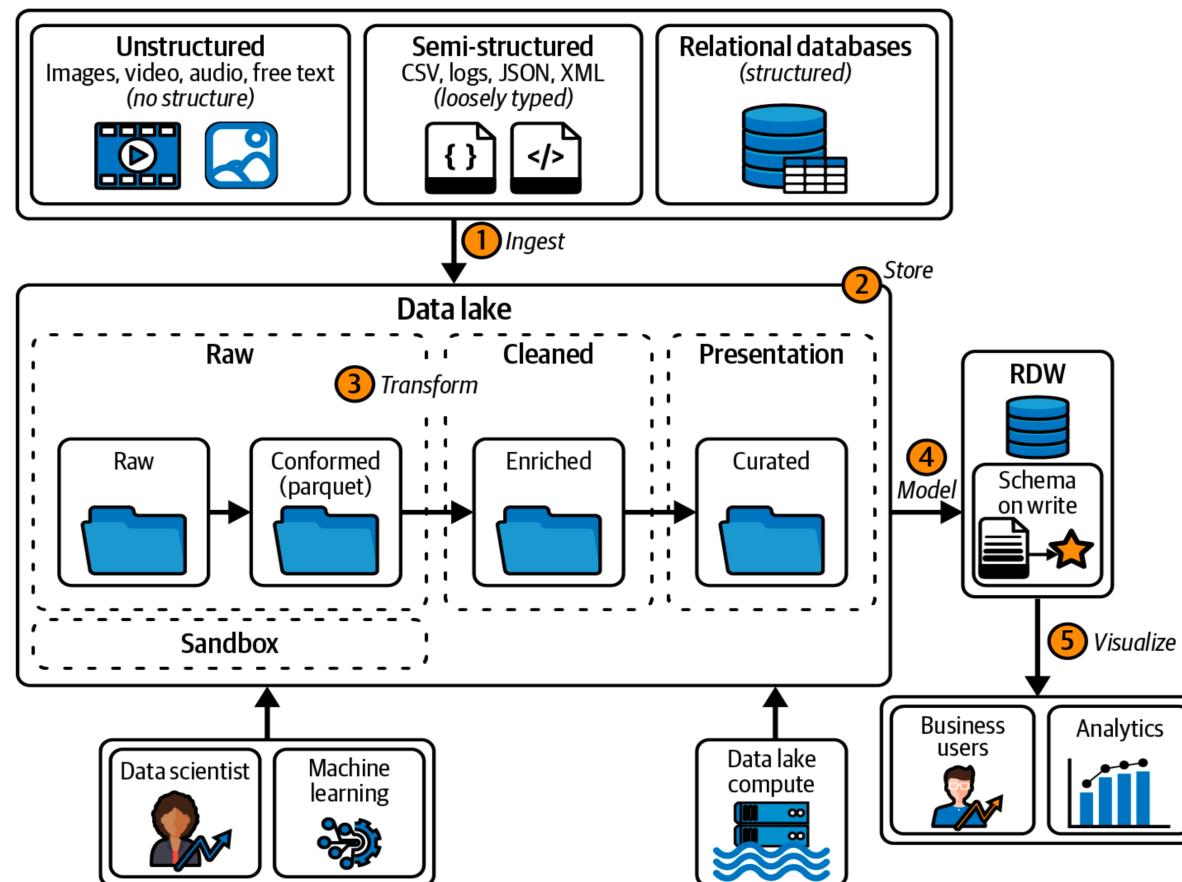
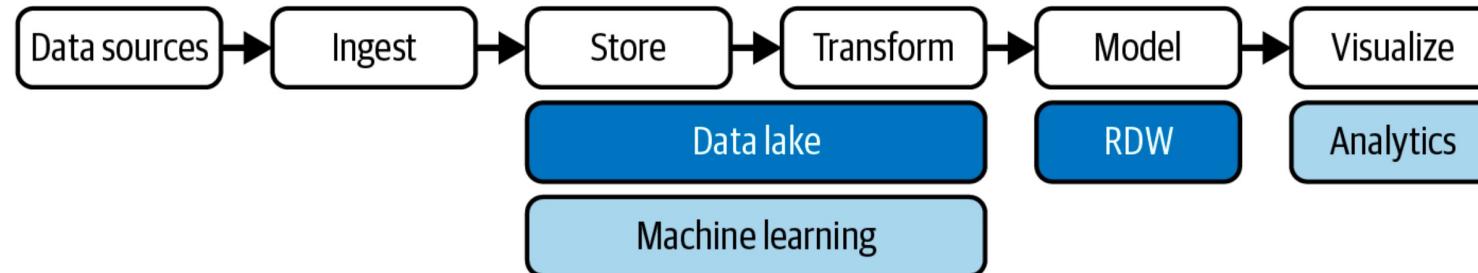


# Data Warehouse vs. Data Lake

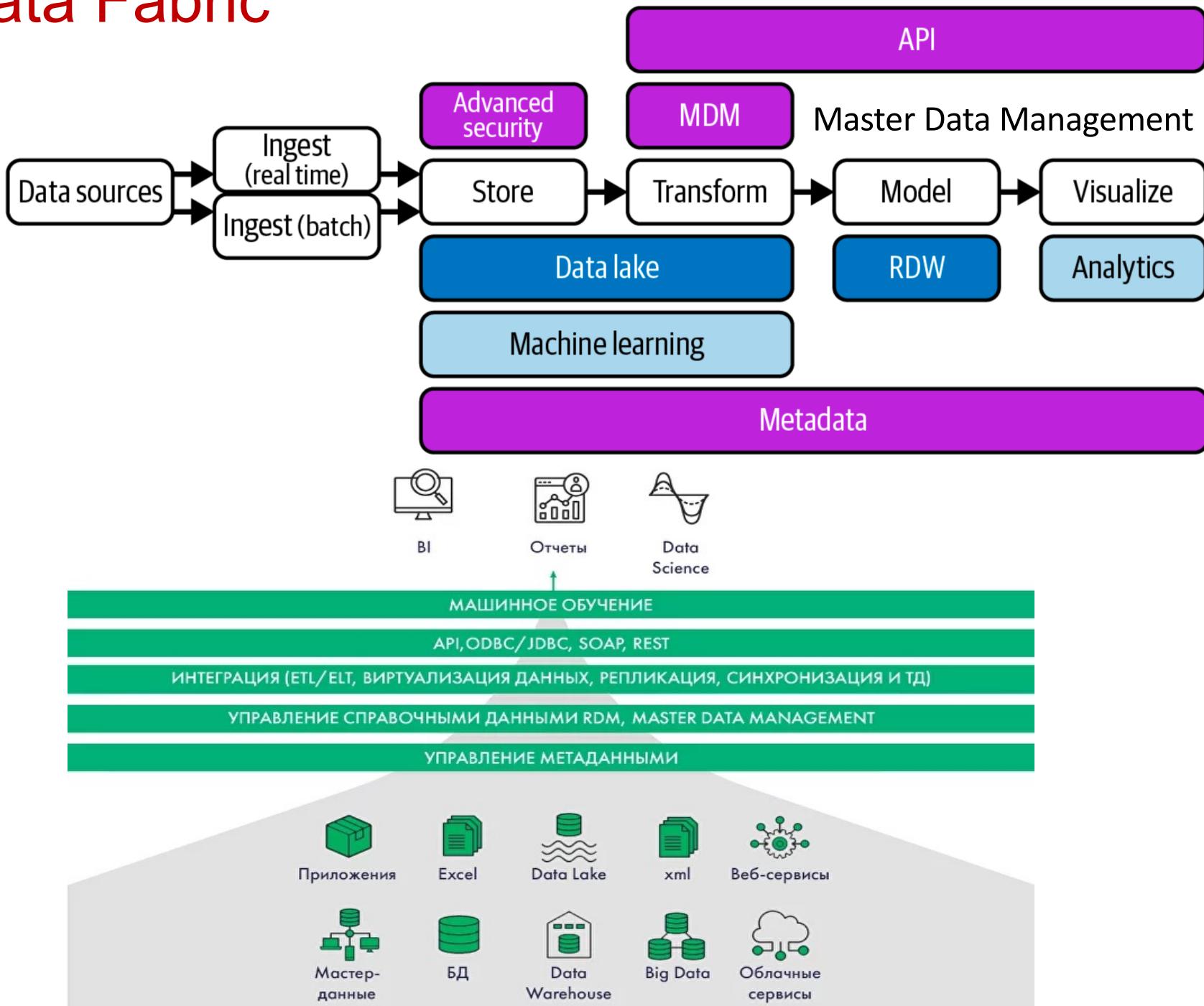
	<b>Data Warehouse</b>	<b>Data Lake</b>
Типы данных	Структурированные, подготовленные к аналитике данные	Данные в необработанном, полуструктурированном или неструктурированном виде и в любых форматах
Актуальность данных	Только необходимые под конкретные бизнес-задачи данные	Все данные компании, часть из которых может не пригодиться никогда
Цели	Визуализация, отчетность, BI	Предиктивная аналитика, машинное обучение, ИИ, BI, аналитика больших данных
Обработка	ETL - данные извлекаются из источника, очищаются, структурируются, на финальном этапе готовы к анализу	ELT - данные извлекаются из источника, хранятся в озере данных и затем трансформируются при необходимости

# Современное хранилище данных

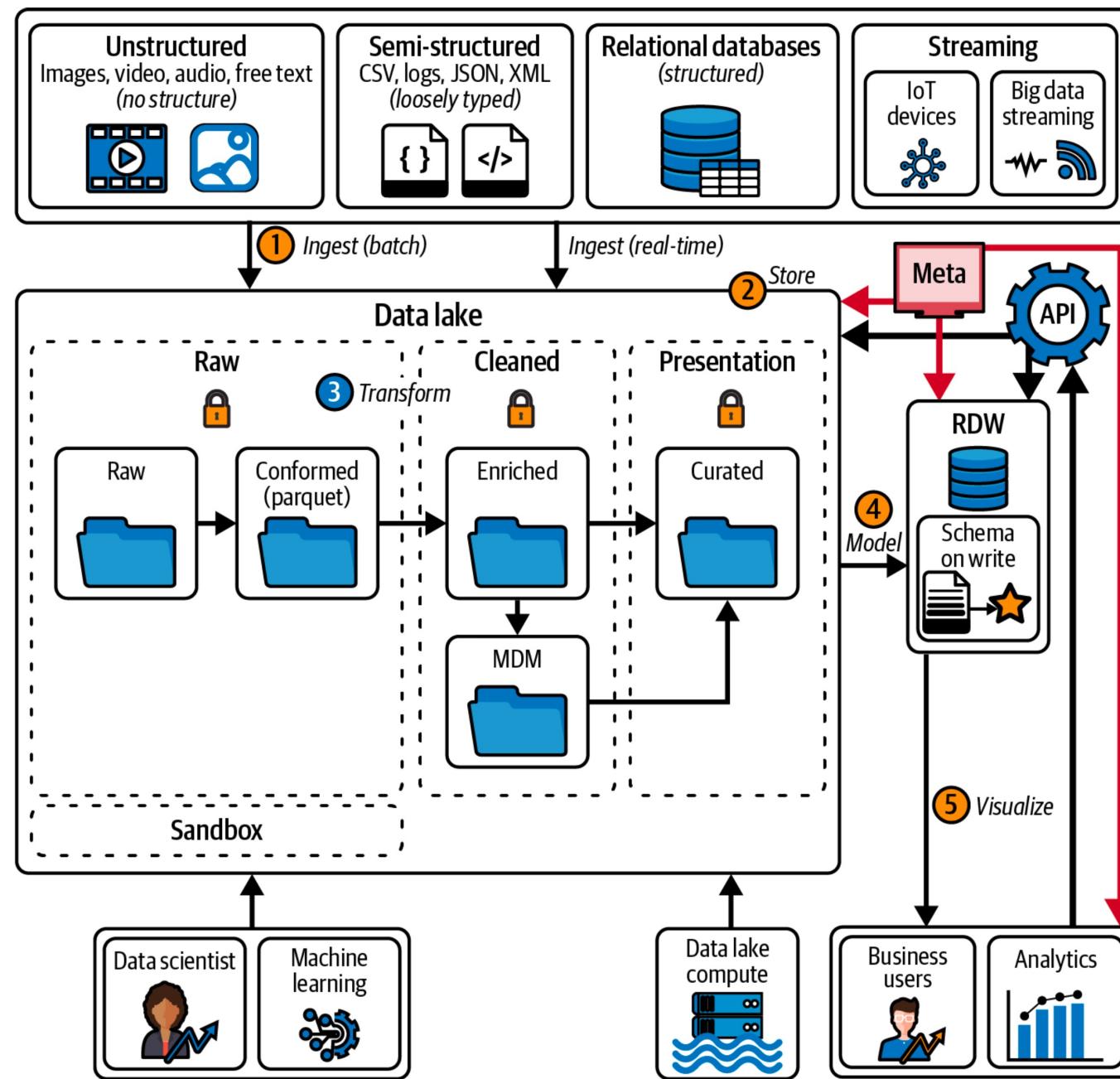
## Modern Data Warehouse



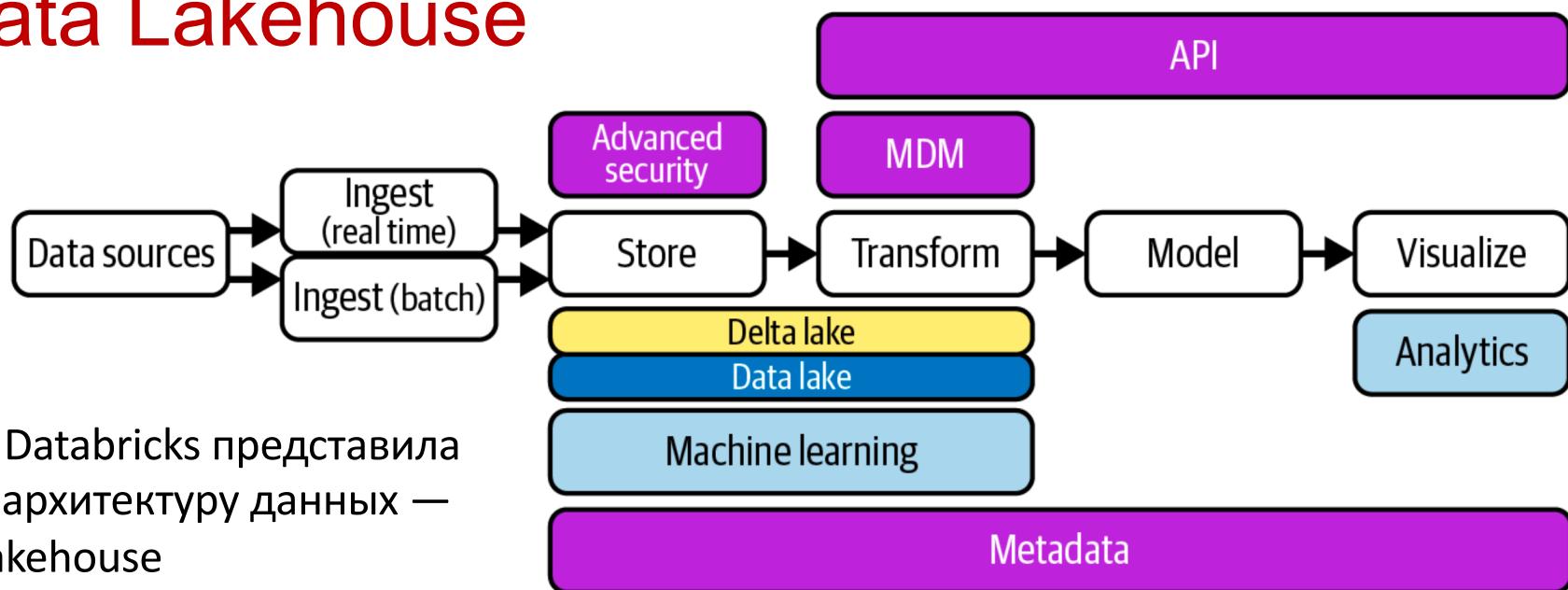
# Data Fabric



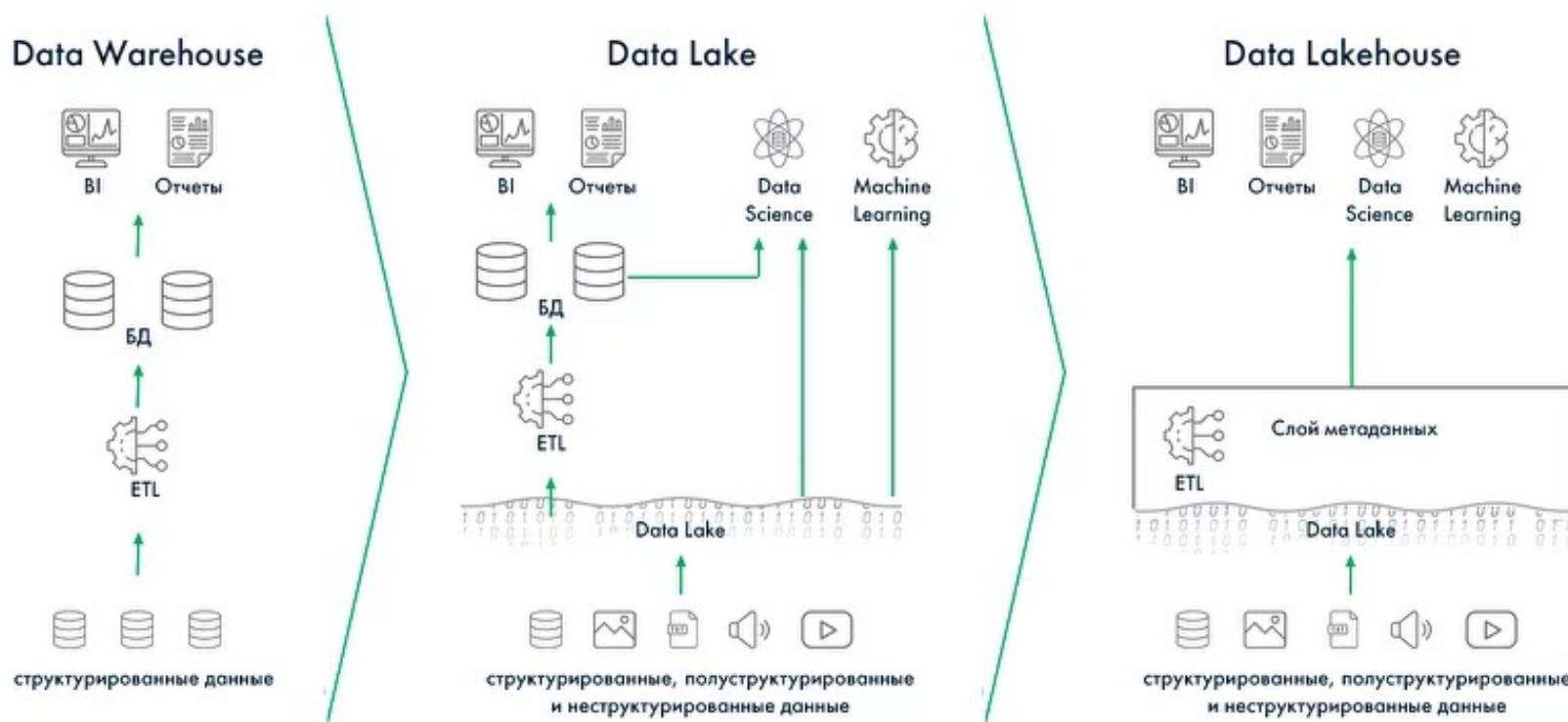
# Data Fabric



# Data Lakehouse

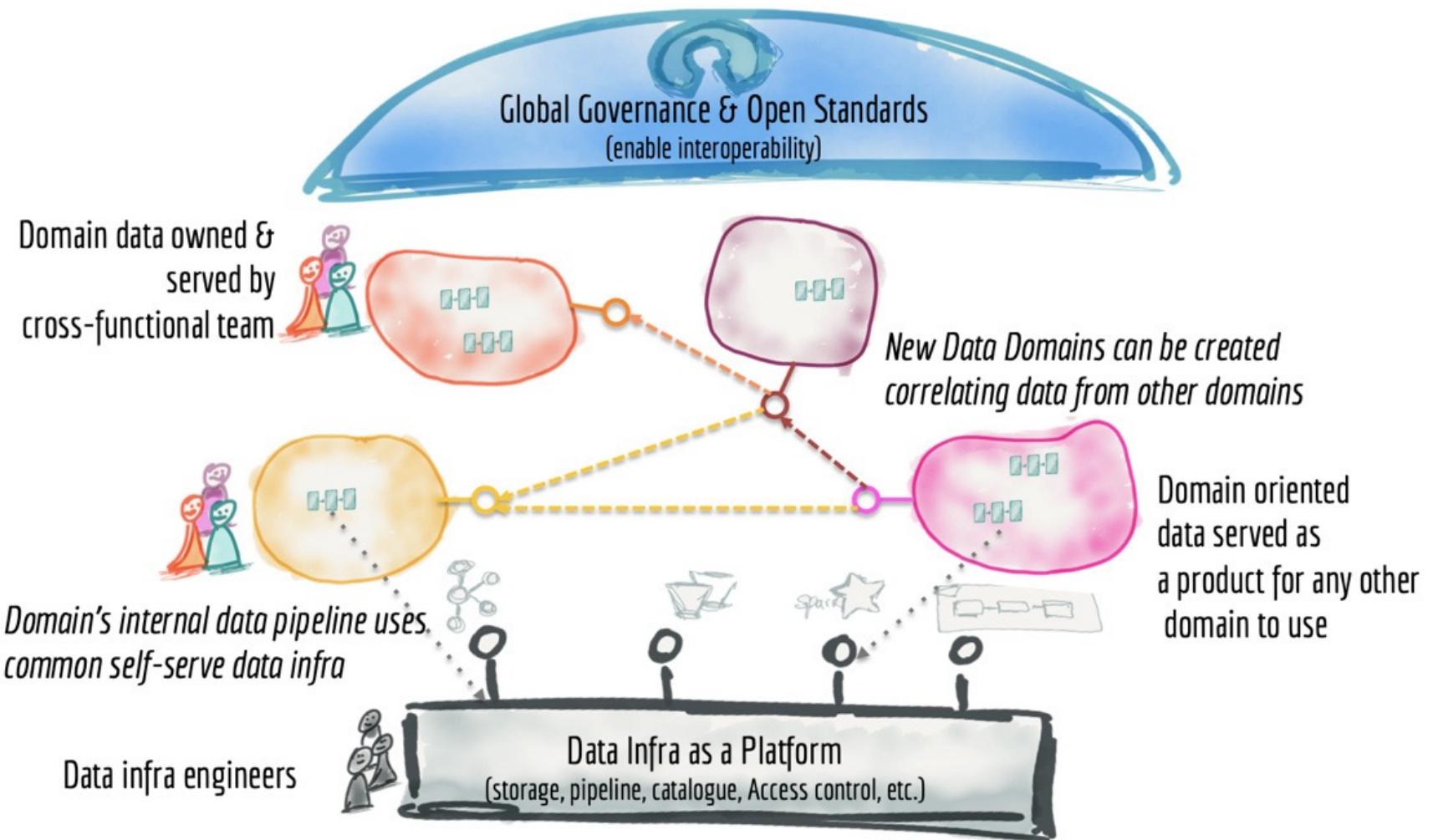
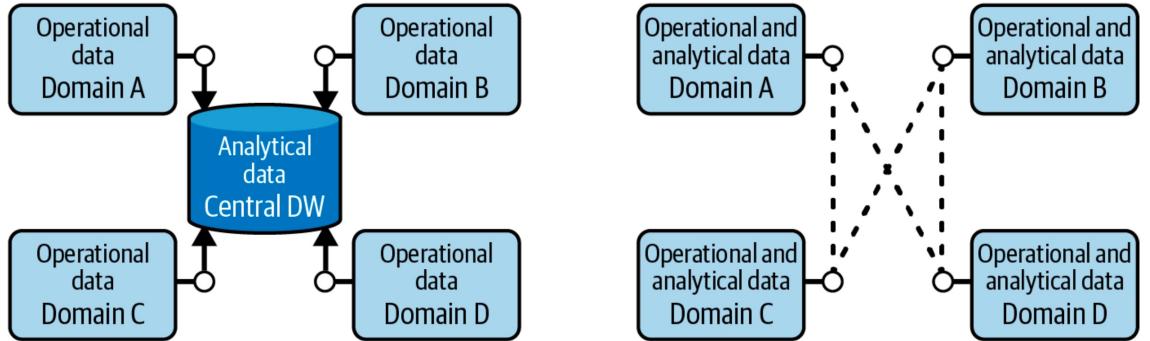


2020 – Databricks представила новую архитектуру данных — Data Lakehouse



# Data Mesh

Snowflake

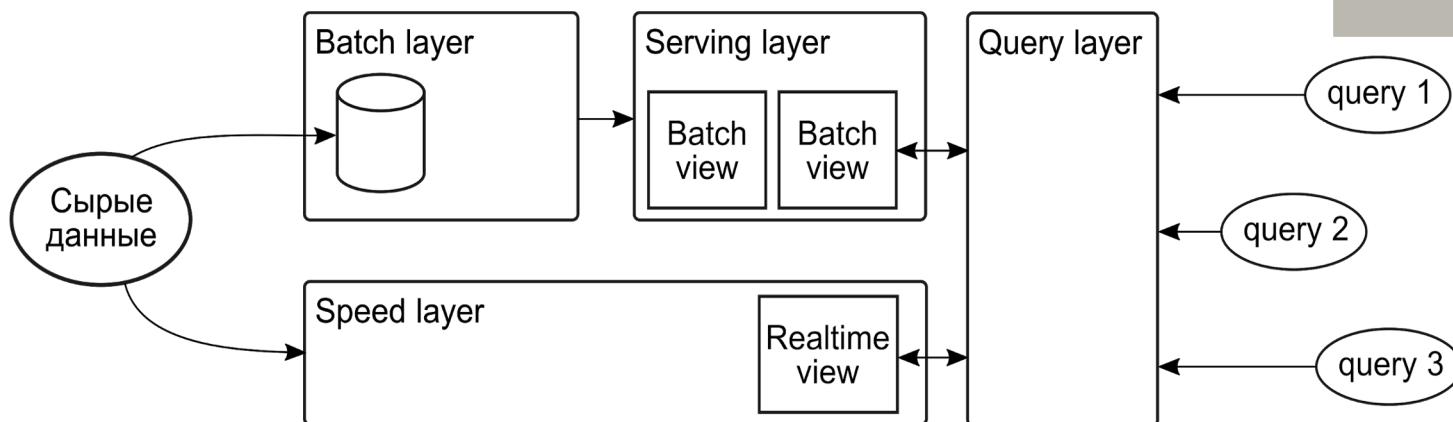
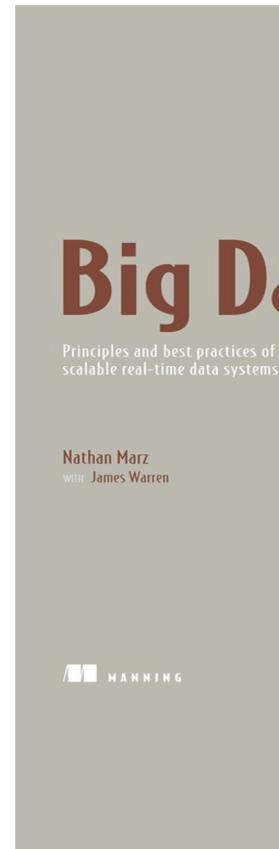


# Lambda Architecture

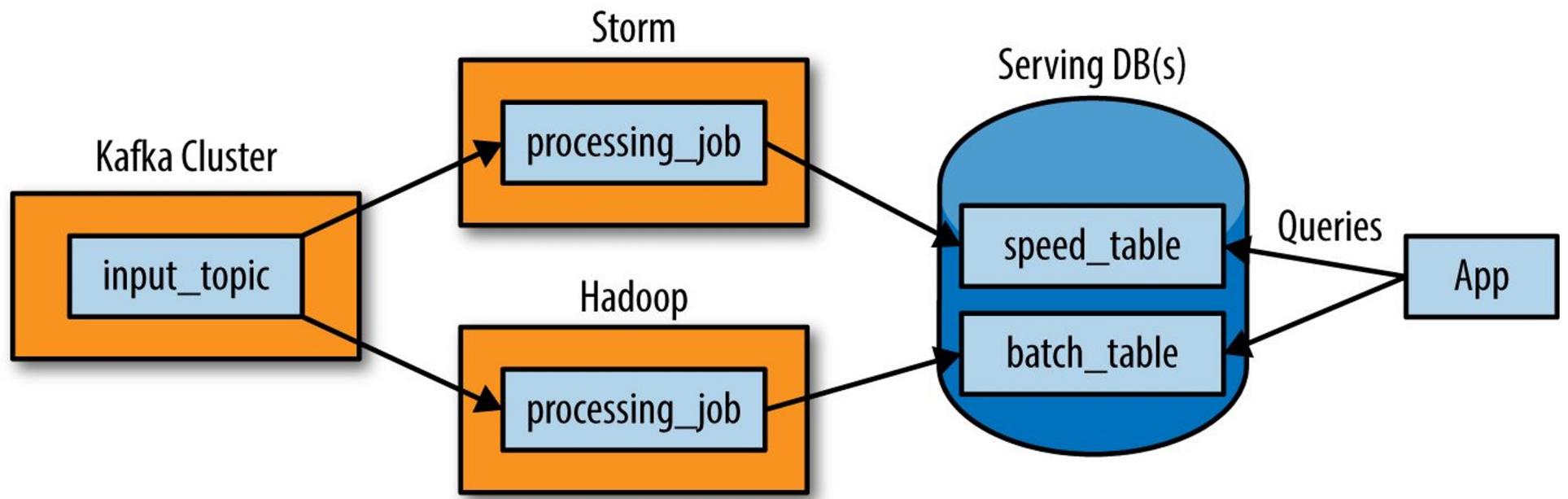
query = function(all data)

batch view = function(all data)  
query = function(batch view)

batch view = function(all data)  
realtime view = function(realtime view, new data)  
query = function(batch view, realtime view)



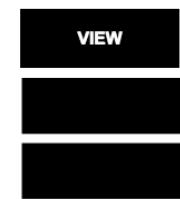
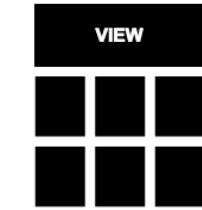
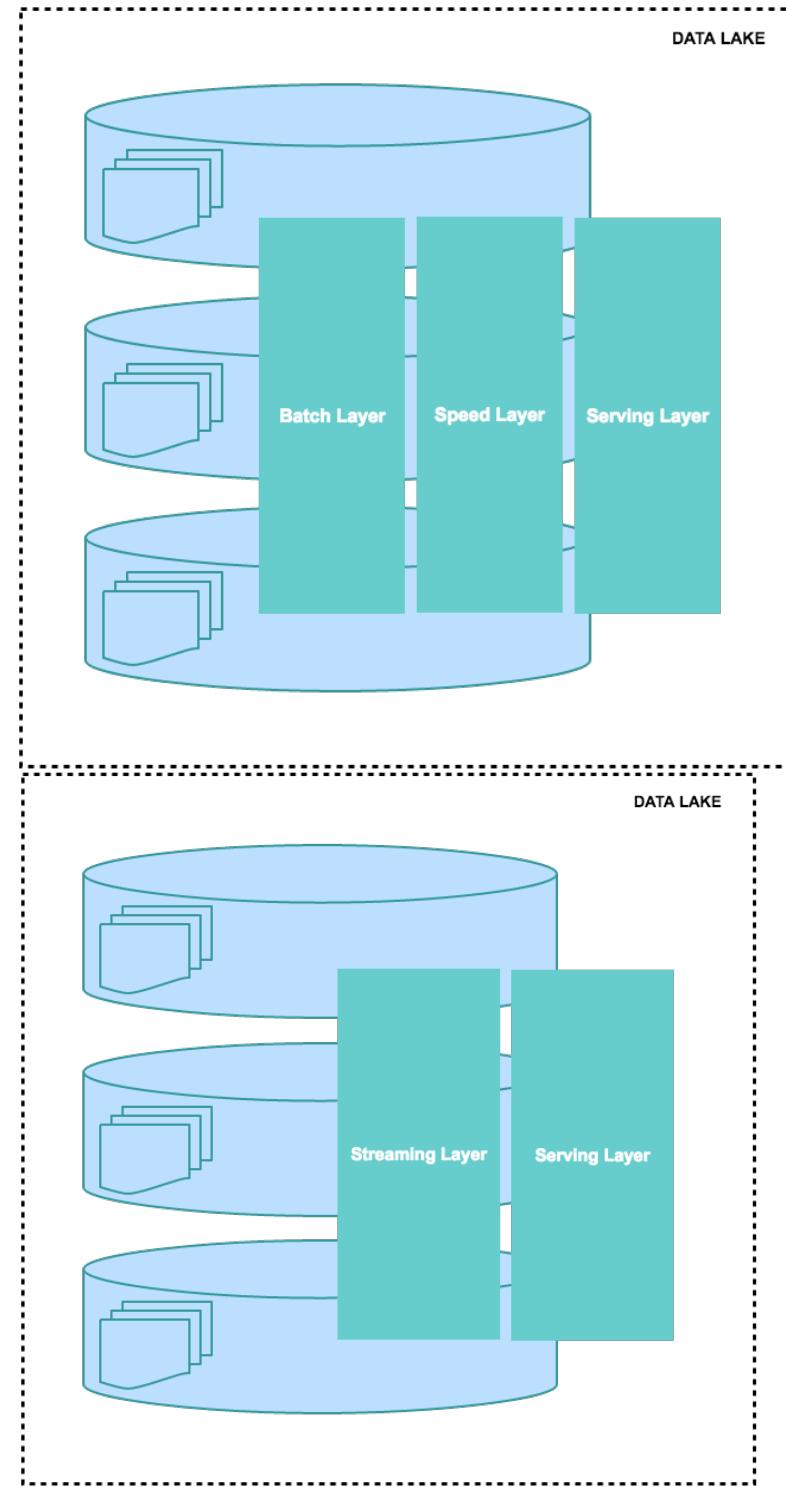
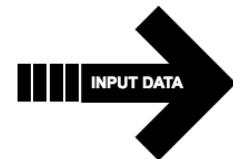
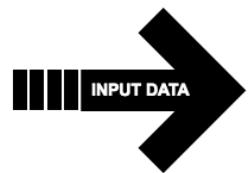
# Реализация Lambda-архитектуры



# Каппа vs. Лямбда архитектуры

- Общее
  - основным хранилищем также является append-only неизменяемый лог,
  - классические базы данных и хранилища типа ключ/значения являются вспомогательными для обслуживания клиентов
- Различие
  - отсутствует слой пакетной обработки, роль которого выполняет потоковая система
- Команда разработчиков использует **единий** фреймворк для обработки данных.

$\lambda$  vs  $\kappa$



# Каппа & Лямбда архитектуры: список литературы

<http://lambda-architecture.net>

<http://kappa-architecture.com>

- Marz N., Warren J. Big Data: Principles and best practices of scalable real-time data systems. – New York; Manning Publications Co., 2015.
- Kleppmann M. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. – " O'Reilly Media, Inc.", 2017.
- Kreps J. I Heart Logs: Event Data, Stream Processing, and Data Integration. – " O'Reilly Media, Inc.", 2014.