DWH

Лекция №1

Введение

Знакомство

Никита Стародубцев

Владислав Алехин

Немного деталей

Характеристики:

- 450 узлов hadoop
- 20 петабайт информации
- 100Тб прирост в день

Основные данные:

- Web Serfing
- Social Networks
- Mobile Data

Цели курса

- Подготовить будущих архитекторов хранилищ данных;
- Дать обширные знания в области проектирования процессов обработки больших объемов данных;
- Научить делать обоснованный выбор архитектуры хранилища;
- Выстраивать процесс внедрения от презентации концепции до полноценного функционирования системы.

Получаемые навыки

- Умение доказать необходимость внедрения ХД;
- Умение выбрать между подходами к построению ХД по Кимбаллу и Инмону;
- Знание основных подходов к проектированию БД (Star Schema, Data Vault, Anchor Modelling) и умение сделать обоснованный выбор между ними;
- Умение проектировать потоки данных с помощью code-driven средств;
- Базовые навыки работы с MPP системами и Hadoop;
- Навык выбора СУБД, модели данных и ETL-инструмента адекватно задаче.

Содержание курса

- 1. Лекция: Введение. Структура курса. Понятие и назначение хранилищ данных.
- 2. Лекция: История развитий подходов к построению Хранилищ данных. Билл Инмон. Ральф Кимбалл.
- 3. Лекция: Проектирование схемы БД по схеме Data Vault, Anchor modeling. Разбор первого ДЗ.
- 4. Семинар: Проектирование модели хранилища данных + ДЗ №1(10 баллов)
- 5. Лекция: MPP-системы
- 6. Смешанное занятие: Hadoop и его основные компоненты: HDFS, MapReduce, YARN. : Практика на кластере: подключение к кластеру, запуск MapReduce задач(15 баллов)
- 7. Смешанное занятие: Hadoop. Знакомство с Hive. Практика на кластере: основы работы с Hive, запуск простых запросов.(доп. задание 5 баллов)
- 8. Смешанное занятие: Hadoop. Знакомство со Spark. Практика на кластере: подключение к кластеру, запуск MapReduce задач + ДЗ №2(35 баллов)
- 9. Смешанное занятие: Потоки данных ETL. Code-Driven ETL: Luigi
- 10. Смешанное занятие: Потоки данных ETL. Code-Driven ETL: Airflow
- 11. Итоговый экзамен (35 баллов)

Итоговые оценки:

отлично от 80 баллов
хорошо от 65 баллов
удовлетворительно от 55 баллов

План лекции

1. Базы данных

2. NoSQL, NewSQL

2. Большие данные

4. Системы обработки данных

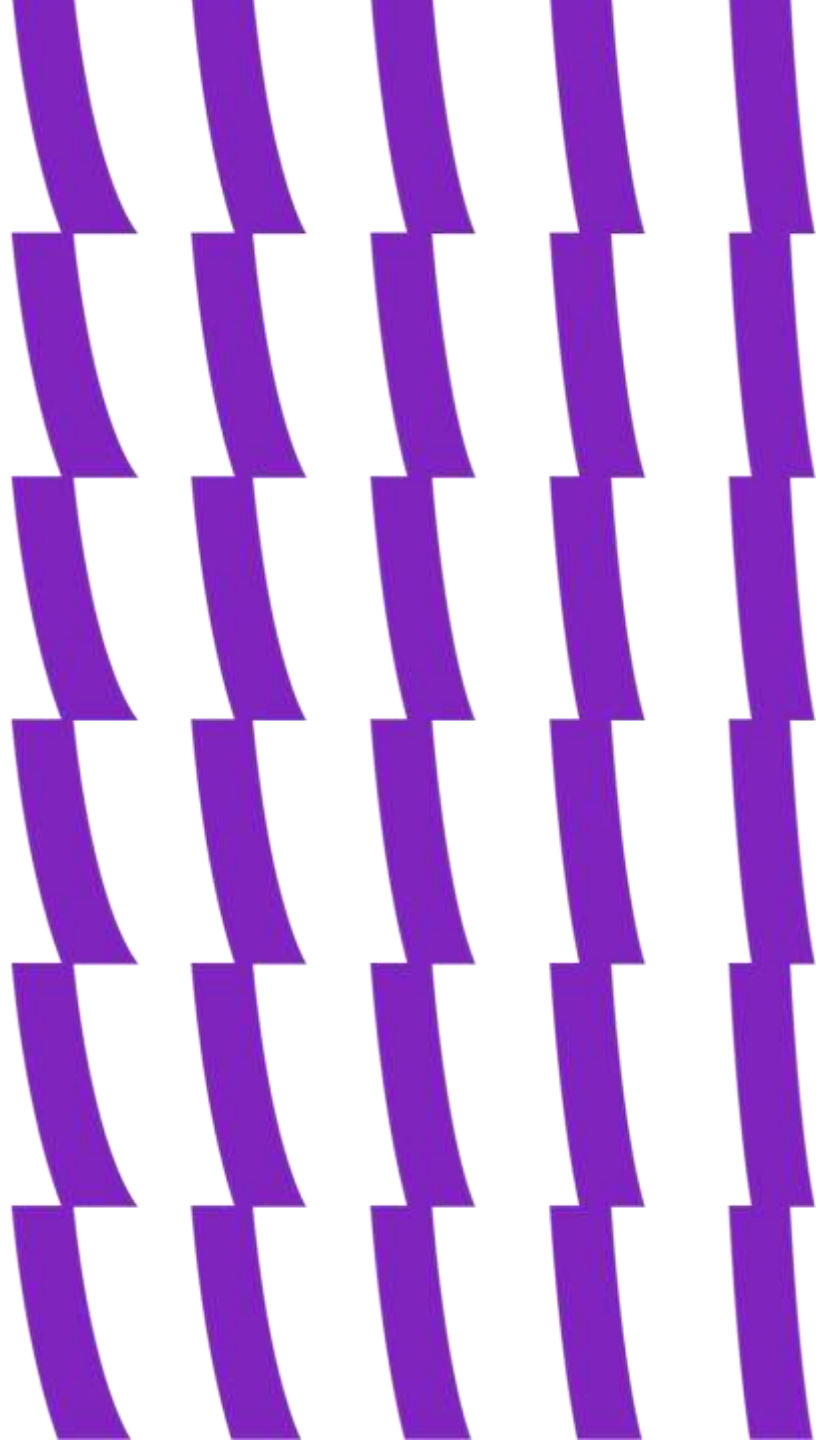
5. DWH

6.Слои данных

Базы данных

База данных

База данных — это некоторый набор перманентных (постоянно хранимых) данных, используемых в процессах и системах какой-либо организации



База данных.
Преимущества электронной версии

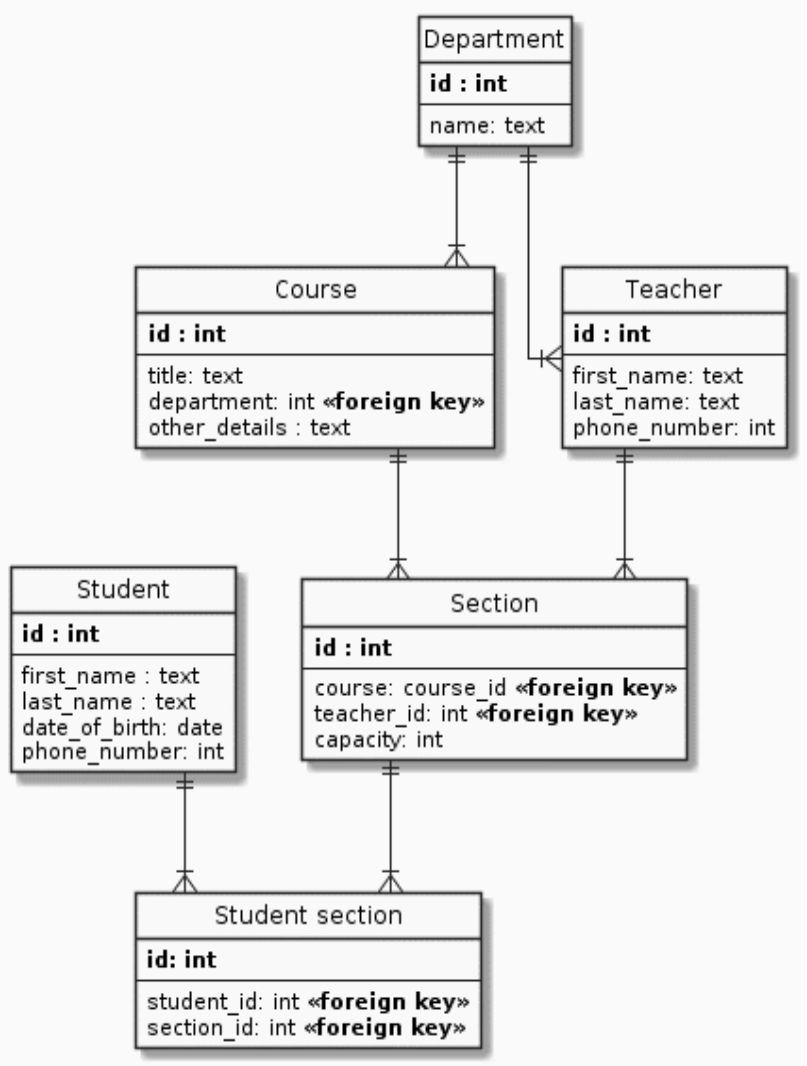
- Компактность. Нет необходимости в создании и ведении, возможно, весьма объемистых бумажных картотек.
- Быстродействие. Компьютер может выбирать и обновлять данные гораздо быстрее человека. В частности, с его помощью можно быстро получать ответы на произвольные вопросы, возникающие в процессе работы, не затрачивая времени на визуальный осмотр или поиск вручную.
- Низкие трудозатраты. Отпадает необходимость в утомительной работе над картотекой вручную. Механическую работу машины всегда выполняют лучше.
- Актуальность. В случае необходимости под рукой в любой момент имеется точная, свежая информация.
- Защита. Данные могут быть лучше защищены от случайной потери и несанкционированного доступа.

Реляционные БД

В 1970-е годы Эдгар Кодд опубликовал свою работу, в которой предложил 8 основных операций реляционной алгебры, после этой статьи активное развитие получили технологии развития реляционных баз данных.

Данные и связи между данными хранятся в таблицах(отношения). Каждая строка – отдельная запись(кортеж). Каждый столбец имеет своё имя и тип данных.

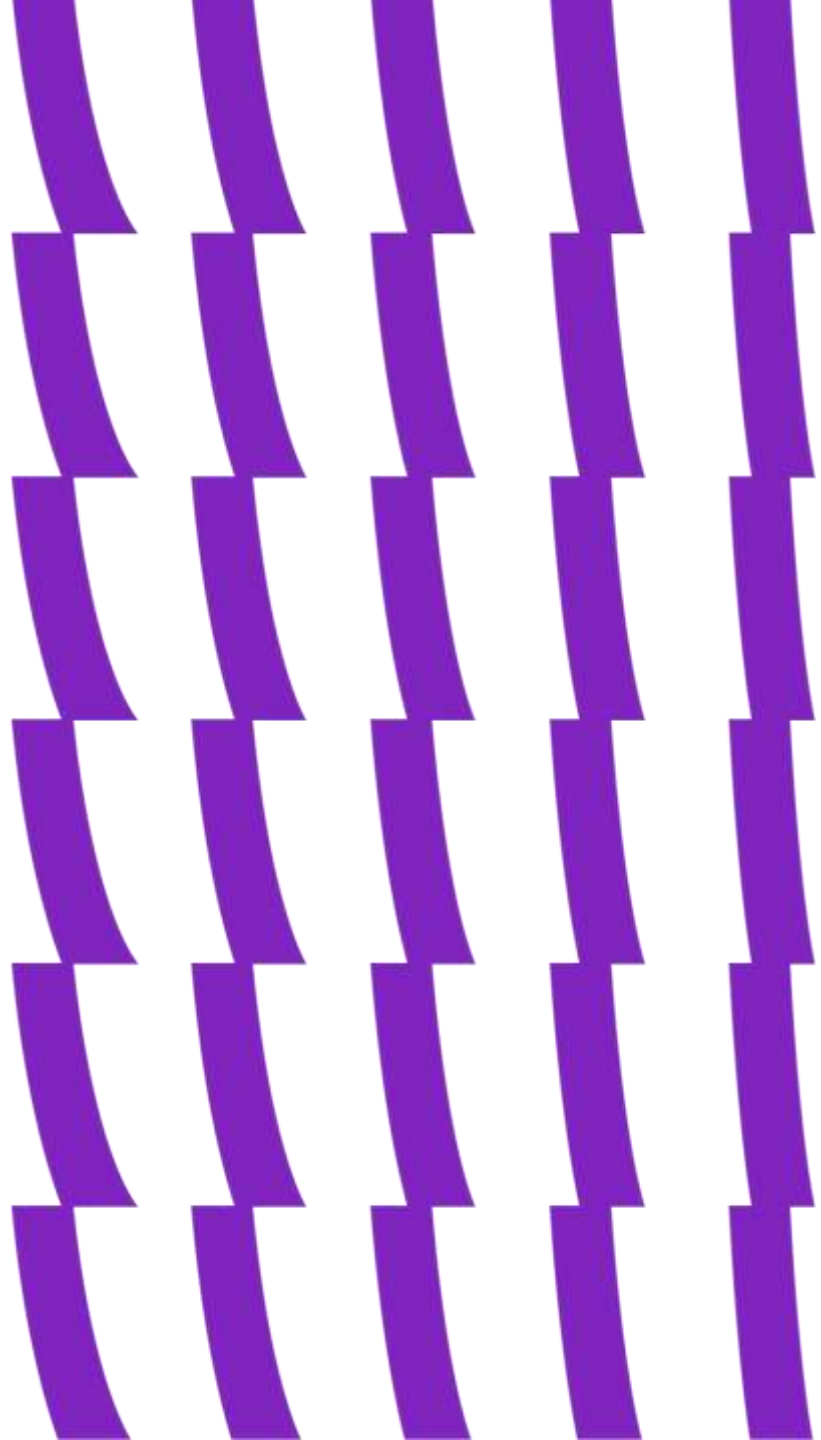
Для связей таблиц используется внешний ключ. Некоторое поле в таблице(множество полей), которое содержит ссылки на столбцы в других таблицах.



СУБД

СУБД – совокупность языковых и программных средств, предназначенных для создания, ведения и использования информации, хранящейся в БД.

Главная цель СУБД – предоставить пользователю возможность оперировать данными в близких ему терминах и понятиях, не связанных с конкретными способами хранения данных в компьютере.



Функции СУБД

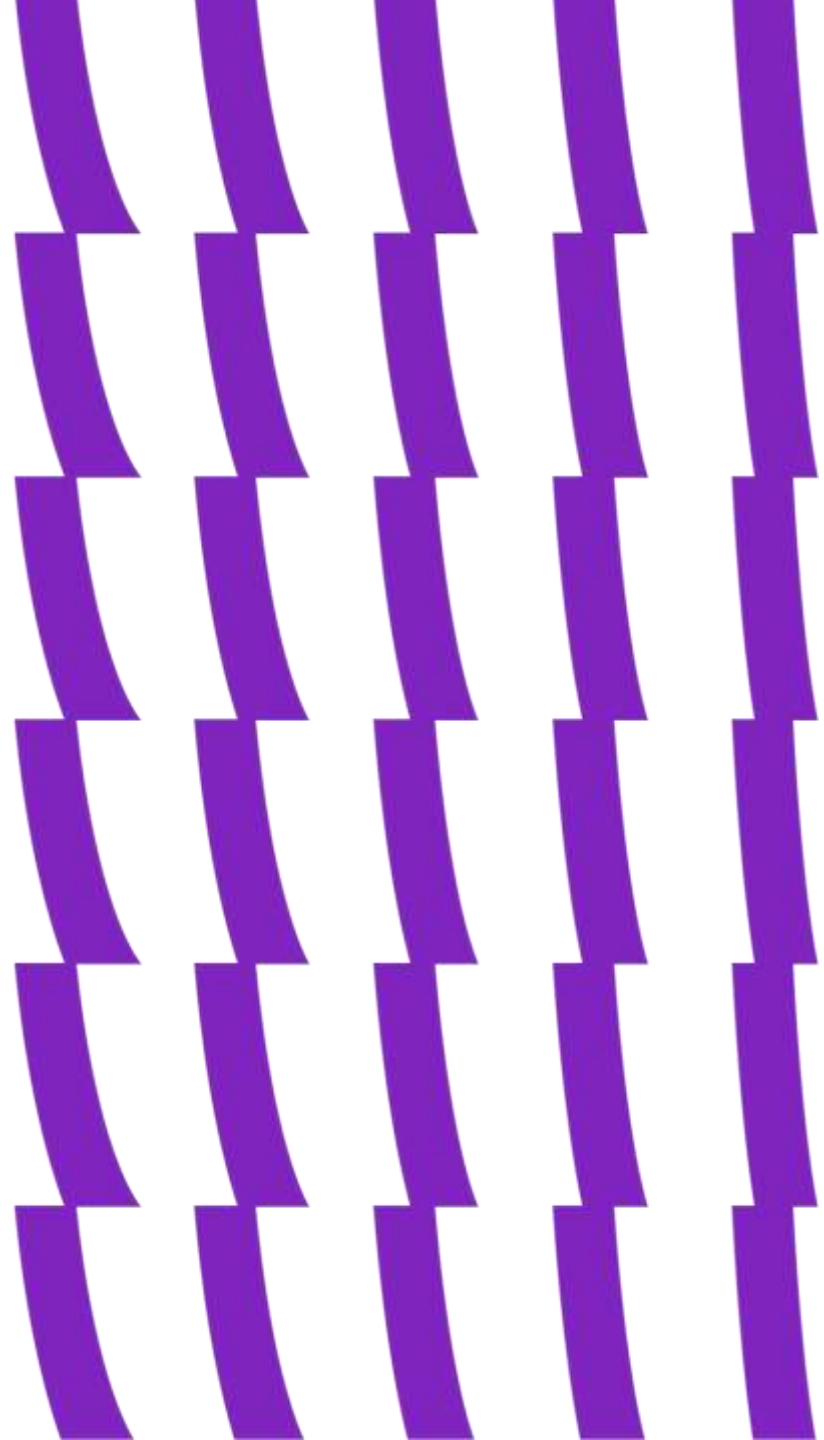
К основным функциям, которые выполняются системами управления базами данных относятся:

1. непосредственное управление данными во внешней памяти. Для сохранности данных и работы с ними, в базе данных нужно иметь в наличии постоянные запоминающие устройства (носители информации), СУБД использует собственный механизм работы с внешней памятью ;
2. управление буферами оперативной памяти. С целью повышения скорости используется буферизация данных в оперативной памяти, обмен информацией осуществляется через этот буфер ;
3. управление транзакциями;
4. ведение журнала или протокола выполненных операций в базе данных, необходимо для обеспечения надежности сохранения данных во внешней памяти;
5. поддержка языков баз данных, для определения схемы данных и манипулирования этими данными.

Свойства СУБД

Для всех существующих систем управления базами данных можно выделить следующие основные свойства:

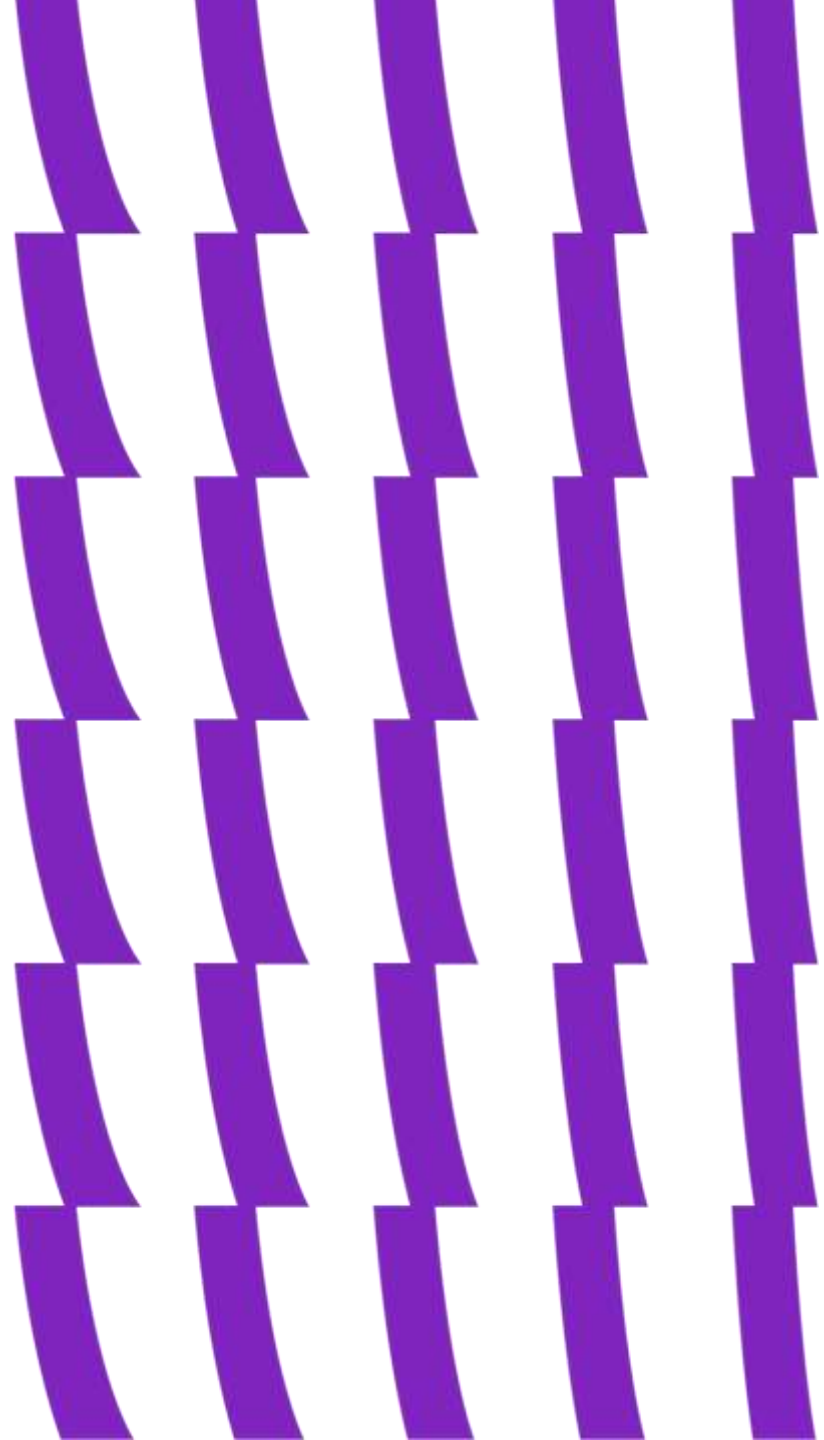
- Поддержка логически согласованного набора файлов. Любая база данных может состоять из одного или нескольких (множества) файлов;
- Обеспечение языка манипулирования данными (Data Manipulation Language или DML). Например, современные системы управления реляционными базами данных поддерживают язык SQL (Structured Query Language) являющийся стандартом;
- Возможность восстановления информации после ее потери в случае сбоев;
- Обеспечение параллельной работы нескольких пользователей.



ACID

Требования к РСУБД можно обозначить с помощью свойств ACID. ACID — набор требований к транзакционным системам, обеспечивающий наиболее надёжную и предсказуемую её работу

- Атомарность (Atomicity) требует, чтобы транзакция выполнялась полностью или не выполнялась вообще.
- Согласованность (Consistency) — транзакция не нарушает согласованности данных после своего завершения. Сразу по завершении транзакции данные должны соответствовать схеме базы данных.
- Изолированность (Isolation) — параллельные транзакции не влияют на работу друг друга.
- Стойкость (Durability) — сбой системы не должен влиять на успешно завершённые транзакции. В случае непредвиденного сбоя в системе или перебоя в подаче питания данные будут восстановлены до последнего сохраненного состояния.



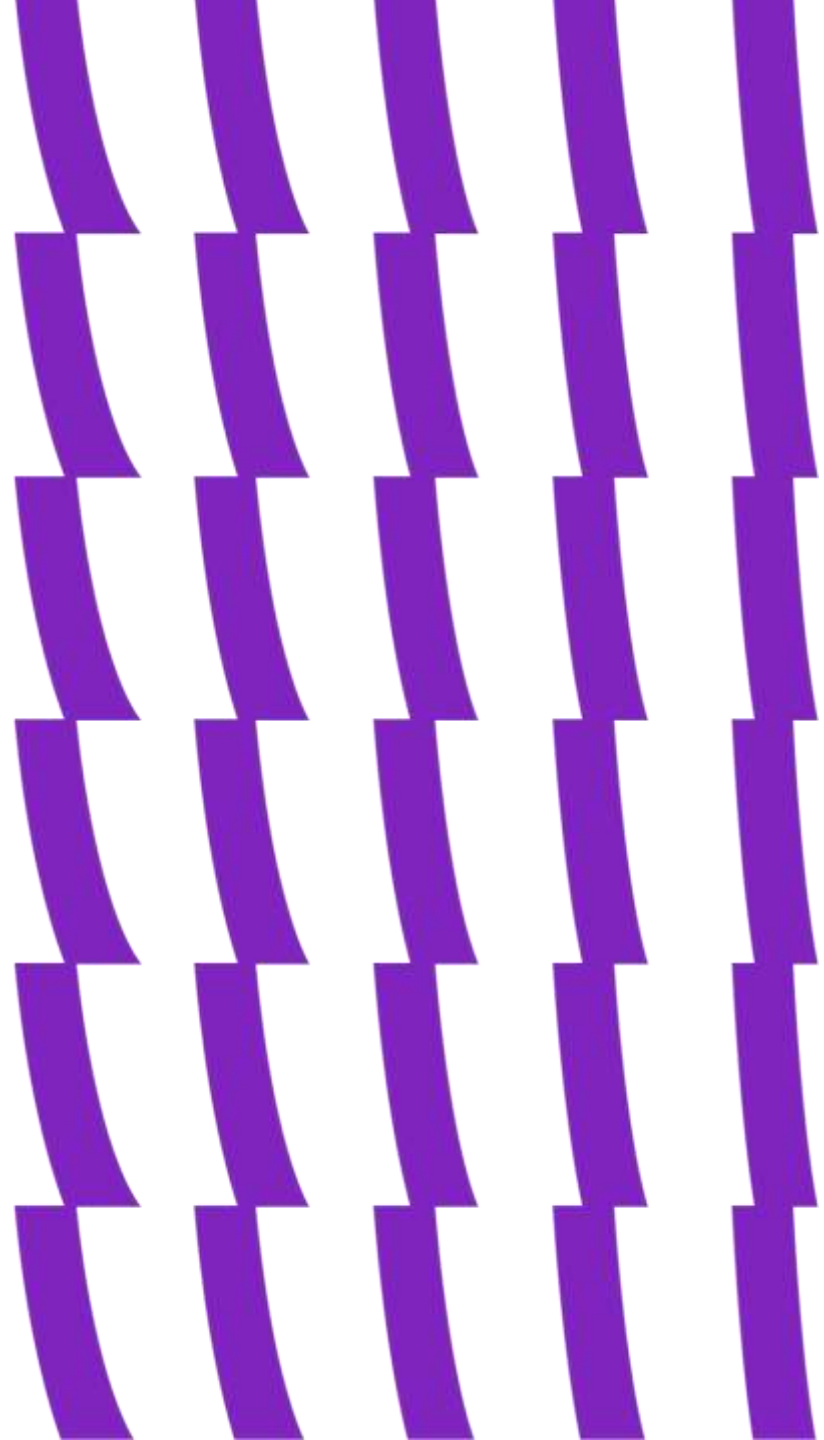
Управление БД

SQL. Операции DDL

- CREATE TABLE
- CREATE TABLE AS SELECT
- CREATE VIEW
- DROP TABLE
- TRUNCATE TABLE

SQL. Операции DML

- INSERT INTO
- INSERT INTO SELECT
- DELETE
- UPDATE

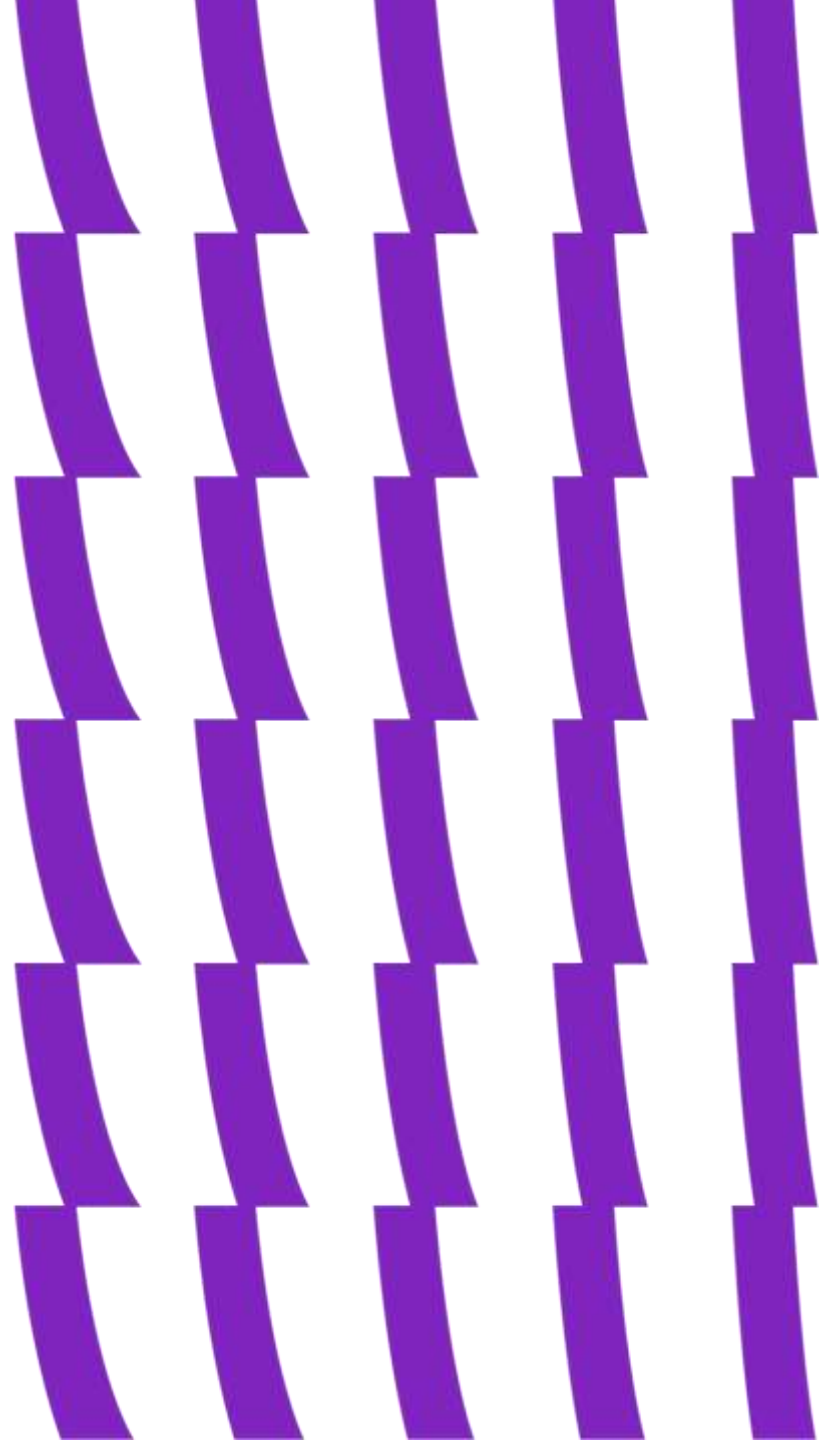


Управление БД

SQL. Операции DQL

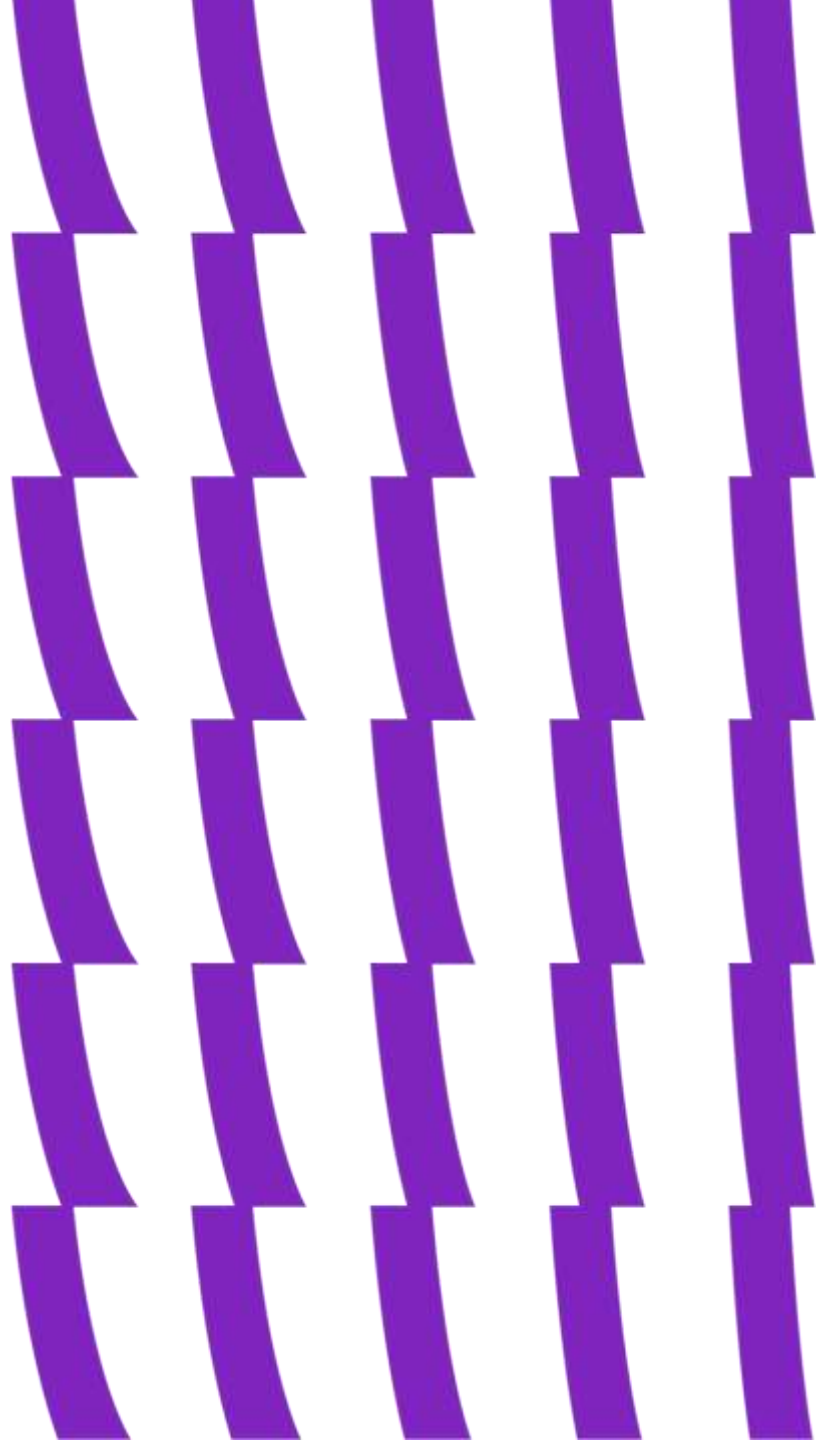
- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY

- JOIN
- UNION
- Подзапрос(subquery)



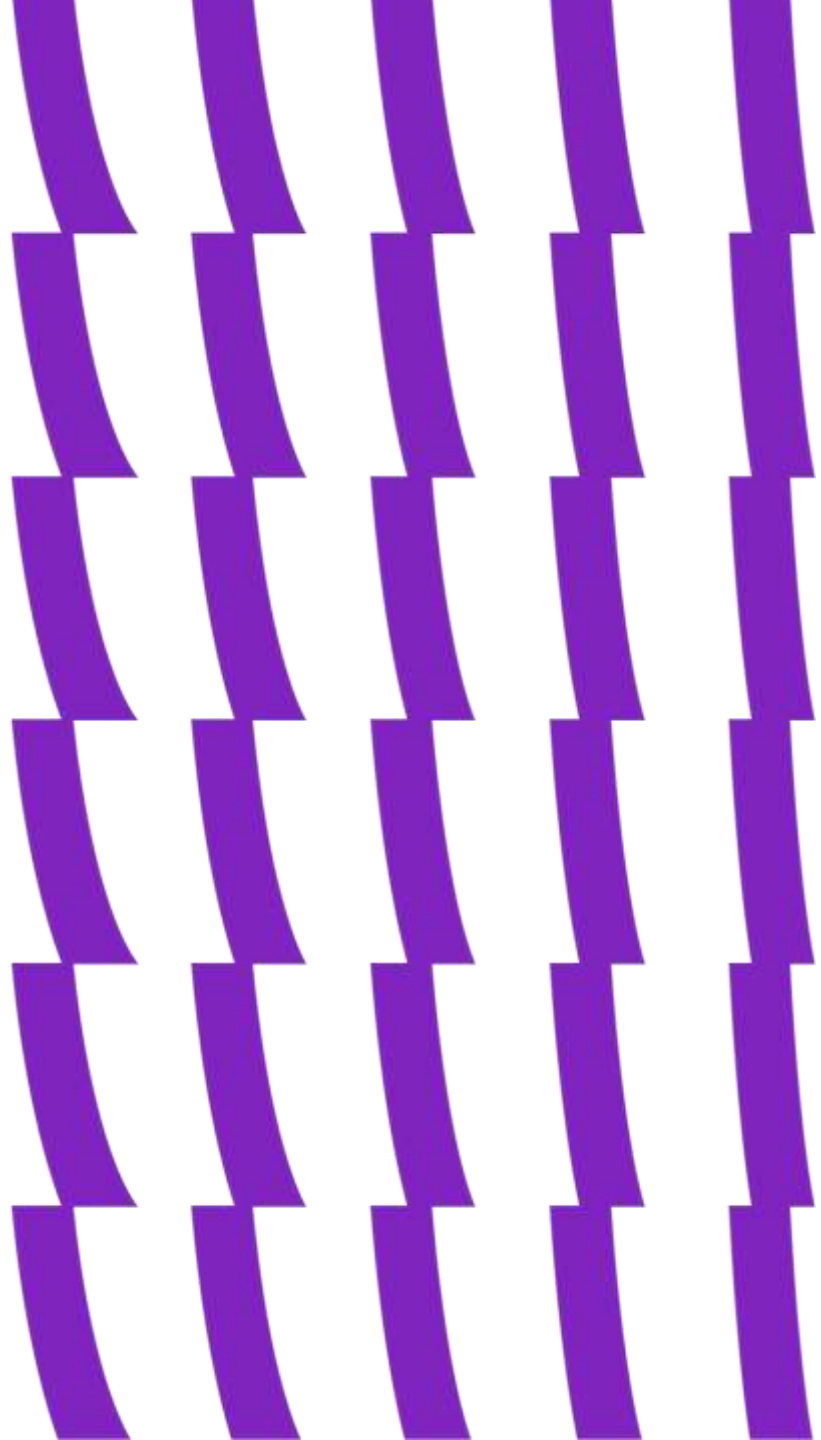
Плюсы Реляционных БД

- Отсутствие дублирования данных;
- Строгие правила проектирования, базирующиеся на математическом аппарате, делают РБД адаптируемыми к различным типам данных;
- Полная независимость данных. Изменения в прикладной программе при изменении реляционной БД минимальны;
- Простота и доступность для понимания пользователем, все данные представлены как факты, хранящиеся в виде отношений (relations) со столбцами (attributes) и строками (tuples).



Минусы Реляционных БД

- ACID свойства не позволяют наращивать производительность реляционных систем;
- Проблемы с горизонтальным масштабированием;
- Работают только с структурированными данными.

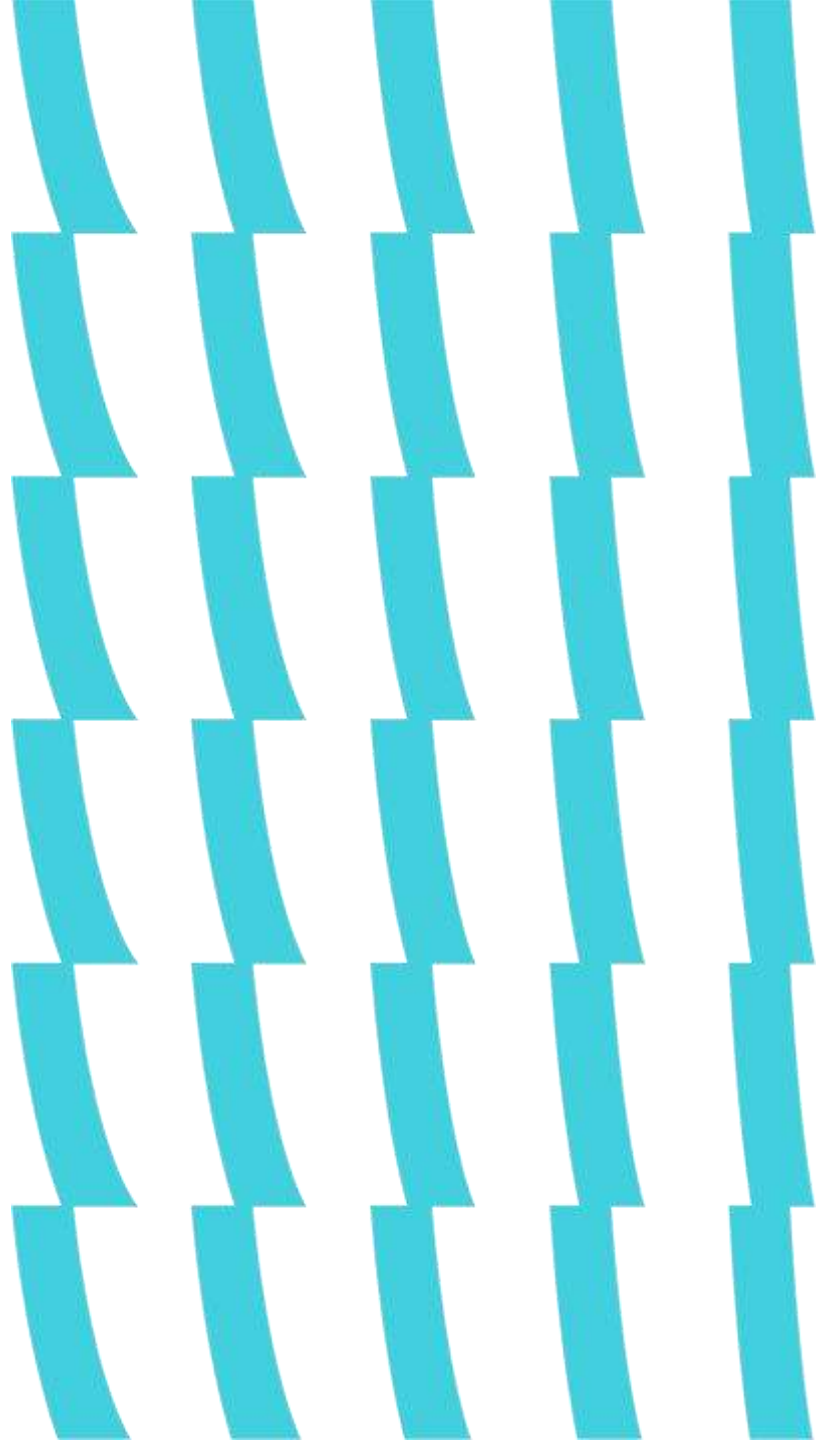


NoSQL, NewSQL

NoSQL — смена парадигмы

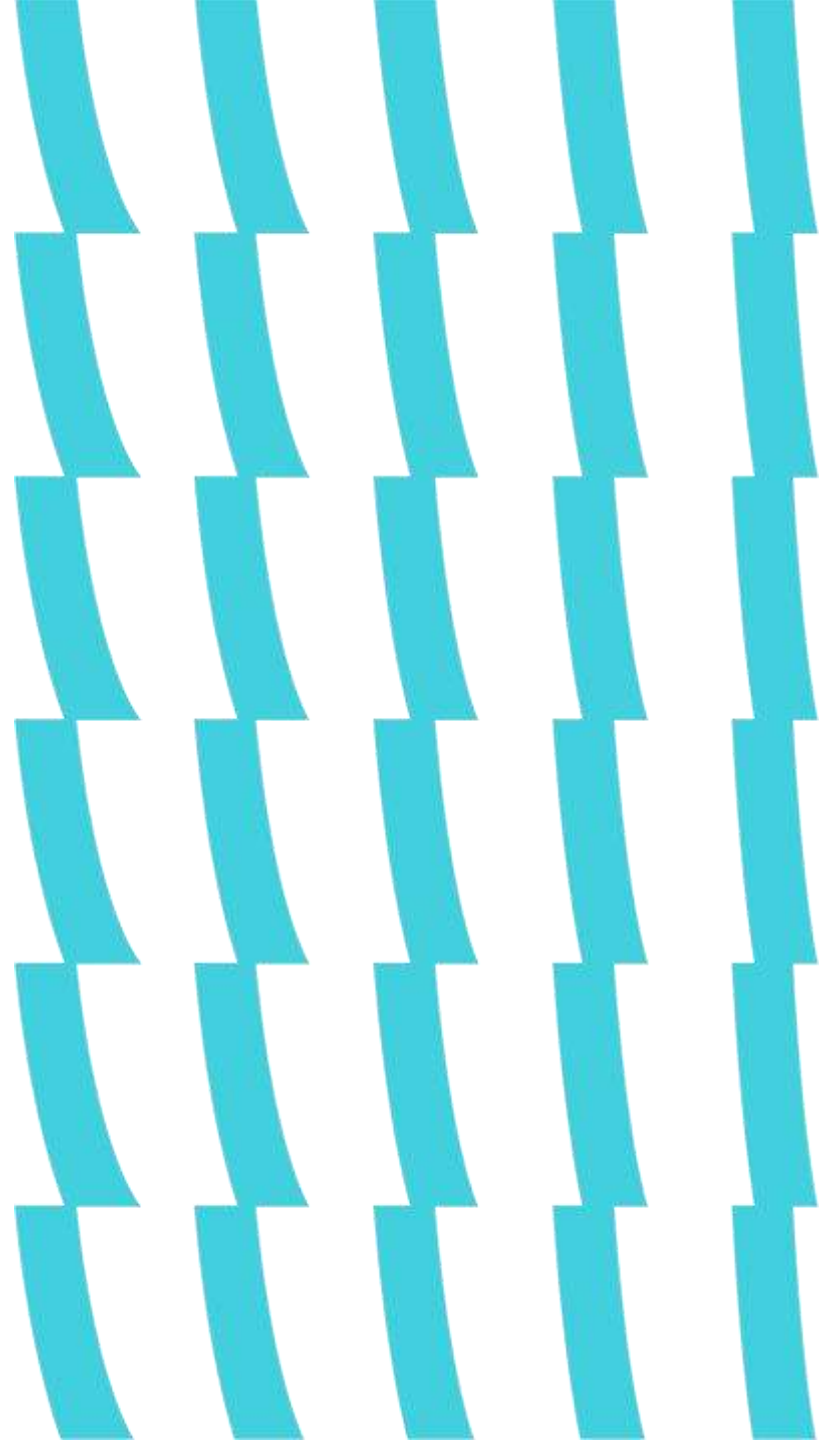
Взрывной рост интернет-технологий привел к появлению большого количества полезных данных, использование и хранение которых традиционными методами становится затруднительно.

Для такого типа данных создаются и развиваются NoSQL решения, которые способны обеспечивать горизонтальную масштабируемость и большую отзывчивость, но жертвуют при этом надежностью транзакций. NoSQL работает с документами без схемы, которые хранят данные в документах, графе, ключе-значении и неупорядоченном виде.



NoSQL — смена парадигмы

Базы данных NoSQL зачастую предлагают компромисс, смягчая жесткие требования свойств ACID ради более гибкой модели данных, которая допускает горизонтальное масштабирование. Благодаря этому БД NoSQL – отличный выбор для примеров использования с высокой пропускной способностью и низкой задержкой.



BASE вместо ACID

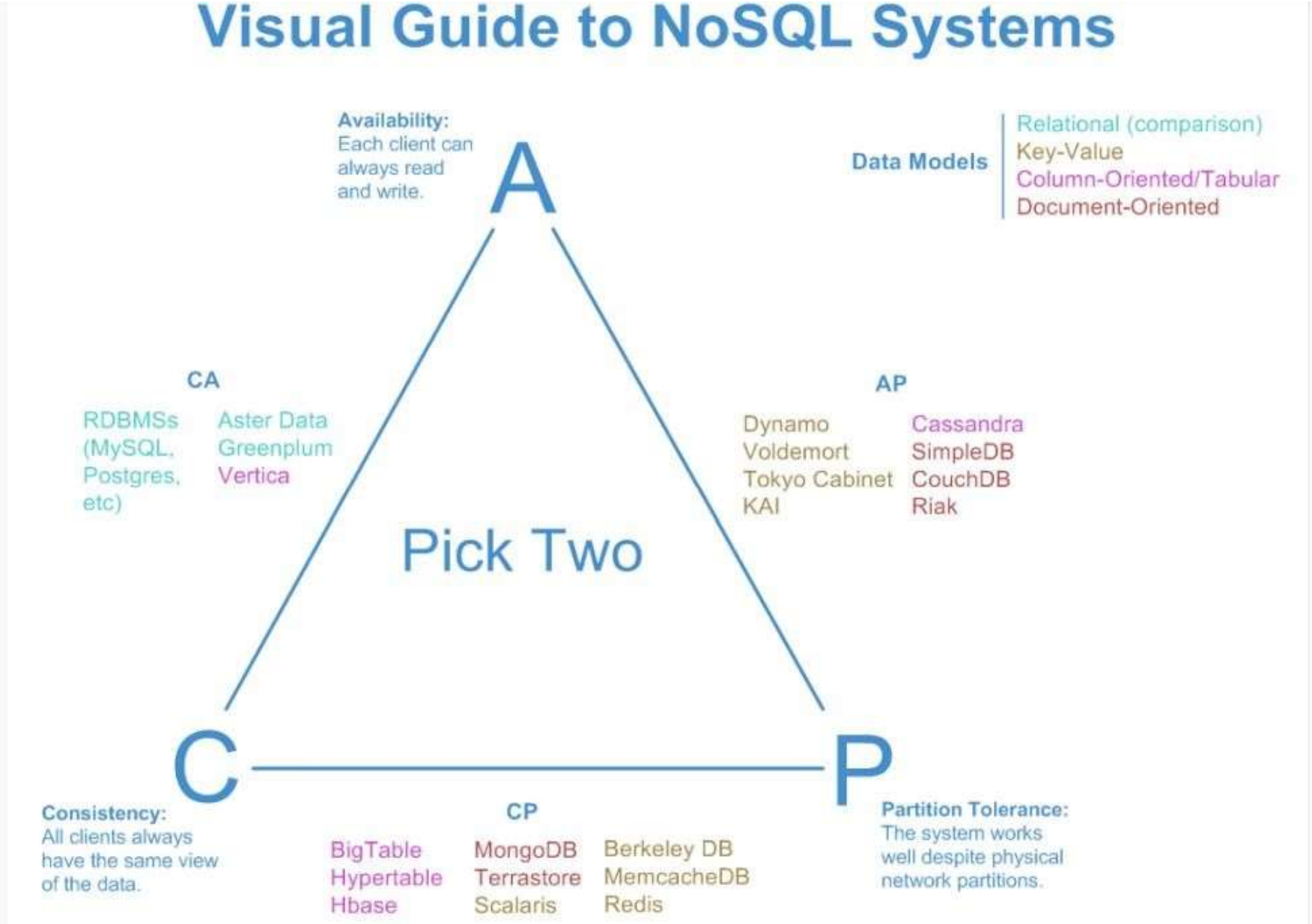
- Basically Available – базовая доступность. Доступность данных даже при наличии множества сбоев, достигается путем использования распределенного подхода к управлению базами данных. Вместо того чтобы поддерживать одно большое хранилище данных базы данных NoSQL распределяют данные по многим системам хранения с высокой степенью репликации.
- Soft State – гибкое состояние. Одна из основных концепций, лежащих в основе BASE, заключается в том, что согласованность данных является проблемой разработчика и не должна обрабатываться базой данных.
- Eventually Consistent – согласованность в конечном счете. Единственное требование, которое базы данных NoSQL имеют в отношении согласованности, – это требование, чтобы в какой-то момент в будущем данные конвертировались в согласованное состояние. Это полное отклонение от требования немедленной согласованности ACID, которое запрещает выполнение транзакции до тех пор, пока предыдущая транзакция не будет завершена и база данных не приблизится к согласованному состоянию.

NoSQL — смена парадигмы

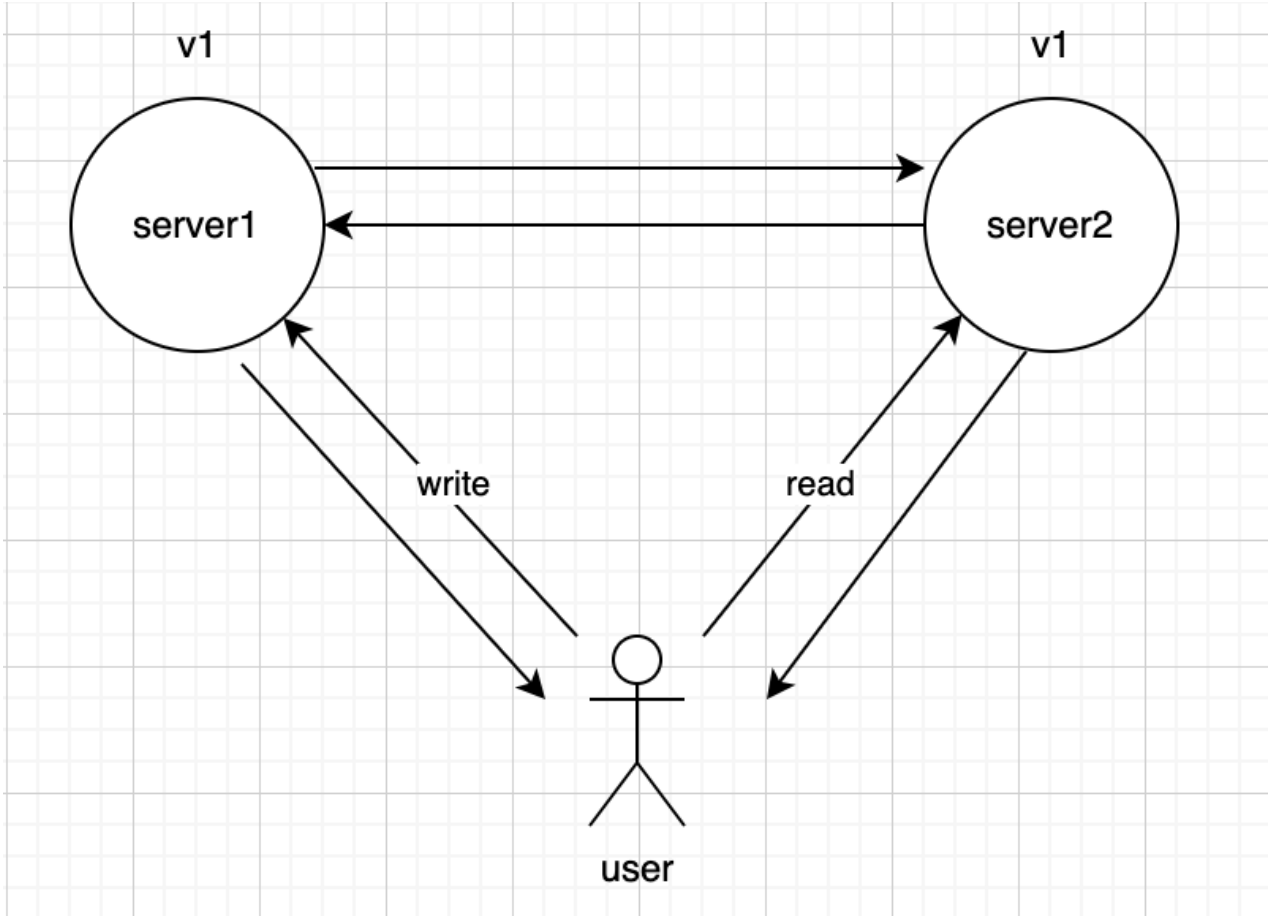
Формулировка была представлена на симпозиуме по принципам распределенных вычислений в 2000 году Эриком Брюером. В 2002 году Сет Гилберт и Нэнси Линч из MIT опубликовали формальное доказательство гипотезы Брюера, сделав её теоремой.
CAP-теорема описывает три ключевых свойства распределённых систем и утверждает, что добиться идеального выполнения всех трёх невозможно:

- C (consistency) — Согласованность. На каждое чтение получаем актуальную запись или ошибку.
- A (availability) — Доступность. На каждый запрос получаем корректный ответ (т. е. без ошибки), но актуальность результата не гарантируется.
- P (partition tolerance) — Устойчивость к распределению. Система способна функционировать независимо от потери данных между узлами в распределённой системе. Разделение распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

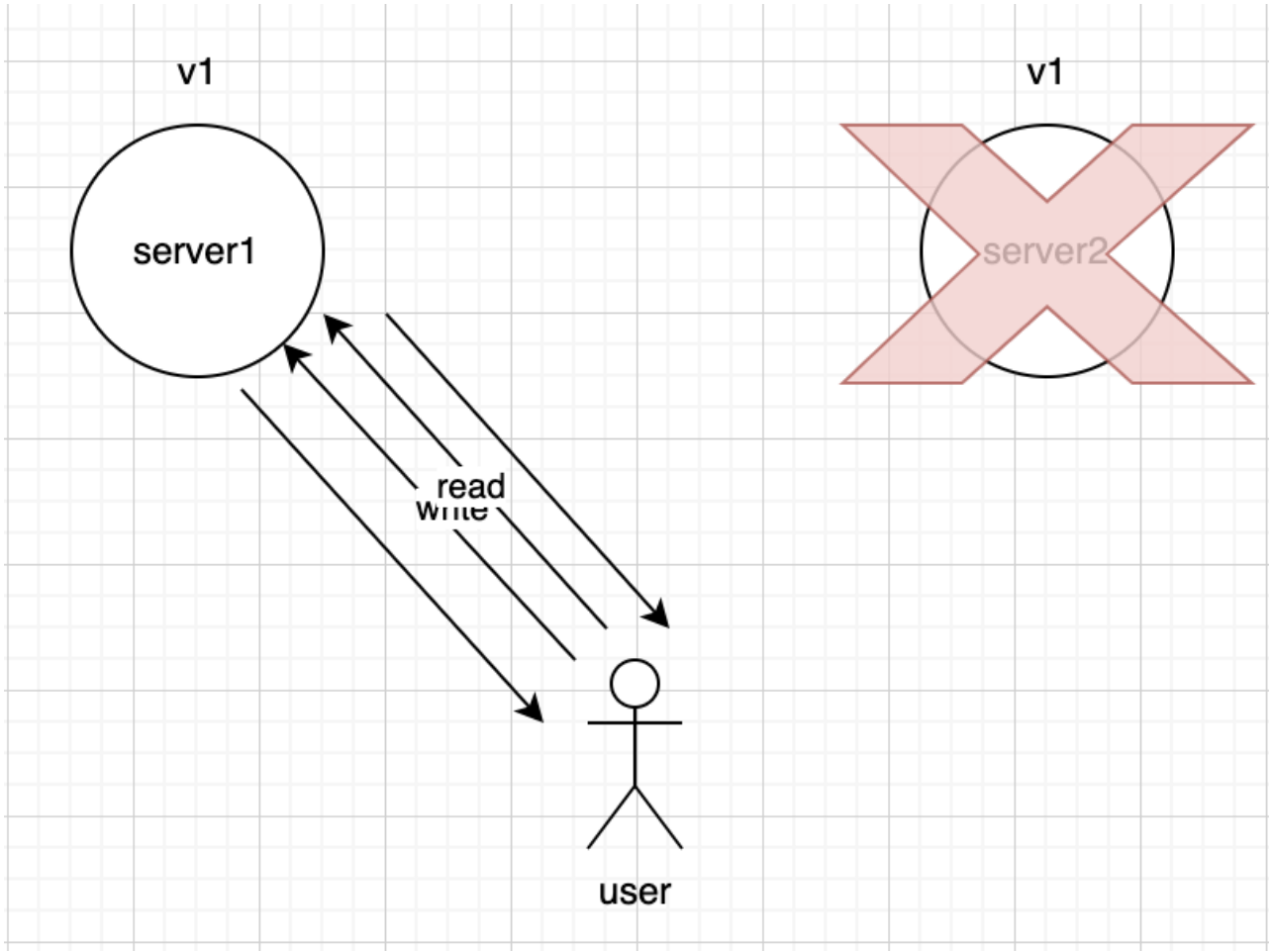
NoSQL. CAP-теорема



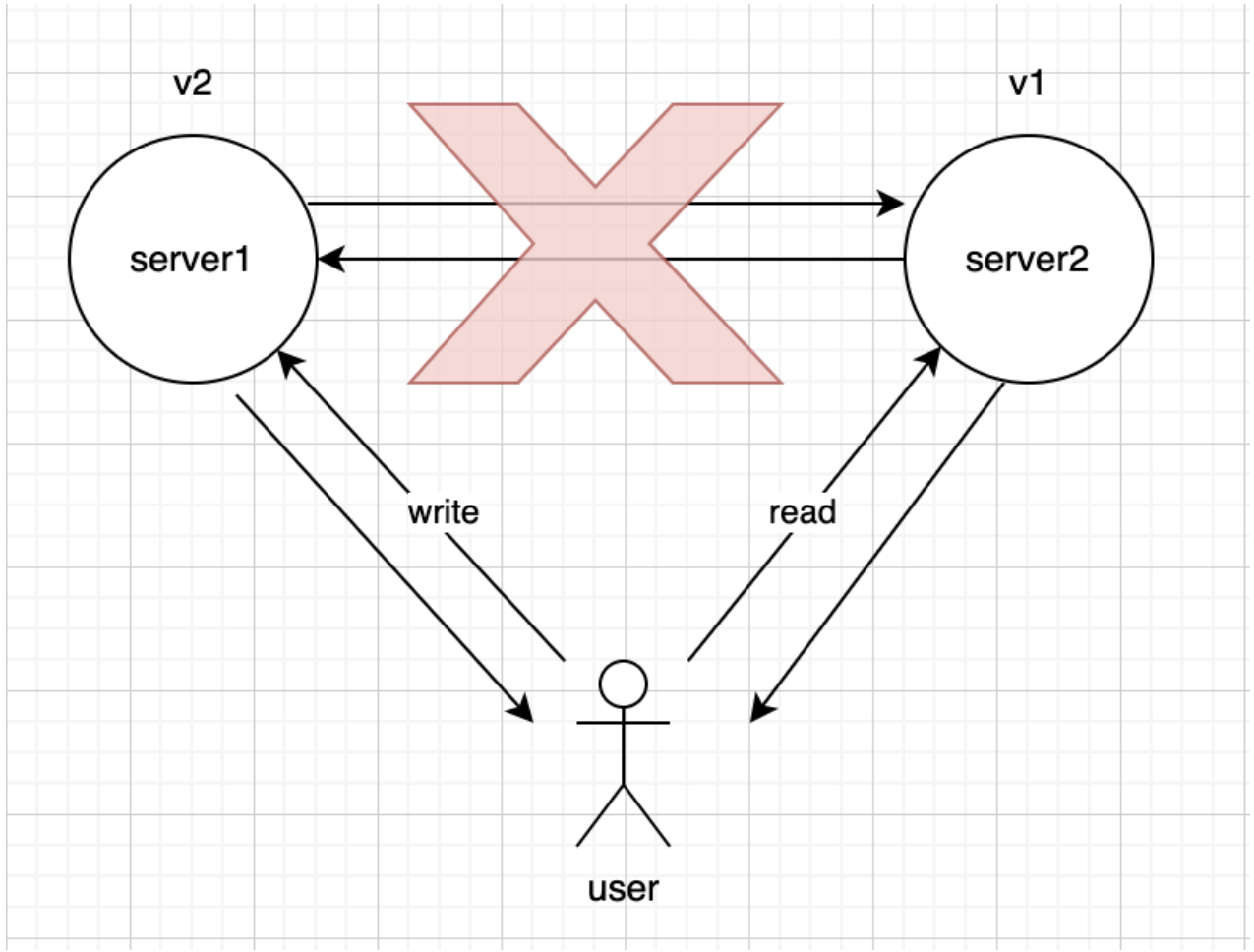
CAP-теорема. Согласованность



CAP-теорема. Доступность



CAP-теорема. Устойчивость к распределению



БД на основе пар «ключ-значение»

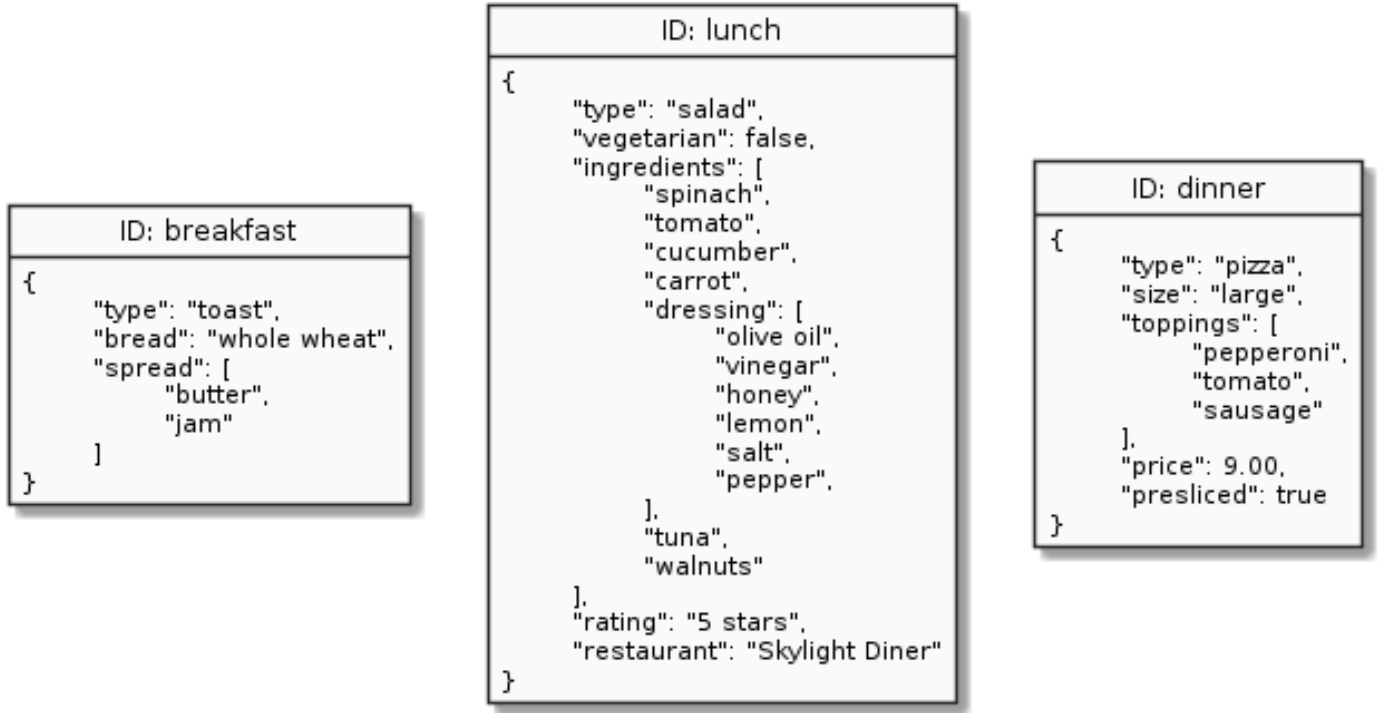
Базы данных с использованием пар «ключ-значение» поддерживают высокую разделяемость и обеспечивают беспрецедентное горизонтальное масштабирование, недостижимое при использовании других типов БД. Хорошими примерами использования для баз данных типа «ключ-значение» являются игровые, рекламные приложения и приложения IoT. Amazon DynamoDB обеспечивает стабильную работу БД с задержкой не более нескольких миллисекунд при любом масштабе.

key:	value
user_id:	f5badc33-5bd7-4b65-a737-b5304675f476
color:	blue
repetitions:	3
text:	hello world
data:	{ ... }

redis

Документная модель

В коде приложения данные часто представлены как объект или документ в формате, подобном JSON, в значении могут храниться только структурированные документы, поскольку для разработчиков это эффективная и интуитивная модель данных. Документные базы данных позволяют разработчикам хранить и запрашивать данные в БД с помощью той же документной модели, которую они используют в коде приложения. Гибкий, полуструктурированный, иерархический характер документов и документных баз данных позволяет им развиваться в соответствии с потребностями приложений. Информацию можно найти по конкретному ключу, а также выделить её из документа. Документная модель хорошо работает в каталогах, пользовательских профилях и системах управления контентом, где каждый документ уникален и изменяется со временем.

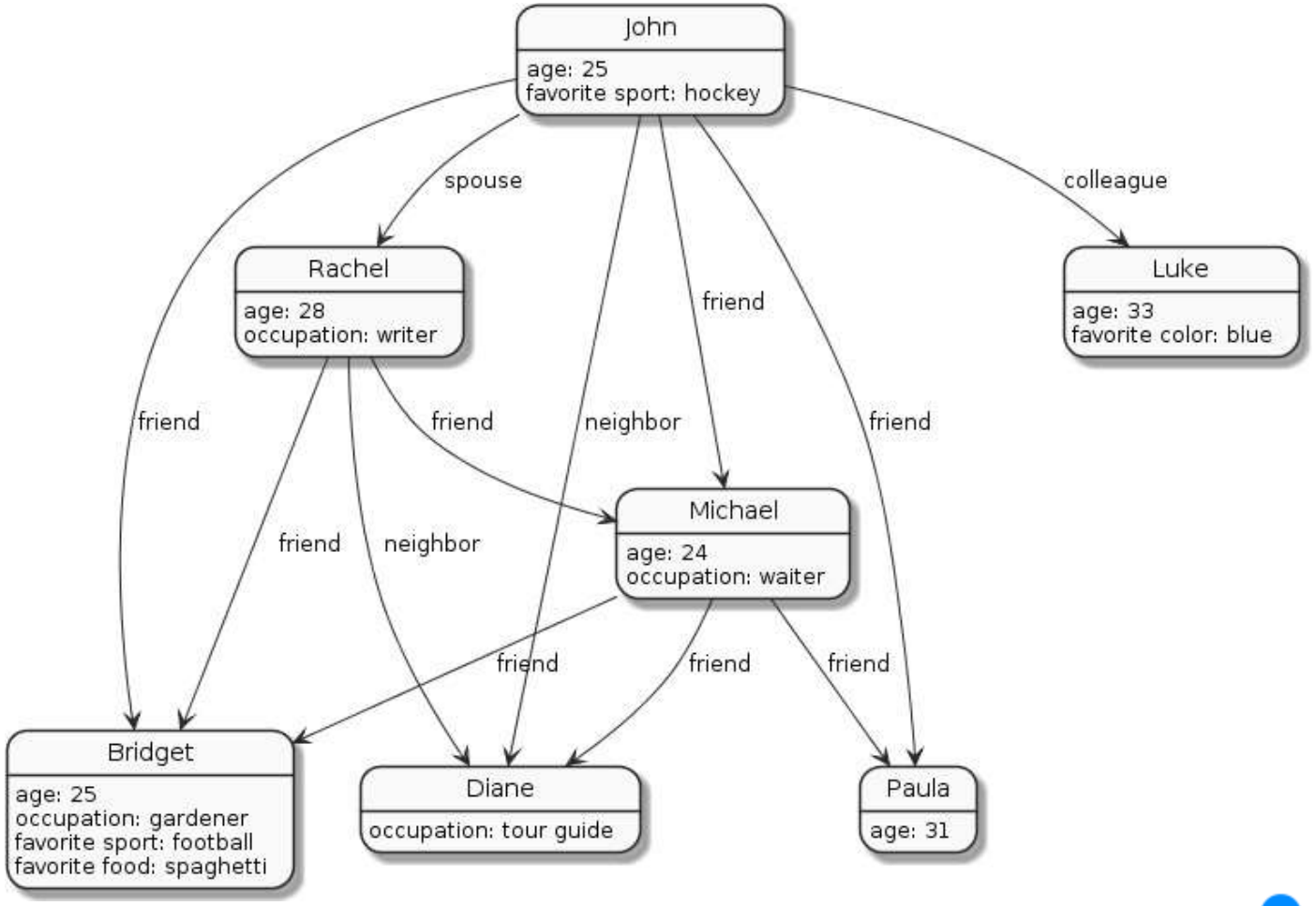


mongoDB®

Графовые БД

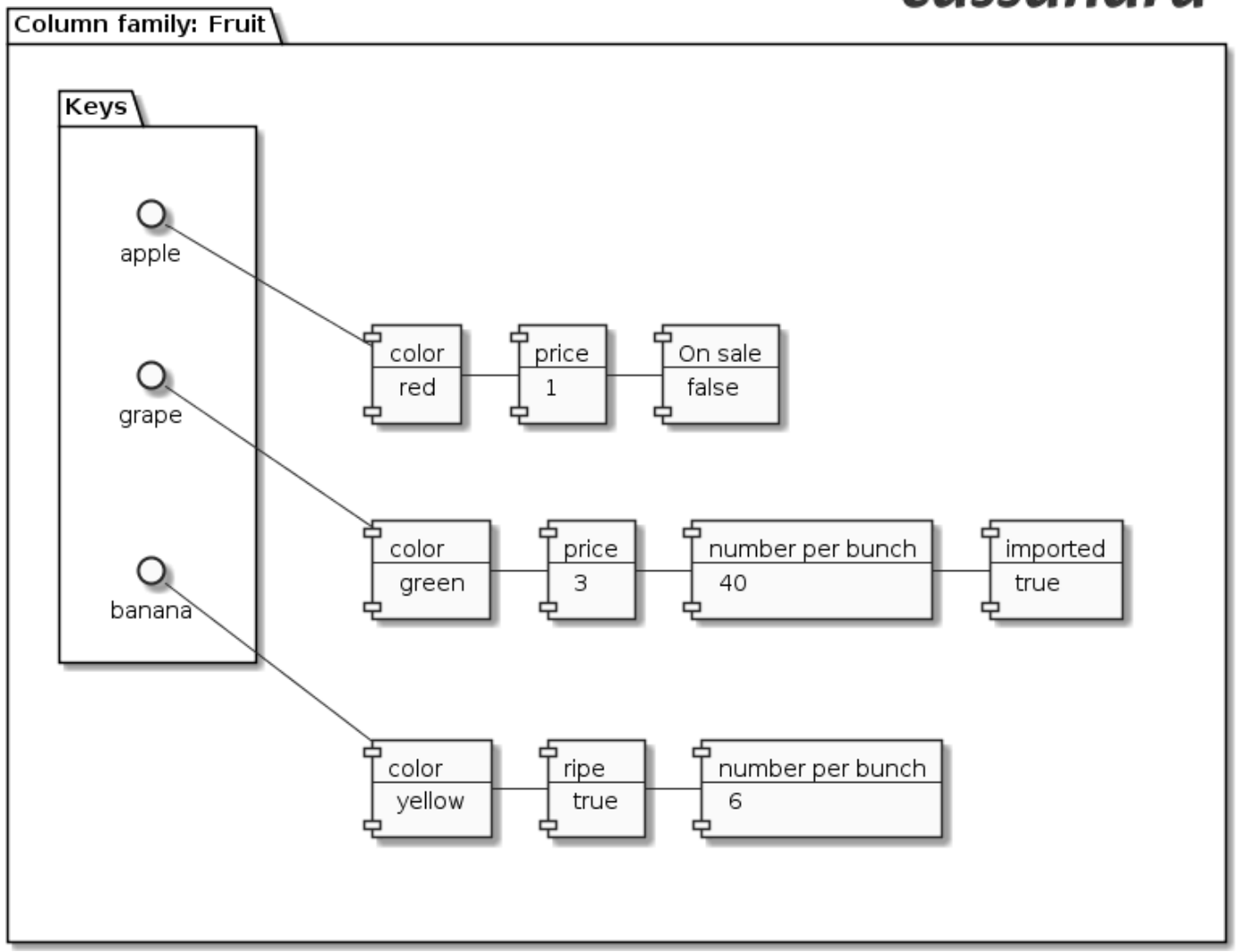
Вместо сопоставления связей с таблицами и внешними ключами, графовые базы данных устанавливают связи, используя узлы, рёбра и свойства. Графовые базы представляют данные в виде отдельных узлов, которые могут иметь любое количество связанных с ними свойств.

Графовые базы данных упрощают разработку и запуск приложений, работающих с наборами сложносвязанных данных. Типичные примеры использования графовых баз данных – социальные сети, сервисы рекомендаций, системы выявления мошенничества и графы знаний.



Колоночные БД

Колоночные базы данных (также нереляционные колоночные хранилища или базы данных с широкими столбцами) принадлежат к семейству NoSQL БД, но внешне похожи на реляционные БД. Как и реляционные, колоночные БД хранят данные, используя строки и столбцы, но с иной связью между элементами.
В реляционных БД все строки должны соответствовать фиксированной схеме. Схема определяет, какие столбцы будут в таблице, типы данных и другие критерии. В колоночных базах вместо таблиц имеются структуры – «колоночные семейства». Семейства содержат строки, каждая из которых определяет собственный формат. Строка состоит из уникального идентификатора, используемого для поиска, за которым следуют наборы имён и значений столбцов.



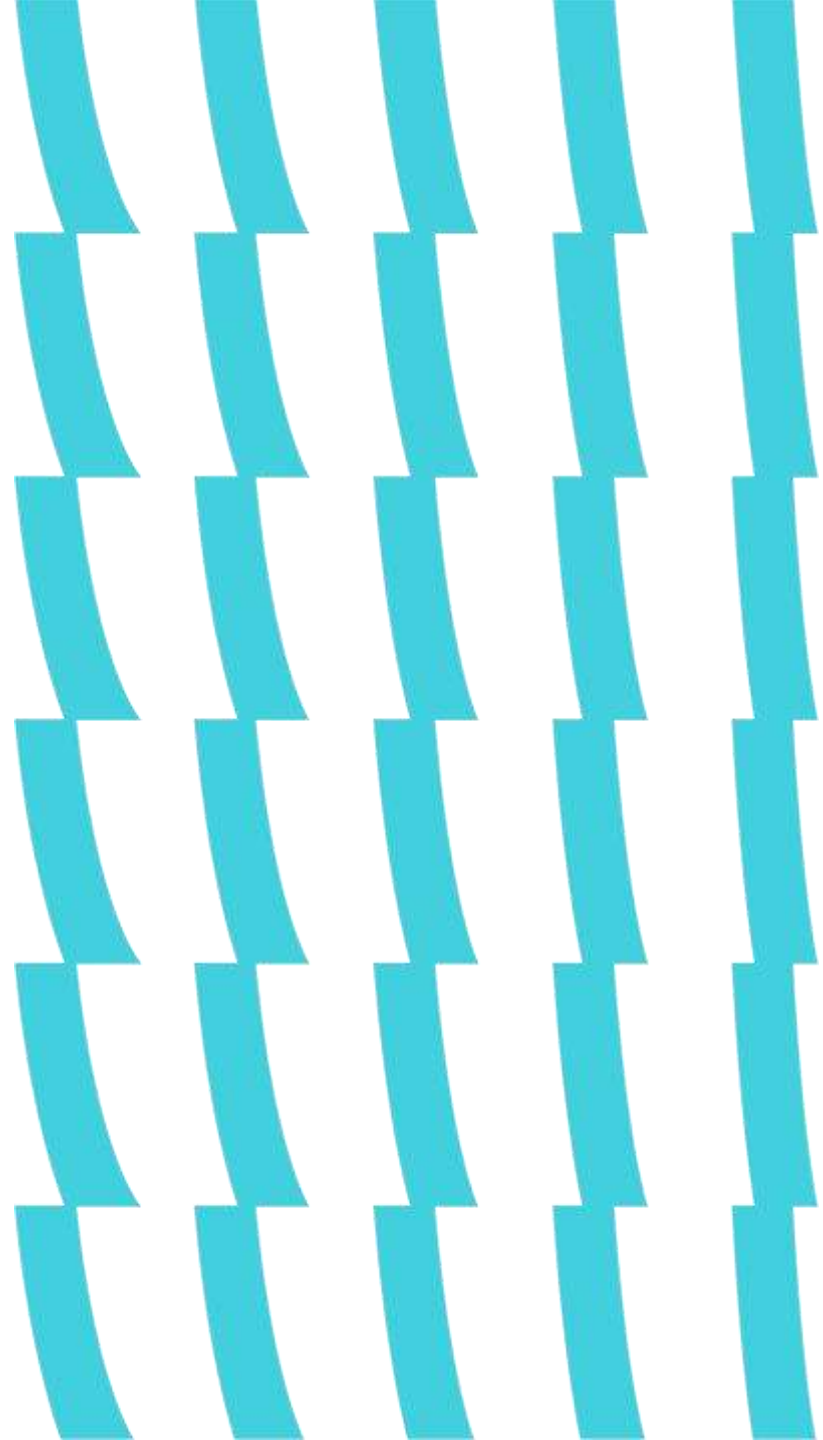
NoSQL. Плюсы и минусы

Плюсы:

- Масштабируются лучше, чем традиционные системы, когда требуется динамическое поведение.
- Эти системы лучше оптимизированы для нереляционных данных.

Минусы:

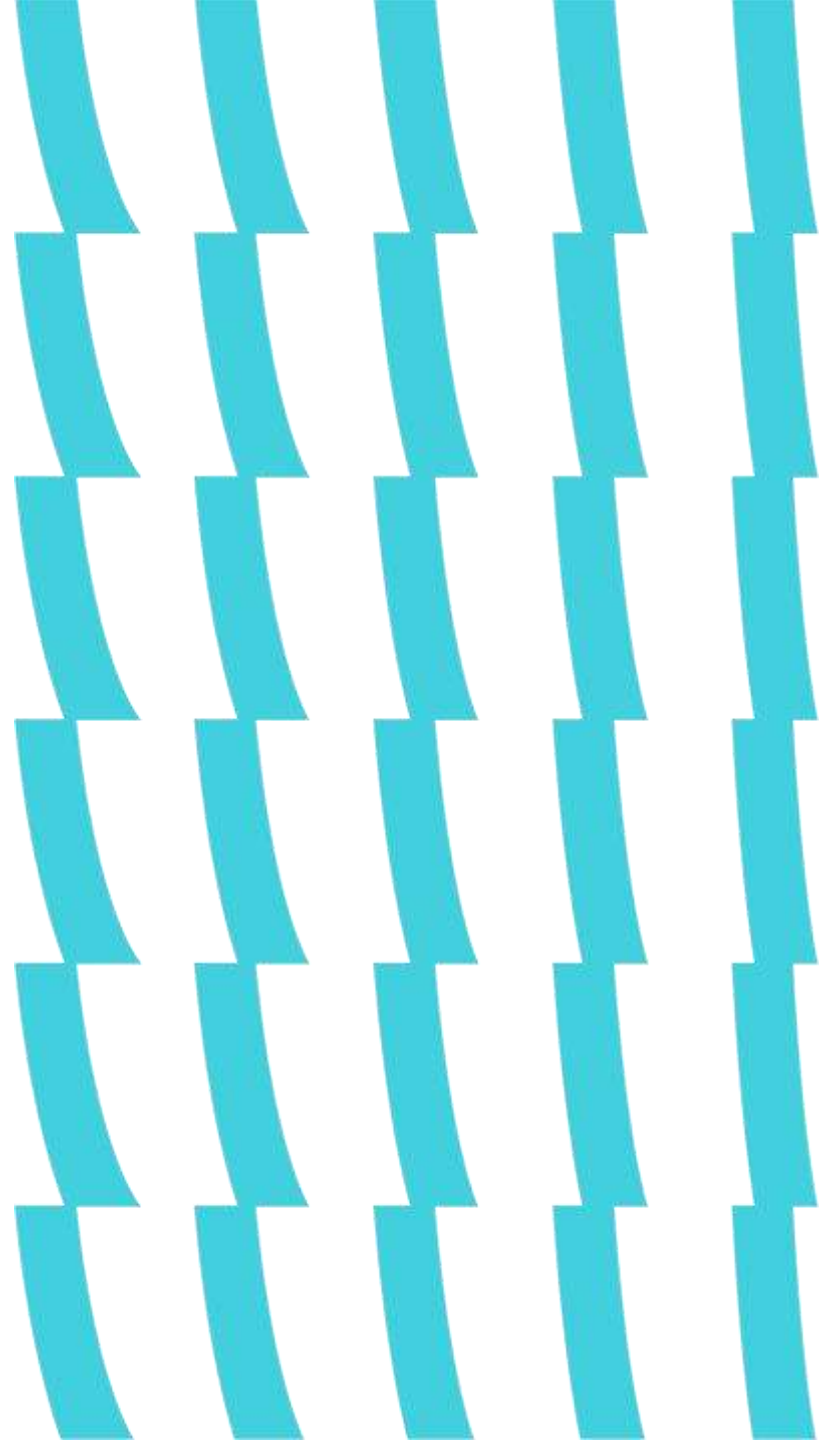
- Система, построенная с помощью NoSQL, принципиально не транзакционная.
- Невозможность обеспечить операции ACID. Это не соответствует согласованности, когда несколько транзакций выполняются одновременно.
- Не имеют таких функций, как сильно структурированные запросы
- Разные БД имеют свои собственные языки запросов, затрудняет стандартизацию интерфейсов прикладных программ



NewSQL — переосмысление

Майкл Стоунбрейкер с коллегами из MIT публикуют работу «Конец архитектурной эпохи, или Наступило время полностью переписывать системы управления данными» (2007)

Авторы говорят о причинах и несостоятельности подхода «универсальности» («one size fits all») классических реляционных СУБД, так как они пытаются продать единый инструмент для всего спектра задач по работе с данными и он становится неконкурентоспособным со специализированными инструментами.
Взамен авторы предлагают новую экспериментальную СУБД для приложений онлайн-обработки транзакций H-Store



NewSQL

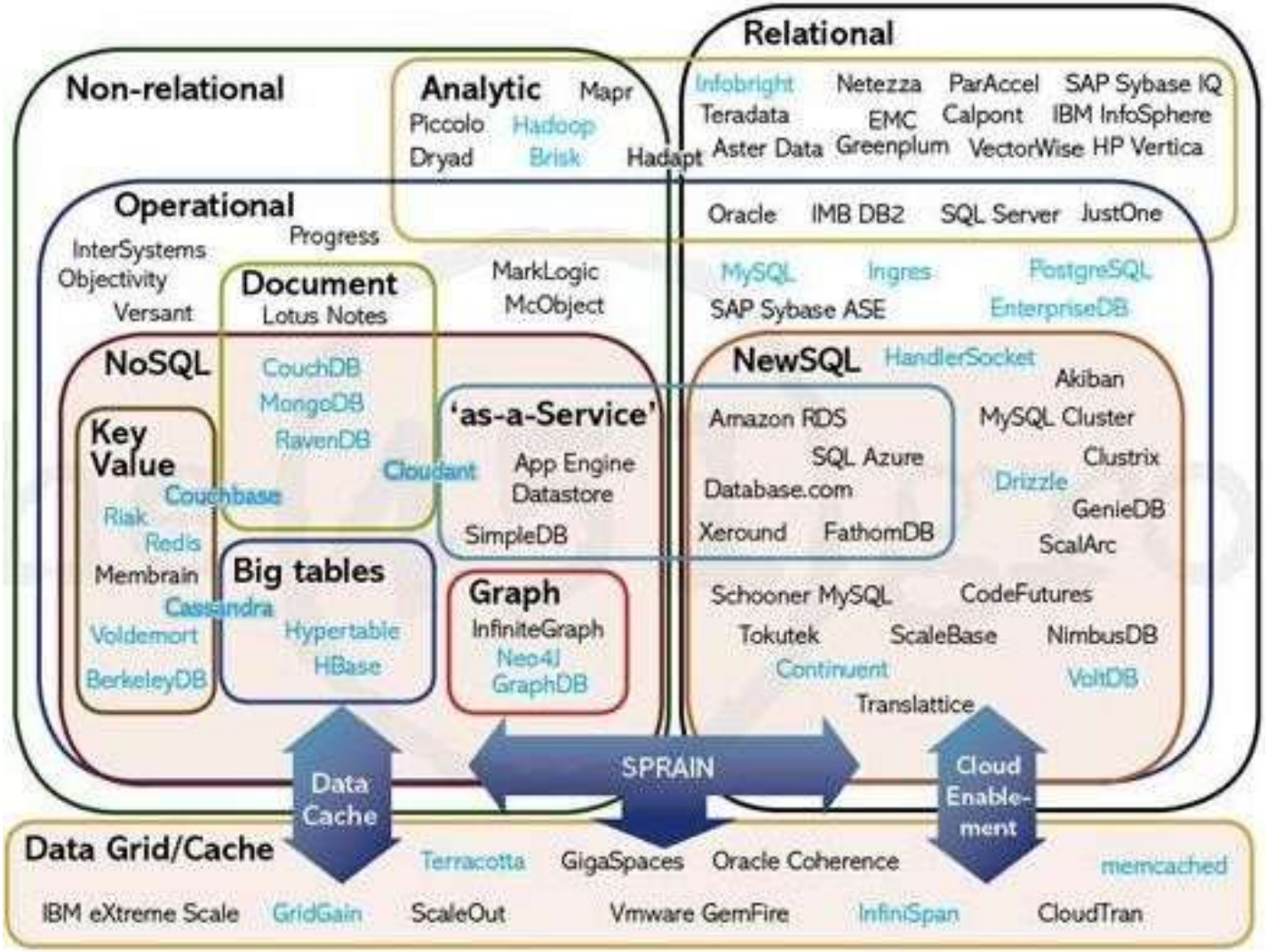
Концепция NewSQL сочетает в себе масштабируемость NoSQL подхода и полную поддержку ACID-транзакционности классических реляционных баз.

Потребность в данных системах возникла в первую очередь у компаний, работающих с критическими данными, например банковская сфера, которым были необходимы масштабируемые решения, но использование NoSQL не могло предоставить гарантий выполнения транзакции и не отвечали требованиям надежности данных.

NewSQL

NewSQL система разрабатывается полностью с нуля с целью достижения масштабируемости и производительности. Подход NewSQL основан на ряде инновационных архитектурных решений. Одним из ключевых факторов в повышении производительности является использование оперативной памяти или новых видов дисков (флэш-память/SSD), которые являются хранилищем первичных данных. Новый метод устраняет потребность в краткосрочных блокировках данных, поскольку система исполняется на сервере строго в виде одиночного потока (хотя иные типы блокировки все же будут создавать определенную нагрузку). А «дорогостоящие» операции восстановления исключаются за счет применения дополнительных серверов для тиражирования и переключения нагрузки при отказе.
Данное решение может осуществляться программно (VoltDB, NuoDB) либо на уровне железа (Clustrix, Inc., TransLattice).

NoSQL vs. NewSQL



SQL vs. NoSQL vs. NewSQL

Характеристики СУБД	Олдскул SQL	NoSQL	NewSQL
Реляционная	Да	Нет	Да
Поддерживает SQL	Да	Нет	Да
Распределенные	Нет	Да	Да
ACID транзакции	Да	Нет (AP по CAP)	Да (CP по CAP)

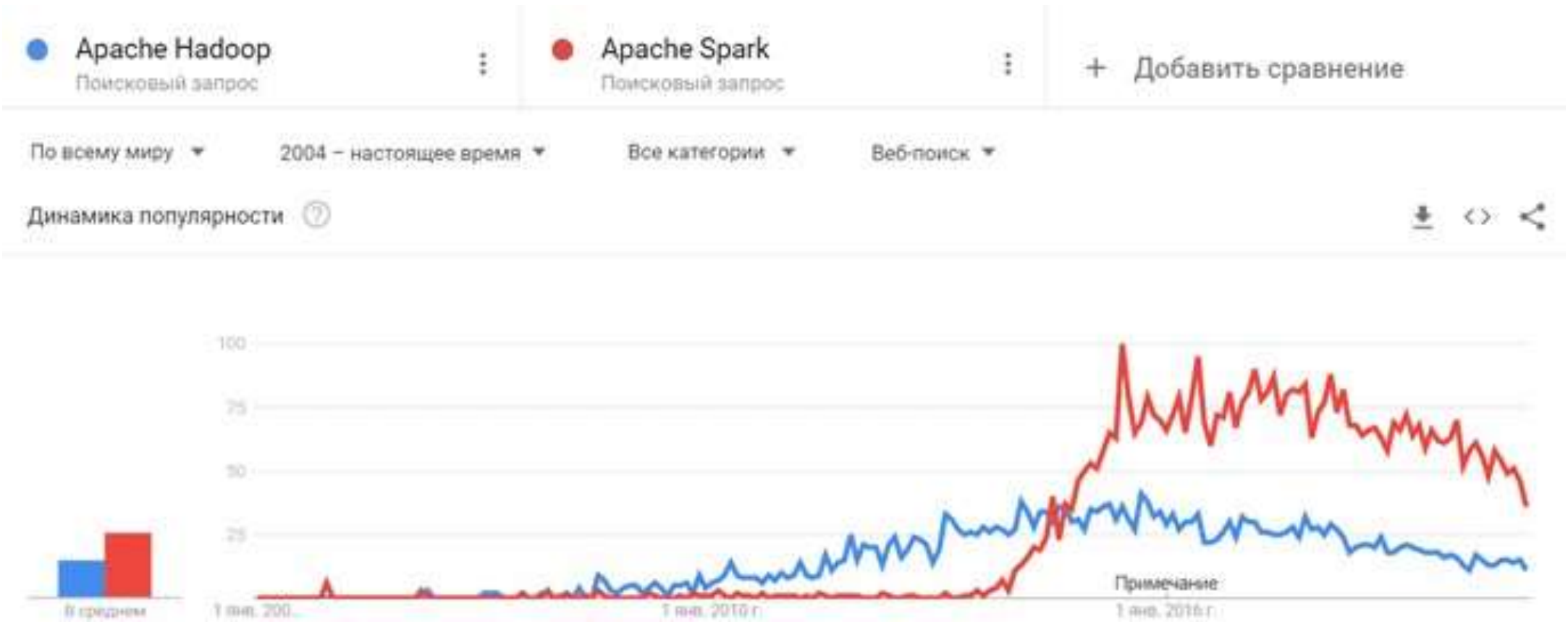
Hadoop и Big Data

Идея родилась в 2004 году: Google публикует работу, в которой рассказывает о технологии BigTable и MapReduce.

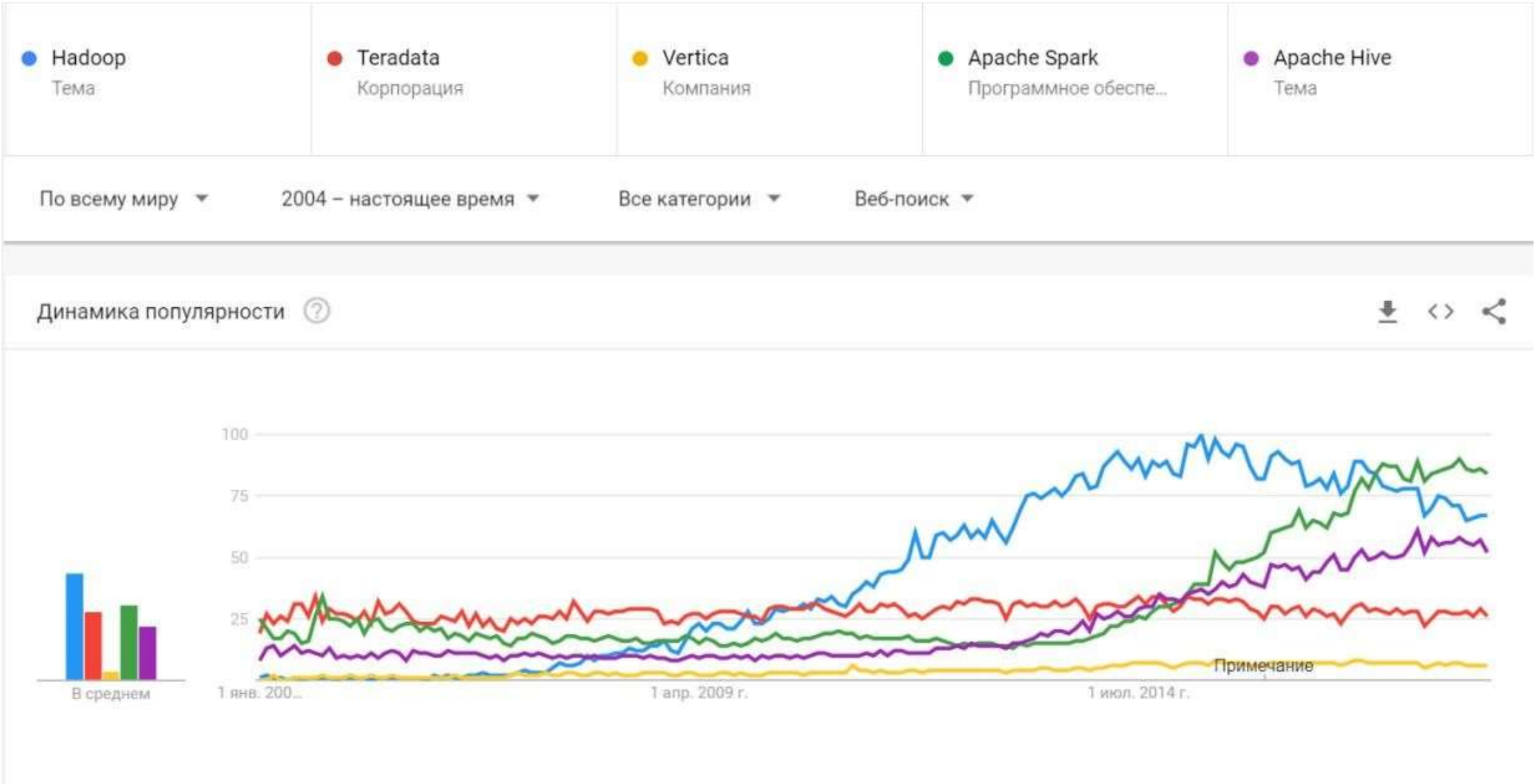
В 2006 Google публикует описание своих технологий — что приводит к созданию open-source проекта Apache Hadoop. Yahoo выпускает открытую реализацию и дает ей имя Hadoop. Появление в 2007 году технологии Hadoop в открытом доступе ознаменовал выход больших данных «в массы».

Открытая система, есть возможность развернуть кластер на любом железе, легко масштабируется горизонтально.

Hadoop vs. Big Data



Hadoop vs. Teradata

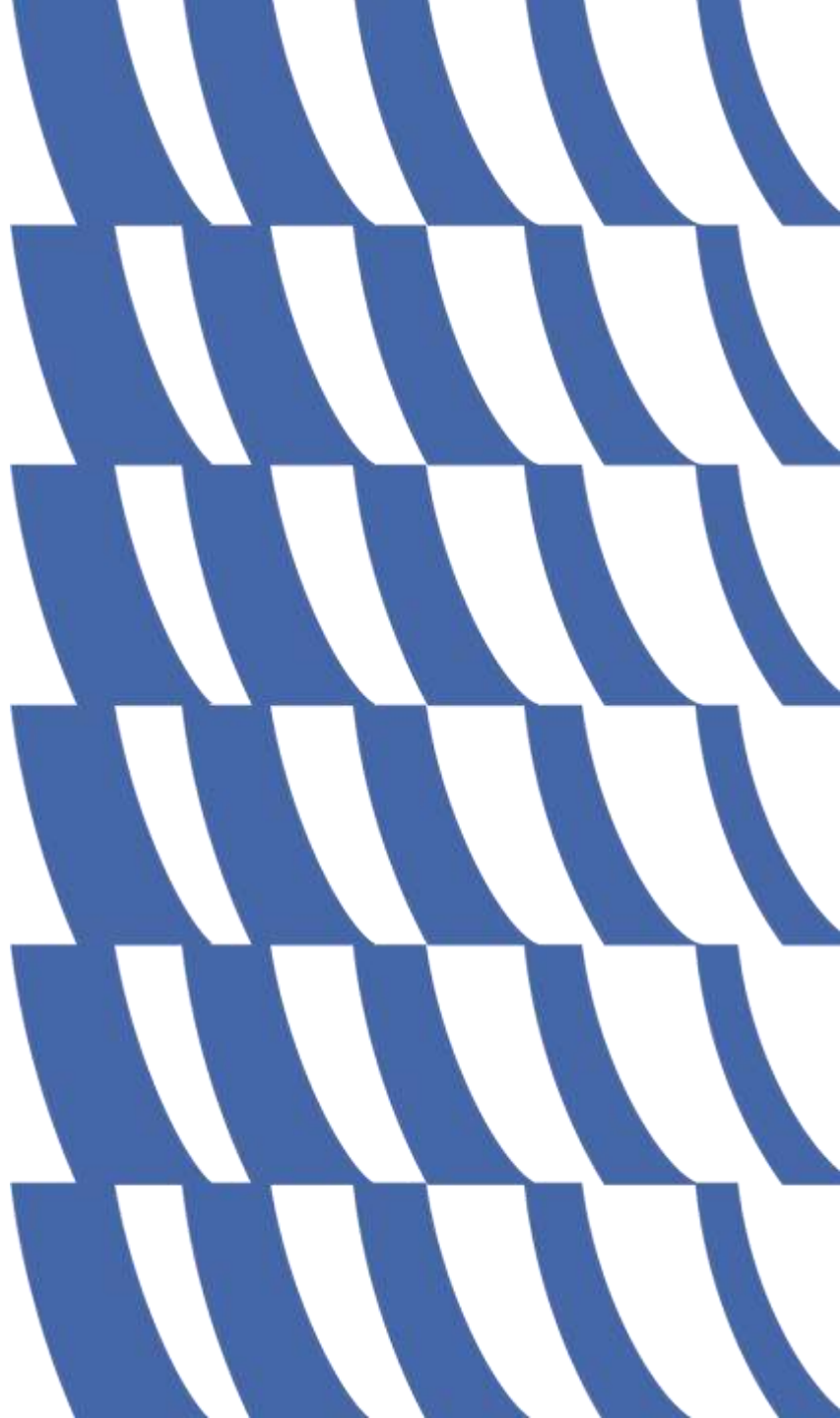


Большие данные

Big Data

Сам термин Big Data впервые был озвучен в 2008 году на страницах спецвыпуска журнала Nature в статье главного редактора Клиффорда Линча. Этот номер издания был посвящен взрывному росту глобальных объёмов данных и их роли в науке.

В русскоязычной среде дополнительно используется понятие «большие данные». Большие данные не имеют строгого определения. Нельзя провести четкую границу — это 10 терабайт или 10 мегабайт? Само название подразумевает неточность и субъективность.



Применение больших данных

медицина

Обработка генетических данных

Анализ анамнеза пациента

Моделирование лекарственных препаратов

Безопасность

Анализ данных фотовидеофиксации

Анализ транзакционных данных

Открытых источников

Транспорт

Пассажиропоток

Треккинг корреспонденции

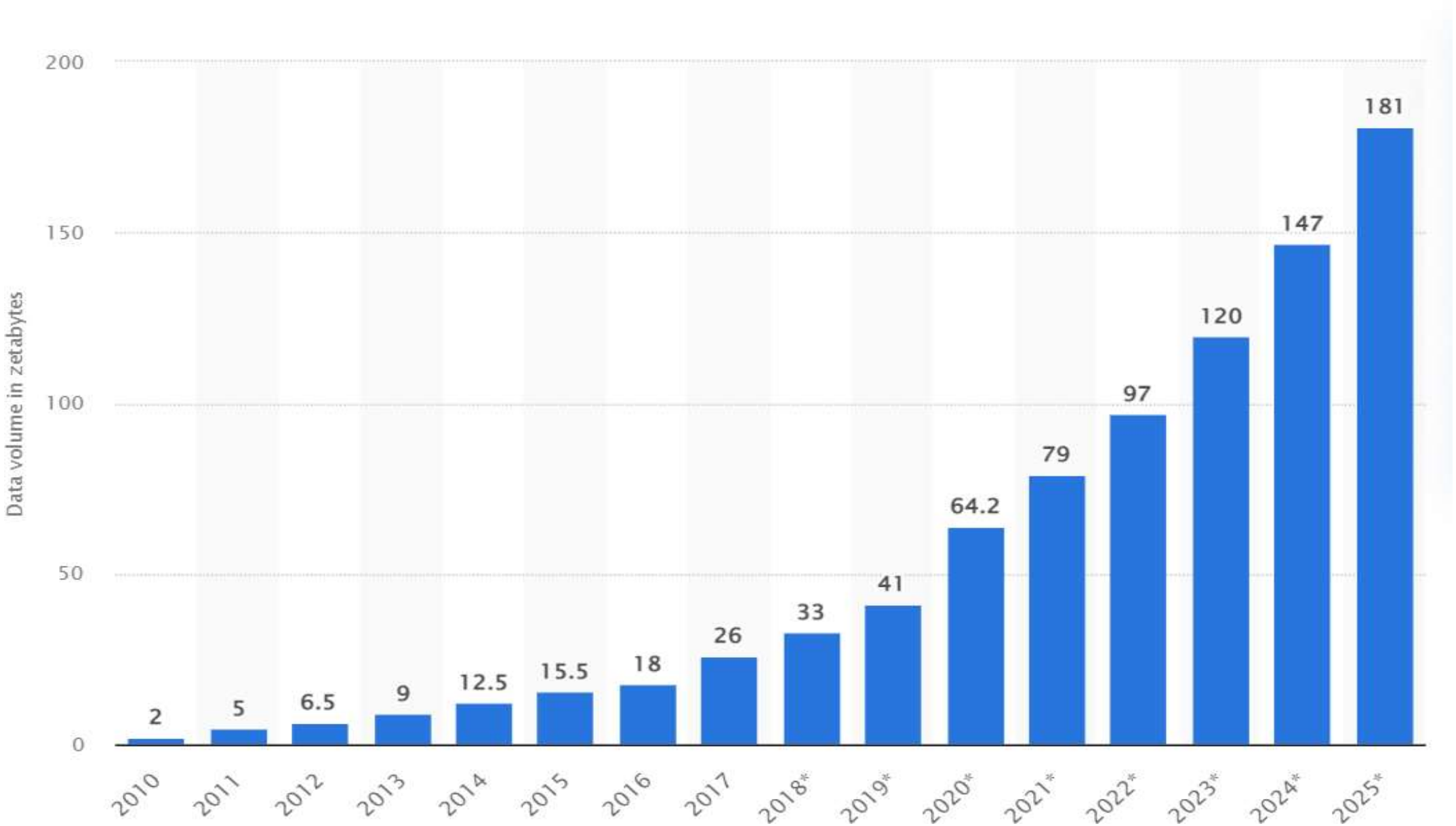
Состояние инфраструктуры

Веб-сайты

Мобильные приложения

Источники больших данных

Информационный взрыв



1

Объём (volume, в смысле величины физического объёма)

2

Скорость (velocity в смыслах как скорости прироста, так и необходимости высокоскоростной обработки и получения результатов)

3

Многообразие (variety, в смысле возможности одновременной обработки различных типов структурированных и полуструктурированных данных)

5V

Позже добавили следующие пункты

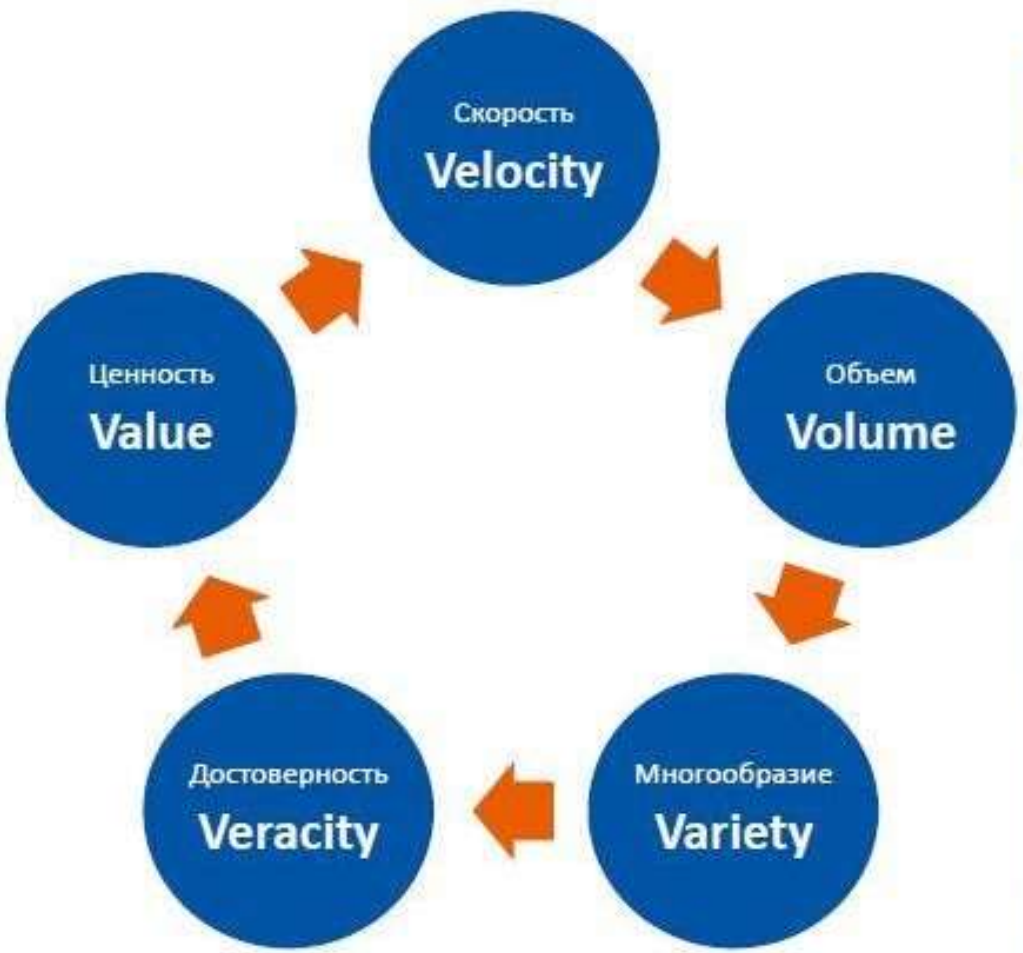
4

veracity — Достоверность. Целостность данных и возможность доверять полученным на их основе результатам.

5

value — Ценность / экономическая целесообразность обработки соответствующие объёмов в конкретных условиях.

Определение Больших данных



Характеристика	Традиционная база данных	База Больших Данных
Объем информации	От гигабайт до терабайт	От петабайт до эксабайт
Способ хранения	Централизованный	Децентрализованный
Структурированность данных	Структурирована	Полуструктурирована или неструктурирована
Модель хранения и обработки данных	Вертикальная модель	Горизонтальная модель
Взаимосвязь данных	Сильная	Слабая

Системы обработки данных

OLTP vs. OLAP

OLTP (Online Transaction Processing) обработка транзакций в режиме реального времени) используется для операционной работы конкретной системы. OLTP-система характеризуется большим потоком коротких транзакций (INSERT, UPDATE, DELETE). Ключевые особенности OLTP-систем: почти мгновенная обработка запросов, обеспечение целостности данных в средах с множественным доступом и высокая нагруженность (определяется числом транзакций в единицу времени)

OLAP (Online Analytical Processing) интерактивная аналитическая обработка имеет дело с историческими (архивными) данными. Такая система характеризуется относительно низкой частотой транзакций, но большими объёмами затрагиваемых данных. Запросы часто очень сложны и включают агрегирование (группировки). Для OLAP-систем показатель эффективности — это время отклика, а данные в OLAP-базе разложены в многомерную схему под конкретный сценарий работы с ними (например, “звезда”)

OLTP vs. OLAP

Характеристика	Требования к OLTP	Требования к OLAP
Степень детализации хранимых данных	Хранение детализированных данных	Хранение детализированных и обобщенных данных
Качество данных	Допускаются неверные данные из-за ошибок ввода	Не допускаются ошибки в данных
Актуальность данных	Доступны актуальные данные	Данные появляются с задержкой
Характер запросов к данным	Доступ к данным осуществляется по заранее составленным запросам	Запросы к данным могут быть произвольны и заранее не оформлены
Управление данными	Возможность в любое время добавлять, удалять и изменять	Возможность периодически добавлять данные
Характер SQL инструкций	Много коротких DML операций	Простые и сложные SELECT-ы

OLTP vs. OLAP

Характеристика	Требования к OLTP	Требования к OLAP
Приоритетность характеристик системы	Высокая производительность и доступность	Обеспечение гибкости системы и независимости работы пользователей
Нормализация данных	Данные нормализованы	Данные для удобства денормализуются
Полнота данных	Должны быть доступны все оперативные данные, требующиеся в данный момент	Должны быть доступны все данные, накопленные в течение продолжительного времени

OLTP vs. OLAP

Необходимость в использовании систем категории OLAP появляется как раз тогда, когда в компании используется несколько OLTP систем или несколько разных способов генерации новых данных (т.е. разных источников новых данных), которые необходимо впоследствии централизованно анализировать.
Технологически правильно для системы OLAP поддерживать отдельную базу данных (DataWarehouse).
Технологически правильно из-за больших объёмов данных, так как для точной аналитики нужна история, из которой можно выводить прогнозы.

Напрямую с источником данных не стоит работать, так как аналитические запросы к данным увеличивают нагрузку на диски, увеличивают время ответа рабочих машин. Не всегда стоит работать с некоторыми чувствительными данными, при загрузки в хранилище персональную информацию можно опускать или шифровать.

Особенности OLTP

- Нормализованные данные;
- Высокая интенсивность добавления и изменения данных;
- Большое количество одновременно активных пользователей;
- Внесение данных и расчеты осуществляют пользователи системы;
- Содержат актуальные данные.

Особенности DWH

- В DWH хранятся консолидированные данные из разнородных внешних источников;
- Данные вручную не вводятся, все данные поступают из внешних источников;
- Источниками данных для DWH могут быть:
  - OLTP системы компании
  - Excel файлы
  - Файлы в другом формате
  - Другие базы данных
- В DWH хранятся полные исторические данные;
- Данные в хранилище появляются с задержкой (например, в 1 день, т.к. закачка данных обычно происходит по ночам).

DWH

Возникновение термина DWH

Две фамилии, которые надо запомнить.

Билл Инмон (Bill Inmon) — отец понятия хранилища данных. Впервые сформулировал его в 1992 году в своей книге «Building the Data Warehouse».

Ральф Кимбалл (Ralph Kimball) — был главным оппонентом Инмона на заре становления традиционных подходов к построению ХД. Сформулировал основные требования к хранилищам данных:

- Полнота — нет потерь данных при наполнении хранилища;
- Консистентность — внутренняя непротиворечивость данных;
- Историчность — доступность временных срезов (point-in-time);
- Производительность — быстрый доступ к данным;
- Доступность — пользователям удобно обращаться к данным;

DWH

1. Предметная ориентированность
Данные хранятся в соответствии с описываемыми бизнес-процессами, а не приложениями. Предметная область важнее технических процессов.

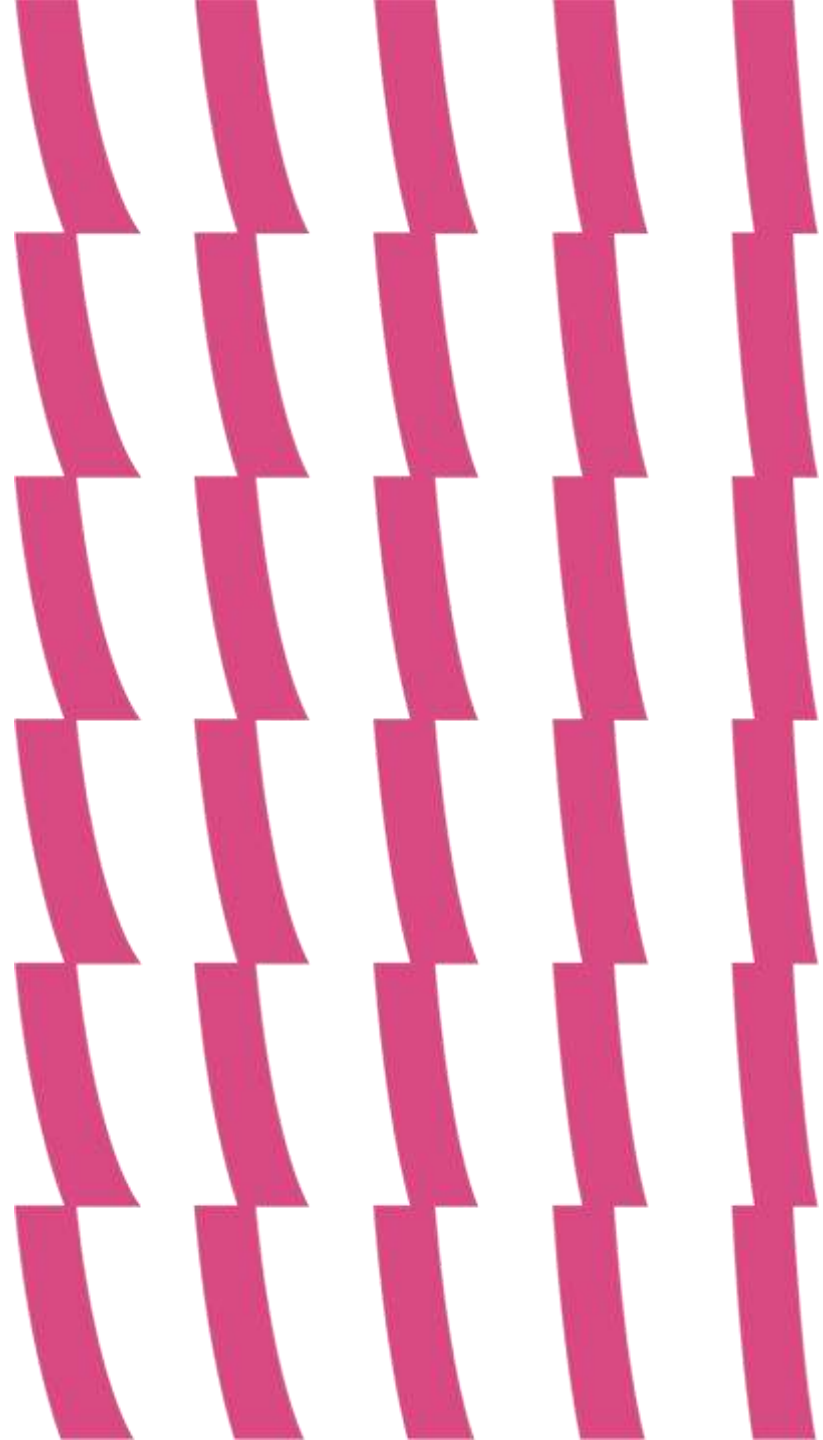
2. Интегрированность
Единая информационная среда Хранилища включает в себя данные из различных систем, удовлетворяя потребностям всего бизнеса в целом.

3. Временная привязка
Данные в Хранилище имеют смысл только при привязке их к некоторому временному интервалу. Такие базы данных называют темпоральными.

4. Некорректируемость
Данные, связанные с бизнес-процессами, не создаются в Хранилище, а поступают из внешних источников, не корректируются и не удаляются. Проблемы с качеством исходных данных решаются уже внутри хранилища.

Классическое определение DWH (Data Warehouse)

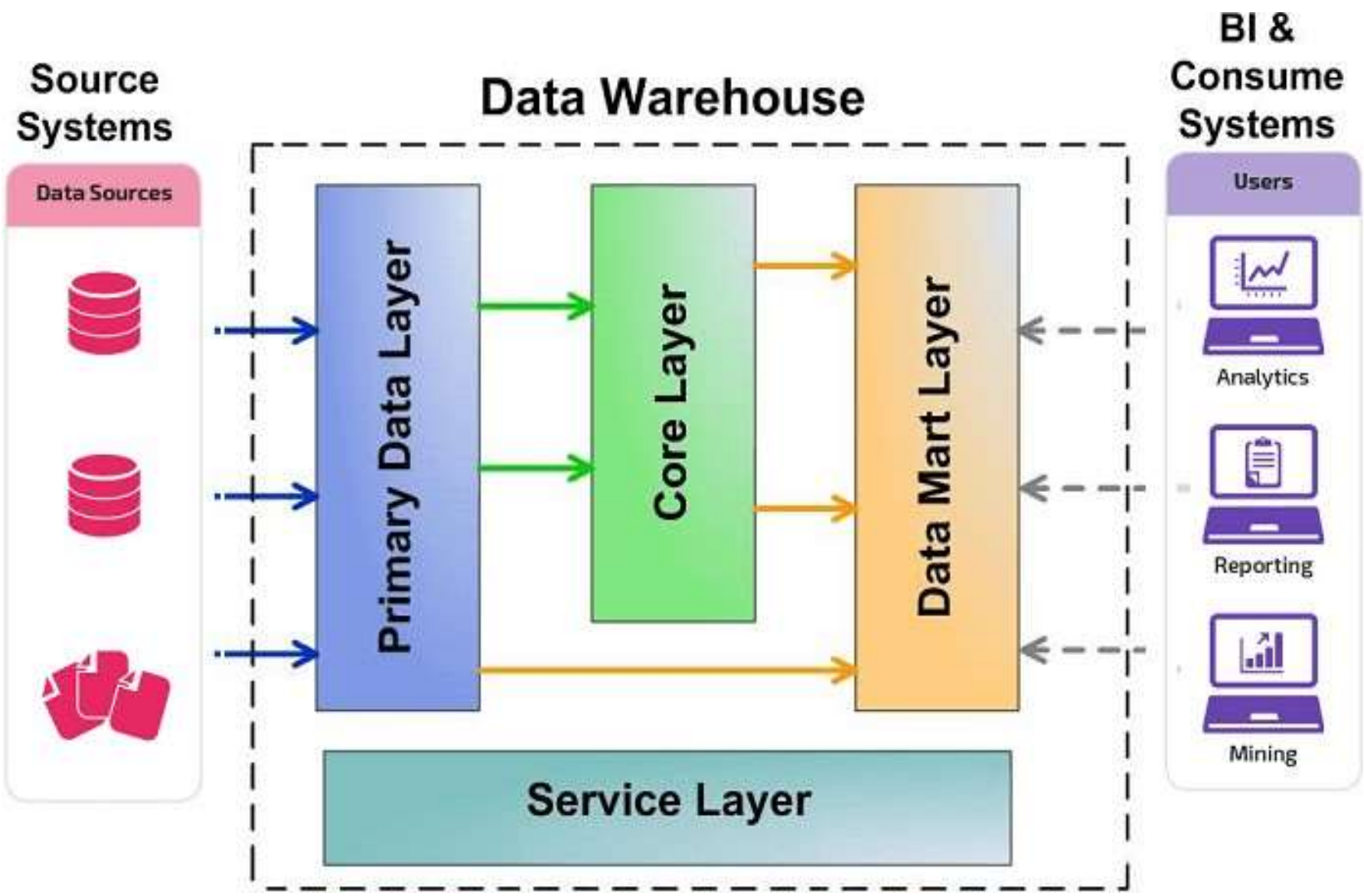
Хранилище данных — предметно-ориентированный, интегрированный, некорректируемый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.



Слои данных

Слои данных в DWH

Уровневая архитектура – это средство борьбы со сложностью системы. Каждый последующий уровень абстрагирован от сложностей внутренней реализации предыдущего.



Стейджинговый слой

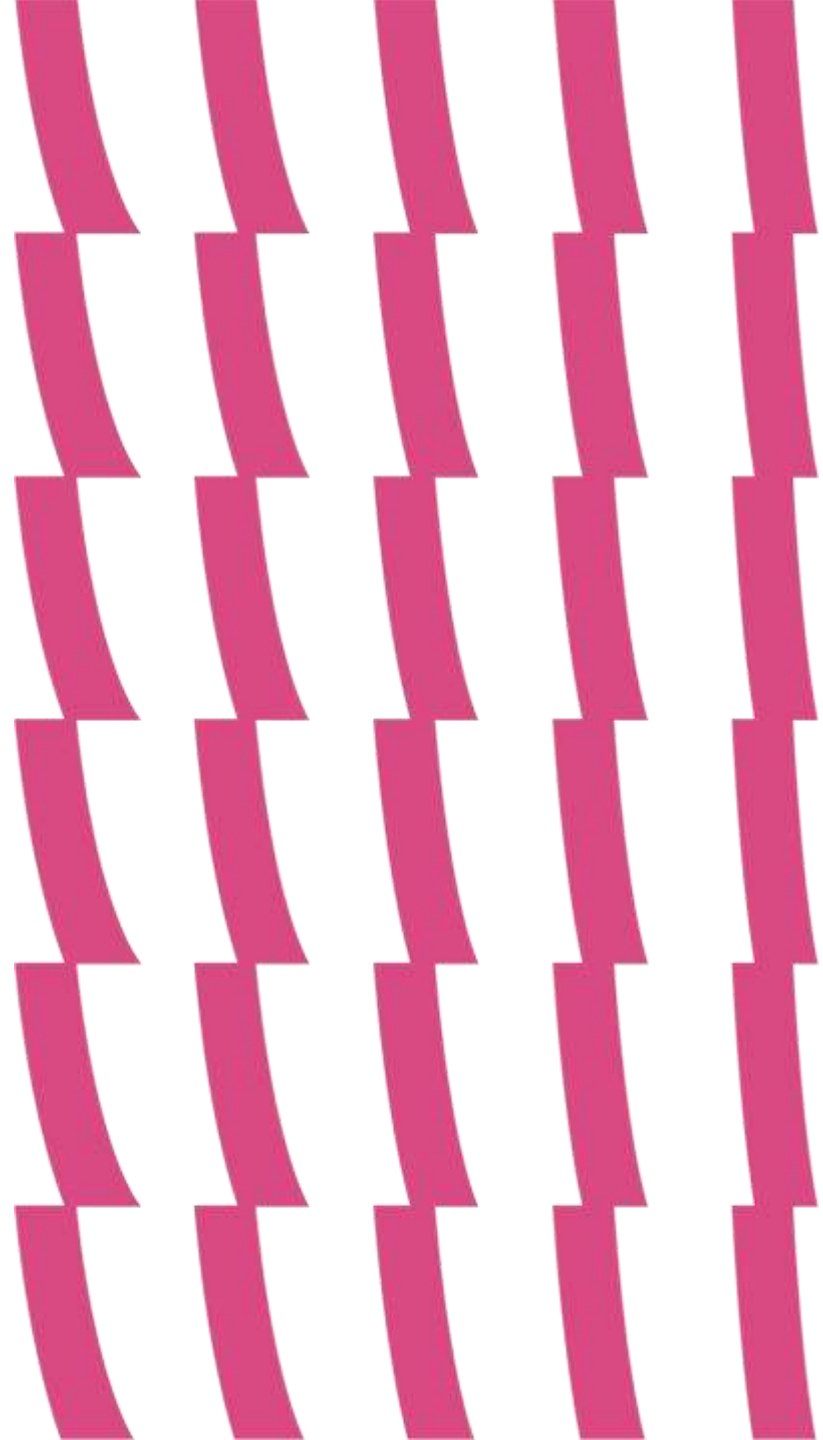
Стейджинговый слой (Primary Data Layer)

- Предназначен для загрузки данных из систем-источников и сохранения первичной информации, без трансформаций
- Абстрагирует последующие слои хранилища от физического устройства источников данных, способов забора данных и методов выделения дельты изменений.

Данные в этом слое хранятся в структурах, максимально близких к системе источнику (чтобы сохранить первичные данные как можно ближе к их первозданному виду.

Что даёт наличие «историзируемого стейджинга»?

- возможность ошибиться при разработке детального слоя;
- возможность подумать и не торопиться с проработкой большого фрагмента детального слоя именно в этой итерации развития хранилища;
- возможность анализа – мы сохраним даже те данные, которых уже нет в источнике;
- возможность информационного аудита – благодаря максимально подробной первичной информации мы сможем потом разобраться, что происходило в хранилище (и не только в нем)



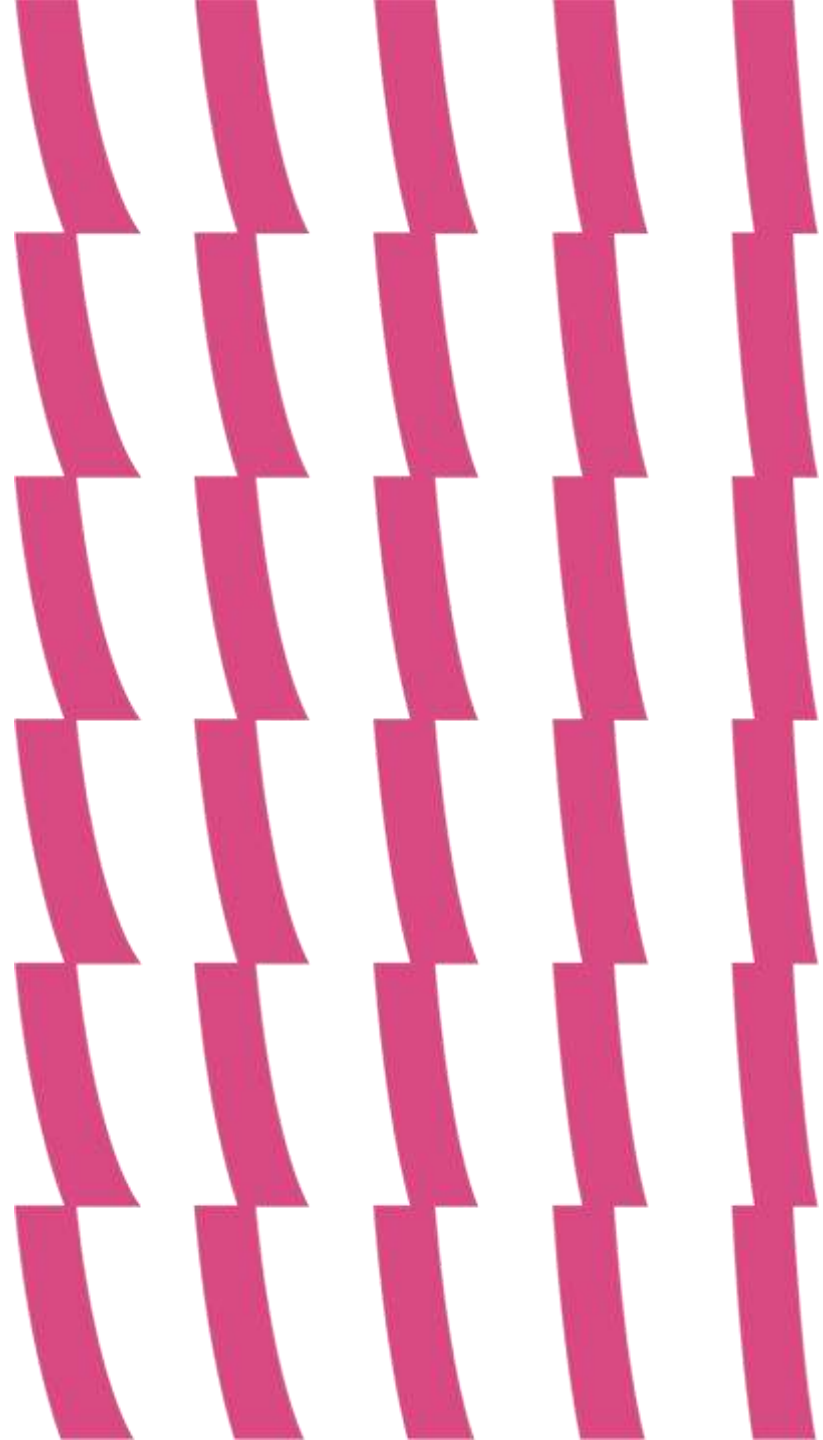
Особенности стейджингового слоя

Возможные сложности при построении:

- Целостность слоя труднодостижима
- Содержит большие объемы данных

Особенности:

- Стейджинговый слой устроен по принципу «конвейера»: взял данные из источника – положил в свой слой – готов брать следующую порцию.
- Процессы загрузки устроены очень просто => их можно очень хорошо оптимизировать и параметризовать, снижая нагрузку на нашу систему и ускоряя процесс подключения источников (время разработки).

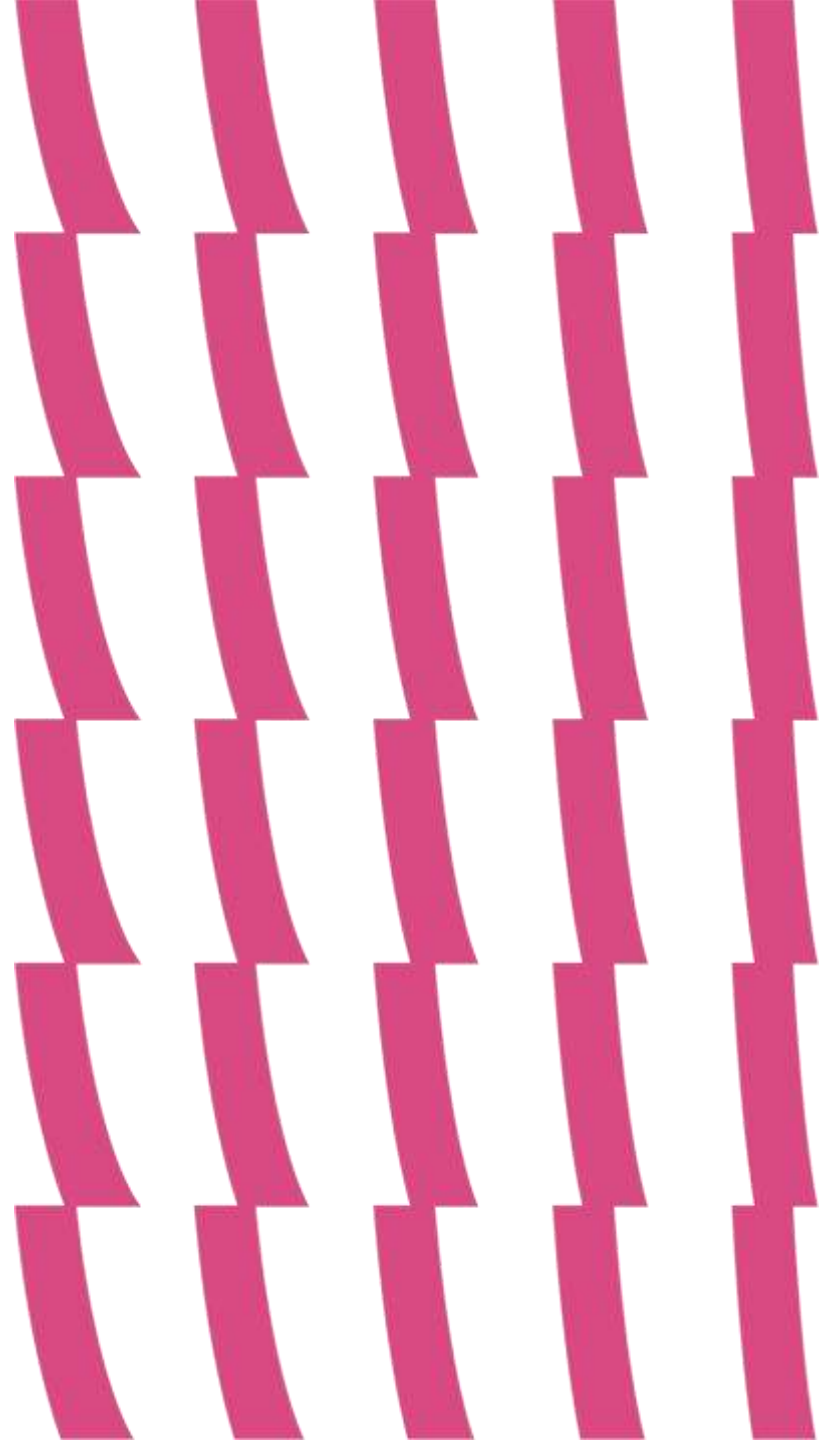


Детальный слой

Детальный слой (Core Data Layer, Detail Data Store) - центральный компонент системы, который отличает хранилище от «большой свалки данных». Абстрагирует дальнейшие слои от:

- Особенностей логического устройства источников данных
- Необходимости консолидировать данные из различных систем
- Необходимости обеспечивать целостность и качество данных
- Необходимости систематизации данных, приведения их к единым структурам, ключам

Здесь осуществляется основная работа с качеством данных – очистка, трансформация, унификация.

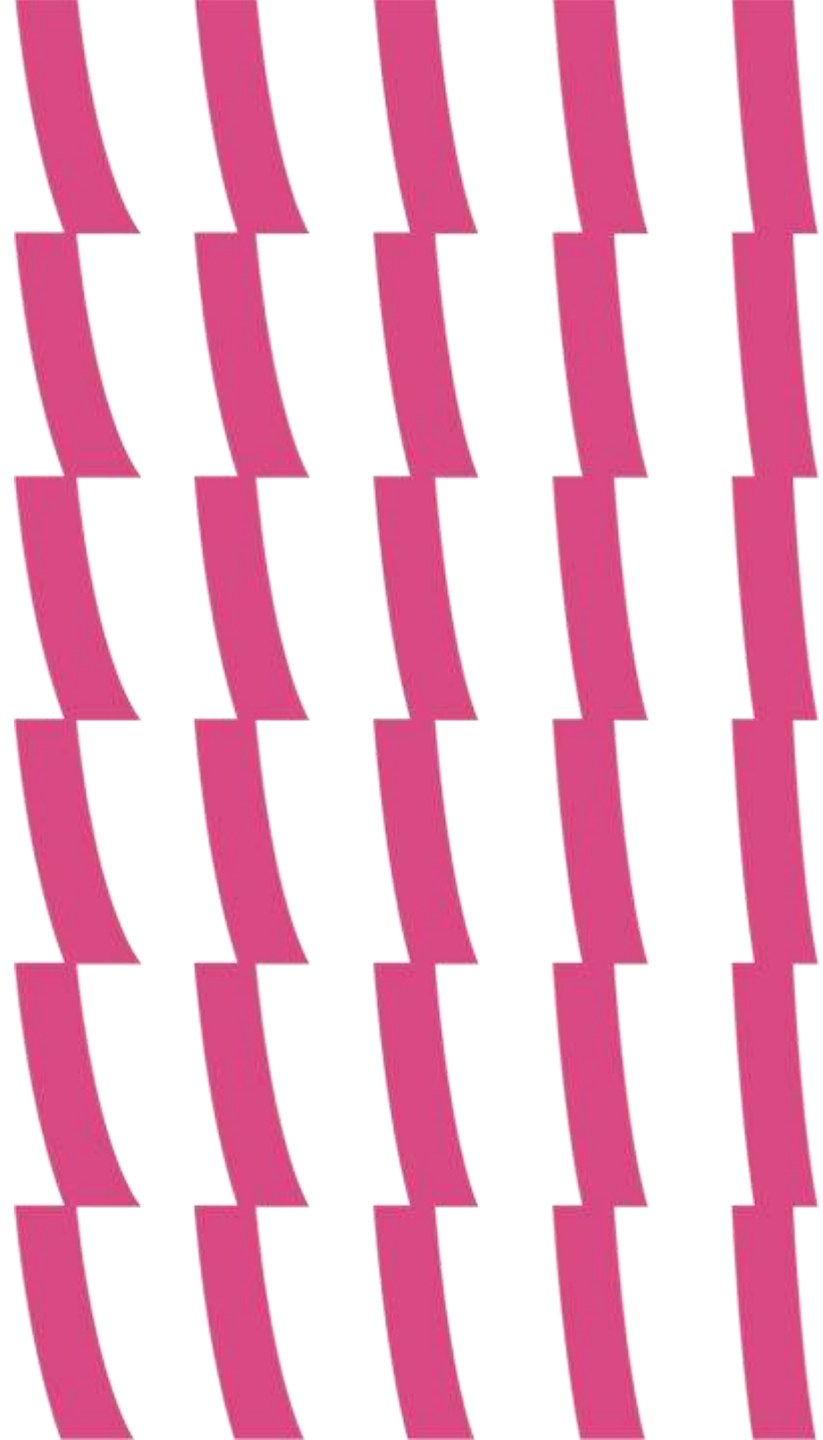


Слой витрин данных

Слой Витрин (Data Mart Layer) – то, ради чего строится хранилище. Отвечает за подготовку и предоставление данных конечным потребителям – людям или системам.

Витрины – денормализованные представления, которые отображают данные о какой-либо сущности с ее связями целиком, так, как это было бы удобно для анализа.

Задача слоя витрин – подготовка данных согласно требованиям конкретного потребителя – BI-платформы, группы пользователей, либо внешней системы.



Пример


```
{
  "order_id":23935
  "order_date":"2022-07-20",
  "price":"3000000",
  "currency":"копейки",
  "address_from":"Вольная 7",
  "address_to":"Смотровая 10",
  "user": {
      "user_id":"214535",
      "uid_type":"VK_ID"
      }
}
  ,
{
  "order_id":23939
  "order_date":"2022-07-20",
  "price":"70000",
  "currency":"копейки",
  "address_from":"Садовая 19",
  "address_to":"Вольная 7",
  "user": {
      "user_id":"69035",
      "uid_type":"OK_ID"
      }
}
,
{
  "order_id":33935
  "order_date":"2022-07-22",
  "price":"2500",
  "currency":"рубли",
  "address_from":"Задодская 4",
  "address_to":"Петронко 7",
  "user": {
      "user_id":"2309734",
      "uid_type":"VK_ID"
      }
}
```


order_id	order_dt	price	depar_addr	dest_addr	ok_id	vk_id
23935	2022-07-20	30000	Вольная 7	Смотровая 10	null	214535
23939	2022-07-20	700	Садовая 19	Вольная 7	69035	null
33935	2022-07-22	2500	Заводская 4	Петренко 7	null	2309734

order_cnt	price_avg	dt
2	15350	2022-07-20
1	2500	2022-07-22

Спасибо
за внимание!