Кафедра «Техника и технологии»

РЕФЕРАТ

по дисциплине:

«Информационные технологии в профессиональной деятельности»

Тема: Основы NoSQL (MongoDB, Redis)

Направление подготовки/специальность 09.03.02 Информационные системы и технологии

(код, наименование)

Обучающийся Кузюков Егор Валерьевич

(ФИО полностью)

Группа И-207 Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(номер группы)

Форма обучения Очная

Проверил Ефимов Матвей Александрович

(Фамилия И.О. преподавателя)

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Челябинск, 2025 г.

Оглавление

[Введение 3](#_Toc209319400)

[Глава 1: От SQL к NoSQL: Эволюция подходов к хранению данных 4](#_Toc209319401)

[1.1. Общие принципы организации данных: реляционная модель 4](#_Toc209319402)

[1.2. Недостатки реляционных БД и причины появления NoSQL 5](#_Toc209319403)

[1.3 Отказ от ACID в пользу CAP/BASE: Почему это важно? 5](#_Toc209319404)

[Глава 2: Многообразие NoSQL: Типы, сравнение и практическое применение 9](#_Toc209319405)

[2.1. Четыре основных типа NoSQL-баз данных 9](#_Toc209319406)

[2.2. Сравнительный анализ и выбор подходящего типа 10](#_Toc209319407)

[2.3. Практическое знакомство с документной СУБД MongoDB 13](#_Toc209319408)

[Заключение 15](#_Toc209319409)

# Введение

В современном мире данные стали новым нефтяным месторождением. Они генерируются с невероятной скоростью: сообщения в соцсетях, лайки, транзакции, показания датчиков, логи веб-серверов. Объем, разнообразие и скорость обработки этой информации — три большие «V» (Volume, Variety, Velocity) — легли в основу термина «Big Data».

Традиционные реляционные системы управления базами данных (SQL-БД), такие как MySQL, PostgreSQL и Oracle, десятилетиями были надежным фундаментом для хранения структурированной информации. Они основаны на строгих принципах: данные organized в таблицы с четко заданными столбцами (жесткая схема), связаны между собой отношениями, а работа с ними обеспечивается транзакциями с гарантиями ACID (Атомарность, Согласованность, Изоляция, Долговечность).

Однако для задач Big Data эти сильные стороны стали слабостями. Жесткая схема не позволяет быстро адаптироваться к изменениям. Сложно горизонтально масштабировать (добавлять новые серверы) реляционную базу без серьезных затрат и сложностей. Обеспечение строгой согласованности в распределенной системе может сильно замедлить работу.

Именно эти вызовы привели к появлению новой парадигмы — NoSQL (Not Only SQL — «не только SQL»). NoSQL — это не одна технология, а общее название для разнообразных нереляционных баз данных, каждая из которых создана для решения специфических задач. Цель данного доклада — классифицировать основные типы NoSQL БД, понять их сильные стороны, показать на практических примерах их применение и дать базовые навыки для начала работы с ними.

# Глава 1: От SQL к NoSQL: Эволюция подходов к хранению данных

## 1.1. Общие принципы организации данных: реляционная модель

Чтобы понять, зачем нужны NoSQL-базы, необходимо кратко вспомнить основы реляционных систем. Базы данных вроде MySQL, PostgreSQL или Oracle основаны на реляционной модели. Данные в них организованы в строгие таблицы (отношения), состоящие из строк и столбцов. Каждая таблица имеет заранее определенную схему — жесткую структуру, которая диктует, какие столбцы могут существовать и какого типа в них должны быть данные.

Сильной стороной SQL-баз является обеспечение целостности данных и поддержка транзакций, следующих принципам ACID (Atomicity — Атомарность, Consistency — Согласованность, Isolation — Изоляция, Durability — Долговечность). Это означает, что операции (например, перевод денег с одного счета на другой) выполняются полностью или не выполняются вовсе, гарантируя, что данные всегда будут в корректном состоянии.

Однако эта строгость и надежность становятся уязвимостью, когда мы сталкиваемся с тремя большими «V» Big Data:

Volume (Объем): Гигабайты данных превращаются в терабайты и петабайты.

Velocity (Скорость): Данные должны не только храниться, но и очень быстро записываться и считываться (например, ленты новостей, показания сенсоров).

Variety (Разнообразие): Данные перестали быть только структурированными. Появились полуструктурированные (JSON, XML) и неструктурированные форматы (изображения, тексты).

Базы данных NoSQL идеально подходят для работы с неопределенными, слабосвязанными или быстро меняющимися данными. Их модель данных, где схема определяется самим приложением, интуитивно понятна разработчикам. Такой подход обеспечивает гибкость схемы, что ускоряет итеративную разработку. NoSQL-решения делают ставку на производительность и горизонтальное масштабирование через сегментирование, иногда в ущерб строгой согласованности данных и ссылочной целостности. Они также отлично справляются с частично структурированными или неструктурированными данными.

При этом не всегда стоит делать жесткий выбор в пользу одной архитектуры. Гибридный подход, сочетающий SQL и NoSQL базы данных в рамках одного приложения, является распространенной практикой. Он позволяет сопоставить каждую задачу с наиболее подходящим инструментом, достигая тем самым оптимального соотношения цены и качества.

## 1.2. Недостатки реляционных БД и причины появления NoSQL

Именно три «V» определили ключевые ограничения реляционных СУБД:

1) Проблема горизонтального масштабирования (Scale-out). Традиционный способ увеличения производительности SQL-базы — это масштабирование вверх (Scale-up): добавление более мощного процессора, больше оперативной памяти, более быстрых дисков. Этот подход имеет физические и финансовые пределы. NoSQL-системы изначально проектировались для дешевого горизонтального масштабирования: чтобы справиться с нагрузкой, вы добавляете не более мощный сервер, а несколько обычных, объединяя их в кластер.

2) Жесткость схемы. Если ваши данные имеют сложную, иерархическую структуру (например, вложенные объекты в JSON) или их структура часто меняется, постоянное изменение схемы таблиц через операторы ALTER TABLE становится узким местом и тормозит разработку.

3) Производительность на больших объемах. Сложные JOIN-операции между огромными таблицами могут выполняться очень долго, что неприемлемо для систем, требующих мгновенного отклика.

NoSQL-базы данных появились как ответ на эти вызовы. Они пожертвовали строгой согласованностью и универсальностью реляционной модели в угоду производительности, масштабируемости и гибкости. Важно понимать: NoSQL — это не полная замена SQL, а другой инструмент, предназначенный для решения иного круга задач.

## 1.3 Отказ от ACID в пользу CAP/BASE: Почему это важно?

Появление и массовое распространение распределенных систем, особенно в эпоху облачных вычислений и Big Data, выявило фундаментальное несоответствие между строгими гарантиями ACID и требованиями к масштабируемости, доступности и отказоустойчивости. Это привело к переосмыслению подходов к проектированию систем и появлению парадигмы BASE как альтернативы ACID, тесно связанной с теоремой CAP.

1) Теорема CAP как фундаментальное ограничение:

Теорема CAP, сформулированная Эриком Брюером (Eric Brewer) и позже доказанная Сетом Гилбертом и Нэнси Линч, утверждает, что в любой распределенной системе возможно обеспечить одновременное выполнение не более двух из трех свойств: Согласованности (Consistency), Доступности (Availability) и Устойчивости к разделению (Partition Tolerance) . Поскольку сетевые разделения в распределенных системах считаются неизбежными, на практике отказ от полной согласованности (C) или доступности (A) становится не выбором, а необходимостью. Это означает, что системы, ориентированные на распределение (Partition Tolerance), вынуждены жертвовать либо мгновенной согласованностью данных (выбирая AP), либо их постоянной доступностью (выбирая CP) в моменты сетевых разделений .

2) Философия BASE как практическая реализация компромиссов:

В ответ на жесткие ограничения ACID и дилеммы CAP была сформулирована философия BASE (Basically Available, Soft-state, Eventually consistent — «Базово доступный, неустойчивое состояние, согласованность в конечном счете») . Она представляет собой прямой идеологический антипод ACID и ориентирована на создание высокодоступных и масштабируемых систем, которые мирятся с временной несогласованностью данных:

Basically Available (Базовая доступность): Система стремится оставаться доступной для обработки запросов даже в условиях сбоев или сетевых разделений, возможно, за счет возврата упрощенных или закешированных данных .

Soft-state (Неустойчивое состояние): Состояние системы может меняться со временем самостоятельно, даже без внешнего воздействия, из-за фоновых процессов синхронизации и репликации. Данные могут быть временно несогласованными .

Eventually consistent (Согласованность в конечном счете): Гарантируется, что если в систему не вносятся новые изменения, то через некоторое время все узлы придут к согласованному состоянию. Это ослабленное требование к согласованности является краеугольным камнем многих NoSQL-систем .

3) Почему этот отказ важен и неизбежен?

Масштабируемость и производительность: Обеспечение строгой согласованности (как в ACID) в распределенной системе требует сложных протоколов (например, двухфазного коммита — 2PC), синхронной репликации и блокировок, что вводит значительные задержки и ограничивает пропускную способность. Отказ от немедленной согласованности в пользу eventual consistency позволяет системам горизонтально масштабироваться, обрабатывая огромные объемы операций чтения и записи с низкой латентностью .

Высокая доступность: В системах, где бесперебойная работа критически важна (например, социальные сети, электронная коммерция), предпочтительнее всегда давать ответ, даже если данные могут быть немного устаревшими (AP-системы), а не полностью блокировать запросы до восстановления согласованности .

Соответствие современным требованиям: Для многих современных приложений (кэширование, аналитика в реальном времени, работа с большими данными) полная транзакционная согласованность ACID является избыточной и слишком дорогой. Гибкая модель BASE лучше подходит для этих сценариев .

4) Важность выбора: не «что лучше», а «что подходит»

Важно понимать, что отказ от ACID в пользу CAP/BASE — это не объявление ACID устаревшим, а расширение арсенала инструментов архитектора. Выбор между этими подходами определяется требованиями конкретного приложения:

ACID незаменим для систем, где критически важна точность и целостность данных каждый момент времени (банковские транзакции, финансовые системы).

CAP/BASE идеален для систем, где важнее масштабируемость, доступность и скорость отклика, а временная несогласованность допустима (социальные ленты, профили пользователей, телеметрия, IoT).

Таким образом, понимание теоремы CAP и философии BASE позволяет осознанно подходить к проектированию распределенных систем, выбирая оптимальный компромисс между согласованностью, доступностью и производительностью в зависимости от бизнес-требований.

Теперь, когда мы понимаем, почему возникла потребность в NoSQL, логично перейти к изучению многообразия этого семейства. Мы увидим, что не существует одной «универсальной» NoSQL-базы, а есть несколько специализированных типов, каждый из которых оптимизирован для своих сценариев.

# Глава 2: Многообразие NoSQL: Типы, сравнение и практическое применение

## 2.1. Четыре основных типа NoSQL-баз данных

NoSQL-базы данных классифицируются по модели хранения данных. Выбор модели определяет, насколько эффективно база будет справляться с той или иной задачей.

**Документные базы данных**

Концепция: Данные хранятся в виде документов. Чаще всего это JSON-, BSON- или XML-документы. Каждый документ содержит пары «ключ-значение», где значения могут быть как простыми типами (строка, число), так и вложенными структурами (массивы, объекты).

Главное преимущество: Гибкая и интуитивно понятная модель данных. Разработчик может легко хранить сложные объекты предметной области без их преобразования в табличный вид.

Яркие представители: MongoDB, Couchbase.

**Базы данных типа «Ключ-Значение» (Key-Value)**

Концепция: Самая простая модель. Данные хранятся как ассоциативный массив (словарь), где каждому уникальному ключу соответствует некоторое значение. Значение является «черным ящиком» для базы данных — она не знает и не анализирует его внутреннюю структуру.

Главное преимущество: Максимальная скорость и возможность простого горизонтального масштабирования для операций чтения/записи по ключу.

Яркие представители: Redis, Amazon DynamoDB, Riak.

**Графовые базы данных**

Концепция: Данные представлены в виде графа, состоящего из узлов (сущностей), ребер (связей) и их свойств (атрибутов). Узлы и связи могут иметь свои атрибуты.

Главное преимущество: Не имеющие аналогов производительность и удобство для работы со сложными и сильно связанными данными. Идеальны для анализа отношений и паттернов.

Яркие представители: Neo4j, Amazon Neptune.

**Колоночные базы данных (Column-Family)**

Концепция: Данные хранятся не по строкам, а по колонкам. Это означает, что все значения одного столбца хранятся рядом на диске, а не значения одной строки.

Главное преимущество: Невероятная скорость выполнения аналитических запросов (OLAP), которые часто требуют агрегации (сумма, среднее, count) данных по определенному столбцу из миллионов записей.

Яркие представители: Apache Cassandra, HBase. ClickHouse — это также колоночная СУБД, оптимизированная specifically для аналитики в реальном времени.

## 2.2. Сравнительный анализ и выбор подходящего типа

Чтобы осознанно выбрать тип базы данных, недостаточно просто знать их названия. Необходимо понимать, как внутреннее устройство каждой системы влияет на её сильные и слабые стороны в различных сценариях. Сравнение мы проведем по пяти ключевым критериям, которые напрямую отвечают на вопрос «подойдет ли эта база данных для моей задачи?».

**Критерий 1:** Модель данных — как организована информация?

Это главный критерий, определяющий все остальные.

Документные БД хранят информацию в виде самодостаточных документов (чаще всего в формате JSON). Это позволяет в одном документе объединить все данные о сущности, например, в карточке пользователя хранить его имя, список адресов и историю заказов. Данные имеют иерархическую структуру.

БД «Ключ-Значение» устроены максимально просто: каждому уникальному ключу (например, user\_session:12345) соответствует некоторое значение (например, сериализованный объект данных сессии). База данных не анализирует структуру значения, что обеспечивает максимальную скорость.

Графовые БД представляют мир как совокупность взаимосвязанных сущностей. Узлы — это объекты (например, «Пользователь», «Товар»), а связи — это отношения между ними («Купил», «Друг»). Это позволяет наглядно моделировать сложные сети.

Колоночные БД хранят данные таблично, но не по строкам, а по колонкам. Это значит, что все значения одного столбца (например, «цена товара») физически хранятся вместе, отдельно от значений других столбцов.

**Критерий 2:** Гибкость схемы — насколько легко изменить структуру данных?

Этот критерий показывает, как база данных реагирует на изменение требований в ходе разработки.

Документные и БД «Ключ-Значение» обладают высокой гибкостью схемы (иногда это называют «схема при чтении»). Вам не нужно заранее объявлять, какие поля будет иметь документ. Вы можете добавлять новые поля к одним документам, не добавляя их к другим. Это идеально для быстроразвивающихся проектов.

Графовые БД имеют умеренно гибкую схему. Узлы и связи имеют типы, но их свойства можно относительно легко добавлять и изменять.

Колоночные БД технически также обладают гибкостью, но их структура все же требует более внимательного проектирования под конкретные аналитические запросы.

**Критерий 3:** Сложность запросов — какие вопросы мы можем задавать данным?

Разные типы БД по-разному позволяют извлекать и анализировать информацию.

БД «Ключ-Значение» — чемпионы по скорости, но аутсайдеры по сложности запросов. Единственный эффективный способ получить данные — обратиться к ним по точному ключу. Поиск по значению невозможен или очень неэффективен.

Документные БД позволяют выполнять довольно сложные запросы внутри документа и по его полям: поиск по значению, использование условий, агрегации. Однако операции, которые в реляционных базах делаются через JOIN (соединение таблиц), здесь либо не поддерживаются, либо требуют неэффективных обходных путей.

Графовые БД специализируются на запросах, исследующих связи. Их язык запросов позволяет находить сложные паттерны: «найди всех друзей друзей пользователя А, которые купили товар Б». Выполнение таких запросов в других типах БД было бы чрезвычайно медленным.

Колоночные БД созданы для молниеносного выполнения аналитических запросов, которые затрагивают огромное количество строк, но лишь несколько столбцов. Операции агрегации — SUM(), AVG(), COUNT() — выполняются здесь быстрее всего, так как система считывает и обрабатывает не всю строку целиком, а только нужные колонки.

**Критерий 4:** Горизонтальное масштабирование — как система растет с нагрузкой?

Способность распределять данные и нагрузку на множество серверов критически важна для современных приложений.

БД «Ключ-Значение» и Колоночные БД лидируют в этом аспекте. Они изначально проектировались для легкого и дешевого горизонтального масштабирования. Добавление новых серверов в кластер позволяет почти линейно наращивать производительность.

Документные БД также хорошо поддерживают шардирование — автоматическое распределение данных по разным серверам на основе ключа.

Графовые БД — самые сложные для распределения. Поскольку запросы часто требуют обхода множества взаимосвязанных узлов, которые могут находиться на разных серверах, это создает высокие накладные расходы и задержки. Кластеризация графовых БД — нетривиальная задача.

**Критерий 5:** Основные сценарии применения — где этот тип сияет brightest?

Итогом сравнения является понимание идеальных use-case для каждого типа.

Документные БД (MongoDB): Каталоги товаров интернет-магазинов, пользовательские профили, системы управления контентом (CMS), данные мобильных приложений. Любые задачи, где данные имеют четкую объектную структуру и требуют гибкой схемы.

БД «Ключ-Значение» (Redis): Кеширование результатов запросов или HTML-страниц для ускорения сайтов, хранение сессий пользователей, корзины покупок, очереди задач (например, для фоновой отправки email), счетчики.

Графовые БД (Neo4j): Социальные сети (анализ социальных графов), рекомендательные системы («люди, смотревшие это, также смотрели...»), обнаружение мошеннических схем (выявление сложных связей между транзакциями), семантические базы знаний.

Колоночные БД (ClickHouse, Cassandra): Это прямо ваша сфера интересов как аналитика. Системы бизнес-аналитики (BI), хранилища данных (Data Warehouse), аналитика в реальном времени (обработка логов, телеметрии, событий с веб-сайтов), построение сложных отчетов и дашбордов на основе гигантских объемов данных.

**Как сделать выбор?** Задайте себе простой вопрос: «Какой основной способ доступа к моим данным?»

* Если это обращение по уникальному идентификатору — выбирайте «Ключ-Значение».
* Если вы работаете со сложными, целостными объектами без необходимости сложных соединений — выбирайте документную БД.
* Если ваша задача — анализ связей и отношений — ваш выбор графовая БД.
* Если вам нужна молниеносная агрегация миллиардов записей для отчетности — ваш инструмент колоночная БД.

Понимание этих фундаментальных различий — ключ к проектированию эффективных и масштабируемых систем. Теперь, чтобы закрепить теорию на практике, давайте посмотрим, как работают простые запросы в самой популярной документной БД — MongoDB.

## 2.3. Практическое знакомство с документной СУБД MongoDB

MongoDB — одна из самых популярных документных NoSQL-баз. Данные в ней хранятся в коллекциях (аналог таблиц), которые содержат документы (аналоги строк, но в формате BSON, бинарном JSON).

Базовые операции:

1) Вставка документа (Insert). Создадим документ, представляющий пользователя.

db.users.insertOne({

    name: "Иван Петров",

    age: 30,

    email: "ivan@mail.ru",

    address: {

        city: "Москва",

        street: "Ленина"

    },

    hobbies: ["программирование", "чтение"]

})

Обратите внимание на гибкость: документ содержит вложенный объект (address) и массив (hobbies). Не нужно заранее описывать схему.

2) Поиск документов (Find). Найдем всех пользователей из Москвы.

db.users.find({ "address.city": "Москва" })

Запрос выглядит очень естественно, обращение к вложенному полю происходит через точку.

3) Обновление документа (Update). Добавим пользователю новое хобби.

db.users.updateOne(

    { name: "Иван Петров" }, // Критерий поиска

    { $push: { hobbies: "гитара" } } // Операция добавления в массив

)

Эти простые примеры демонстрируют мощь и простоту работы с документной моделью. Для веб-разработки, где данные часто передаются в формате JSON, MongoDB подходит идеально.

## Заключение

Проведенное исследование позволило всесторонне изучить фундаментальные основы и практические аспекты нереляционных баз данных NoSQL. Отправной точкой стало понимание объективных ограничений реляционных СУБД, которые, несмотря на свою надежность и структурированность, столкнулись с вызовами современных объемов, скорости и разнообразия данных. Это закономерно привело к появлению и активному развитию альтернативной парадигмы — NoSQL, которая предлагает не единое решение, а целый спектр специализированных инструментов.

Ключевым выводом данной работы является то, что каждый из четырех основных типов NoSQL-баз — документные, хранилища «ключ-значение», графовые и колоночные — занимает свою четко определенную нишу и решает уникальный круг задач. Не существует «лучшей» базы данных вообще; существует оптимальный выбор для конкретной цели. Документная MongoDB идеальна для объектно-ориентированной разработки и работы с гибкими схемами, Redis обеспечивает непревзойденную скорость для кеширования и работы с простыми структурами, графовая Neo4j не имеет равных в анализе сложных взаимосвязей, а колоночные решения вроде ClickHouse становятся незаменимым фундаментом для высокопроизводительной аналитики и Data Warehousing.

Таким образом, современный разработчик и аналитик должны владеть не одной конкретной технологией, а понимать всю палитру доступных инструментов. Принцип «полиглотного персистентства», при котором в рамках одного приложения используются разные типы баз данных для решения различных подзадач, становится новым стандартом индустрии. NoSQL — это не слепая замена SQL, а мощное расширение арсенала, позволяющее строить высоконагруженные, масштабируемые и эффективные системы, отвечающие вызовам эпохи big data.

Изучение основ NoSQL открывает путь к проектированию архитектуры, которая не просто хранит данные, но делает это оптимальным образом, обеспечивая быстроту, гибкость и мощность, необходимые для успеха современных digital-проектов.

Список используемых источников

1. Всё, что вы не знали о CAP теореме / Habr. 2025. [Электронный ресурс]. URL: https://habr.com/ru/articles/328792/

2. Выбор между SQL и NoSQL: ACID и CAP, схема и транзакции Medium. 2025. [Электронный ресурс]. URL: https://medium.com/nuances-of-programming/выбор-между-sql-и-nosql-acid-и-cap-схема-и-транзакции-6c0e0a861c3b

3. Что такое NoSQL? Amazon. 2025. [Электронный ресурс]. URL: https://aws.amazon.com/ru/nosql/

4. Выбираем решение для NoSQL Habr. 2025. [Электронный ресурс]. URL: https://habr.com/ru/companies/otus/articles/882598/

5. NoSQL базы данных: понимаем суть Habr. 2025. [Электронный ресурс]. URL: https://habr.com/ru/articles/152477/