



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень

бакалавриат

(бакалавриат, магистратура, специалитет)

Форма обучения

очная

(очная, очно-заочная, заочная)

Направление(-я)
подготовки

10.05.04 Информационно-аналитические системы безопасности

(код(-ы) и наименование(-я))

Институт

Кибербезопасности и цифровых технологий (ИКБ)

(полное и краткое наименование)

Кафедра

КБ-2 «Прикладные информационные технологии»

(полное и краткое наименование кафедры, реализующей дисциплину (модуль))

Лектор

к.т.н., Селин Андрей Александрович

(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года

2024/2025

(учебный год цифрами)

Проверено и согласовано «___» _____ 2024 г.

А.А. Бакаев

*(подпись директора Института/Филиала
с расшифровкой)*

Москва 2024 г.



Технологии хранения в системах кибербезопасности

2024 год



Лекция 6. NoSQL БД

Учебные вопросы лекции:

- 
- The background of the slide features a complex network diagram with nodes and connecting lines, overlaid on a faint globe. The network diagram is composed of white lines and dots on a blue-toned background.
1. Нереляционные БД
 2. Типы нереляционных БД
 3. MongoDB

Введение

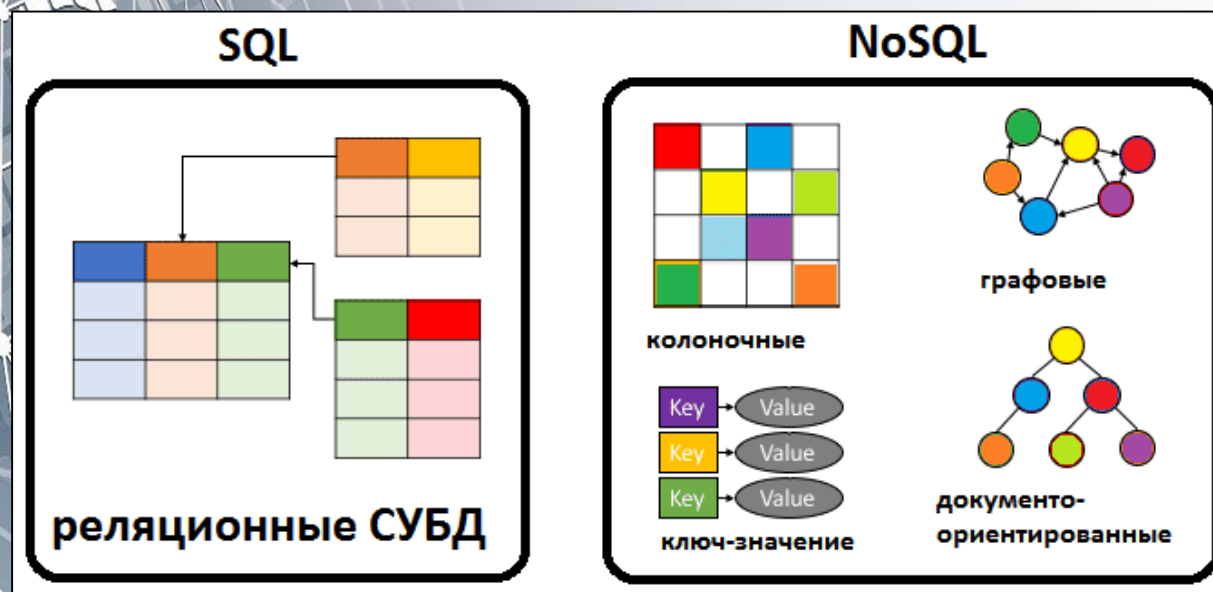
Традиционные системы управления реляционными базами данных (БД) предоставляют мощные механизмы для хранения данных и построения запросов к ним. За годы использования эти системы сильно развились и сейчас гарантируют стабильность работы и надежность транзакций.

Однако в последние годы в некоторых областях стало храниться слишком много полезных данных, использование и хранение которых традиционными методами становится затруднительно. К этим областям можно отнести пользовательский контент социальных сетей и данные, полученные из других обширных сетей (Big Data).

Для такого типа данных создаются и развиваются NoSQL решения, которые способны обеспечивать горизонтальную масштабируемость и большую отзывчивость, но жертвуют при этом надежностью транзакций.

Нереляционные БД

Нереляционная БД – это БД, в которой не используется табличная схема строк и столбцов. В этих БД применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных. Например, данные могут храниться как простые пары "ключ — значение", документы JSON или граф, состоящий из ребер и вершин.



Термин *NoSQL* применяется к хранилищам данных (ХД), которые не используют язык запросов SQL. Вместо этого они запрашивают данные с помощью других языков программирования и конструкций.

Нереляционные БД

На практике NoSQL означает "нереляционная БД", даже несмотря на то, что многие из этих БД поддерживают запросы, совместимые с SQL. Тем не менее базовая стратегия выполнения запросов SQL обычно значительно отличается от применяемой в системе управления реляционной БД (реляционная СУБД).

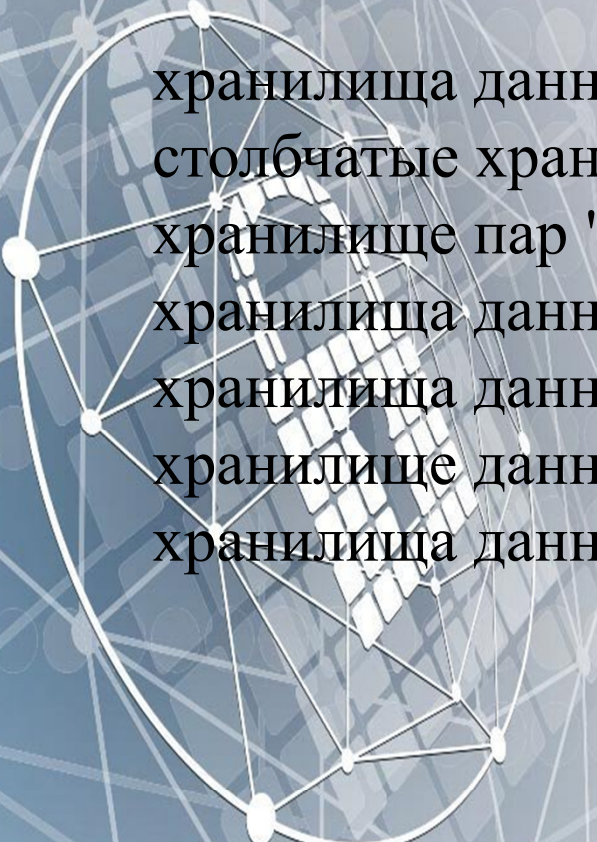
NoSQL – это подход к реализации масштабируемого хранилища (базы) информации с гибкой моделью данных, отличающийся от классических реляционных СУБД. В нереляционных базах проблемы масштабируемости (scalability) и доступности (availability), важные для Big Data, решаются за счёт атомарности (atomicity) и согласованности данных (consistency)

NoSQL-базы оптимизированы для приложений, которые должны быстро, с низкой временной задержкой (low latency) обрабатывать большой объем данных с разной структурой. Таким образом, нереляционные хранилища непосредственно ориентированы на Big Data.



Основные типы нереляционных БД

Основные категории нереляционных БД или БД NoSQL:



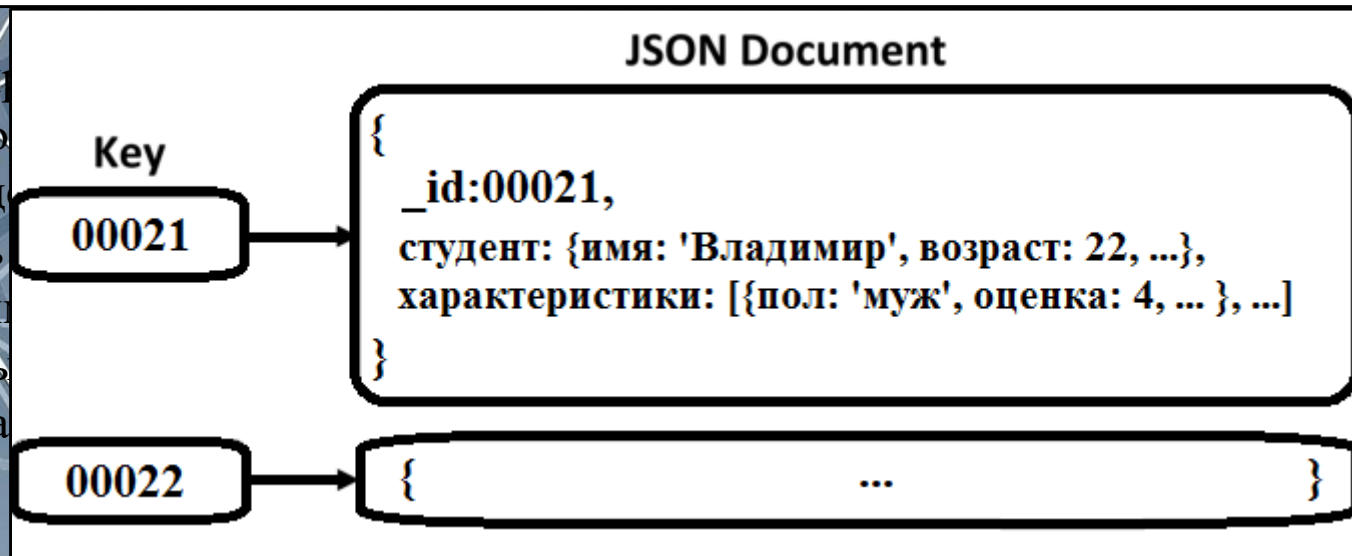
- хранилища данных документов;
- столбчатые хранилища данных;
- хранилище пар "ключ — значение";
- хранилища данных графов;
- хранилища данных временных рядов;
- хранилище данных объектов;
- хранилища данных внешних индексов.

Хранилища данных документов

Приложение может получать документы по ключу документа. Ключ – это уникальный идентификатор документа. Часто к нему применяется хэширование для равномерного распределения данных. Некоторые БД документов автоматически создают ключ документа. Другие позволяют указать атрибут документа, который будет использоваться в качестве ключа. Приложение также может запрашивать документы на основе значения одного или нескольких полей. Некоторые БД документов поддерживают индексирование, чтобы облегчить быстрый поиск документов по одному или нескольким индексированным полям.

запросы и

Как при
составляю
может сод
содержать
несколько
документы
документа



Элементы,
сущность
ент может
яются по
чтобы все
хранить в
ии.

Многие базы данных документов поддерживают обновления "на месте", то есть позволяют приложению изменять значения отдельных полей без перезаписи всего документа. Операции чтения и записи в нескольких полях одного документа обычно являются атомарными.

Столбчатые хранилища данных

Столбчатое хранилище данных или хранилище семейств столбцов упорядочивает данные по столбцам и строкам. Семейства данных в простейшей форме почти неотличимы от семейств строк в хранилище строк.

Данные одной сущности имеют одинаковые ключи строк во всех семействах столбцов. Такая структура, в которой строки любого объекта в семействе столбцов могут динамически изменяться, определяет важное преимущество этой категории хранилищ. Семейства столбцов очень хорошо подходят для хранения данных с различными схемами.

Семейства столбцов могут рассматриваться как единое целое. Другие данные, используемые в процессах, хранятся отдельно в других семействах столбцов. В семействе столбцов можно динамически добавить новые столбцы, а строки могут быть разнотипными (то есть строки не обязаны иметь значение для каждого столбца).

CustomerID	Column Family: Identity	CustomerID	Column Family: Contact Info
001	First name: Mu Bae Last name: Min	001	Phone number: 555-0100 Email: someone@example.com
002	First name: Francisco Last name: Vila Nova Suffix: Jr.	002	Email: vilanova@contoso.com
003	First name: Lena Last name: Adamczyk Title: Dr.	003	Phone number: 555-0120

Столбчатые хранилища данных

В отличие от хранилища пар "ключ – значение" и БД документов, большинство столбчатых БД упорядочивают хранимые данные с помощью самих значений ключей, а не хэш-кодов от них. Ключ строки рассматривается как первичный индекс и обеспечивает доступ на основе определенного ключа или их диапазона. Некоторые реализации позволяют создавать вторичные индексы по определенным столбцам в семействе столбцов. Вторичные индексы позволяют получать данные по значениям столбцов, а не ключам строки.

Все столбцы одного семейства хранятся на диске в одном файле. Каждый файл содержит определенное число строк. При использовании больших наборов данных этот подход позволяет повысить производительность за счет снижения объема данных, которые необходимо считывать с диска, когда отправляется запрос на получение нескольких столбцов за раз.

Чтение и запись строки из одного семейства столбцов – это обычно атомарные операции. Однако некоторые реализации поддерживают атомарность всей строки, распределенной по нескольким семействам столбцов.

Хранилище пар "ключ – значение"

Хранилище пар "ключ – значение" представляет собой таблицу. Каждое значение ключей используется для функции хэширования, обеспечивающей равномерное распределение значений по таблице.

Большинство хранилищ пар "ключ – значение" поддерживают простые операции: добавить запись, изменить значение, удалить запись целиком. В большинстве хранилищ пар "ключ – значение" запись одного значения занимает относительно долгое время.

Приложение может хранить в наборе значений произвольные данные, но некоторые хранилища пар "ключ – значение" накладывают ограничения на максимальный размер значений. Программное обеспечение хранилища ничего не знает о значениях, которые в нем хранятся. Все сведения о схеме поддерживаются и применяются на уровне приложения. Эти значения по существу являются большими двоичными объектами, которые хранилище извлекает и сохраняет по соответствующему ключу.

Ключ	Значение
AAAAA	1101001111010100110101111...
AABAB	1001100001011001101011110...
DFA766	0000000000101010110101010...
FABCC4	1110110110101010100101101...

ольшую хэш-функцию и хранилище пар "ключ – значение" для некоторой функции хэширования, обеспечивающей равномерное распределение значений по таблице.

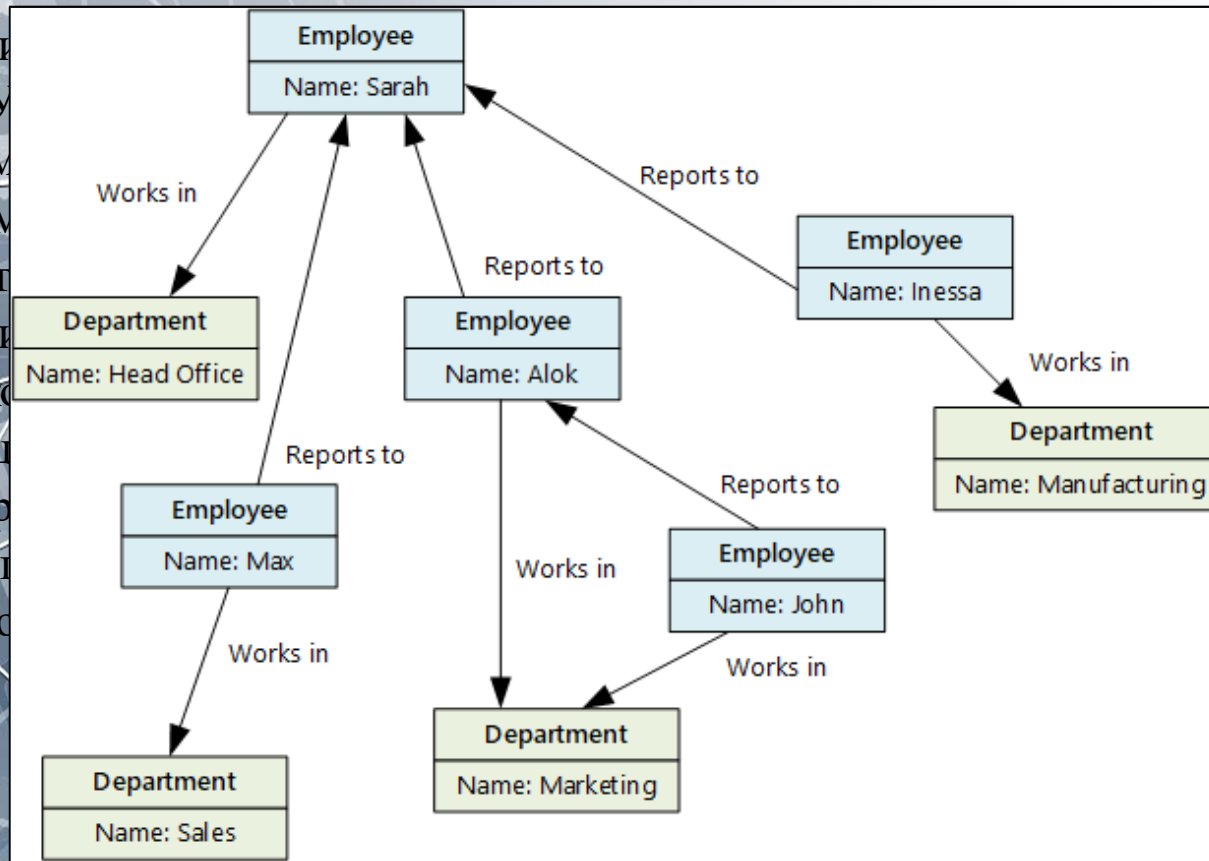
Большинство хранилищ пар "ключ – значение" поддерживают простые операции: добавить запись, изменить значение, удалить запись целиком. В большинстве хранилищ пар "ключ – значение" запись одного значения занимает относительно долгое время.

Приложение может хранить в наборе значений произвольные данные, но некоторые хранилища пар "ключ – значение" накладывают ограничения на максимальный размер значений. Программное обеспечение хранилища ничего не знает о значениях, которые в нем хранятся. Все сведения о схеме поддерживаются и применяются на уровне приложения. Эти значения по существу являются большими двоичными объектами, которые хранилище извлекает и сохраняет по соответствующему ключу.

Приложение может хранить в наборе значений произвольные данные, но некоторые хранилища пар "ключ – значение" накладывают ограничения на максимальный размер значений. Программное обеспечение хранилища ничего не знает о значениях, которые в нем хранятся. Все сведения о схеме поддерживаются и применяются на уровне приложения. Эти значения по существу являются большими двоичными объектами, которые хранилище извлекает и сохраняет по соответствующему ключу.

Хранилища данных графов

Хранилища данных графов позволяют хранить и обрабатывать данные, связанные между собой. У них есть ребра, которые соединяют узлы. Узлы могут быть конкретными объектами, а ребра — связями между ними. Хранилища данных графов позволяют выполнять запросы, которые требуют поиска информации о связях между объектами. Например, можно найти всех сотрудников, которые работают в определенном отделе, или всех сотрудников, которые подчиняются определенному руководителю. Хранилища данных графов также позволяют выполнять запросы, которые требуют поиска информации о связях между объектами. Например, можно найти всех сотрудников, которые работают в определенном отделе, или всех сотрудников, которые подчиняются определенному руководителю.



в: узлами и
деляют связи
т сведения о
блице. Грани
о выполнять
ировать связи
организации,
отрудники и
ром работает
направление

Такая структура позволяет легко выполнять такие запросы, как "найти всех сотрудников, которые прямо или косвенно подчиняются Светлане" или "найти всех, кто работает в одном отделе с Дмитрием". Процессы сложного анализа выполняются быстро даже на больших графах с большим количеством сущностей и связей. Многие базы данных графов предоставляют язык запросов, который можно использовать для эффективного обхода сети связей.

Хранилища данных временных рядов

Данными временных рядов называются наборы значений, которые упорядочены по времени. Соответственно хранилища данных временных рядов оптимизированы для хранения данных именно такого типа. Хранилища данных временных рядов должны поддерживать очень большое число операций записи, так как обычно в них в режиме реального времени собирается большой объем данных из большого количества источников. Эти хранилища также хорошо подходят для хранения данных телеметрии. Обновления в таких базах данных выполняются редко, а удаление чаще всего является массовой операцией.

timestamp	deviceid	value
2017-01-05T08:00:00.123	1	90.0
2017-01-05T08:00:01.225	2	75.0
2017-01-05T08:01:01.525	2	78.0

Хранилище данных объектов

Хранилища данных объектов оптимизированы для хранения и извлечения больших двоичных объектов, например изображений, текстовых файлов, видео- и аудиопотоков, объектов данных и документов приложений большого размера, образы дисков виртуальных машин. Объект состоит из сохраненных данных, метаданных и уникального идентификатора доступа к объекту. Хранилища объектов поддерживают отдельные большие файлы, а также позволяют управлять всеми

Не
двоичн
паралл
масшта
как не
одновр

path	blob	metadata
/delays/2017/06/01/flights.csv	0XAABBCCDDEEF...	{created: 2017-06-02}
/delays/2017/06/02/flights.csv	0XAADDCCDDEEF...	{created: 2017-06-03}
/delays/2017/06/03/flights.csv	0XAEBBDEDDEEF...	{created: 2017-06-03}

большой
быстрое
ализовать
йлах, так
х, могут

Часто хранилища данных объектов используют как сетевые общие папки. Доступ к файлам, хранящимся в этих папках, можно получить через компьютерную сеть с использованием стандартных сетевых протоколов, например SMB. Если созданы необходимые механизмы поддержки безопасности и одновременного доступа, такое совместное использование данных позволяет распределенным службам с высокой степенью масштабируемости предоставлять доступ к данным для базовых низкоуровневых операций, то есть для простых запросов на чтение и запись.

Хранилища данных внешних индексов

Хранилища данных внешних индексов позволяют искать информацию, содержащуюся в других хранилищах данных и службах. Внешний индекс выступает в роли вторичного индекса любого хранилища данных. Кроме того, с его помощью можно индексировать большие объемы данных и предоставлять доступ к этим индексам почти в реальном времени.

Например, в файловой системе могут храниться текстовые файлы. По пути файл можно найти быстро, но поиск на основе содержимого выполняется медленно, так как сканируются все файлы. Внешний индекс позволяет создавать вторичные индексы, а затем быстро искать путь к файлам, соответствующим заданным условиям.

id	search-document
233358	{"name": "Pacific Crest National Scenic Trail", "county": "San Diego", "elevation": 1294, "location": {"type": "Point", "coordinates": [-120.802102, 49.00021]}}
801970	{"name": "Lewis and Clark National Historic Trail", "county": "Richland", "elevation": 584, "location": {"type": "Point", "coordinates": [-104.8546903, 48.1264084]}}
1144102	{"name": "Intake Trail", "county": "Umatilla", "elevation": 1076, "location": {"type": "Point", "coordinates": [-118.0468873, 45.9981939]}}

Хранилища данных внешних индексов

Индексы создаются в процессе индексирования, который может выполняться по модели извлечения, то есть по требованию хранилища данных, или по модели передачи, то есть по команде из кода приложения. В некоторых системах поддерживаются многомерные индексы и полнотекстовый поиск по большим объемам текстовых данных.

Часто хранилища данных внешних индексов используют для реализации полнотекстового поиска и поиска в Интернете. В этих случаях поддерживается точный или нечеткий поиск. Нечеткий поиск находит документы, которые соответствуют набору условий, и вычисляет для них коэффициент совпадения с этим набором.

Некоторые внешние индексы также поддерживают лингвистический анализ, который возвращает соответствия с учетом синонимов, категорий (например, при поиске по запросу "собаки" соответствием считается "питомцы") и морфологии (например, при поиске по запросу "бег" соответствием считается "бегущий").

Согласованность и доступность данных: CAP

Другое определяющее свойство БД, помимо того, как данные хранятся, это уровень их согласованности. Одни базы создаются с учетом высокой согласованности и сериализации данных (ACID), другие делают упор на доступность данных (BASE). Этот компромисс присущ каждой распределенной системе БД, и большое число NoSQL баз показывает широкий спектр решений между этими двумя парадигмами. Известны две теоремы: CAP и PACELC, согласно которым системы БД можно классифицировать по их положению в этом спектре.

Теорема CAP (теорема Брюера) – эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

- согласованность данных (**c**onsistency) – во всех вычислительных узлах в один момент времени данные не противоречат друг другу;
- доступность (**a**vailability) – любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;
- устойчивость к разделению (**p**artition tolerance) – расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

Согласованность и доступность данных: RACELC

– Couchbase предоставляет ряд вариантов согласованности и доступности во время разделения, а также диапазон параметров задержки и согласованности без разделения. В отличие от большинства других БД, Couchbase не имеет единого набора API и не реплицирует все службы данных однородно. Для записи Couchbase предпочитает согласованность, а не доступность, что делает ее формально CP, но при чтении появляется больше управляемой пользователем изменчивости в зависимости от желаемого уровня согласованности и типа операции. В зависимости от диапазона против полноты согласованности и типа операции, Couchbase предлагает диапазон согласованности и доступности. В отличие от большинства других БД, Couchbase не имеет единого набора API и не реплицирует все службы данных однородно. Для записи Couchbase предпочитает согласованность, а не доступность, что делает ее формально CP, но при чтении появляется больше управляемой пользователем изменчивости в зависимости от желаемого уровня согласованности и типа операции. В зависимости от диапазона против полноты согласованности и типа операции, Couchbase предлагает диапазон согласованности и доступности.

DDBS	P+A	P+C	E+L	E+C
Dynamo	Да		Да ^[a]	
Cassandra	Да		Да ^[a]	
Cosmos DB	Да		Да	
Couchbase		Да	Да	Да
Riak	Да		Да ^[a]	
VoltDB/H-Store		Да		Да
Megastore		Да		Да
MongoDB	Да			Да
PNUTS		Да	Да	

– Cosmos DB поддерживает варианты согласованности, позволяющих выбирать между согласованностью и доступностью. В базовом случае система гарантирует, что в любой момент времени только один узел будет доступен для записи. PNUTS – это система

й согласованности, которая обеспечивает согласованность и L/C в случае разделения. В базовом случае система гарантирует, что в любой момент времени только один узел будет доступен для записи.

MongoDB

MongoDB – это документоориентированная нереляционная СУБД, которая распространяется по лицензии SSPL и имеет открытый исходный код.

Базы состоят из коллекций и документов – иерархических структур, содержащих пары «ключ – значение» (поля). Если проводить аналогии с реляционной базой, коллекции при таком способе хранения соответствуют таблицам, а документы – строкам.

- Информация отформатирована в BSON – двоичной кодировке JSON-подобных документов. Это позволяет поддерживать данные типа Date и двоичных файлов, что невозможно в JSON.

У документов нет строгой структуры. Они могут содержать разные наборы полей, причём различающиеся как по типу, так и по количеству. Например, документ может выглядеть так:

```
test> db.smartphones.find()
[
  {
    _id: ObjectId("64084a222063b8732a692d5a"),
    model: 'DEXP G450 One',
    color: 'Blue',
    screen: '5 inch',
    year: '2021'
  }
]
```


MongoDB

Значениями могут быть даже другие документы – их называют встроеными.

Подобно тому, как у строк в реляционных БД есть первичный ключ, у каждого документа в MongoDB есть уникальный идентификатор (в наших примерах это `_id`). Он формируется автоматически или задаётся пользователем.

Например, все эти документы принадлежат одной коллекции **smartphones**:

Перед добавлением данных необязательно создавать коллекцию, документ можно делать сразу.

```
test> db.smartphones.find()
[
  {
    _id: ObjectId("64084a222063b8732a692d5a"),
    color: 'Blue',
    model: 'DEXP G450 One',
    screen: '5 inch',
    year: '2021'
  },
  {
    _id: ObjectId("DEXP"),
    model: 'DEXP G450 One',
    mpn: '[MZB0CBJRU]',
    material: 'plastic',
    year: 2022
  },
  {
    _id: ObjectId("640f7e114f9f59d9dcda468d"),
    model: 'Xiaomi Redmi Note 10 Pro',
    color: 'Grey',
    screen: {
      diagonal: '6.67 inch',
      resolution: '2400x1080',
      refresh_rate: '120 Hz'
    }
  }
]
```

Преимущества и недостатки MongoDB

Преимущества, которые особенно актуальны при работе с большими объёмами данных:

- гибкая система хранения информации: в приложениях не обязательно преобразовывать объекты в элементы таблиц, не нужно пересоздавать схему базы при изменении структуры данных, например при добавлении нового поля. В документах хранится информация разных типов, что важно при работе с большими данными, которые имеют разную структуру и взяты из разных источников;
- базы легко масштабируются;
- большинство языков имеют специальные инструменты для работы с Mongo – например, в JavaScript это Mongoose;
- благодаря индексации, системе запросов и другим особенностям можно быстро искать, читать и записывать данные в базах;
- базы MongoDB могут работать сразу на нескольких серверах: сегментирование позволяет распределять нагрузку, а репликация – создавать копии. Поэтому система работает быстро и без перебоев.

Недостатки:

- в базах нет хранимых процедур, триггеров и внешних ключей, поэтому невозможно полностью автоматизировать работу;
- нет полного соответствия ACID;
- есть сложности при работе с транзакциями, хотя разработчики стараются это исправить, и в скором времени это у них должно получиться.

Сфера использования MongoDB

MongoDB используется в веб-программировании, big data и аналитике, там, где приходится работать с большим количеством не связанных друг с другом фрагментов информации.

Также её нередко применяют в стартапах, где ещё нет чётко определённой структуры хранения данных и могут потребоваться постоянные её изменения.

Там, где требуются гибкие и масштабируемые базы:

- в каталогах товаров электронной коммерции;
- при хранении событий в системе (логировании);
- для записи информации с датчиков мониторинга;
- в управлении контентом;
- в играх;
- в платёжных системах;
- в приложениях интернета вещей;
- в мобильных приложениях;
- для кэширования;
- в приложениях, обрабатывающих временные ряды;
- и др.



СПАСИБО ЗА ВНИМАНИЕ!

