УДК 004.65 EDN KGDJMH

# Изучение причин и преимуществ использования баз данных NoSQL

**Д.В. Сотников***

ФГБОУ ВО «МИРЭА - Российский технологический университет», г. Москва, Россия

*E-mail: dmitrij.sotnickow@yandex.ru

**Аннотация.** В настоящее время большое количество компаний пользуется системами управления реляционными данными. Они превосходно справлялись со своей задачей на протяжении последних 20 лет, но объем данных непрерывно растет. Их обработка из-за вариативности типов и структур данных и общего объема стала излишне трудоемкой задачей. Реляционные базы данных не подходят для обработки таких больших данных из-за своих жестких ограничений данных, структуры, отношений и прочих факторов. В результате агрегация данных становится невозможной. Базы данных NoSQL предлагают практичную и понятную основу для объединения огромных объемов данных, структур и взаимодействий. Главными проблемами перехода от реляционных баз данных являются моделирование данных и миграция. Пока еще нет инструмента для переключения с реляционных баз данных на базы данных без SQL. Для миграции запросы к реляционным (SQL) базам данных должны быть преобразованы в запросы к базе данных NoSQL. Эта статья направлена на освещение и оценку возможностей NoSQL и процедуры миграции базы данных NoSQL.

**Ключевые слова:** NoSQL базы данных, Аналитика больших данных, большие данные, MongoDB.

# Exploring the reasons and benefits of using NoSQL databases

**D.V. Sotnikov***

MIREA – Russian Technological University, Moscow, Russia

*E-mail: dmitrij.sotnickow@yandex.ru

**Abstract.** Currently, a large number of companies use relational data management systems. They have done a good job for the last 20 years, but the amount of data is constantly growing. Their processing due to the variability of data types and structures and the total volume has become an unnecessarily time-consuming task. Relational databases are not suitable for handling such big data due to their severe data constraints, structure, relationships, and other factors. As a result, data aggregation becomes impossible. NoSQL databases offer a practical and understandable framework for aggregating vast amounts of data, structures and interactions. The main challenges of transition from relational databases are data modeling and migration. There is not yet a tool to switch from relational databases to NoSQL databases. For migration, relational (SQL) database queries must be converted to NoSQL database queries. This article aims to highlight and evaluate NoSQL features and NoSQL database migration procedures.

**Keywords:** NoSQL database, Big Data Analytics, Big Data, MongoDB.

## 1. Введение

NoSQL ─ это сокращение от Not Only SQL(не только SQL), представляет собой механизм хранения и извлечения данных, в котором используются методы моделирования, отличные от табличных отношений, встречающихся в реляционных базах данных. NoSQL лучше всего понимать как базу данных, которая не соответствует структуре обычной системы управления реляционными базами данных (RDMS). Она не изменяет данные с помощью SQL. Несмотря на то, что она не может полностью гарантировать атомарность, согласованность, изоляцию и надежность. Архитектура NoSQL обладает своими преимуществами, она является распределенной и отказоустойчивой. Из-за этих свойств она и потребовалась таким интернет-гигантам, как Facebook, Google, Amazon, которые приложили руку к разработке и внедрению технологии NoSQL. Им требовалась система управления базами данных, которая могла бы записывать и считывать данные в любой точке земного шара, масштабируя и обеспечивая производительность для огромных наборов данных и миллионов пользователей. NoSQL — это система баз данных, созданная для удовлетворения потребностей облачных приложений и выхода за пределы реляционных баз данных (RDBMS) с точки зрения масштаба, производительности, модели данных и распространения данных.

## 2. Цель исследования

Целью статьи является изучение возможностей, предоставляемых NoSQL системами и преимущества использования, совместно с освящением проблемы миграции при помощи MongoDB.

## 3. Анализ баз данных NoSQL систем

Данные NoSQL предлагают моделирование без строгой схемы, более свободный подход к моделированию данных, при котором семантика данных включается в гибкую топологию соединения и связанную модель хранени [1]. В результате управление большими наборами данных становится более гибким. Гибкая модель обеспечивает автономное распределение данных и эластичность использования вычислительных ресурсов, хранилища и пропускной способности сети, не требуя точной привязки данных для постоянного хранения в определенных физических местах. Кроме того, базы данных NoSQL предлагают встроенное кэширование данных, что снижает задержку доступа к данным и повышает производительность. Реляционная структура была ослаблена, чтобы

упростить применение различных моделей к разным видам исследований. Например, некоторые из них разработаны как хранилища ключей и значений, что хорошо вписывается в парадигмы программирования больших данных, такие как MapReduce. Однако он не требует соблюдения строго определенных структур, а сами модели не обязательно налагают какие-либо правила валидности. Несмотря на то, что «расслабленный» подход к моделированию и управлению прокладывает путь к повышению производительности аналитических приложений. В результате могут возникнуть опасности нерегулируемой деятельности по управлению данными, такие как непреднамеренное непоследовательное дублирование данных, семантическая неверная интерпретация и проблемы с актуальностью и своевременностью.

В этой статье рассматриваются четыре различные модели NoSQL:

- Хранилище «ключ-значение»,
- Документо-ориентированное хранилище,
- Колоночное хранилище,
- Хранилище на основе графов.

*Хранилище «ключ-значение»*. в котором объекты данных связаны с уникальными строками символов, называемыми ключами. Уникальные ключи используются как для идентификации сущностей, так и для поиска атрибутивной информации об этих сущностях во многих проектах NoSQL, что является вариацией темы «ключ-значение». Этот фундаментальный подход к бессхемной модели получил некоторое доверие благодаря широкому использованию уникальных ключей.

Семантика организации данных должна интерпретироваться бизнес-приложениями, которые будут ее использовать [2, 3]. Фундаментальные операции, выполняемые с хранилищем «ключ-значение», состоят из:

- Get(key), дает значение, связанное с предоставленным ключом.
- Put(key, value) связывает ключ и значение вместе.
- Функция multi-get(key1, key2,.., keyN), которая возвращает список значений, связанных со списком ключей.
- Delete(key) удаляет запись ключа из хранилища данных.

Ключ должен быть уникальным при использовании хранилища "ключ-значение", чтобы гарантировать доступность значений. Разработчик должен учитывать представления этих значений и то, как они должны быть связаны с ключом, чтобы сопоставить множество значений с одним ключом (например, список моделей

автомобилей из примера). Хранилища «ключ-значение» могут быть проиндексированы. по значению ключа для ускорения запросов данных.

Этот подход имеет определенные потенциальные недостатки, несмотря на тот факт, что пары ключ-значение особенно полезны как для хранения, так и для получения результатов аналитических алгоритмов. Потенциальный недостаток заключается в том, что сохранение уникальных ключей по мере увеличения объема данных может стать более сложной задачей.

*Документо-ориентированное хранилище.* Его можно сравнить с хранилищем «ключ-значение» в том смысле, что сохраненные элементы связаны с ключами в виде строк символов, которые используются для доступа к ним. Отличие состоит в том, что сохраняемые значения, известные как «документы», придают управляемым данным некоторую структуру и кодировку [4]. Существуют различные популярные общепринятые кодировки, такие как XML (расширяемый язык разметки), JSON (обозначение объектов сценария Java) и BSON (который представляет собой двоичное кодирование объектов JSON). Структура модели встроена в представление документа, что позволяет выводить семантику значений документа. Одно из существенных различий между документо-ориентированным хранилищем и хранилищем «ключ-значение» заключается в том, что первое встраивает метаданные атрибутов, связанные с сохраненным содержимым, что позволяет эффективно запрашивать данные на основе содержимого. Например, поиск всех документов, в которых для параметра «MallLocation» задано значение «Wheaton Mall», будет возвращен набор результатов со всеми документами, относящимися к любому «розничному магазину», расположенному в этом конкретном торговом центре.

*Колоночное хранилище.* В некотором смысле оно соединяют мир бессхемного управления данными и обычные реляционные модели. Колоночное хранилище данных представляет собой гибридный подход к хранению и администрированию данных. С одной стороны, оно напоминает методы для документо-ориентированного хранилища, за исключением того, что колоночные базы данных сохраняют структуры объектов на месте. Использование колоночных баз данных с большей вероятностью обеспечит стандартное соответствие ACID, чем некоторые другие модели NoSQL, связанные с надежностью базы данных [5]. Крайне важно иметь в виду, что колоночные базы данных не являются реляционными и поиск в них не осуществляется с помощью SQL.

*Хранилище на основе графов*. Вместо строгого формата SQL или представления таблиц и колонок, используется гибкое графическое представление. Графовые структуры используются вместе с ребрами, узлами и свойствами, что обеспечивает безиндексовую смежность, при этом данные могут преобразовываться из одной модели в другую.

Реляционные модели могут скопировать поведение графовых, но для этого потребуется использовать Join, что будет неэффективно.

## 4. Использование NoSQL. Миграция при помощи MongoDB

Для большинства предприятий собранные данные являются одним из их самых ценных активов. Компании время от времени переключаются с одной информационной системы на другую в зависимости от потребностей клиентов и технологических достижений. В результате данные должны быть перенесены из устаревшей системы в новую. Согласно Ф. Маттесу и К. Шульцу, миграция данных означает: «поддерживаемый инструментами одноразовый процесс, который переносит форматированные данные из исходной структуры данных в целевую структуру данных, когда обе структуры различаются концептуально и/или физически».

Будет использоваться следующая общая стратегия миграции:

A. Подготовка данных в формате файла JSON.

B. Извлечение данных в плоские файлы с использованием формата файла JSON или извлечение данных с помощью настраиваемых загрузчиков данных из хранилища обработанных данных.

В следующих разделах более подробно рассматриваются многочисленные действия для каждого этапа миграции [6].

### 4.1. Подготовка и извлечение данных

Извлечение, преобразование и загрузка данных известны как ETL, и согласование играет важную роль в завершении процесса. Перед загрузкой данных в промежуточные таблицы процесс ETL также включает проверку и обогащение данных. Подготовка данных. Во время подготовки данных будут выполнены следующие действия:

1) Создаются объекты базы данных,
2) В зависимости от требований должны быть созданы необходимые промежуточные таблицы, имитирующие типичный открытый интерфейс или структуру базовой таблицы,

3) Проверка и преобразование данных перед их загрузкой из указанного источника (дампы или плоские файлы),
4) Очистка данных Использование рекомендаций по макету файла JSON, чтобы отфильтровать неточные данные,
5) Используя рекомендации по файловой структуре JSON, удалим всю ненужную информацию,
6) Поместим данные в промежуточную область,
7) Обогащение данных,
8) Определение неадекватных данных по умолчанию,
9) Определение отсутствующих данные с помощью сопоставления или поиска,
10) Определение данных, которые структурированы по-другому

*Действия по извлечению данных.* В процессе извлечения данных в форматы файлов JSON будут выполняться следующие операции:

1) Выбор данных на основе файловой структуры JSON
2) Разработка приложений SQL с использованием формата файлов JSON
3) PL/SQL приложения или сценарии, создаваемые на основе процедур ETL и спецификаций отображения данных. Эти приложения будут использоваться для различных задач, таких как загрузка данных в промежуточные таблицы и стандартные таблицы с открытым интерфейсом.
4) Преобразование данных перед извлечением в соответствии со спецификациями отображения и макета файлов JSON.

*Загрузка данных.* Доступ к структурам данных NoSQL можно получить с помощью различных языков программирования, включая (.net, Java, Python, Ruby и др.). Используя эти языки программирования, данные можно загружать непосредственно из реляционных баз данных (таких как Access, SQL Server, Oracle, MySQL, и т.д.). В зависимости от правил принятия, уровня настройки и типа обработки данных для загрузки данных в структуры NoSQL могут использоваться специальные загрузчики.

## 5. Выводы

В этой статье основное внимание уделялось таксономии NoSQL с описанием его методов моделирования и миграции, а также причин и преимуществ использования NoSQL. Также было продемонстрировано, как NoSQL может эффективно использоваться для хранения больших объемов данных с отличной масштабируемостью.

Данная статья должна поспособствовать в принятии баз данных NoSQL там, где базы данных сильно расширяются по мере их роста.

Методы NoSQL предназначены для высокопроизводительных вычислений для создания отчетов и анализа. Многочисленные новые предприятия перенимают различные концепции NoSQL и выпускают на рынок свои собственные версии.

Решения для управления данными NoSQL обладают заманчивыми характеристиками производительности; однако они не могут полностью заменить системы управления реляционными базами данных. Выбор использования NoSQL не всегда прост.

Лучшая архитектура будет зависеть от потребностей разрабатываемых вами приложений, поскольку как SQL, так и NoSQL имеют свои преимущества и недостатки. Реляционная база данных SQL по-прежнему невероятно мощна и способна удовлетворить ваши потребности в транзакциях. Когда вы приближаетесь к предельным возможностям реляционных баз данных, а размер ваших данных или масштаб действий требуют более рассредоточенной системы, обратите внимание на возможности NoSQL.

**Список литературы**

1. Порохненко, Ю.С. Сравнительный анализ NoSQL баз данных / Ю.С. Порохненко, П.Н. Полежаев // актуальные направления научных исследований XXI века: теория и практика. – 2017. – № 9(35). – С. 25-30.
2. Рахимова, А.Р. NoSQL как механизм реализации ресурсного потенциала предприятия / А.Р. Рахимова // Информационные технологии в науке, управлении, социальной сфере и медицине Сборник научных трудов VI Международной научной конференции. / Под редакцией О.Г. Берестневой, В.В. Спицына, А.И. Труфанов, Т.А. Гладковой. – Томск: НИТПУ, 2019. – С. 588-591.
3. Industrial Internet of Things: Persistence for Time Series with NoSQL Databases/ Di Martino, S., Fiadone, L., Peron, A., Riccabone, A., & Vitale, V. N. // In 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 340-345). IEEE.
4. Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics / Ali, W., Shafique, M. U., Majeed, M. A., & Raza, A. // Asian Journal of Research in Computer Science. – 2019. – № 4(2). – P. 1-10.

5. Analysis of the Unexplored Security Issues Common to All Types of NoSQL Databases / Reddy, H.B.S., Reddy, R.R S., Jonnalagadda, R., Singh, P., Gogineni, A. // Asian Journal of Research in Computer Science. – 2022. – № 14(1). – P. 1-12.
6. Mongodb.com: сайт. - 2009. – URL: https://www.mongodb.com/docs/ (дата обращения 15.02.2023)