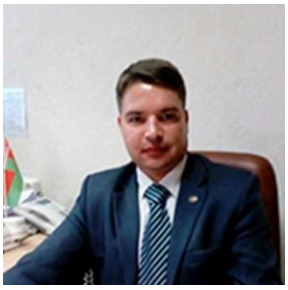
УДК 004.6

# NOSQL - РЕШЕНИЕ ПРОБЛЕМ ХРАНЕНИЯ БОЛЬШИХ ДАННЫХ

***И.В. Чибисов***
*Студент БГУИР*

***В.В. Шиманский***
*Ассистент кафедры Информатики БГУИР*

***И.А. Клапаток***
*Студент БГУИР*



***Ю.А.Чернявский***
*Доцент кафедры информатики.Доцент, кандидат технических наук.*

*Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь*
*E-mail: knigaykinjan@gmail.com, klapatok1980@gmail.com*

**Аннотация.** Из-за роста объемов данных и усложнения структуры хранимых данных потребовались новые методы построения инфраструктуры баз данных. Сейчас использование одного сервера является дорогостоящим и сложным. В связи с этим используют облачные хранилища данных. Также новые гибкие методы позволил обрабатывать запросы быстрее. Это достигается с помощью двух подходов: ручной шардинг и распределенный кэш. Таким образом появились системы NoSQL. Они имеют ряд преимуществ перед традиционными базами данных. NoSQL – общий термин, который описывает множество технологий, которые обладают общими характеристиками. NoSQL является общим решением некоторых проблем, которые возникают при организации хранения данных.

**Ключевые слова:** NoSQL, ручной шардинг, распределенный кеш, Динамические схемы, Авто-шардинг, Автоматическая репликация, Интегрированное кеширование.

Идея разработки реляционных баз данных состояла в том, чтобы обеспечить подход к хранению данных, который использует язык структурированных запросов или SQL [1]. Введение этих баз данных восходит к 1970-м годам, когда схемы данных были не такие сложные, как сегодня. Кроме того, хранение было дорогостоящей операцией, и не все данные считались архивными. С ростом социальных платформ объем хранящихся данных о событиях, объектах и людях вырос в геометрической прогрессии. Использование данных в наше время не ограничивается только архивированием данных, также необходим частый поиск и обработка данных, чтобы достигать такие цели, как генерирование в реальном времени индивидуальной подачи материала [2] и индивидуальных рекламных объявлений [3], в дополнение ко многим другим объявлениям и рекламам.

Из-за сложности обрабатываемой информации и необходимости обрабатывать несколько, порядка сотен, запросов к базе данных просто для ответа на один запрос API или

для отображения веб-страницы, требования к современной системе баз данных постоянно растут. Некоторые из ключевых факторов в этой области - это потребность в интерактивности, увеличении сложности и постоянное развитие сети пользователей [4]. Для удовлетворения этих растущих потребностей используются сложные стратегии развертывания и улучшенная вычислительная инфраструктура [7]. С учетом сказанного, использования одного сервера являются дорогостоящими и очень сложными, что вызвало сдвиг в сторону использования облачных аппаратных средств хранения [5] для этой цели. Помимо этого, использование гибких методов также сократило время разработки и развертывания [6], что позволяет быстрее реагировать на потребности пользователей.

Было бы неправильно утверждать, что реляционные базы данных не были созданы для управления требованиями гибкости и масштабируемости современных систем. Кроме того, они также не приспособлены для работы с облачным средством хранения и принимаю главное преимущество своего более дешевого хранения и возможностей обработки. Эти недостатки могут быть устранены с помощью двух основных технических подходов, которые обсуждаются ниже:

– ***Ручной шардинг***

Чтобы использовать распределенную парадигму, таблицы должны быть сегментированы на более мелкие единицы, которые затем должны храниться на разных машинах. Этот процесс расщепления называется ручным шардингом [8]. Однако эта функция недоступна в традиционной базе данных и должна быть реализована разработчиком. Кроме того, хранение данных в каждом экземпляре выполняется в анонимном режиме. Код приложения несет ответственность за сегментацию данных, их хранение в распределенном режиме, а также управление запросами и агрегирование результатов для представления пользователю. Дополнительный код необходим для поддержки перебалансировки данных, выполнения операций объединения, обработки сбоев ресурсов и репликации. Важно отметить, что ручное разбиение может снизить некоторые преимущества реляционных баз данных, таких как целостность транзакций

– ***Распределенный кэш***

Кэширование [9] является широко используемым процессом, который в основном используется для улучшения производительности чтения системы. Следует отметить, что использование кэша не влияет на производительность записи и способно существенно увеличить сложность всей системы. Поэтому, если требования к системе это интенсивное чтение, то следует рассмотреть использование распределенного кэша. С другой стороны, приложения, интенсивно использующие запись или чтение / запись, не требуют распределенного кэша [10].

Известно, что базы данных NoSQL [11] смягчают проблемы, связанные с традиционными базами данных. Кроме того, они также раскрывают истинную мощь облачных средств хранения, используя обычное аппаратное обеспечение, которое снижает стоимость и упрощает развертывание, делая жизнь разработчика намного проще, поскольку больше нет необходимости поддерживать несколько уровней кэша.

Некоторые из преимуществ решений NoSQL перед традиционными базами данных следующие:

– *Масштабируемость*

NoSQL позволяет системам масштабироваться горизонтально [11]. Более того, это можно сделать быстро, не влияя на общую производительность системы с помощью облачных технологий хранения данных. Масштабирование традиционных баз данных требует ручного разделения, что связано с большими затратами и сложностью. С другой стороны, решения NoSQL предлагают автоматическое разбиение, уменьшая сложность и стоимость системы [11].

– *Производительность*

Как упоминалось ранее, системы NoSQL можно масштабировать по мере необходимости. С увеличением количества систем производительность системы NoSQL также соответственно улучшается. Тот факт, что в этих системах используется автоматическое разбиение, означает, что накладные расходы, связанные с ним, также устраняются, что дополнительно способствует повышению производительности системы.

– *Высокая и глобальная доступность*

Реляционные базы данных зависят от первичного и вторичного узлов для выполнения требований доступности. Это не только увеличивает сложность системы, но и делает ее в меру доступной. Напротив, решения NoSQL используют архитектуру без хозяина, и данные распределяются по нескольким системам [11]. Следовательно, даже после сбоя узла доступность приложения остается неизменной как для операций чтения, так и для операций записи. Решения NoSQL предлагают репликацию данных между ресурсами [11]. Следовательно, пользовательский опыт является последовательным

независимо от местонахождения пользователя. Кроме того, это также играет важную роль в снижении задержки с дополнительным преимуществом переноса внимания разработчика с администрирования баз данных на бизнес-приоритеты.

– *Гибкое моделирование данных*

В NoSQL можно реализовать постоянно меняющиеся и гибкие модели данных [11]. Это позволяет разработчикам реализовывать параметры запросов и типы данных, которые подходят для приложения, а не те, которые соответствуют схеме. При этом взаимодействие между базой данных и приложением упрощается, что делает этот подход лучшим вариантом для гибкой разработки.

NoSQL – это общий термин, используемый для описания множества технологий, каждая из которых влечет за собой следующие общие характеристики [12]

***Динамические схемы***

Реляционные базы данных [13] имеют внутреннее требование заранее создавать схемы. Данные добавляются в базу данных только после выполнения этого требования. Например, если системе необходимо хранить данные о сотрудниках, такие как имя, отдел, возраст, пол и зарплата, то таблица, созданная для нее, должна иметь соответствующую схему.

Такое требование неприемлемо для гибкой среды разработки, поскольку поля данных могут со временем меняться. В качестве новой итерации может быть добавлено новое требование, и впоследствии, возможно, придется изменить схему. Это трудоемкая задача, если база данных велика. В результате база данных может быть заблокирована для любого использования в течение значительного периода времени для внесения необходимых изменений. Более того, если процесс разработки требует нескольких итераций, базу данных, возможно, придется блокировать довольно часто на значительное количество времени. Очевидно, что реляционные базы данных не подходят для хранения больших, неструктурированных и неизвестных данных [13].

NoSQL удовлетворяет этому требованию, поскольку не имеет предопределенных схем. Кроме того, для вставки данных не требуется, чтобы разработчик заблаговременно определял схему. В результате этого изменения в данных и структуре данных могут быть сделаны в режиме реального времени без необходимости блокировать базу данных для любого другого использования [12]. Есть несколько преимуществ использования этого подхода. Помимо того, что он сокращает время администратора, такой подход также уменьшает время, необходимое для разработки, и упрощает процесс интеграции кода.

***Авто-шардинг***

Реляционные базы данных структурированы таким образом, что им необходим сервер, который контролирует остальные системы, чтобы обеспечить требования к надежности и

доступности решения для базы данных. Поэтому такая система может поддерживать только вертикальное масштабирование [12], что не просто дорого, но приводит к появлению небольшого количества точек отказа. Помимо этого, это также накладывает ограничение на масштабирование, которое может поддерживать система.

С учетом системных требований решение для базы данных должно поддерживать горизонтальное масштабирование [11]. Следовательно, должна быть возможность добавить серверы в *ensemble* и избавиться от ограничения, которое фокусируется на тестировании мощности одного сервера. Облачные средства хранения предлагают лучшее решение в этом отношении, предоставляя услуги по требованию и неограниченные возможности масштабирования [14]. Таким образом, системе больше не нужно полагаться на один сервер для выполнения своих задач. Еще одним важным аспектом использования облачных средств является его встроенное администрирование базой данных. Более того, разработчику больше не нужно создавать сложные платформы, и он может просто сосредоточиться на написании кода приложения. И наконец, использование облачных многопользовательских серверов обходится значительно дешевле, чем использование одного сервера большой емкости.

Чтобы выполнить сегментирование базы данных, охватывающей несколько серверов, необходимо создать сложные механизмы, позволяющие нескольким серверам работать в одной системе. С другой стороны, базы данных NoSQL поддерживают авто-шардинг. Другими словами, база данных автоматически распределяет данные по нескольким системам без необходимости администратора знать о составе пула серверов. Балансировка нагрузки [15] для данных и запросов также автоматически выполняется системой. Это позволяет системе предлагать высокую доступность. Когда сервер выходит из строя, его можно легко заменить, а операции остаются без изменений.

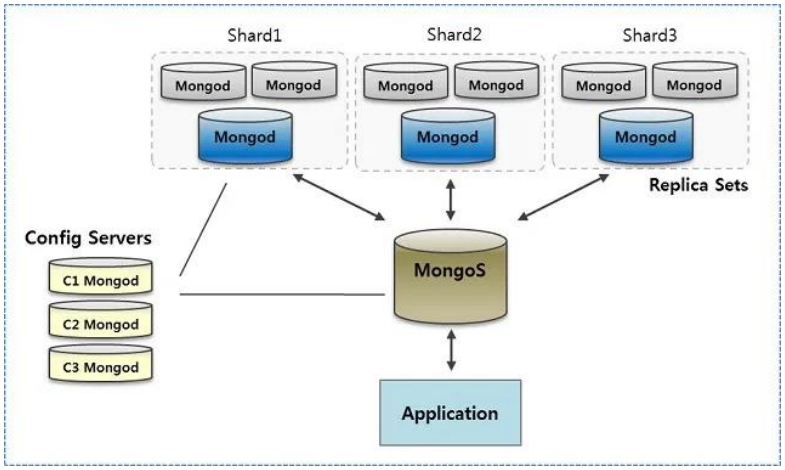



Рисунок *1*. – Авто-шардинг

**Автоматическая репликация**

Репликация выполняется автоматически для любой системы NoSQL [12]. Таким образом, система может достаточно легко восстанавливаться после аварий, что также обеспечивает высокую степень доступности. С точки зрения разработчика, программисту больше не нужно учитывать эти аспекты разработки в коде приложения.

**Интегрированное кеширование**

Интегрированные возможности кэширования [12] систем NoSQL достаточно хорошо оснащены, и большинство часто используемых данных хранятся в системной памяти для обеспечения быстрого доступа. Следовательно, нет необходимости поддерживать несколько уровней кэширования на уровне приложения.

Таким образом была рассмотрена система NoSQL, как вариант нереляционной системы баз данных. В целом такая система имеет широкое применение там, где необходима гибкая и динамично меняющаяся схема баз данных. В этом плане системы NoSQL имеют явное преимущество над стандартными реляционными базами данных. Также в качестве преимущества систем NoSQL можно выделить авто-шардинг, он делает возможным продолжение работы системы даже при сбое на одном из серверов, что в реляционных базах данных было бы явной проблемой и остановило бы все процессы на неопределенное время. Подводя итоги можно сказать, что в современном мире при увеличении сложности структуры данных, а также потребностей приложений в гибкости, такие системы, как NoSQL будут набирают популярность и всегда будут иметь место на рынке систем баз данных.

***Список литературы***

[1.] Köhler, H., & Link, S. (2018). SQL schema design: foundations, normal forms, and normalization. Information Systems, 76, 88-113.

[2.] Rathore, P., Rao, A. S., Rajasegarar, S., Vanz, E., Gubbi, J., & Palaniswami, M. (2018). Real-time urban microclimate analysis using internet of things. IEEE Internet of Things Journal, 5(2), 500-511.

[3.] Albayrak, N., Özdemir, A., & Zeydan, E. (2019, February). An Artificial Intelligence Enabled Data Analytics Platform for Digital Advertisement. In 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN) (pp. 239-241). IEEE.

[4.] Rao, T. R., Mitra, P., Bhatt, R., & Goswami, A. (2018). The big data system, components, tools, and technologies: a survey. Knowledge and Information Systems, 1-81.

[5.] Stergiou, C., Psannis, K. E., Kim, B. G., & Gupta, B. (2018). Secure integration of IoT and cloud 28 computing. Future Generation Computer Systems, 78, 964-975.

[6.] Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., & Ebdrup, A. (2018, May). From Scrum to Agile: a journey to tackle the challenges of distributed development in an Agile team. In Proceedings of the 2018 International Conference on Software and System Process (pp. 11-20). ACM.

[7.] Khan, S., Shakil, K. A., Alam, M. (2017). Big Data Computing Using Cloud-Based Technologies: Challenges and Future Perspectives In: Networks of the Future: Architectures, Technologies, and Implementations. Chapman and Hall/CRC.

[8.] Cordeiro, J. R., & Postolache, O. (2018, September). Big Data Storage for a Health Predictive System. In 2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)(pp. 1-6). IEEE.

[9.] Zhao, J., Lai, M., Tian, H., & Chang, Y. (2019). U.S. Patent Application No. 10/205,673. [10] Reddy, K. S., Moharir, S., & Karamchandani, N. (2018, May). Effects of storage heterogeneity in distributed cache systems. In 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) (pp. 1-8). IEEE.

[10.] Strauch, C., Sites, U. L. S., & Kriha, W. (2011). NoSQL databases. Lecture Notes, Stuttgart Media University, 20

[11.] Acharya, B., Jena, A. K., Chatterjee, J. M., Kumar, R., & Le, D. N. (2019). NoSQL Database Classification: New Era of Databases for Big Data. International Journal of Knowledge-Based Organizations (IJKBO), 9(1), 50-65.

[12.] Batra, R. (2018). A History of SQL and Relational Databases. In SQL Primer (pp. 183-187). Apress, Berkeley, CA.

[13.] Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. Future Generation Computer Systems, 79, 849-861.

[14.] Ragmani, A., El Omri, A., Abghour, N., Moussaid, K., & Rida, M. (2018). A performed load balancing algorithm for public Cloud computing using ant colony optimization. Recent Patents on Computer Science, 11(3), 179-195

# NOSQL – A SOLUTION FOR BIG DATA STORAGE ISSUES

***I.V. Chybisov***
*Student of the BSUIR*

***V.V. Shimanskiy***
*Assistant of the Department of Informatics of the BSUIR*

***I.A. Klapatok***
*Student of the BSUIR*

***Y.A.Cherniavskiy***
*Associate Professor at the Department of Informatics.*
*Associate Professor, Candidate of Technical Sciences.*

*Belarusian State University of Informatics and Radioelectronics, Republic of Belarus,*
*E-mail: knigaykinjan@gmail.com, klapatok1980@gmail.com*

**Abstract.** Due to the growth of data volumes and the complexity of the structure of stored data, new methods for building a database infrastructure were required. Using one server is expensive and complicated right now. In this regard, they use cloud data storage. Also, new flexible methods allowed processing requests faster. This is achieved using two approaches: manual sharding and distributed cache. Thus, NoSQL systems appeared. They have several advantages over traditional databases. NoSQL is a generic term that describes many technologies that share common characteristics. NoSQL is a common solution to some of the problems that arise when organizing data storage.

**Keywords:** NoSQL, Manual Sharding, Distributed Cache, Dynamic Schemas, Auto-Sharding, Automatic Replication, Integrated Caching.