ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК004.021, 004.42

DOI: 10.21822/2073-6185-2024-51-1-87-94 Оригинальная статья/ Original article

Исследование эффективности методов компрессии данных в реляционных и NoSQL СУБД

В.А. Егунов, В.С. Сурин, П.С. Ступницкий, Р.Д. Ахметова
Волгоградский государственный технический университет,
400005, г.Волгоград, проспект им.Ленина, 28, Россия

Резюме. Цель. Исследование, представленное в работе, направлено на изучение эффективности современных СУБД. **Метод.** Наряду с традиционными реляционными решениями все большую популярность приобретают СУБД на основе NoSQL. В первую очередь это связано с возможностью подобных систем хранить и обрабатывать огромные объемы данных. При этом для эффективной работы с этими данными необходимо обеспечить их сжатие (компрессию). Компрессия позволяет сократить объем хранимых данных и обеспечить быстрый доступ к ним. **Результат.** Выполнен сравнительный анализ компрессии данных в СУБД MySQL и OpenSe. **Вывод.** Компрессия данных в OpenSearch более эффективна, чем в MySQL при использовании стандартных алгоритмов. OpenSearch будет отличным выбором при необходимости сохранить ресурсы хранилища, оптимально используя процессор, в противном случае обе системы хорошо справятся с поставленной задачей.

Ключевые слова: системы управления базами данных, СУБД, реляционные СУБД, NoSQL, компрессия данных, zlib.,lz4, deflate

Для цитирования: В.А. Егунов, В.С. Сурин, П.С. Ступницкий, Р.Д. Ахметова. Исследование эффективности методов компрессии данных в реляционных и NoSQL СУБД. Вестник Дагестанского государственного технического университета. Технические науки. 2024; 51(1):87-94. DOI:10.21822/2073-6185-2024-51-1-87-94

Research on the effectiveness of data compression methods in relational and NoSQL DBMS

V.A. Egunov, V.S. Surin, P.S. Stupnitskiy, R.D. Akhmetova
Volgograd State Technical University,
28 Lenin Ave., Volgograd 400005, Russia

Abstract. Objective. The research presented in the paper is aimed at studying the effectiveness of modern DBMSs. **Method.** Along with traditional relational solutions, NoSQL-based DBMSs are becoming increasingly popular. This is primarily due to the ability of such systems to store and process huge volumes of data. At the same time, to work effectively with this data, it is necessary to ensure its compression. Compression allows you to reduce the amount of stored data and provide quick access to it. **Result.** A comparative analysis of data compression in the MySQL and OpenSe DBMS was performed. **Conclusion.** Data compression in OpenSearch is more efficient than in MySQL when using standard algorithms. OpenSearch is an excellent choice if you need to conserve storage resources while making optimal use of the processor, otherwise both systems will do the job well.

Keywords: database management systems, DBMS, relational DBMS, NoSQL, data compression, zlib.,lz4, deflate.

For citation: V.A. Egunov, V.S. Surin, P.S. Stupnitskiy, R.D. Akhmetova. Research on the effectiveness of data compression methods in relational and NoSQL DBMS. Herald of Daghestan State Technical University. Technical Sciences. 2024; 51(1):87-94. DOI:10.21822/2073-6185-2024-51-1-87-94

Введение. Проблема повышения производительности вычислительных систем всегда привлекала внимание исследователей. Данная проблема приобрела особую актуальность в последние годы вместе с резким ростом количества обрабатываемых данных, увеличением сложности решаемых задач. Одним из способов решения данной проблемы является повышение эффективности программного обеспечения [1]. Данная тематика достаточно широко освещается в научной литературе.

Для повышения эффективности компьютерных программ могут применяться различные методы, такие как распараллеливание и векторизация вычислений [2], изменение стратегии кэширования данных [3], распределение нагрузки между узлами гетерогенной вычислительной системы [4]. Вместе с ростом объемов хранимых и обрабатываемых данных потребность в эффективном хранении и ускорении их обработки также увеличилась.

Постановка задачи. В настоящее время все большую популярность приобретают системы управления базами данных (СУБД) на основе NoSQL (not only SQL). В первую очередь это связано с возможностью данных систем хранить и обрабатывать огромные объемы данных. При этом для эффективной работы с этими данными необходимо обеспечить их сжатие (компрессию).

Компрессия позволяет сократить объем хранимых данных и обеспечить быстрый доступ к ним. Это особенно важно для баз данных, где данные хранятся на диске и читаются в оперативную память при запросах. Сжатие данных также позволяет уменьшить нагрузку на сеть при передаче данных между узлами кластера. Однако, компрессия данных при работе с NoSQL СУБД может стать проблемой. Некоторые NoSQL СУБД, например Cassandra, не поддерживают сжатие данных непосредственно на уровне базы данных. В таких случаях необходимо проводить компрессию данных на уровне приложений. Это может привести к увеличению нагрузки на процессор и меньшей эффективности по сравнению с сжатием данных на уровне базы данных.

Методы исследования. В статье будет выполнен сравнительный анализ компрессии данных в MySQL и OpenSearch, который в результате может быть полезным для принятия обоснованных решений при выборе систем управления базами данных при необходимости эффективного использования ресурсов хранилища. Данный анализ был проведен в процессе работы над программной системой управления мероприятиями. Исходя из особенностей разрабатываемой системы, были использованы реляционная СУБД MySQL и NoSQL СУБД OpenSearch.

1. Современные СУБД. Традиционно базы данных делят на два класса: реляционные – SQL и нереляционные - NoSQL (рис.1). При этом стоит отметить, что существуют и другие критерии классификации.

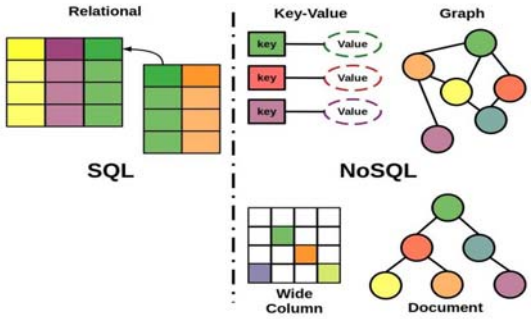

Рис.1. Реляционные и нереляционные базы данных
Fig.1. Relational and non-relational databases

Реляционная база данных (реляционная система управления базами данных, РСУБД) представляет собой набор отношений (сущностей, таблиц), множества кортежей для каждого отношения (строк), множества атрибутов для каждого кортежа (столбцов). При этом между сущностями могут быть отношения различных типов.

РСУБД выполняют требования транзакционной целостности ACID: атомарность (atomicity), согласованность (consistency), изолированность (isolation) устойчивость (durability). Выполнение данных требований обеспечивает надежную и предсказуемую работу системы. Реляционные базы данных имеют фиксированную структуру, которая получается в процессе нормализации, т.е. процесса структурирования базы данных в соответствии с последовательностью нормальных форм.

Для создания, модификации и управления данными в РСУБД используется Structured Query Language (SQL). SQL представляет собой мощный язык запросов, позволяющий создавать сложные запросы, выполняющие агрегацию данных из множества таблиц. Обработка сложных запросов, выполнение требований ACID приводит к тому, что РСУБД перестают удовлетворять требованиям к СУБД, обрабатывающим большие объемы данных в силу своей относительно невысокой производительности. Стоит также отметить, что РСУБД масштабируются вертикально (за счет увеличения мощности «железа») и практически не масштабируются горизонтально (за счет увеличения числа узлов).

Как альтернатива РСУБД активно развиваются NoSQL СУБД. NoSQL не означает конкретную технологию и включает в себя широкий спектр программных продуктов. Общим свойством данных СУБД является частичный или полный отказ от принципов ACID в пользу BASE:

- basic availability; базовая доступность – каждый запрос гарантированно завершается, успешно или нет (без гарантии, что будет получена последняя версия данных);
- soft state; гибкое состояние – состояние системы может изменяться со временем, даже без ввода новых данных для достижения согласования данных (репликация, eventual consistency);
- eventual consistency; согласованность в конечном счете – данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время (один процесс редактирует, данные не блокируются, другой читает «старые данные»).

Кроме того, для NoSQL баз данных характерен отказ от строгой структуры данных, структура данных может изменяться, более того, хранимые данные могут быть неструктурированными. Подобные СУБД хорошо масштабируются горизонтально, лучше подходят для распределенного хранения данных и широко используются в системах обработки больших данных.

2. Компрессия данных в современных СУБД.

Как уже было сказано выше, данное исследование было проведено в рамках работы над программной системой управления мероприятиями, где были использованы реляционная СУБД MySQL и NoSQL СУБД OpenSearch. Реляционная СУБД использовалась для хранения данных, т.к. ввиду особенностей проектируемой системы данная функция была нереализуема в NoSQL. Однако отказываться от всех преимуществ NoSQL, а это полнотекстовый поиск, обширный набор агрегаций и т.д., было нерационально. Именно поэтому часть данных реплицируется в OpenSearch.

MySQL является одной из наиболее популярных реляционных баз данных, которая применяется во многих областях. Преимущества MySQL:

1. Широкий функционал: MySQL обладает широким спектром функций, позволяя работать с большим количеством данных. Она может обрабатывать большие объемы данных в многопользовательском режиме.
2. MySQL поддерживает множество языков программирования, поэтому внедрение ее в сервис или приложение не составит труда.
3. Как и большинство других реляционных баз данных, MySQL имеет возможность контроля доступа и управления привилегиями пользователей.

Кроме описанных выше общих недостатков РСУБД, MySQL присущи собственные недостатки:

1. Настройка MySQL может быть довольно сложной и затратной задачей. Она может требовать дополнительных знаний и инструментов, чтобы получить максимальную производительность.
2. MySQL может использовать довольно большое количество ресурсов для обработки больших объемов данных. Это может привести к снижению производительности при работе с нагрузкой.
3. MySQL масштабируется хуже других новых технологий и не может обрабатывать такое большое количество данных, как NoSQL и другие базы данных новых поколений.

MySQL поддерживает несколько типов таблиц, в частности ISAM и InnoDB, однако в настоящее время рекомендуется использовать таблицы InnoDB. В таблицах InnoDB используется два вида компрессии данных [5]: Compressed row format (ROW_FORMAT); Page compression (PAGE_COMPRESSED).

Первый тип компрессии позволяет сократить размер данных, сохраненных в таблице, до 50%. Это достигается за счет использования алгоритма сжатия данных zlib [6, 7]. Как правило, этот тип компрессии используется для таблиц, с большим количеством повторяющихся данных [8].

Второй тип компрессии позволяет значительно сократить размер индексов таблицы, сохраненных на диске, без потери производительности при выполнении запросов. Это достигается за счет использования алгоритма сжатия данных LZ4 [9].

Выбор типа компрессии данных зависит от характеристик данных, хранящихся в таблице, а также от требований к производительности системы. Рекомендуется экспериментировать с различными значениями ROW_FORMAT и PAGE_COMPRESSED и выбрать то, что лучше всего соответствует конкретной ситуации.

В качестве NoSQL СУБД использовалась OpenSearch. OpenSearch представляет собой систему поискового сервера, созданную как форк Elasticsearch. Преимущества OpenSearch:

1. OpenSearch – это форк Elasticsearch, который имеет преимущества оригинального проекта и позволяет легко переключаться между двумя системами, что делает ее популярным решением, особенно для пользователей, уже знакомых с Elasticsearch.
2. OpenSearch является проектом с открытым исходным кодом, что позволяет пользователям выбирать и изменять их собственным образом, а также разрабатывать плагины и надстройки.
3. OpenSearch разработана для масштабирования и может работать с большим количеством данных, обеспечивая высокую производительность и надежность.

К недостаткам OpenSearch можно отнести следующее :

1. OpenSearch может быть сложным и требовать знания языка запросов для эффективного использования. Это делает его менее доступным для новых пользователей.
2. OpenSearch имеет меньшее количество полезных плагинов и инструментов, чем Elasticsearch. Может потребоваться больше усилий, чтобы настроить и оптимизировать OpenSearch для своих нужд.
3. Сообщество OpenSearch является новым и менее развитым, чем сообщество Elasticsearch. Это может означать, что пользователи могут столкнуться с проблемами документации или отсутствием ответов на интересующие их вопросы.

OpenSearch предлагает два варианта сжатия [10]: default; best_compression.

Default указывает OpenSearch использовать блоки по 16 КБ, сжатые с помощью LZ4, а best_compression — использовать блоки по 60 КБ, сжатые с помощью DEFLATE [11].

LZ4 представляет собой алгоритм сжатия данных без потерь, ориентированный на скорость сжатия и распаковки. Он принадлежит к семейству байт-ориентированных схем сжатия LZ77 [12]. Алгоритмы LZ4 призваны обеспечить хороший компромисс между скоростью и степенью сжатия. Как правило, он имеет меньшую (т.е. худшую) степень сжатия, чем аналогичный алгоритм LZO [13], который, в свою очередь, хуже, чем алгоритмы типа

DEFLATE. Однако скорость сжатия LZ4 аналогична LZO и в несколько раз выше, чем у DEFLATE, а скорость распаковки значительно выше, чем у LZO.

Deflate представляет собой формат файла сжатия данных без потерь, который использует комбинацию LZ77 и кодирования Хаффмана. Он был разработан Филом Кацем (Phil Katz) для версии 2 его инструмента архивирования PKZIP [14].

Важным компонентом этих двух алгоритмов сжатия является дедупликация строк. Всякий раз при обнаружении строки, которая уже встречалась ранее в потоке, эта строка будет заменена ссылкой на предыдущее появление этой строки. На самом деле это единственное, что делает LZ4, в то время как DEFLATE сочетает дедупликацию строк с кодированием Хаффмана для дальнейшего сжатия данных.

Для выполнения анализа эффективности компрессии в MySQL и OpenSearch был сгенерирован набор данных в 2 миллиона записей. Загрузка в обе системы была итеративной по 500 тысяч записей. На каждом этапе производилась фиксация размера таблицы/индекса. Для каждой системы загрузка осуществлялась дважды, OpenSearch:

1. В индекс с настроенным сжатием LZ4
2. В индекс с настроенным сжатием DEFLATE

Для MySQL:

1. Таблица с сжатием LZ4
2. Таблица с сжатием ZLIB

Обе системы были развернуты и настроены с помощью Docker. Используемое программное обеспечение: MySQL 8.0.33 (arm64v8); OpenSearch 2.7.0 (arm64v8); Docker 4.9.0 (arm64v8). Датасет представлял из себя следующую структуру: id – уникальный идентификатор пользователя (целое число); name – имя пользователя (строка); savings – сбережения пользователя (целое число).

Обсуждение результатов. В результате выполнения вычислительных экспериментов были получены следующие данные (табл. 1).

Таблица 1. Полученные размеры хранимых данных
Table 1. Resulting sizes of stored data

Объем записей Record volume	OpenSearch LZ4 (МБ)	MySQL LZ4 (МБ)	OpenSearch DEFLATE (МБ)	MySQL ZLIB (МБ)
500 тысяч	12.6	19.5	9.9	10.3
1 млн	25.2	36.6	20.1	19.8
1.5 млн	38.4	54.6	30.4	30.3
2 млн	52.1	75.6	40.7	41.3

На основании полученных данных были построены графики зависимости размера хранимых данных от объема данных. Ось X – количество записей, ось Y – размер в мегабайтах. На рис. 1 и 2 представлено сравнение эффективности применения алгоритма LZ4 в обеих системах.

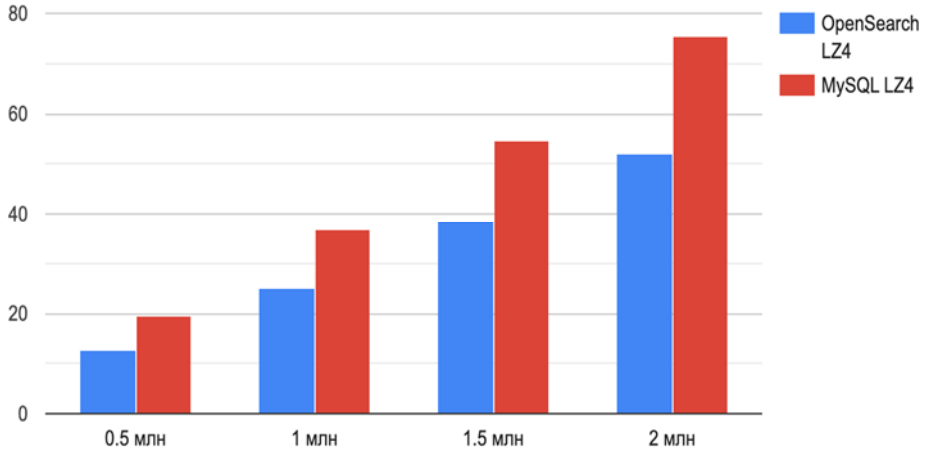

Рис. 1. Эффективность алгоритма LZ4 (столбчатая диаграмма)
Fig. 1. Efficiency of LZ4 algorithm (bar chart)

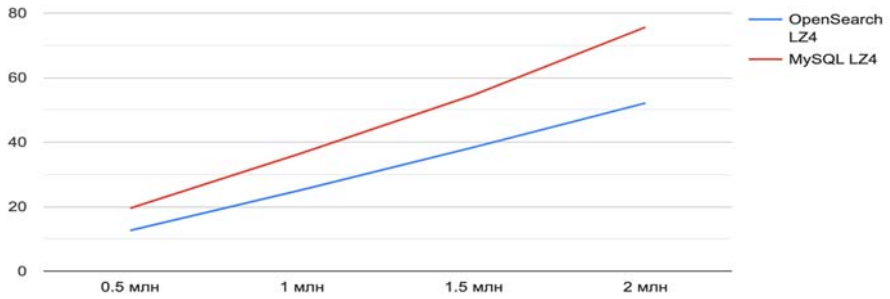

Рис. 2. Эффективность алгоритма LZ4 (линейная диаграмма)
Fig. 2. Efficiency of the LZ4 algorithm (line diagram)

На рис. 3 и 4 представлено сравнение эффективности применения продвинутых алгоритмов сжатия (ZLIB и DEFLATE) в обеих системах.

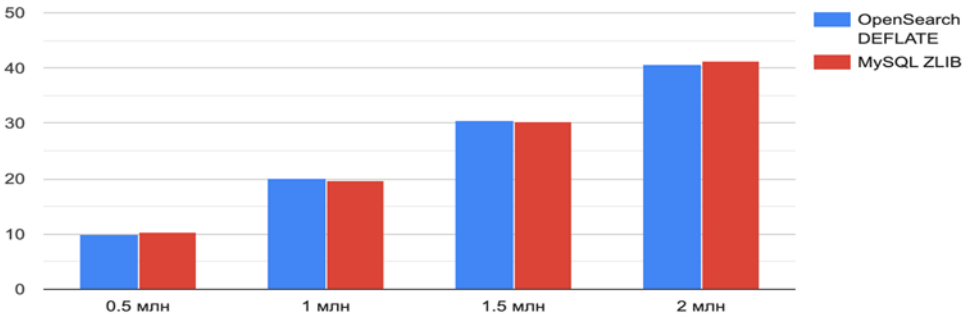

Рис. 3. Эффективность продвинутых алгоритмов (столбчатая диаграмма)
Fig. 3. Efficiency of advanced algorithms (bar chart)

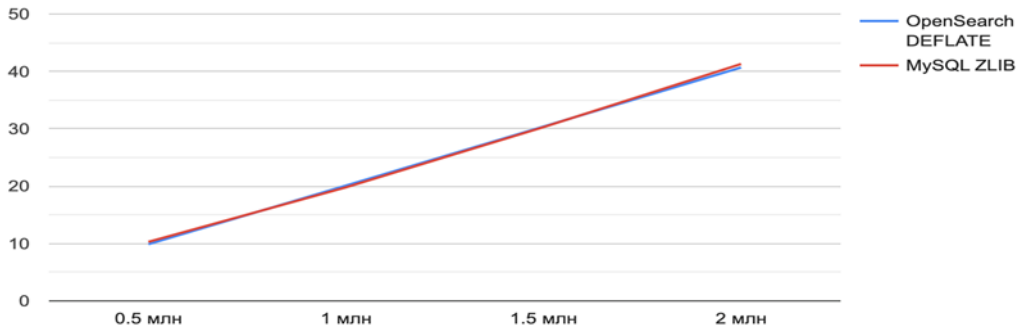

Рис. 4. Эффективность продвинутых алгоритмов (линейная диаграмма)
Fig. 4. Efficiency of advanced algorithms (line chart)

На рис. 5 представлено полное сравнение примененных алгоритмов сжатия.

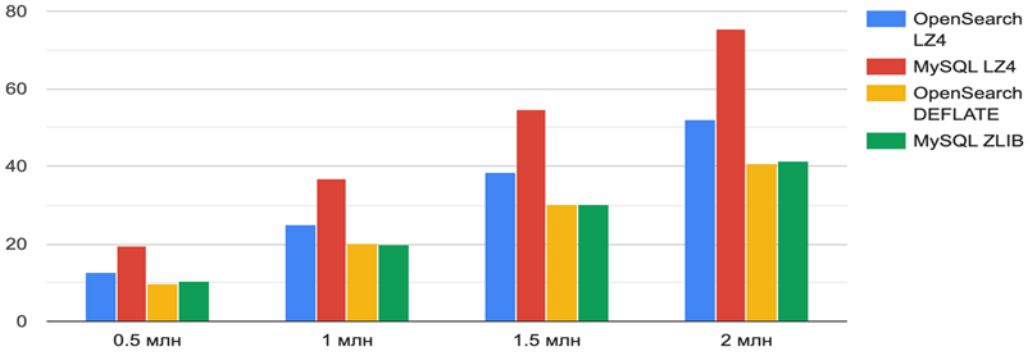

Рис. 5. Эффективность алгоритмов (столбчатая диаграмма)
Fig. 5. Algorithm efficiency (bar chart)

Вывод. По полученным данным можно сделать следующие выводы:

1. OpenSearch лучше работает с алгоритмом сжатия LZ4, чем MySQL.
2. Размер сжатых данных растет линейно, при этом MySQL к двум миллионам записей начал хуже справляться с сжатием. Это означает, что для больших объемов данных предпочтительнее использовать OpenSearch.
3. Алгоритмы DEFLATE и ZLIB работают одинаково хорошо и в OpenSearch и в MySQL.

Общая картина показывает достаточную эффективность применения алгоритмов компрессии DEFLATE и ZLIB.

Таким образом, можно утверждать, что компрессия данных в OpenSearch более эффективна, чем в MySQL при использовании стандартных алгоритмов. Но системы схожи при использовании более продвинутых способов сжатия. При этом стоит учитывать, что использование вторых интенсивнее нагружает процессор, так как процесс декомпрессии сложнее, чем у стандартных алгоритмов, поэтому можно сделать вывод, что OpenSearch будет отличным выбором при необходимости сохранить ресурсы хранилища, оптимально используя процессор, в противном случае обе системы хорошо справятся с поставленной задачей.

Библиографический список:

1. Low, T.M., Igual, F.D., Smith, T.M., Quintana-Orti, E.S.: Analytical modeling is enough for high-performance BLIS. ACM Trans. Math. Softw. 43, 1–18 (2016).https://doi.org/10.1145/2925987
2. Егунов, В.А. Повышение эффективности векторизации вычислений / В.А. Егунов, А.Г. Кравец // Математические методы в технологиях и технике. - 2023. - № 3. - С. 65-68. - DOI: 10.52348/2712-8873_MMTT_2023_3_65.
3. Егунов, В.А. Метод улучшения стратегии кеширования для вычислительных систем с общей памятью / В.А. Егунов, А.Г. Кравец // Программная инженерия. - 2023. - Т. 14, № 7. - С. 329-338. - DOI: 10.17587/prin.14.329-338
4. Бычков И. В. Поддержка вычислений в распределенных средах на основе непрерывной интеграции / И. В. Бычков, С. А. Горский, А. Г. Феоктистов, Р. О. Костромин // Информационные технологии. – 2021. – Т. 27. – № 12. – С. 619-625. – DOI 10.17587/it.27.619-625. – EDN QSGWAI.
5. Краткий обзор движков таблиц MySQL // habr.com – Режим доступа : https://habr.com/ru/articles/64851/ (дата обращения 20.09.2023)
6. Zlib 1.3 Manual // zlib.net – Режим доступа: https://zlib.net/manual.html (дата обращения 20.09.2023)
7. Koranne, S. (2011). Compression Engines. In: Handbook of Open Source Tools. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-7719-9_8
8. Черняк Л., Большое хранилище для больших данных, Л. Черняк // Открытые системы. СУБД. 2010. № 5. С. 60.
9. Новый алгоритм сжатия LZ4 от TOAST в PostgreSQL 14. Насколько быстрым он может быть? // shunlongwei.com – Режим доступа : https://www.shunlongwei.com/ru/toasts-new-compression-algorithm-lz4-in-postgresql-14-how-fast-can-it-be/ (дата обращения 20.09.2023)
10. Save space and money with improved storage efficiency in Elasticsearch 7.10 // www.elastic.co – Режим доступа : https://www.elastic.co/blog/save-space-and-money-with-improved-storage-efficiency-in-elasticsearch-7-10 (дата обращения 20.09.2023)
11. Алгоритм Deflate на примере формата PNG // habr.com – Режим доступа : https://habr.com/ru/articles/274825/ (дата обращения 20.09.2023)
12. Yan, H., Lu, H., Gao, Q. (2012). A BP-LZ77 Compression Algorithm Based on BP Network. In: Jin, D., Lin, S. (eds) Advances in Electronic Engineering, Communication and Management Vol.2. Lecture Notes in Electrical Engineering, vol 140. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27296-7_34
13. LZO use and presentation // Programmer AI – Режим доступа: https://programmerall.com/article/99471030002/ (дата обращения 20.09.2023)
14. PKZIP // PKWARE – Режим доступа: https://www.pkware.com/products/pkzip/ (дата обращения 20.09.2023)

References

1. Low, T.M., Igual, F.D., Smith, T.M., Quintana-Orti, E.S.: Analytical modeling is enough for high-performance BLIS. ACM Trans. Math. Softw. 2016; 43:1–18.https://doi.org/10.1145/2925987
2. Egunov, V.A. Povyshenie effektivnosti vektorizacii vychislenij [Improving the efficiency of vectorization of calculations]/ V.A. Egunov, A.G. Kravec. Matematicheskie metody v tekhnologiyah i tekhnike [Mathematical methods in technology and engineering]. 2023;3:65-68. - DOI: 10.52348/2712-8873_MMTT_2023_3_65. (In Russ)
3. Egunov, V.A. Metod uluchsheniya strategii keshirovaniya dlya vychislitel'nyh sistem s obshchej pamyat'yu [A method for improving the caching strategy for computing systems with shared memory] / V.A. Egunov,

A.G. Kravec. *Programmnaya inzheneriya [Software Engineering]*. 2023;14(7):329-338. - DOI: 10.17587/prin.14.329-338.(In Russ)

4. Bychkov I. V. Podderzhka vychislenij v raspredelennyh sredah na osnove nepreryvnoj integracii [Support for computing in distributed environments based on continuous integration] / I. V. Bychkov, S. A. Gorskij, A. G. Feoktistov, R. O. Kostromin // Informacionnye tekhnologii [Information technology]. 2021; 27(12): 619-625. – DOI 10.17587/it.27.619-625. – EDN QSGWAI. (In Russ)
5. Kratkij obzor dvizhkov tablic MySQL [A brief overview of MySQL table engines] // habr.com – Rezhim dostupa [Access mode] : https://habr.com/ru/articles/64851/ (data obrashcheniya [date of application] 20.09.20273) .(In Russ)
6. Zlib 1.3 Manual // zlib.net – Rezhim dostupa [Access mode]: https://zlib.net/manual.html (data obrashcheniya [date of application] 20.09.2023)
7. Koranne, S. (2011). Compression Engines. In: Handbook of Open Source Tools. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-7719-9_8
8. CHernyak L., Bol'shoe hranilishche dlya bol'shih dannyh [Large storage for big data], L. CHernyak // Otkrytye sistemy. SUBD. [Open systems. DBMS.] 2010;5: 60. (In Russ)
9. Novyj algoritm szhatiya LZ4 ot TOAST v PostgreSQL 14. Naskol'ko bystrym on mozhet byt'? [TOAST's new LZ4 compression algorithm in PostgreSQL 14. How fast can it be?] // shunlongwei.com – Rezhim dostupa [Access mode] : https://www.shunlongwei.com/ru/toasts-new-compression-algorithm-lz4-in-postgresql-14-how-fast-can-it-be/ (data obrashcheniya [date of application] 20.09.2023) (In Russ)
10. Save space and money with improved storage efficiency in Elasticsearch 7.10 // www.elastic.co – Rezhim dostupa [Access mode]:https://www.elastic.co/blog/save-space-and-money-with-improved-storage-efficiency-in-elasticsearch-7-10 (data obrashcheniya [date of application] 20.09.2023)
11. Algoritm Deflate na primere formata PNG [Deflate algorithm on the example of PNG format] // habr.com – Rezhim dostupa [Access mode] : https://habr.com/ru/articles/274825/ (data obrashcheniya [date of application] 20.09.2023) (In Russ)
12. Yan, H., Lu, H., Gao, Q. (2012). A BP-LZ77 Compression Algorithm Based on BP Network. In: Jin, D., Lin, S. (eds) Advances in Electronic Engineering, Communication and Management Vol.2. Lecture Notes in Electrical Engineering, vol 140. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27296-7_34
13. LZO use and presentation//Programmer AI – Rezhim dostupa [Access mode]: https://programmerall.com/article/99471030002/ (data obrashcheniya [date of application] 20.09.2023)
14. PKZIP // PKWARE – Rezhim dostupa [Access mode]: https://www.pkware.com/products/pkzip/ (data obrashcheniya [date of application] 20.09.2023)

Сведения об авторах:

Егунов Виталий Алексеевич, кандидат технических наук, доцент, кафедра ЭВМ и систем; vegunov@mail.ru

Сурин Валерий Сергеевич, магистрант, кафедра ЭВМ и систем; surin190899@gmail.com

Ступницкий Павел Сергеевич, магистрант, кафедра ЭВМ и систем; p.stupnitskiy@gmail.com

Ахметова Руфина Джигировна, магистрант, кафедра ЭВМ и систем; rufina_akhmetova@list.ru

Information about the authors:

Vitaly A. Egunov, Cand. Sci. (Eng.), Assoc. Prof., Computers and Systems Department; vegunov@mail.ru

Valeriy S. Surin, Master Student, Computers and Systems Department; surin190899@gmail.com

Pavel S. Stupnitskiy, Master Student, Computers and Systems Department; p.stupnitskiy@gmail.com

Rufina D. Akhmetova, Master Student, Computers and Systems Department, rufina_akhmetova@list.ru

Конфликт интересов/Conflict of interest.

Авторы заявляют об отсутствии конфликта интересов/The authors declare no conflict of interest.

Поступила в редакцию/ Received 25.12.2023.

Одобрена после рецензирования / Reviced 20. 01.2024.

Принята в печать /Accepted for publication 20.01.2024.