Вестник НГУЭУ. 2023. № 3. С. 144–153
Vestnik NSUEM. 2023. No. 3. P. 144–153

Научная статья
УДК 004.65
DOI: 10.34020/2073-6495-2023-3-144-153

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ И МЕТОДОВ ХРАНЕНИЯ JSON-ОБЪЕКТОВ В СУБД POSTGRESQL

Широков Иван Алексеевич

Университет ИТМО

ivan.a.shirokov@ya.ru

Аннотация. На сегодняшний день существует несколько методов хранения данных, таких как табличное и объектное, в виде JSON-объектов. Отечественными и зарубежными учеными были проведены исследования по сравнению производительности реляционных СУБД и NoSQL СУБД, однако не было работ, которые сравнивали производительность при вставке и хранении данных с использованием обычных записей и JSON-объектов внутри реляционной СУБД. На примере реальных случаев корпоративного использования СУБД был проведен эксперимент, результаты которого показывают релевантность использования JSON-объектов при хранении большого объема данных.

Ключевые слова: СУБД, PostgreSQL, хранение данных, JSON, производительность

Для цитирования: Широков И.А. Исследование производительности и методов хранения JSON-объектов в СУБД PostgreSQL // Вестник НГУЭУ. 2023. № 3. С. 144–153. DOI: 10.34020/2073-6495-2023-3-144-153.

Original article

THE STUDY OF PERFORMANCE AND METHODS OF STORAGE OF JSON OBJECTS IN POSTGRESQL DATABASE MANAGEMENT SYSTEM

Shirokov Ivan A.

ITMO University

ivan.a.shirokov@ya.ru

Currently there are several methods of data storage, such as table and object, as JSON objects. National and foreign scientists conducted studies on the comparison of performance of relational database management systems and NoSQL database management system, but there were no studies that compared the performance in data insertion and storage using ordinary records and JSON objects within the relational database management system. An experiment was carried out in terms of the real examples of corporate use of the database

management system; the results show the relevance of the use of JSON objects in storage of large volumes of data.

Ключевые слова: database management system, PostgreSQL, data storage, JSON, productivity

For citation: Shirokov I.A. The study of performance and methods of storage of JSON objects in PostgreSQL database management system. *Vestnik NSUEM*. 2022; (3): 144–153. (In Russ.). DOI: 10.34020/2073-6495-2023-3-144-153.

Для транспортировки данных организации, в том числе финансового сектора, используют различные методы хранения, которые позволяют целостно переносить большие объемы данных. Сегодня в СУБД используются различные методики хранения, такие как таблицы и объекты. Использование объектов обуславливается тем, что при разработке, например, frontend компонентов проекта необходимо хранить данные в виде ключ-значения, что позволяет быстро менять значения состояния компонента. Главным приоритетом компаний является получение большего количества данных за меньшее количество занимаемого пространства на хранилище. Следуя к цели уменьшения занимаемого пространства, компании часто прибегают к методикам, которые нацелены, как минимум, уменьшить количество записей в таблицах. Однако уменьшение количества записей за счет увеличения количества атрибутов или более глубокого описания конкретной записи не приносит желаемый результат, так как может возникнуть дополнительная проблема в виде доступности данных или их ненормализованность. Одной из проблем доступности является извлечение данных из объектов, например JSON-объектов. Такое извлечение влечет за собой более сложную структуру запроса. В случае возникновения ненормализованности данных, может произойти дублирование или исчезновение данных из запроса, что соответствует нарушению целостности данных.

В статье будут рассмотрены реальные случаи, которые используются в компаниях с целью определения релевантного метода хранения данных. Объекты как вариант переноса данных применяются в некоторых банках данных для хранения характеристик пользователей. Однако нет определенного ответа на вопрос об эффективности такой методики хранения непосредственно данных.

В работах отечественных и зарубежных ученых отмечается, что NoSQL базы данных на примере MongoDB [1, 2] работают в ряде сценариев быстрее, чем аналоги в виде СУБД MS SQL [6] и PostgreSQL [3].

Однако в работах [3, 4, 6] авторы сравнивали производительность разных архитектур СУБД. В PostgreSQL реализована возможность хранения данных в виде JSON, что позволяет провести прямой эксперимент по возможностям данного типа данных в СУБД.

Объекты могут представлять собой следующие разновидности:

1) данные типа ключ-значение,

2) список объектов,

3) многоуровневые данные в виде вложения в пункт 1 дополнительных объектов и/или списка данных.

Такой метод хранения, особенно при использовании списка объектов, повторяет связь в реляционной СУБД «один-ко-многим», предполагая, что к одной записи относится множество других записей. Метод хорош тем, что не требуется создавать дополнительные сущности и дополнительные связи между сущностями. Примером использования такого метода хранения данных можно назвать аналитику компаний по разным показателям, таким как, например, вид экономической деятельности, поскольку организации могут иметь несколько видов деятельности параллельно с основным. Исходным форматом поставки данных для аналитики, в большинстве случаев, является использование объекта, который следует перевести в состояние таблицы для более оптимизированного доступа к данным.

В работе представлены результаты проведенного исследования, которые показывают релевантность использования объектов как метод хранения данных и сравнение его с обычным методом хранения.

Для эксперимента использовалась последняя версия СУБД PostgreSQL и язык программирования Python. Выбор СУБД PostgreSQL обусловлен тем, что иностранные компании-поставщики СУБД прекращают поддержку своих продуктов на территории Российской Федерации, однако PostgreSQL является open-source продуктом, который постоянно поддерживается разными разработчиками. Открытый исходный код PostgreSQL позволяет организациям делать свои образы СУБД, что фактически дает организациям самостоятельно поддерживать СУБД. В СУБД было создано три таблицы с использованием разных методов хранения:

1) обычный,
2) JSON-объект с первичным ключом,
3) JSONB-объект с первичным ключом.

Для эксперимента использовался выделенный Windows сервер со следующими характеристиками:

1) процессор: 64-битный Intel Xeon Gold 6128 3,4GHz (6 ядер 12 потоков),
2) ОЗУ: 16ГБ,
3) постоянная память на HDD,
4) ОС: Windows Server 2016.

При обращении к официальной документации PostgreSQL разница между JSON и JSONB описывается как «типы данных json и jsonb принимают в качестве входных данных почти идентичные наборы значений. Основное практическое отличие заключается в эффективности. Тип данных json хранит точную копию входного текста, который обрабатывающие функции должны повторно обрабатывать при каждом выполнении. Данные jsonb хранятся в разложенном двоичном формате, что немного замедляет ввод из-за дополнительных затрат на преобразование, но значительно ускоряет обработку, поскольку повторный анализ не требуется. Jsonb также поддерживает индексацию, что может быть значительным преимуществом» [7].

В ходе эксперимента с помощью Python генерировались случайные записи разной длины в количестве 10 штук в соответствии с количеством атрибутов. Оценка производительности не включала время, потраченное на генерацию атрибутов, что позволяет более точно определить время записи строк. Всего проведено 10 тестов, показывающих производительность 1000, 2000, 5000, 10 000, 20 000, 50 000, 100 000, 200 000, 500 000,

1 000 000 записей. Все таблицы имеют одинаковое содержание за счет того, что производилась генерация данных с последующей очередной записью данных в таблицы.

Для визуализации эксперимента были выведены графики с разным поведением вставки. По оси *X* указаны порядковые метки записей, по оси *Y* – время, затраченное на вставку данных.

На рис. 1 показан график записи сгенерированных строк при различных сценариях.

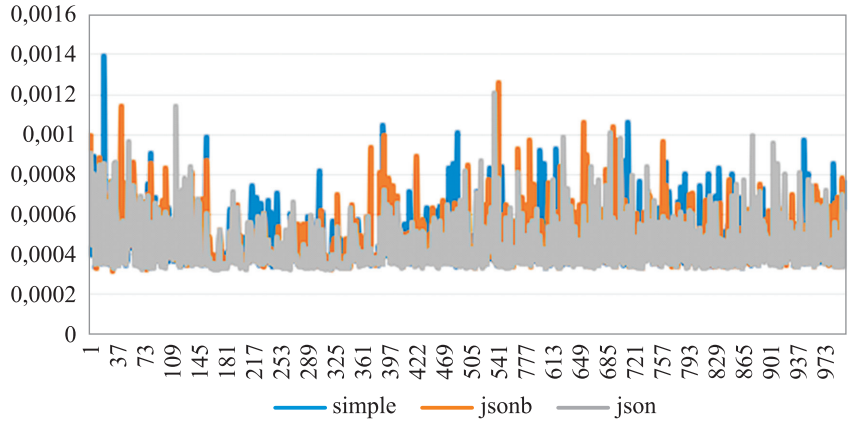

Рис. 1. График производительности вставки 1000 строк
1000 row insert performance graph

На графике видно, что все методы вставки имеют примерно одну и ту же сезонность, однако в большинстве случаев выбросы не совпадают, скорее всего существует зависимость вводимых данных от времени, затраченного на вставку данных.

Явные признаки выбросов присутствуют при вставке 10 000–50 000 строк (рис. 2–4). При этом в выбросах участвуют методы вставки объектов, однако при 50 000 строках начинают выделяться явные выбросы по обычному типу вставки.

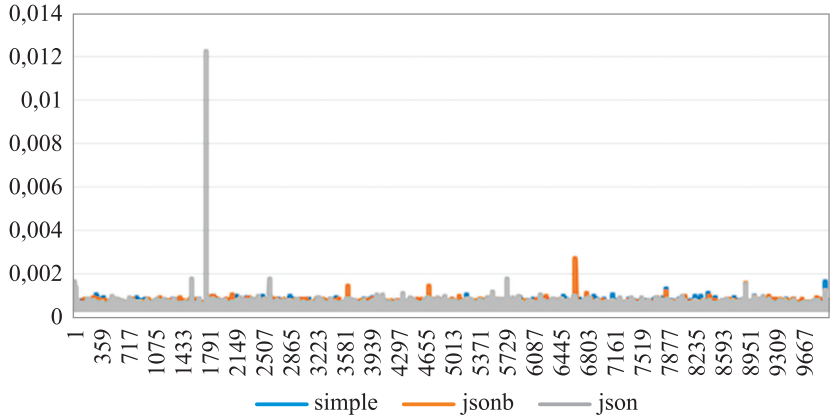

Рис. 2. График производительности вставки 10 000 строк
10,000 row insert performance graph

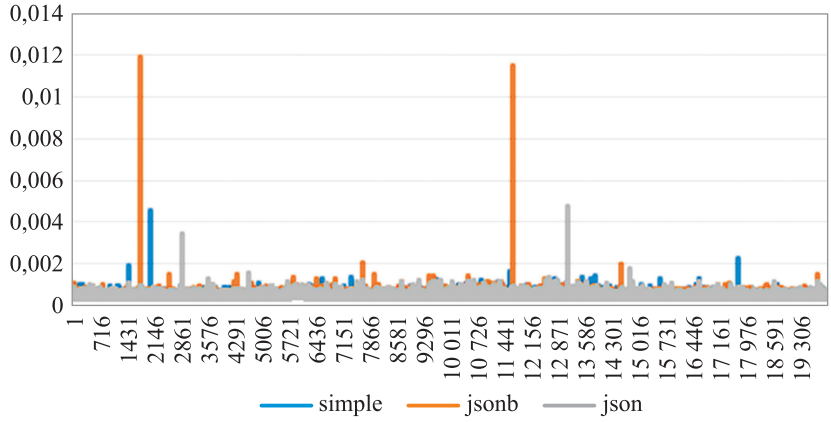

Рис. 3. График производительности вставки 20 000 строк
20,000 row insert performance graph

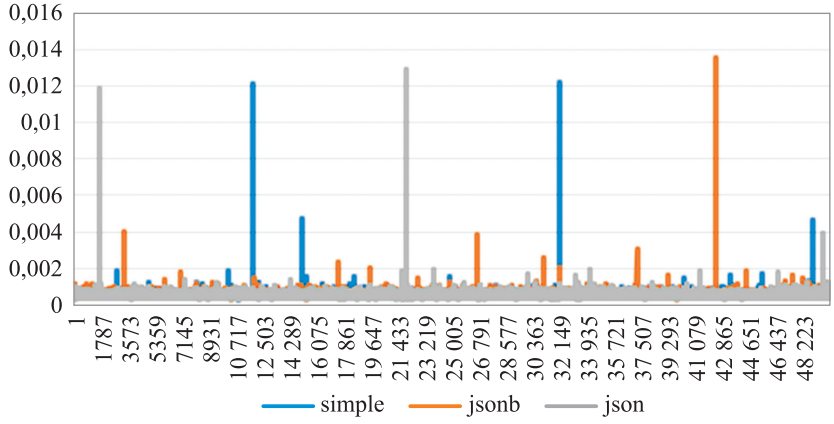

Рис. 4. График производительности вставки 50 000 строк
50,000 row insert performance graph

При вставке 100 000–500 000 строк (рис. 5–7) наблюдается общая тенденция на синхронное замедление.

Рис. 5. График производительности вставки 100 000 строк
100,000 row insert performance graph

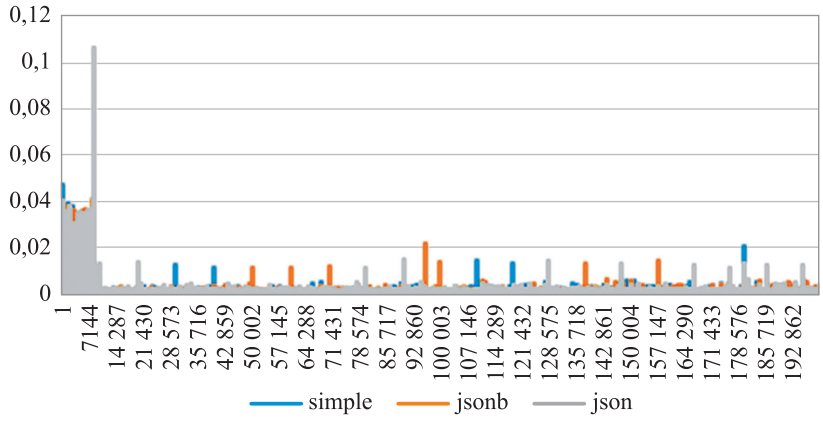

Рис. 6. График производительности вставки 200 000 строк
200,000 row insert performance graph

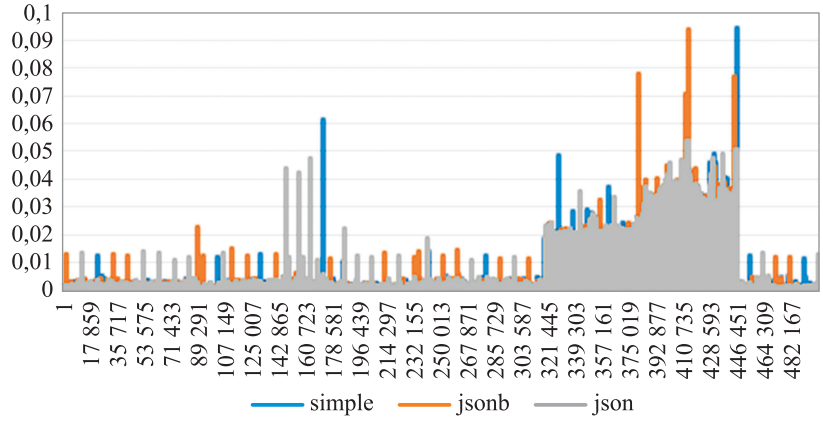

Рис. 7. График производительности вставки 500 000 строк
500,000 row insert performance graph

При вставке 1 000 000 строк наблюдается синхронное замедление всех типов вставки, показанное на рис. 8.

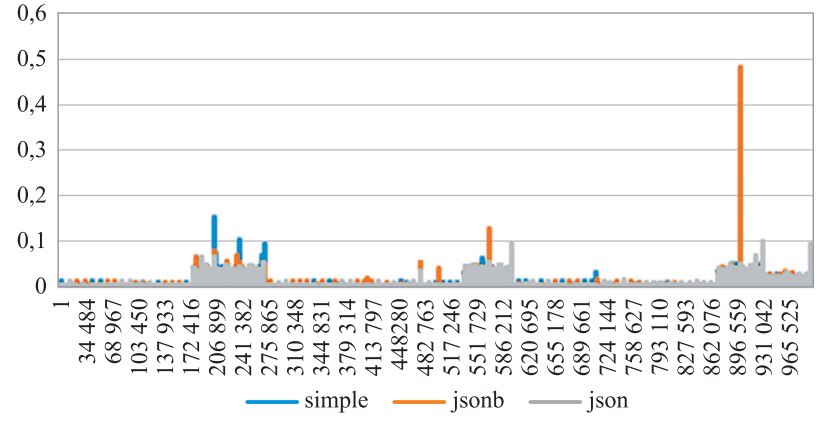

Рис. 8. График производительности вставки 1 000 000 строк
Performance graph for inserting 1,000,000 rows

При проведении эксперимента были выявлены выбросы на разных этапах исследования, однако их наличие не повлекло за собой существенное изменение результатов исследования. Выбросы показали различное поведение одного и того же контента в разных сценариях записи.

По результатам проведенных экспериментов были получены результаты, представленные в табл. 1, 2. В табл. 1 представлены результаты по времени вставки данных по типам и количеству строк. Данные были округлены до тысячных.

Таблица 1

Сводная таблица результатов вставки данных

Summary table of data insertion results

Строки/ эксперимент	SIMPLE	JSONB	JSON	Delta simple/ JSONB	Delta simple/ JSON	Delta JSON/ JSONB
1000	0,430	0,443	0,430	2,826	0,021	2,806
2000	0,885	0,923	0,893	4,114	0,919	3,225
5000	2,268	2,349	2,278	3,445	0,455	3,004
10 000	4,082	4,160	4,087	1,861	0,116	1,747
20 000	8,393	8,624	8,411	2,680	0,214	2,472
50 000	20,623	21,105	20,640	2,288	0,087	2,202
100 000	42,440	43,390	42,474	2,189	0,078	2,112
200 000	84,277	86,076	84,276	2,090	–0,001	2,092
500 000	223,542	227,550	222,852	1,762	–0,309	2,065
1 000 000	439,304	450,719	440,008	2,533	0,160	2,377

Таблица 2

Сводная таблица сравнения занимаемого пространства

Space comparison summary table

Строки/ эксперимент	SIMPLE	JSONB	JSON	Delta simple/ JSONB	Delta simple/ JSON	Delta JSON/ JSONB
1000	0,51	0,67	0,67	23,881	23,881	0,000
2000	0,97	1,30	1,30	25,385	25,385	0,000
5000	2,33	3,16	3,16	26,266	26,266	0,000
10 000	4,61	6,28	6,28	26,592	26,592	0,000
20 000	9,41	13,00	13,00	27,615	27,615	0,000
50 000	24,00	31,00	31,00	22,581	22,581	0,000
100 000	48,00	62,00	62,00	22,581	22,581	0,000
200 000	96,00	125,00	125,00	23,200	23,200	0,000
500 000	240,00	311,00	311,00	22,830	22,830	0,000
1 000 000	481,00	623,00	623,00	22,793	22,793	0,000

Из вышеприведенных данных можно сделать вывод, что вставка обычным способом осуществляется быстрее, при этом также существует менее заметная разница между вставкой JSON и JSONB записей.

В табл. 2 представлена разница по занимаемому общему пространству таблиц после их заполнения. Исследование показало, что при обычном методе вставки объем занимаемой памяти на более чем 20 % меньше, чем при использовании JSON и JSONB. Однако разницы по занимаемому пространству между JSON и JSONB нет и объясняется это тем, что они хранятся в одинаковой конфигурации. Дельта по занимаемому пространству между обычным типом и объектными произошла из-за того, что в объектных типах помимо самих данных хранятся ключи данных, что увеличивает размер одной записи.

Была проведена дополнительная серия экспериментов, связанная с отношением количества атрибутов к разнице занимаемого пространства. Результат показал, что разница между обычным методом и объектным колеблется в пределе 25 и 30 %. Таким образом можно сделать вывод, что использование объектного метода хранения данных также увеличивает занимаемое пространство на носителе.

Использование объектного метода хранения данных будет релевантно в следующих случаях:

1) таблицы не имеют большой объем данных,

2) используемые в объектах данные лучше подходят для хранения в них свойств, например, состояния компонентов React.

Доступ к объектным данным осуществляется сложнее, поскольку требуется дополнительная задача по развертке объекта в табличный тип, что приводит к дополнительным вычислениям [5]. Обновление объекта также сопровождается дополнительными сложностями в виде того, что, в отличие от обычной записи, нужно указывать объект целиком с изменяемой частью что может стать следствием нарушения целостности данных в будущем [5].

Предлагается использование конвертации объектной таблицы в обычную, что оптимизирует работу с данными и улучшит характеристики таблицы как в разрезе производительности, так и в разрезе занимаемого дискового пространства. На нативном уровне проблема конвертации решается с помощью операторов и функций запроса, которые получают доступ к данным по ключу. Оптимизация может происходить во время получения данных из запроса, что делает возможным на начальном этапе создать нормализованную таблицу с упрощенным доступом к данным. Пример описанной оптимизации показан на рис. 9.

```
SELECT id,first_name,patronymic, citizenship,date_of_bitrh
    FROM test.json_tbl,
    jsonb_to_record(data_json) as x(last_name text,
  first_name text,
  patronymic text,
  citizenship text,
  date_of_bitrh text);
```

Data Output | Messages | Notifications

	id [PK] numeric	first_name text	patronymic text	citizenship text	date_of_bitrh text
1	1	GVIHHYZKWAATWZXUKDSJKJDAQ	RKOODIJYWTNZFBEXWWKZWF...	SH	YTKIJCXSWI
2	2	DGOWNFPPVSYJCZKDNDSHUUQ...	YPMBFUYDUNMEEIBELWUFBOM...	UT	OQNHKEUREP
3	3	XEPCMQBTHGGZKFCYIVVODQZVN	ISHGUEXBMZIKNTOFDRPDPOW...	BX	WVFIDXZBFR
4	4	XMKITRTJFQDVYTHNNZZKGTQSH	DLDCHEOGZARDCUDWFLHKGJE...	WV	DPWNPLOLLZ
5	5	RDDKFYRAZYJIVJCIZPRVPDOFU	PRUYBARJDRVIELCYWAWTRTJPL	NG	NBXZTZORJM

Рис. 9. Пример преобразования объекта в таблицу
Example of converting an object to a table

Функция jsonb_to_text( ) получает объект в записи и представляет его в виде полей с указанными типами данных. Если использовать операторы, показанные на рис. 10, то каждый атрибут получает возможность использовать внутреннее вложение, что позволяет более гибко хранить данные.

```
SELECT
    JS ->> 'lastname' AS LASTNAME,
    JS ->> 'firstname' AS FIRSTNAME,
    JS -> 'university'  ->> 'bachelor' AS BACHELOR,
    JS -> 'university'  ->> 'master' AS MASTER,
    JS -> 'university'  ->> 'phd' AS PHD
FROM JSONTEST
```

Рис. 10. Пример преобразования объекта с вложенными объектами в таблицу [5]
An example of converting an object with nested objects into a table [7]

Использование каждого из этих методов зависит от сценария, который будет в таблице-источнике, однако эти методы можно комбинировать в одном запросе, что позволяет максимально эффективно преобразовать объекты в таблицы.

Заявленные компаниями методы оптимизации пространства не являются релевантными, как показано в эксперименте статьи. Использование JSON-объектов приводит к увеличению занимаемого пространства за счет хранения в записях ключей и их значений, в то время как для хранения данных в табличном виде исключает хранение ключей в записях, что сразу уменьшает занимаемое пространство относительно вышеописанного метода хранения. Также использование JSON-объектов не подходит и для оптимизации рабочего процесса в компании, специализирующейся на аналитике, так как на примере изменения записи операцией UPDATE требуется полный ввод полученной записи с учетом необходимых изменений, что повышает риск человеческого фактора. Хранение данных с помощью JSON-объектов сегодня имеет перспективу для разработчиков, так как реализация полноценного хранения параметров внутри реляционной СУБД позволяет более широко использовать объекты в виде свойств, делая объекты и их свойства записями в базе данных. Для Российской Федерации такая реализация в PostgreSQL – СУБД с открытым исходным кодом создает альтернативу NoSQL СУБД, таким как MongoDB, который частично заблокирован в облачной реализации.

Список источников

1. *Волушкова В.Л., Волушкова А.Ю.* Структура данных для хранения информации в социальных сетях // Образовательные ресурсы и технологии. 2014. № 2 (5).
2. *Дьяконов А.В., Козлова Ю.Б.* О современных тенденциях хранения данных в документо-ориентированных СУБД // Актуальные проблемы авиации и космонавтики. 2015. № 11. С. 394–396.
3. *Новиков Б.А.* Сравнительный анализ производительности SQL И NOSQL СУБД // КИО. 2017. № 4.
4. *Рудометкин В.А.* Проектирование высоконагруженных систем // Труды ИСП РАН. 2020. № 6.

5. *Широков И.А*. Исследование возможностей хранения и обработки многоуровневых данных в PostgreSQL // Культура, наука, образование: проблемы и перспективы: материалы X Всерос. науч.-метод. конф. с международным участием (г. Нижневартовск, 10–11 ноября 2022 г.). Нижневартовск, 2022. С. 451–457.
6. *Győrödi C., Sotoc R*. A comparative study: MongoDB vs. MSSQL // Conference: 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). 2015.
7. JSON Types. [Электронный ресурс]. URL: https://www.postgresql.org/docs/current/datatype-json.html#JSON-CONTAINMENT (дата обращения: 24.03.2023).

References

1. Volushkova V.L., Volushkova A.Ju. Struktura dannyh dlja hranenija informacii v social'nyh setjah [Data structure for information storage in social networks], *Obrazovatel'nye resursy i tehnologii* [*Educational resources and technologies*], 2014, no. 2 (5).
2. D'jakonov A.V., Kozlova Ju.B. O sovremennyh tendencijah hranenija dannyh v dokumento-orientirovannyh SUBD [About modern trends of data storage in document-oriented DBMS], *Aktual'nye problemy aviacii i kosmonavtiki* [*Actual problems of aviation and cosmonautics*], 2015, no. 11, pp. 394–396.
3. Novikov B.A. Sravnitel'nyj analiz proizvoditel'nosti SQL I NOSQL SUBD [Comparative analysis of SQL AND NOSQL DBMS performance], *KIO* [*KIO*], 2017, no. 4.
4. Rudometkin V.A. Proektirovanie vysokonagruzhennyh sistem [Designing of the highly loaded systems], *Trudy ISP RAN* [*Proceedings of ISP RAS*], 2020, no. 6.
5. Shirokov I.A. Issledovanie vozmozhnostej hranenija i obrabotki mnogourovnevyh dannyh v PostgreSQL [Research of possibilities of storage and processing of multilevel data in PostgreSQL]. Kul'tura, nauka, obrazovanie: problemy i perspektivy: materialy X Vseros. nauch.-metod. konf. s mezhdunarodnym uchastiem (g. Nizhnevartovsk, 10–11 nojabrja 2022 g.). Nizhnevartovsk, 2022. Pp. 451–457.
6. Győrödi C., Sotoc R. A comparative study: MongoDB vs. MSSQL. Conference: 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). 2015.
7. JSON Types [Electronic resource]. Available at: https://www.postgresql.org/docs/current/datatype-json.html#JSON-CONTAINMENT (accessed: 24.03.2023).

Сведения об авторе:

И.А. Широков – Университет ИТМО, Санкт-Петербург, Российская Федерация.

Information about the author:

I.A. Shirokov – ITMO University, St. Petersburg, Russian Federation.

Статья поступила в редакцию	*03.05.2023*	*The article was submitted*	*03.05.2023*
Одобрена после рецензирования	*19.06.2023*	*Approved after reviewing*	*19.06.2023*
Принята к публикации	*21.06.2023*	*Accepted for publication*	*21.06.2023*