Компьютерные инструменты в образовании, 2017
№ 4: 48–63
УДК: 004.65
http://ipo.spb.ru/journal

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ SQL И NOSQL СУБД*

Новиков Б. А.[1,2], Левин М. Ю.[1]

[1]Санкт-Петербургский государственный университет, Санкт-Петербург, Россия
[2]Научный институт университета Аалто, Эспоо, Финляндия

Аннотация

NoSQL системы управления базами данных находятся в поле зрения специалистов области уже довольно давно. Однако на текущий момент очень мало работ связано с данной темой, а в особенности — со сравнением таких систем с традиционными реляционными СУБД. Что касается существующих исследований, то одни статьи являются обзорными, другие используют небольшое количество записей в таблицах в качестве нагрузки или рассматривают только одно окружение для проведения экспериментов, что может давать преимущество одному из объектов сравнения. Данная работа посвящена PostgreSQL и MongoDB. Первая система, несмотря на то, что является свободным программным обеспечением, стремительно набирает популярность в том числе и в корпоративном сегменте, а вторая отличается от большого числа NoSQL решений хорошей проработанностью и поддержкой. С целью обеспечения полноты сравнения, эксперименты проводились в различных окружениях и с различной нагрузкой.

Ключевые слова: *анализ прозводительности, время выполнения запросов, системы управления базами данных, MongoDB, NoSQL, OLAP, OLTP, PostrgeSQL, SQL.*

Цитирование: Новиков Б. А., Левин М. Ю. Сравнительный анализ производительности SQL и NoSQL СУБД // Компьютерные инструменты в образовании, 2017. № 4. С. 48–63.

1. ВВЕДЕНИЕ

NoSQL системы управления базами данных активно растут в последнее десятилетие. В 1998 году итальянский ученый Карло Строцци использовал этот термин для обозначения СУБД его собственной разработки, предназначенной для работы в контексте распределенных архитектур. В ней он не только отказался от языка структурированных запросов SQL, но и от важнейшего принципа реляционных СУБД — ACID (Atomicity, Consistency, Isolation, Durability — Атомарность, Согласованность, Изолированность, Постоянство хранения) [1]. Впрочем, на сегодняшний день данный термин в силу разных причин трактуют в менее радикальном смысле, а именно как «Не-Только-SQL».

Если говорить о причинах популярности такого рода систем, то среди прочих стоит выделить неизбежную порой необходимость расширения вычислительных архитек-

*Работа поддержана грантом РФФИ 16-57-48001.

тур «по горизонтали» (англ. scale-out), то есть не наращивания мощи отдельных узлов, а добавления новых к уже существующим. Сложности реализации полной поддержки транзакционности в распределенной среде крайне затрудняют соответствующие изменения в системах, ядром которых являются SQL СУБД. Тем удивительнее видеть, что NoSQL-системы в данный момент представлены крайне небольшим количеством исследований (как самостоятельных, так и посвященных сравнению их производительности с реляционными системами). Имеющиеся работы зачастую не позволяют получить полной картины, так как либо описывают эксперименты узкой направленности (например, сравнение временных затрат только на операции вставки данных), либо имеют в качестве объектов исследования специализированные и редко применяемые СУБД.

В рамках данной работы предлагается рассмотреть довольно известные PostgreSQL и MongoDB. MongoDB [2] — документно-ориентированная СУБД, разрабатываемая одноименной компанией (ранее — 10gen). MongoDB является одной из самых популярных NoSQL систем и имеет широкий функционал. Аналогом привычных таблиц из реляционных СУБД здесь являются коллекции, в которых хранятся JSON-документы (Java Script Object Notation — текстовый формат для обмена и хранения данных, представляющий информацию в виде пар «ключ-значение»). Для работы с ними предусмотрены операции поиска, вставки, удаления и обновления. Метод запросов по образцу используется для поиска документов в коллекциях, поддерживаются проекция, сортировка, просмотр результатов запроса через курсор. Масштабируемость в MongoDB достигается за счёт разделения документов из коллекции по узлам на основании выбранного ключа (shard key). Поддерживается асинхронная репликация в режиме «главный-подчиненный» (англ. master-slave): операции записи обрабатываются только главным узлом, а операции чтения могут осуществляться как с главного узла, так и с одного из подчиненных. Клиент может работать в разных режимах: асинхронном (не дожидаясь отклика) или блокирующем (ожидая подтверждения от существующих в распределенной сети узлов). Таким образом, MongoDB поддерживает различные модели согласованности в зависимости от того, разрешены ли чтения с вторичных узлов и от скольких узлов ожидаются подтверждения при записи. Данная СУБД используется в большом числе крупных компаний и проектов, среди которых SourceForge, Foursquare, The Guardian, Forbes, The New York Times и другие [3].

В свою очередь, PostgreSQL [4] — это объектно-реляционная система управления базами данных, которая была разработана в научном компьютерном департаменте Беркли Калифорнийского Университета. Это продукт с открытым исходным кодом, он поддерживает большую часть стандарта SQL: комплексные запросы, внешние ключи, триггеры, транзакционная целостность, многоверсионное управление параллельным доступом и т. д.

2. ОБЗОР СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ

В этом разделе осуществляется рассмотрение некоторых иных исследований из области сравнения производительности SQL и NoSQL систем управления базами данных. Группа ученых из университетов Коимбры подготовила работу по сравнению MongoDB, Cassandra, HBase, OrientDB и Redis с помощью фреймворка Yahoo! Cloud System Benchmark [5]. Наряду с традиционным многократным выполнением запроса к хранимым данным для получения точной информации о временных затратах, YCSB позволяет комбинировать нагрузку, выполнив в рамках одной «задачи» операции как чтения, так и обновления данных. При достаточной проработке сценариев такой подход позволяет

получить очень подробную информацию о возможностях СУБД. К недостаткам работы можно отнести небольшой объем тестовых данных и отсутствие в рассмотрении SQL СУБД.

Статья Рика Каттелла (Rick Cattell) носит обзорный характер и делает акцент на горизонтальной масштабируемости как SQL, так и NoSQL систем хранения и обработки данных [6]. Автор справедливо отмечает, что платой за выбор такого пути повышения производительности является частичный или полный отказ от неотъемлемых атрибутов традиционных СУБД: механизмов обеспечения целостности, гарантий долговечности и доступности. В статье большое внимание уделяется классификации NoSQL решений, приводится справочная информация по многим из них, осуществляется их сравнение по способу организации совместного доступа, размещению в памяти и репликации. В работе группы ученых из университета Алабамы в качестве конкурента продукта компании MongoDB использовался Microsoft SQL Server Express [7]. Сами базы хранились на SSD-дисках для увеличения скорости чтения и записи. Была проведена серия экспериментов со вставкой, обновлением данных и выборкой по полям с индексами и без таковых для разного количества строк. MongoDB проявила себя лучше в задачах вставки данных и несложных запросов, а используемая SQL-система — при изменении неиндексированных полей и выполнении запросов с агрегированием. Проведенные эксперименты также показывают существенное уменьшение времени выполнения запросов в MongoDB после построения индекса. Из-за небольшого количества записей в базах данных время выполнения запросов в рамках этой работы не превышало секунду, а зачастую составляло единицы миллисекунд, поэтому авторам приходилось выполнять запросы сотни раз для повышения точности оценок.

В сентябре 2014 года компания Enterprise DB, являющаяся самым крупным в мире поставщиком продуктов и услуг корпоративного уровня на основе PostgreSQL, опубликовала на своем сайте заметку о том, каких результатов удалось добиться PostgreSQL в плане улучшения производительности в сравнении с MongoDB благодаря внедрению нового типа данных JSONB, призванного ускорить доступ к хранимым JSON-документам [8]. Сравнение, которое с учетом условий проведения эксперимента должно быть комплементарным по отношению к MongoDB, обернулось победой PostgreSQL практически по всем статьям: скорости выполнения операций вставки данных, доступа к ним и потребления дискового пространства.

Главным итогом данного обзора стало принятие решения о подготовке разнообразных сценариев запросов, а также тестовых данных в количестве, достаточном для получения статистически значимых результатов.

3. ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ В ОКРУЖЕНИИ, СОСТОЯЩЕМ ИЗ ОДНОЙ ЭВМ

На сегодняшний день можно с уверенностью говорить, что процесс информатизации касается всех сфер жизнедеятельности человека, а значит, что те или иные хранилища информации используются практически повсеместно. Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов — атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов). В рамках данной работы для СУБД PostgreSQL организована структура «интернет-магазин-склад» (см. рис. 1).

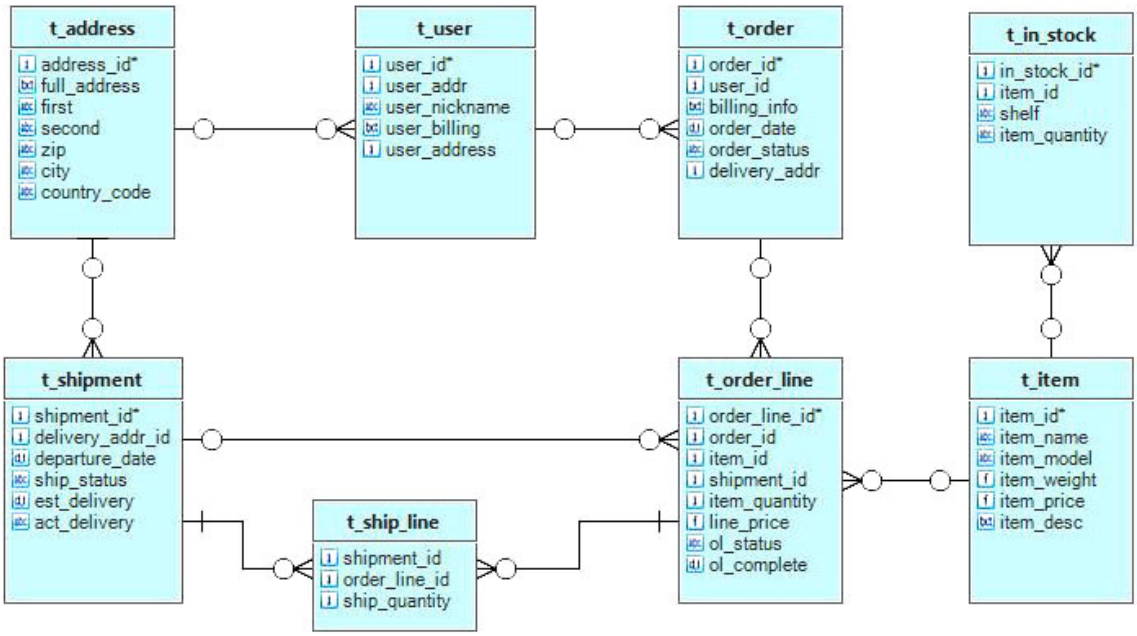


Рис. 1. ER-диаграмма базы данных для PostgreSQL

Что касается MongoDB, то она является нереляционной базой данных, а значит, строить произвольные запросы по имеющимся данным невозможно. Данную проблему решают, как правило, двумя способами. Первый из них состоит в проектировании коллекций на манер таблиц из реляционных СУБД. Само присоединение при этом осуществляется в рамках приложения. Второй способ связан с денормализацией данных. Поместив, например, коллекцию t_address внутри коллекции t_user (оставив при этом отдельную копию таблицы t_address), можно обеспечить возможность предварительной организации запросов присоединения по этим сущностям. Данный подход, впрочем, связан с очень серьезными трудностями по обеспечению согласованности данных, ведь изменения, произошедшие с конкретной записью в одной коллекции, должны произойти и во всех копиях. Таким образом, следует быть крайне внимательным при реализации «prejoin» и принимать во внимание связанные с ним сложности при анализе экспериментов, описываемых в данной работе.

В качестве рабочей станции был использован персональный компьютер с характеристиками:

— операционная система: Windows 10;
— процессор: Intel Core i7 2.6 ГГц;
— оперативная память: 8 ГБ.

Версия PostgreSQL – 9.6.1, версия MongoDB – 3.4.2. Для генерации данных использовался интернет-сервис [9]. Каждый эксперимент проводился для 10000, 100 000, 500 000, 1 000 000, 2 000 000 и 5 000 000 записей с вычислением среднего времени выполнения по тридцати попыткам. Подсчеты и построение гистограмм осуществлялись в программном продукте Microsoft Excel. Для измерения времени выполнения в PostgreSQL использовалась директива /timing, в MongoDB применялись методы профилирования журнала операций, использование метода explain() там, где это возможно, а также расстановка временных меток с последующим вычислением разницы между их значениями. Стоит сделать замечание, что внутренний анализатор запросов MongoDB имеет точность 1 мс,

поэтому при проведении экспериментов с небольшим количеством данных точную информацию о времени выполнения запроса получить не удастся.

В эксперименте по вставке записей (см. рис. 2) использовалась сущность t_item, хранящая строки вида:

Таблица 1. Пример строки из таблицы t_item

item_id (integer)	item_name (varchar(30))	item_model (varchar(30))
1	vitae consectetuer	adipiscing
item_weight (float)	**item_price (float)**	**item_desc (text)**
52,902	9959,346	ante ipsum primis in faucibus

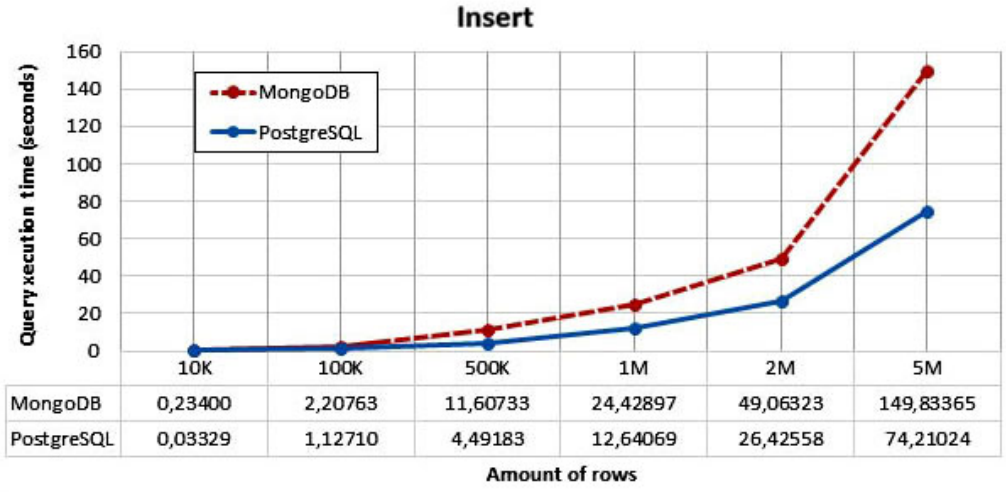


Рис. 2. Сравнение времени выполнения операций вставки

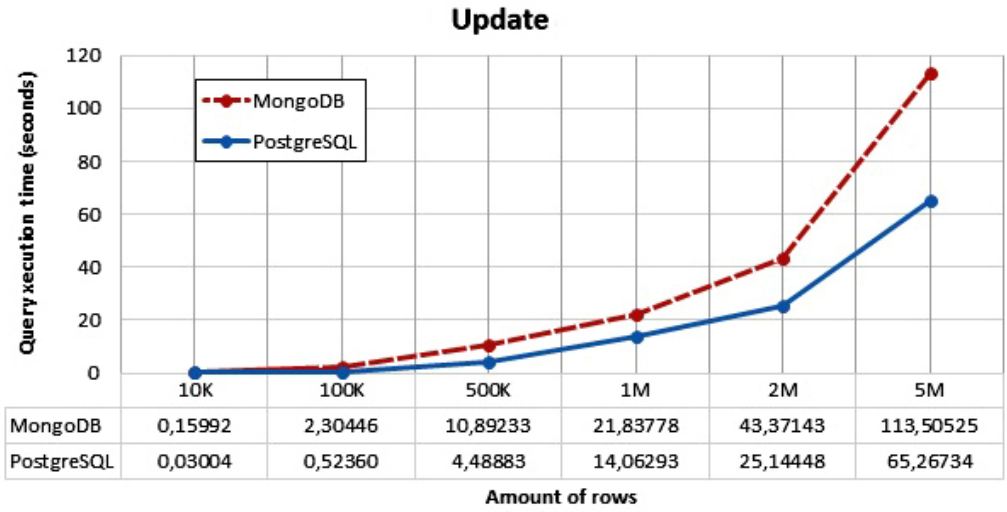
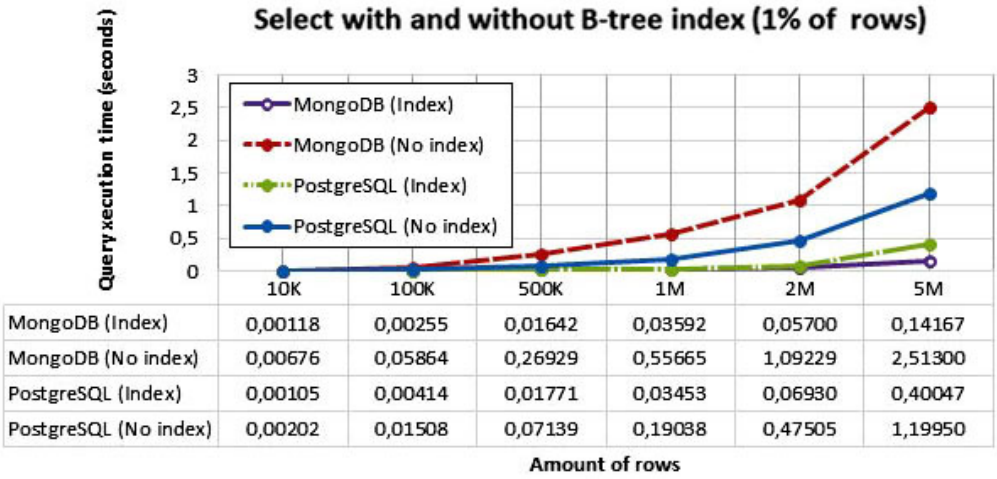


Рис. 3. Сравнение времени выполнения операций обновления

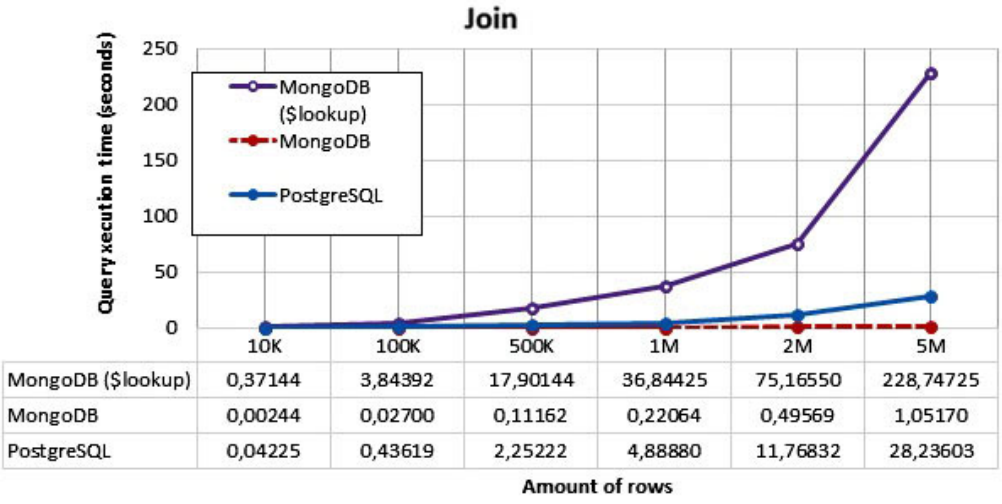
Эксперимент по обновлению данных (см. рис. 3) проводился в рамках таблицы t_address, осуществлялось изменение числового поля zip (почтовый индекс).

Для экспериментов с операцией выборки (см. рис. 4) применялся следующий сценарий: измерялось время выполнения операций с условием выборки для одних и тех же записей с наличием построенного по соответствующему полю индекса на основе двоичного дерева поиска и без него. Использовалось условие по полю item_price типа float. Поисковому выражению удовлетворял один процент записей. На рис. 5 приводятся результаты экспериментов с операцией выборки с присоединением таблицы (t_user и t_address по полям user_address_id и address_id соответственно).



	10K	100K	500K	1M	2M	5M
MongoDB (Index)	0,00118	0,00255	0,01642	0,03592	0,05700	0,14167
MongoDB (No index)	0,00676	0,05864	0,26929	0,55665	1,09229	2,51300
PostgreSQL (Index)	0,00105	0,00414	0,01771	0,03453	0,06930	0,40047
PostgreSQL (No index)	0,00202	0,01508	0,07139	0,19038	0,47505	1,19950

Рис. 4. Сравнение времени выполнения операций выборки с индексом и без использования индекса



	10K	100K	500K	1M	2M	5M
MongoDB ($lookup)	0,37144	3,84392	17,90144	36,84425	75,16550	228,74725
MongoDB	0,00244	0,02700	0,11162	0,22064	0,49569	1,05170
PostgreSQL	0,04225	0,43619	2,25222	4,88880	11,76832	28,23603

Рис. 5. Сравнение времени выполнения операций присоединения данных

Как уже было упомянуто выше, на сегодняшний день под словом NoSQL понимают не те СУБД, управление которыми осуществляется при помощи языка, не принадлежащего стандарту SQL, а, скорее, те, которые не являются реляционными. Тем удивительнее видеть, что в версии MongoDB – 3.2 — компания-разработчик предоставила возможность

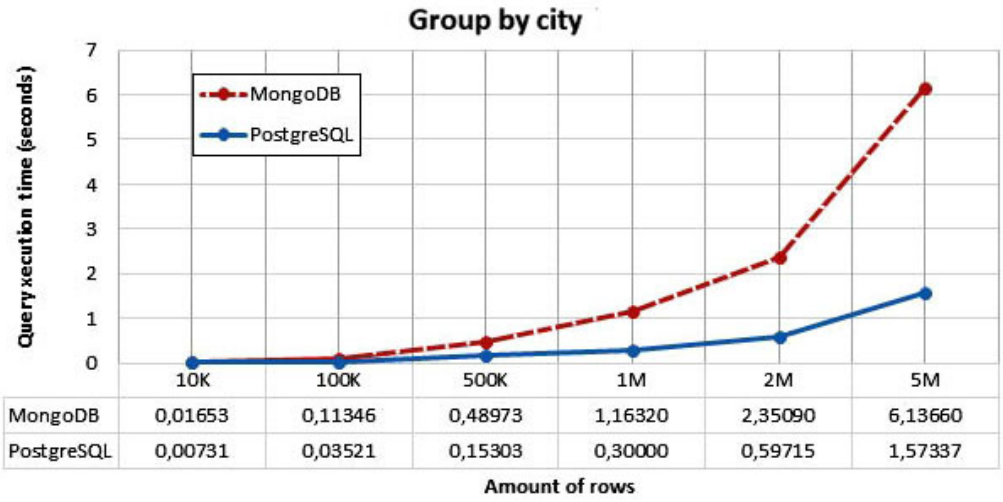
организации соединений по общим полям таблицы [10]. Команда соединения выглядит следующим образом:

```
db.t_user.aggregate([{$lookup:{ from: 't_address', localField: 'user_address_id',
                                foreignField: 'address_id', as: 'find_address'}}]),
```

где from — наименование внешней коллекции, localField — поле присоединения из рассматриваемой коллекции, foreignField — поле присоединения из внешней коллекции, as — alias, наименование для получившегося присоединения записей. На данный момент можно осуществлять присоединение только по одному полю и только в формате left outer join. Таким образом, исходя из существующих ограничений и временных характеристик выполнения данного запроса при организации присоединений таблиц в выбранном NoSQL-решении, следует пользоваться иными методами.

Эксперимент (см. рис. 5) основан на запросе, вычисляющем, сколько пользователей ресурса представляют тот или иной город (на основе таблицы t_address; результат отсортировать по убыванию агрегирующей функции).

Последний эксперимент в этой главе связан с поиском максимального значения поля item_price в таблице t_item (см. рис. 6). Таким образом, эксперименты показывают преимущество PostgreSQL во всех задачах кроме индексированного поиска.



	10K	100K	500K	1M	2M	5M
MongoDB	0,01653	0,11346	0,48973	1,16320	2,35090	6,13660
PostgreSQL	0,00731	0,03521	0,15303	0,30000	0,59715	1,57337

Рис. 6. Сравнение времени выполнения операций группировки данных

Что касается операции присоединения, то принятие решения о денормализации данных для NoSQL СУБД напрямую зависит от конкретной задачи с учетом затрат на поддержание согласованности и хранение избыточной информации.

4. ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ В РАСПРЕДЕЛЕННОМ ОКРУЖЕНИИ

СУБД MongoDB изначально спроектирована в расчете на горизонтальное масштабирование, которое является одним из наиболее разумных методов повышения мощности систем хранения данных. Для этого в ней реализован механизм автоматического шардирования (autosharding), который управляет распределением хранимой информации между узлами системы. При этом код приложения ничего не знает об инфраструктуре и

взаимодействует с сегментированным кластером так, как будто это один узел (см. пример на рис. 7).

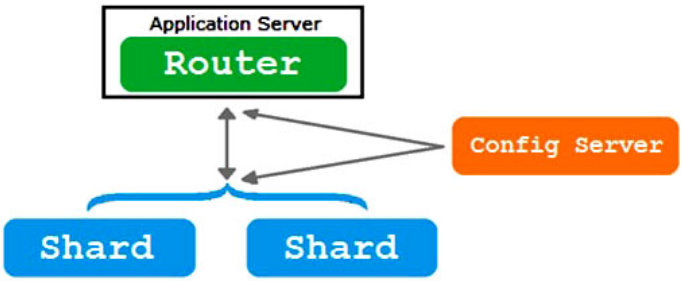


Рис. 7. Пример топологии распределенной сети для MongoDB

На каждом из шардов (англ. shard — осколок), которые являются отдельными машинами, хранится некоторая часть всего множества имеющихся данных. Благодаря им и реализуется масштабирование операций записи и чтения. Конфигурационный сервер отвечает за хранение метаинформации о глобальной конфигурации кластера, физическом местоположении каждой базы данных, коллекции и журнале изменений, в который записывается история миграции данных между узлами. Без этой информации невозможно получить согласованное представление сегментированного кластера, поэтому в производственных условиях для повышения отказоустойчивости рекомендуется иметь конфигурационные серверы в количестве, равном количеству шардов. Маршрутизатор (он же «мастер») реализует интерфейс взаимодействия с кластером. Именно ему адресует запросы драйвер приложения. Процессы mongos не требуют большого количества ресурсов и зачастую располагаются на одном сервере с приложением.

Стоит отметить, что шардирование данных носит, главным образом, логический характер. Для сегментируемой коллекции специфицируется ключ шардирования (shard key), который обязан являться индексом или его частью и присутствовать во всех документах коллекции. Далее диапазон всевозможных значений ключа делится на порции (chunks). Для всех JSON-документов однозначно определяется, к какой порции они относятся. Каждый шард хранит у себя некоторое количество порций коллекции, причем соответствующие им диапазоны значений ключа не обязательно являются смежными, а порядок следования документов в порции ничего не говорит об особенностях их физического расположения в шарде. По достижению максимального размера порции (определяемого в настройках), происходит логическое расщепление порции на два поддиапазона. Внутренний балансировщик MongoDB осуществляет физический процесс миграции данных между шардами для управления нагрузкой на отдельные узлы системы. Конкретные цифры зависят от общего объема данных, но зачастую миграция осуществляется, когда разница между максимальным и минимальным количеством порций среди всех шардов системы становится больше восьми.

В это же самое время в PostgreSQL, как и в прочих реляционных СУБД, изначально предусмотрены только такие возможности балансировки нагрузки, как партицирование, то есть разбиение таблиц на логические части по выбранным критериям в рамках одной машины, и репликация, заключающаяся в построении распределенной системы с рабочими узлами, хранящими идентичную информацию и обеспечивающими, таким образом, горизонтальное масштабирование операций чтения. Возможности шардинга изначально не предусмотрены, поэтому некоторые компании, взяв в качестве основы

PostgreSQL, разрабатывают для этих целей отдельные программные продукты. Ниже рассмотрены некоторые из них.

Postgres-XL предназначен для организации кластерных OLTP-систем и обработки сложных аналитических запросов. Полностью соответствует требованиям ACID на уровне всего кластера, предоставляет методы массивной параллельной обработки данных (MPP, Massively Parallel Processing). По своей структуре кластер Postgres-XL состоит из балансировщика нагрузки, узла управления глобальными транзакциями, узлов координации выполнения запросов и узлов хранения данных [11]. ToroDB — это крайне интересная разработка компании 8KData. Ядро СУБД, размещенное поверх PostgreSQL, работает по протоколу MongoDB, что позволяет с легкостью обрабатывать данные, хранящиеся в формате JSON. Каждый уровень JSON-документа формирует отдельную таблицу, поэтому можно сказать, что, наряду с исходными возможностями шардинга MongoDB, обеспечивается еще и партицирование по типам хранимых данных [12].

CitusDB (ранее — pg_shard) является расширением для PostgreSQL, позволяющим распределять таблицы в кластере серверов PostgreSQL. Механизм разбиения данных очень похож на то, что можно наблюдать в MongoDB: есть ключ шардирования (быть его составной частью или полностью совпадать с ним должен первичный ключ распределенной таблицы), по интервалам его значений формируются порции, которые с двукратным реплицированием хранятся на рабочих узлах системы [13]. Управление распределением данных осуществляется с помощью специальных пользовательских функций. Например, запрос вида

SELECT master_create_distributed_table(‘t_item’, ‘item_id’, ‘hash’);

преобразует таблицу t_item в распределенную с ключом шардирования по полю item_id. Разбиение на порции будет осуществлено по диапазонам значений не ключа, а его хэш-преобразования. Следующий запрос сообщает мастеру, что данные таблицы нужно разделить на 16 порций и распределить их между двумя узлами:

SELECT master_create_worker_shards(‘t_item’, 16, 2);

Разумеется, на текущий момент у данного программного продукта существуют определенные ограничения: не поддерживается union и оконные функции, в системе возможно наличие только одного мастера, не являются возможными транзакции, затрагивающие несколько фрагментов данных.

Несмотря на это, в роли реляционного конкурента MongoDB в рамках данной работы будет использоваться именно CitusDB.

После изучения предложений на рынке сервисов по предоставлению услуг в сфере облачных вычислений было принято решение воспользоваться услугами Amazon Elastic Compute Cloud (EC2) [14]. На данный момент это очень развитый ресурс с гибким инструментарием по организации распределенных систем, широким диапазоном конфигураций машин и продолжительной дисконтной тарификацией для новых пользователей, что актуально при проведении длительных исследований. Кроме того, существует большое количество руководств и рекомендаций по работе с данным сервисом.

Каждый отдельный узел организованного в рамках проведения экспериментов кластера EC2 представлен инстансом t2.large.

Как и прежде, в эксперименте по вставке записей (см. рис. 8) в безусловной выборке всех значений таблицы, а также в выборках с условным оператором с наличием индекса и без него использовалась сущность t_item.

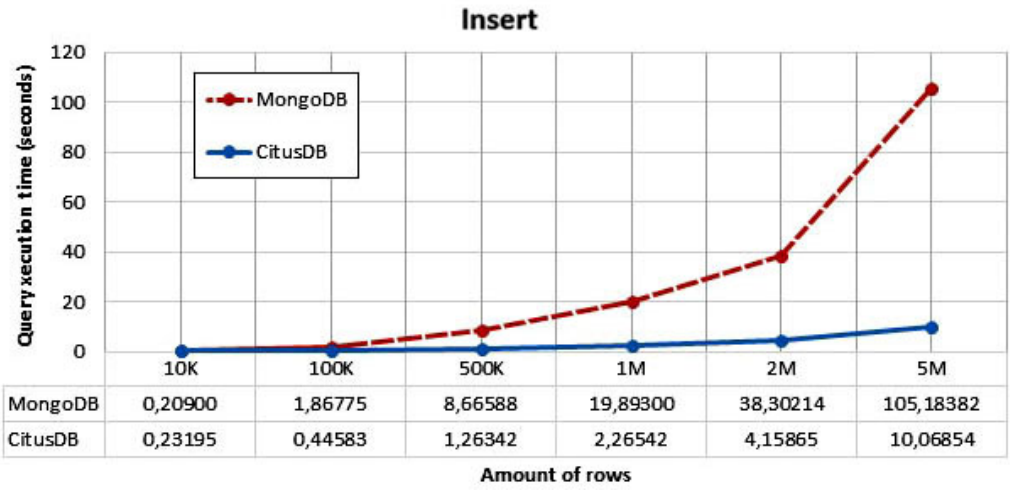


Рис. 8. Сравнение времени выполнения операций вставки данных

Для экспериментов с индексом (см. рис. 9) измерялось время выполнения операций с условием выборки для одних и тех же записей с наличием построенного посоответствующему полю индекса и без него.

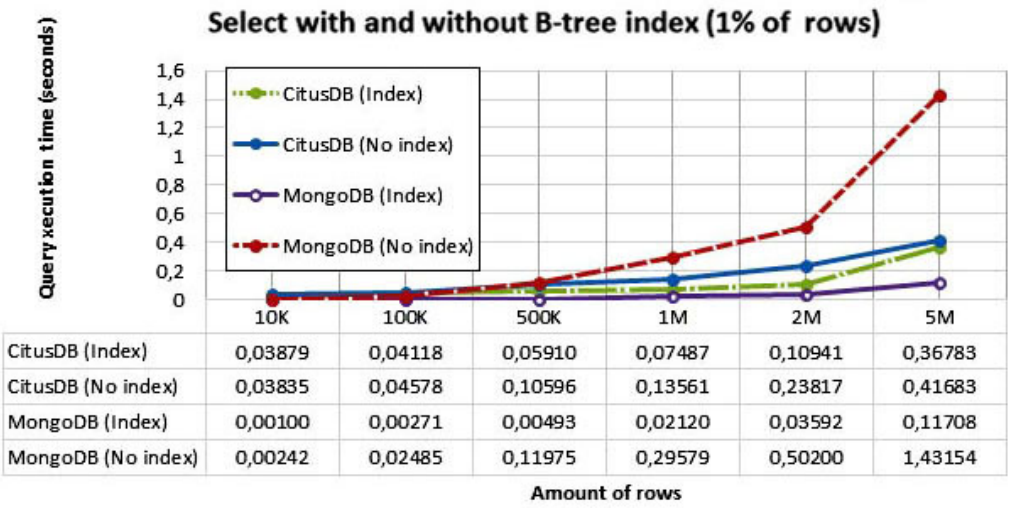
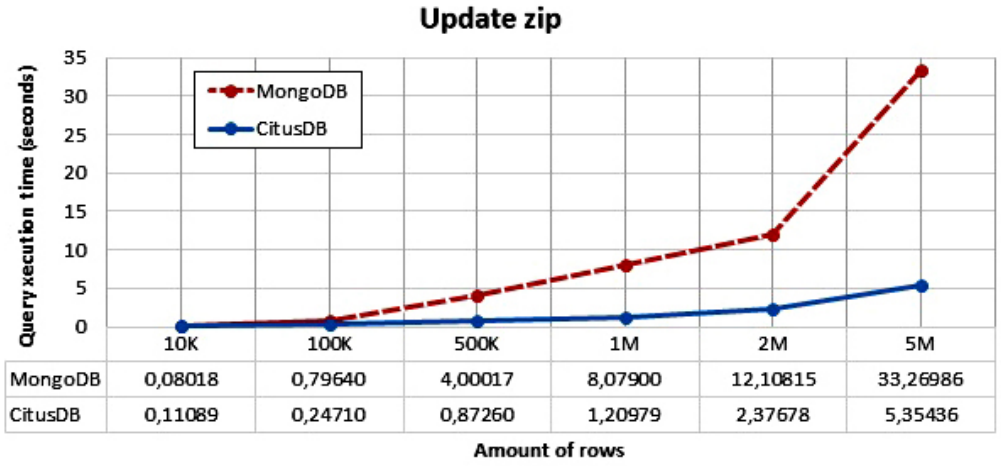


Рис. 9. Сравнение времени выполнения операций выборки данных с индексом и без использования индекса

Использовалось условие по полю item_price типа float. Результаты эксперимента по обновлению данных показаны на рис. 10.

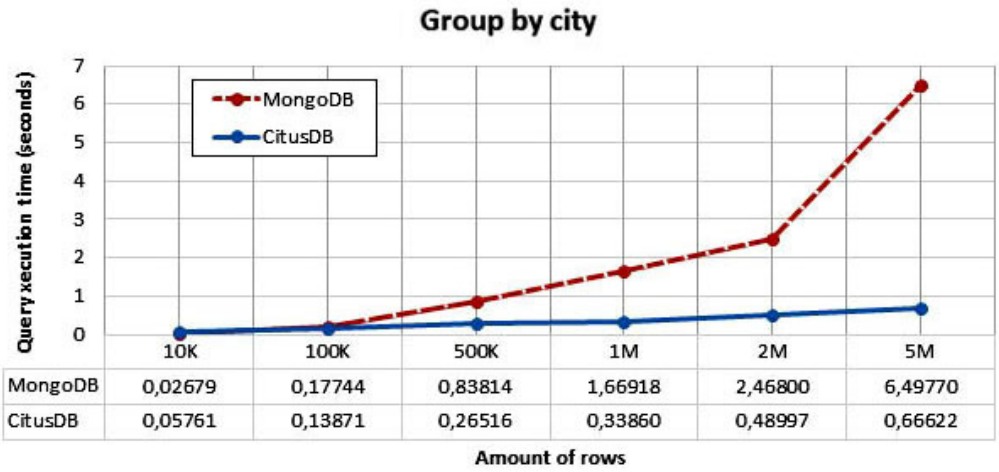
Следующий эксперимент (см. рис. 11) основан на запросе, вычисляющем, сколько пользователей ресурса представляют тот или иной город (группировать данные, результат отсортировать по убыванию агрегирующей функции).

С учетом вывода предыдущей главы было принято решение не проводить экспериментов по присоединению таблиц. Кроме того, опция не поддерживается при работе с шардированными коллекциями, не говоря даже о том, что в проведенных ранее экспериментах на одном узле $lookup оказался в разы медленнее традиционного join'а в PostgreSQL. Как и прежде, наблюдается лидерство SQL СУБД в большей части задач.



	10K	100K	500K	1M	2M	5M
MongoDB	0,08018	0,79640	4,00017	8,07900	12,10815	33,26986
CitusDB	0,11089	0,24710	0,87260	1,20979	2,37678	5,35436

Рис. 10. Сравнение времени выполнения операции обновления данных



	10K	100K	500K	1M	2M	5M
MongoDB	0,02679	0,17744	0,83814	1,66918	2,46800	6,49770
CitusDB	0,05761	0,13871	0,26516	0,33860	0,48997	0,66622

Рис. 11. Сравнение времени выполнения операций выборки с группировкой

5. ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ С JSON-ДОКУМЕНТАМИ

JSON — открытый формат обмена данными, основанный на JavaScript [15]. Информация в JSON представлена в виде пар «ключ-значение». Ключом является строка, в качестве значения могут использоваться строки, цифры, литералы (true, false, null), а также массивы таких элементов и вложенные JSON-документы. JSON определяет набор правил построения документов, например, фигурные скобки ({ }) обозначают границы отдельного JSON-документа, квадратные ([ ]) — границы массива значений, а кавычки (" ") — строку. Кодировкой по умолчанию является UTF-8. JSON не зависит от используемого языка программирования и хорошо подходит для чтения как компьютером, так и человеком. Основной сферой использования JSON является обмен данными между браузером и веб-сервером, а также межсерверная передача данных. MongoDB хранит документы в формате BSON, расширяющем JSON. В качестве значений здесь также могут использоваться даты, двоичные данные и регулярные выражения [16]. Кроме того, на диске BSON-документы представляются в двоичном виде, а не в текстовом, как JSON. Что каса-

ется PostgreSQL, то данная СУБД поддерживает работу с JSON уже в течение нескольких лет [17]. В версии 9.3 было реализовано большое количество функций по работе с такими документами, однако скорость их выполнения была довольно низкой. В версии 9.4 был представлен специальный формат JSONB, крайне похожий на BSON. Он также является двоичным (благодаря этому, оказывается возможным отбросить пробелы между полями документа), сортировка полей в рамках документа не сохраняется (вместо этого положение поля рассчитывается на основе хэш-функции), допускается построение индекса по отдельным полям документа. Вследствие предобработки документов JSONB может быть медленнее оригинального JSON на операциях вставки, но обработка и извлечение данных будут осуществляться быстрее [18].

Как и прежде, серия экспериментов начинается с операций вставки данных (см. рис. 12). Исходя из специфики используемой нагрузки, преимущество PostgreSQL представляется довольно неожиданным. Далее следует обновление коллекций JSON-документов (см. рис. 13). В отличие от предыдущего эксперимента, на этот раз наблюдается преимущество MongoDB.

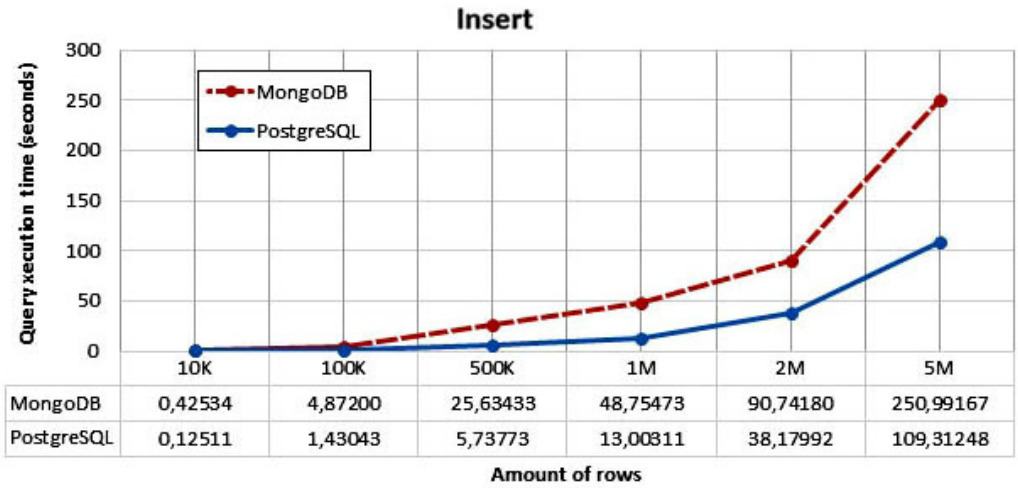


Рис. 12. Сравнение времени выполнения операции вставки данных

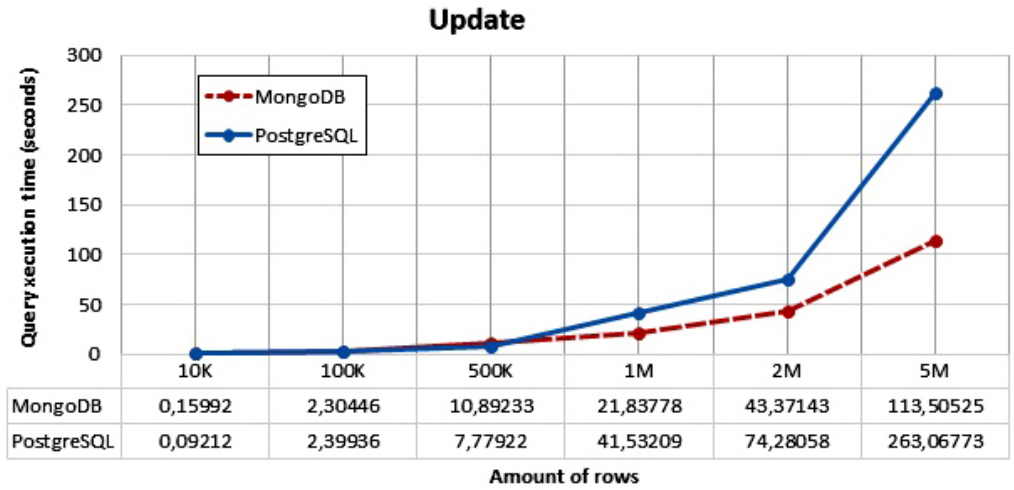
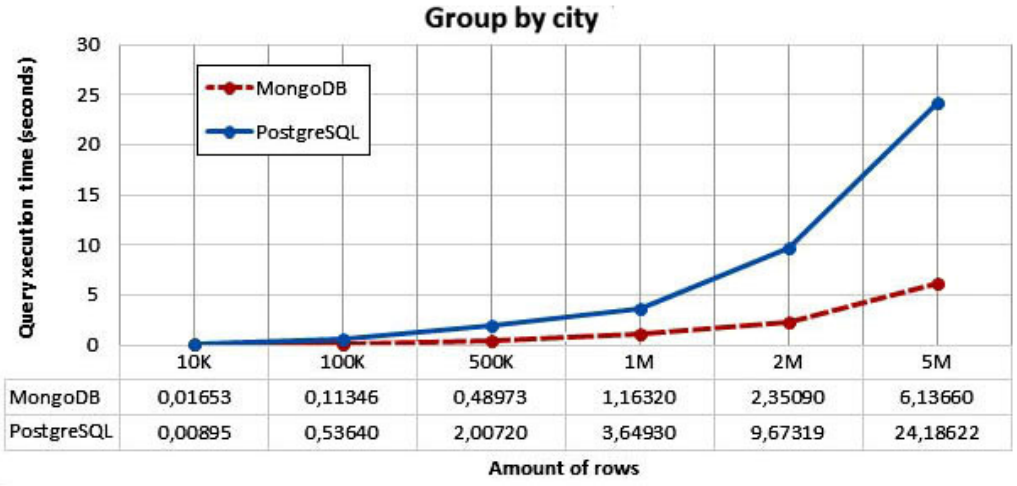


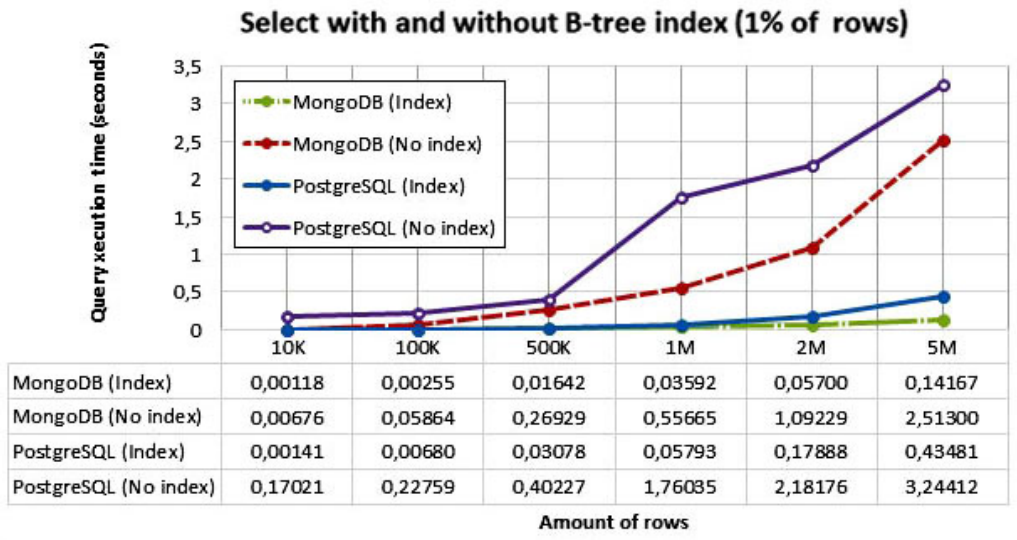
Рис. 13. Сравнение времени выполнения операции обновления данных

Группировка данных, несмотря на аналитический характер такого запроса, наилучшим образом удается MongoDB (см. рис. 14). Наконец, результаты эксперимента с использованием индекса показаны на рис. 15.



	10K	100K	500K	1M	2M	5M
MongoDB	0,01653	0,11346	0,48973	1,16320	2,35090	6,13660
PostgreSQL	0,00895	0,53640	2,00720	3,64930	9,67319	24,18622

Рис. 14. Сравнение времени выполнения операции группировки данных



	10K	100K	500K	1M	2M	5M
MongoDB (Index)	0,00118	0,00255	0,01642	0,03592	0,05700	0,14167
MongoDB (No index)	0,00676	0,05864	0,26929	0,55665	1,09229	2,51300
PostgreSQL (Index)	0,00141	0,00680	0,03078	0,05793	0,17888	0,43481
PostgreSQL (No index)	0,17021	0,22759	0,40227	1,76035	2,18176	3,24412

Рис. 15. Сравнение времени выполнения операций выборки данных по условию с индексом и без индекса в документах JSON

6. ЗАКЛЮЧЕНИЕ

В процессе подготовки данной статьи были решены задачи обзора существующих исследований схожей тематики, подготовки тестовых данных, ознакомления с техническими аспектами работы с СУБД PostgreSQL, MongoDB, распределенными средами и обертками внешних данных. Анализ результатов соответствующих экспериментов привел к следующим выводам:

1. PostgreSQL опережает MongoDB на операциях вставки данных даже в распределенном окружении и с использованием JSON-документов, что опровергает существующее среди специалистов мнение о преимуществе MongoDB в задачах логирования информации.
2. MongoDB функционирует быстрее PostgreSQL в подавляющем большинстве задач по работе с JSON-документами, однако в целом разница во временных показателях не настолько значительная, насколько это можно было бы ожидать при сравнении реляционной СУБД и СУБД, для которой JSON-документ является фактически единственным форматом представления и хранения данных.
3. Результаты экспериментов показали лидерство MongoDB в индексированном поиске. Таким образом, применение данной СУБД является оправданным для хранения редко изменяющихся и часто читаемых данных (например, в базах знаний, словарях, справочниках).
4. Денормализация данных как способ превентивного присоединения кортежей в MongoDB является вполне допустимым решением задачи, что особенно ярко проявляется на больших объемах данных. Однако стоит отметить, что решение о денормализации нужно принимать исходя из оценок затрат на поддержание согласованности.

MongoDB оказалась достойным конкурентом PostgreSQL в отдельных задачах с точки зрения времени выполнения запросов. Этот результат подтверждает существующее среди специалистов в сфере СУБД мнение о том, что MongoDB является одним из наиболее сильных представителей семейства NoSQL систем. В то же самое время эксперименты показали, что, несмотря на заверения разработчиков и евангелистов данной СУБД, на сегодняшний день нельзя говорить о ней как о заменителе более традиционных систем обработки и хранения данных.

Список литературы

1. *Strozzi C.* NoSQL: A Relational Database Management System [Электронный ресурс]. URL: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page (дата обращения: 15.06.2017).
2. *Бэнкер К.* MongoDB в действии. М.: ДМК Пресс, 2012.
3. MongoDB Production Deployments [Электронный ресурс]. URL: http://www.mongodb.org/about/production-deployments (дата обращения: 15.06.2017).
4. *Plattner H.* A common database approach for OLTP and OLAP using an in-memory column database // Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. P. 1–2.
5. *Abramova V. et al.* Which NoSQL Database? A Performance, Overview // Open Journal of Databases (OJDB), Vol. 1, Is. 2, 2014. Lübeck: RonPub. P. 17–24.
6. *Cattell R.* Scalable SQL and NoSQL data stores // ACM SIGMOD Record, Vol. 39 Is. 4, December 2010. NY: ACM New York. P. 12–27.
7. *Parker Z. et al.* Comparing NoSQL MongoDB to an SQL DB, Proceedings of the 51st ACM Southeast Conference. NY: ACM New York, 2013.
8. Postgres Outperforms MongoDB and Ushers in New Developer Reality [Электронный ресурс]. URL: http://www.enterprisedb.com/postgres-plus-edb-blog/marc-linster/postgres-outperforms-mongodb-and-ushers-new-developer-reality/ (дата обращения: 15.06.2017).
9. Data generator: free tool to generate test data. [Электронный ресурс]. URL: http://www.yandataellan.com/ (дата обращения: 15.06.2017).
10. Release Notes for MongoDB 3.2 — MongoDB Manual 3.2 [Электронный ресурс]. URL: https://docs.mongodb.org/manual/release-notes/3.2/#aggregation-framework-enhancements (дата обращения: 15.06.2017).

11. Overview. Postgres-XL [Электронный ресурс]. URL: http://www.postgres-xl.org/overview/ (дата обращения: 15.06.2017).
12. ToroDB. 8KData [Электронный ресурс]. URL: https://www.8kdata.com/torodb (дата обращения: 15.06.2017).
13. Доклады конференции PgConfRussia 2016: CitusDB: расширение для масштабирования PostgreSQL [Электронный ресурс]. URL: http://www.postgres-xl.org/overview/ (дата обращения: 15.06.2017).
14. Облачные серверы и хостинг Elastic Compute Cloud (EC2) — AWS [Электронный ресурс]. URL: http://aws.amazon.com/ru/ec2/ (дата обращения: 15.06.2017).
15. Internet Engineering Task Force (IETF). The JavaScript Object Notation (JSON) Data Interchange Format [Электронный ресурс]. URL: https://tools.ietf.org/html/rfc7159 (дата обращения: 15.06.2017).
16. JSON and BSON | MongoDB [Электронный ресурс]. URL: https://docs.mongodb.com/v3.0/reference/bson-types/ (дата обращения: 15.06.2017).
17. *Lerner R. M.* At the forge: PostgreSQL, the NoSQL database // Linux Journal. 2014. № 247. P. 3–6.
18. PostgreSQL 9.6: Documentation: 9.6: JSON Types [Электронный ресурс]. URL: https://www.postgresql.org/docs/9.6/static/datatype-json.html/ (дата обращения: 15.06.2017).

Поступила в редакцию 16.06.2017, окончательный вариант — 21.07.2017.

Computer tools in education, 2017
№ 4: 48–63
http://ipo.spb.ru/journal

COMPARATIVE ANALYSIS OF THE PERFORMANCE OF SQL AND NOSQL DBMS

Novikov B. A.[1,2], Levin M. Y.[1]

[1]Saint Ptersburg state University, Saint Petersburg, Russia
[2]Aalto University School of Science, Espoo, Finland

Abstract

NoSQL database management systems have been under examination by industry specialists for quite some time. However, at the moment there are very few works connected with this topic, and in particular — with the comparison of such systems with traditional relational DBMSs. As for the existing studies, some articles are plain overviews, some use a small number of records in tables as a workload or they only consider one environment for conducting experiments, which can give advantage to one of the comparison objects. This paper is concerned with PostgreSQL and MongoDB. The first system, despite being free and open-source software, is rapidly gaining popularity even in the corporate world, whereas the second one differs from a large number of NoSQL solutions in its meticulousness and support. In order to ensure the completeness of the comparison, the experiments were carried out in different environments and with different loads.

Keywords: *perfomance, analysis, database management systems, query execution time, MongoDB, NoSQL, OLAP, OLTP, PostrgeSQL, SQL.*

Citation: B. A. Novikov & M. Y. Levin, "Sravnitel'nyi analiz proizvoditel'nosti SQL i NoSQL SUBD" [Comparative Analysis of the Performance of SQL and NOSQL DBMS], *Computer tools in education*, no. 4, pp. 48–63, 2017 (in Russian).

Received 16.06.2017, the final version — 21.07.2017.

Boris A. Novikov, D.Sc., proffesor, Dept. Analytical Information systems, borisnov@acm.org
Maksim Yu. Levin, PhD student, Dept. Analytical Information systems, levin-m@list.ru

Новиков Борис Асенович,
доктор физико-математических наук,
профессор, профессор кафедры
информационно-аналитических систем,
приглашенный лектор,
borisnov@acm.org

Левин Максим Юрьевич,
аспирант кафедры
информационно-аналитических систем,
levin-m@list.ru