Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Кафедра информационно-аналитических систем

Левин Максим Юрьевич

Сравнительный анализ производительности СУБД с использованием OLAP и OLTP запросов

Магистерская диссертация

Научный руководитель:

д.ф.-м.н., профессор Новиков Б.А.

Рецензент:

к.ф.-м.н. Соловьев А.А.

Санкт-Петербург

2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Sub-Department of Analytical Information Systems

Levin Maksim

Comparative analysis of DBMS performance using OLAP and OLTP queries

Master's Thesis

Scientific supervisor:

D.Sc., professor Novikov B.

Reviewer:

Ph.D., Solovev A.

Saint-Petersburg

2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Последнее десятилетие связано с появлением и стремительным развитием класса систем управления данными под названием NoSQL. Впервые этот термин использовал Карло Строцци в 1998 году для названия небольшой СУБД его собственной разработки, предназначенной для работы в контексте распределенных архитектур. В ней он отказался от языка SQL и поддержки ACID-транзакций (Atomicity, Consistency, Isolation, Durability – Атомарность, Согласованность, Изолированность, Постоянство хранения) [1]. На данный момент чаще всего это слово применяют в смысле «Not-Only-SQL», т.е. «Не-Только-SQL».

Одной из главных причин популярности такого рода систем является необходимость расширения вычислительных архитектур «по горизонтали», т.е. не наращивания мощи отдельных узлов, а добавления новых к уже существующим. Представить работу традиционных SQL-, а точнее реляционных систем в таких условиях довольно трудно из-за сложностей реализации транзакций в распределенной среде.

Несмотря на то, что NoSQL-системы существуют довольно продолжительное время, на сегодняшний день существует относительно небольшое число исследований на тему сравнения их производительности с реляционными системами. Имеющиеся работы зачастую не позволяют получить полной картины, т.к. 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] – документная СУБД, разрабатываемая однойименной компанией (ранее – 10gen). MongoDB имеет широкий функционал и на

данный момент является одной из самых популярных NoSQL систем. MongoDB позволяет оперировать JSON-документами, хранящимися в коллекциях, которые являются аналогом привычных SQL-таблиц. Для работы с документами предусмотрены операции поиска, вставки, удаления и обновления. Для поиска документов в коллекции используется метод запросов по образцу, поддерживаются сортировка, проекция, просмотр результатов запроса с помощью курсора. 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»: операции записи обрабатываются только главным узлом, а 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. Эта система 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.

Целью работы является исследование производительности баз данных PostgreSQL и MongoDB с точки зрения выполнения запросов к хранимым данным для определения типов задач, в которых применение данных СУБД является оправданным.

Поэтому были обозначены следующие задачи:

- изучение технической документации по обозначенным СУБД и имеющихся исследований схожей тематики;
- проектирование структуры таблиц/коллекций, в которых будут храниться данные;
- заполнение полученной структуры данными;
- проведение экспериментов на основе различных запросов с измерением времени выполнения;
- анализ результатов.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ

В этой главе осуществляется рассмотрение некоторых иных исследований из области сравнения производительности SQL и NoSQL систем управления базами данных.

В работе группы ученых из университета Алабамы в качестве конкурента продукта компании MongoDB использовался Microsoft SQL Server Express [5]. 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 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 PostgresSQL в плане улучшения производительности в сравнении с MongoDB благодаря внедрению нового типа данных JSONB, призванного ускорить доступ к хранимым JSON-документам (Java Script Object Notation – текстовый формат для обмена и хранения данных, представляющий информацию в виде пар «ключ-значение») [6]. 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.

Статья Рика Каттелла (Rick Cattell) носит обзорный характер и делает акцент на горизонтальной масштабируемости как SQL, так и NoSQL систем хранения и обработки данных [7]. 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.

Группа ученых из университетов Коимбры подготовила работу по сравнению MongoDB, Cassandra, HBase, OrientDB и Redis с помощью фреймворка Yahoo! Cloud System Benchmark [8]. Наряду с традиционным многократным выполнениме 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.

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

ГЛАВА 2. 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.).

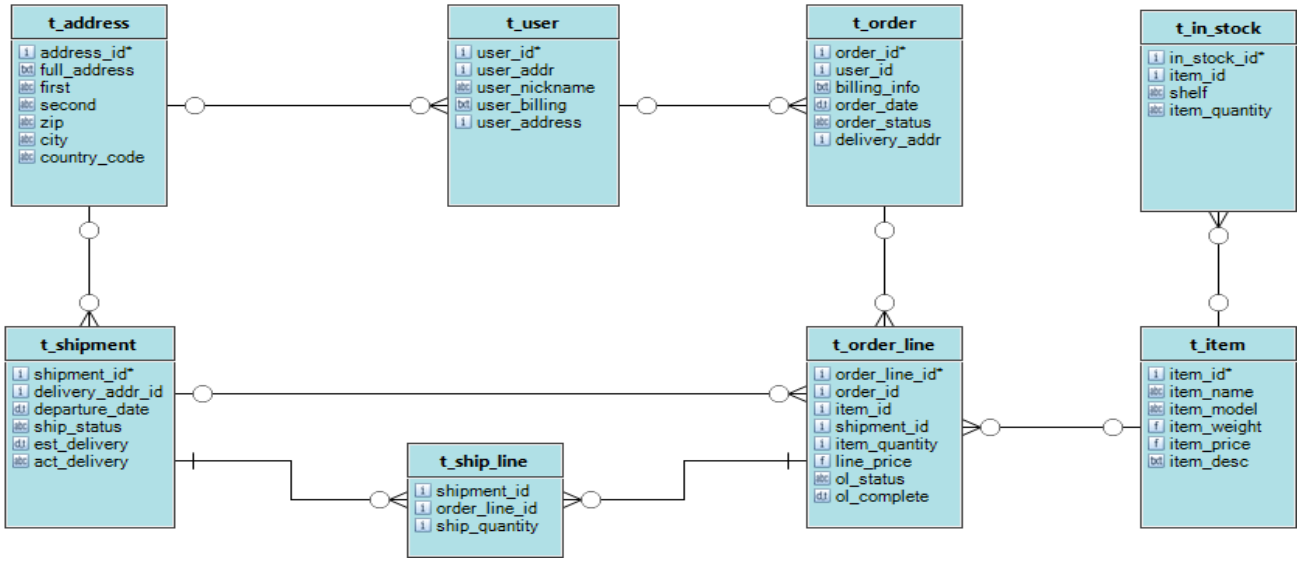


Рис. 1. ER-диаграмма базы данных для PostgreSQL

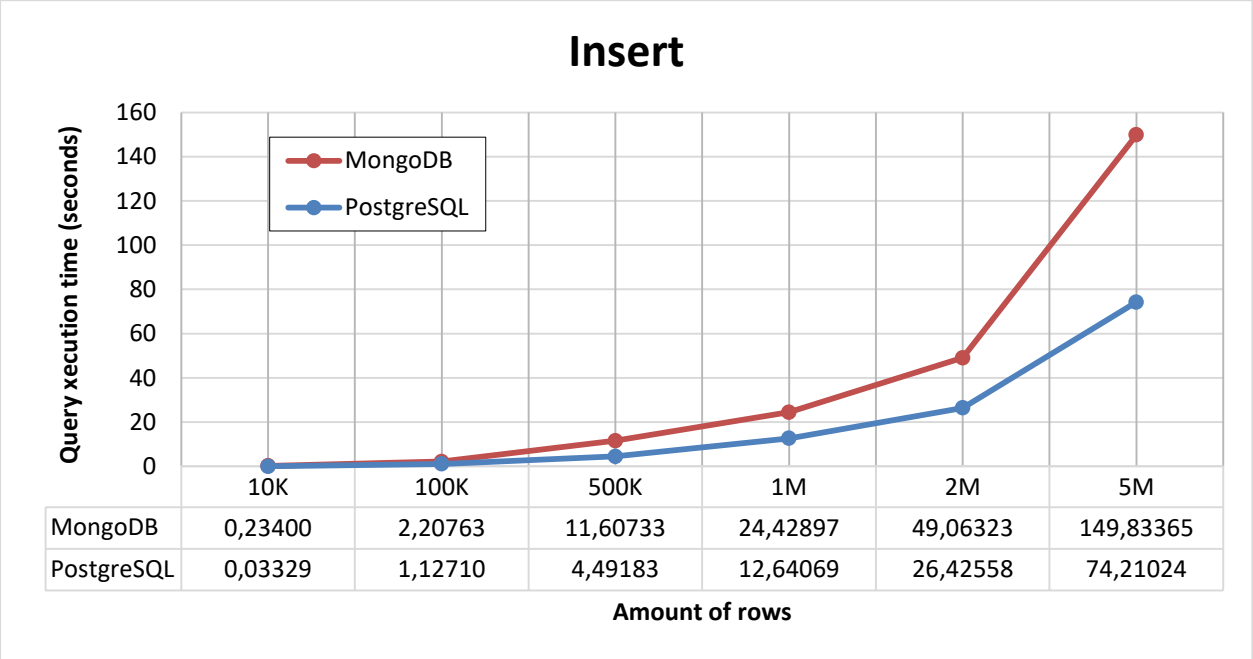
Что касается MongoDB, то она является нереляционной базой данных, а значит строить произвольные запросы по имеющимся данным невозможно. Данную проблему решают, как правило, двумя способами. Первый из них состоит в проектировании коллекций на манер таблиц из реляционных СУБД. Само присоединение при этом 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 «pre-join» 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.



	10K	100K	500K	1M	2M	5M
MongoDB	0,23400	2,20763	11,60733	24,42897	49,06323	149,83365
PostgreSQL	0,03329	1,12710	4,49183	12,64069	26,42558	74,21024

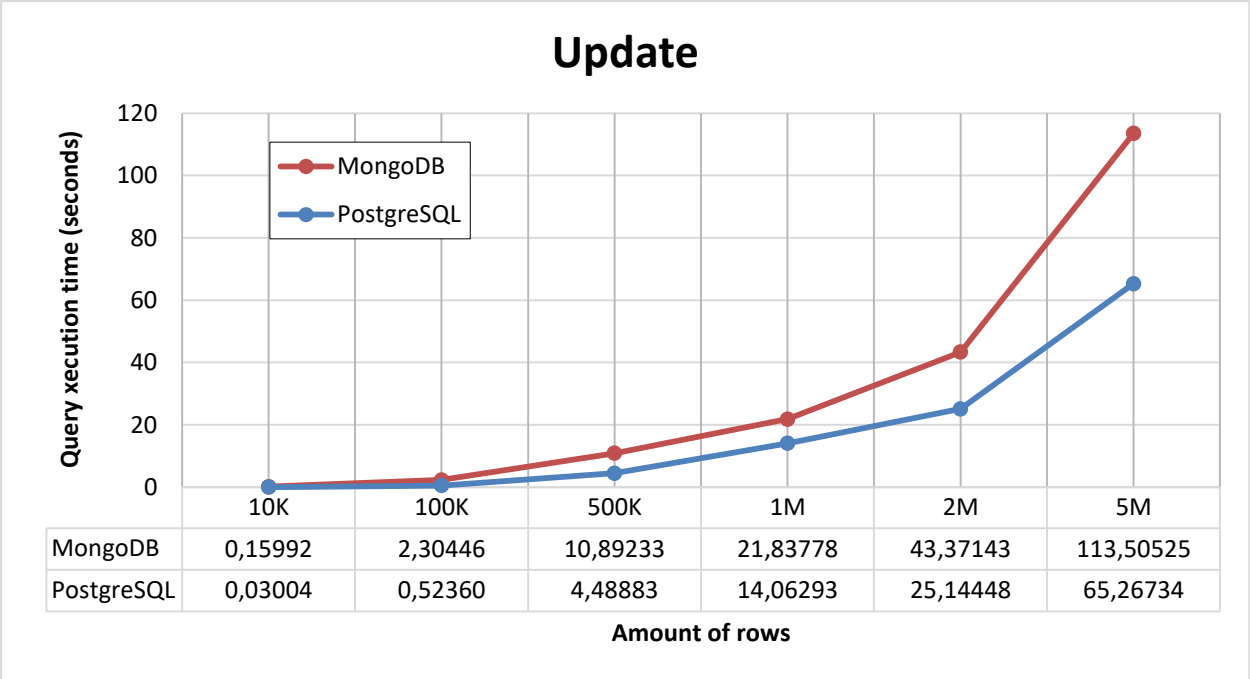
Рис. 2. Сравнение времени выполнения операций вставки

В эксперименте по вставке записей использовалась сущность t_item, хранящая строки вида:

Таблица 1. Пример строки из таблицы t_item.

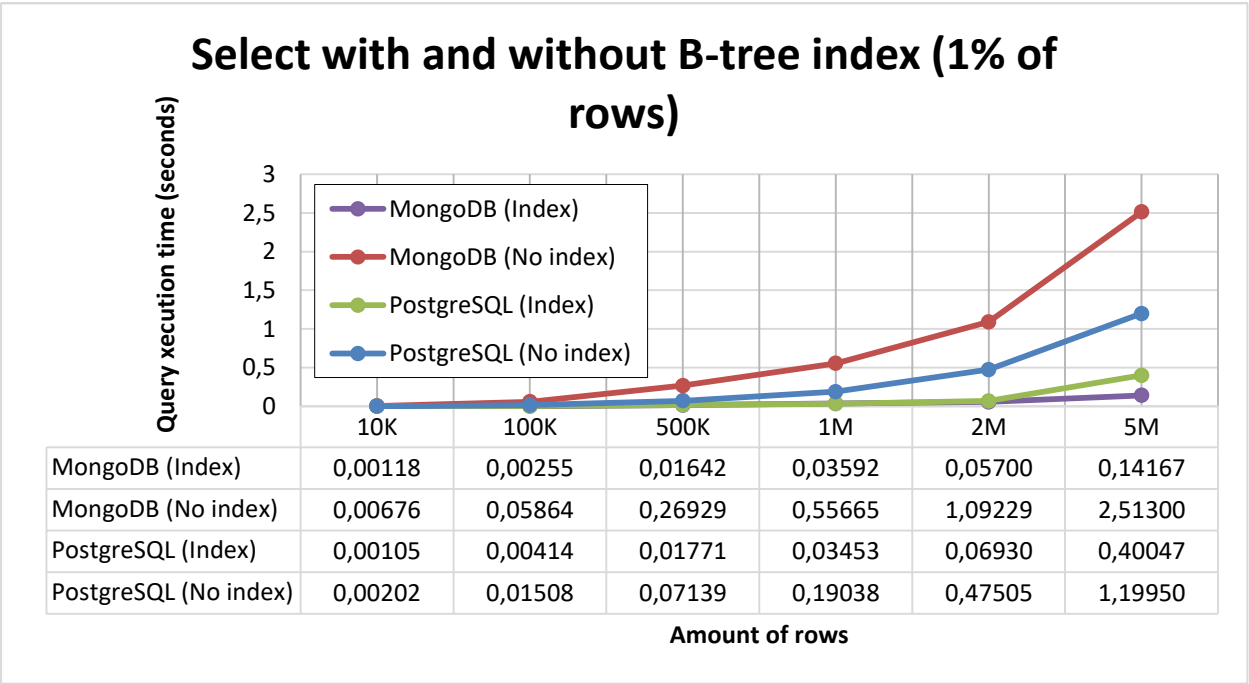
item_id (integer)	item_name varchar(30)	item_model varchar(30)	item_weight float	item_price float	item_desc text
1	vitae consectetuer	adipiscing	52,902	9959,346	ante ipsum primis in faucibus

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



	10K	100K	500K	1M	2M	5M
MongoDB	0,15992	2,30446	10,89233	21,83778	43,37143	113,50525
PostgreSQL	0,03004	0,52360	4,48883	14,06293	25,14448	65,26734

Рис. 3. Сравнение времени выполнения операций обновления



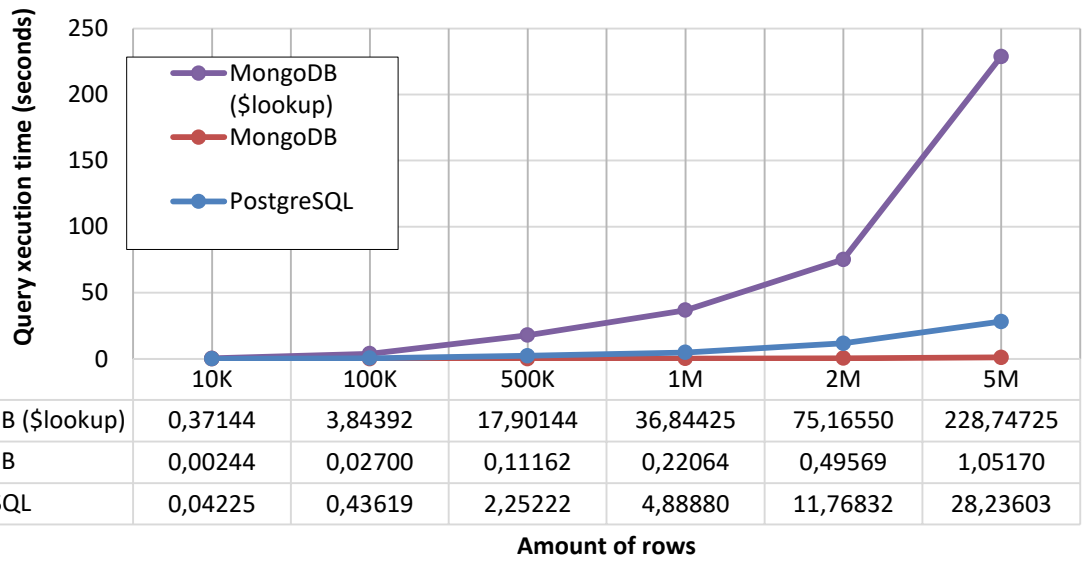
	10K	100K	500K	1M	2M	5M
MongoDB (Index)	0,00118	0,00255	0,01642	0,03592	0,05700	0,14167
MongoDB (No index)	0,00676	0,05864	0,26929	0,55665	1,09229	2,51300
PostgreSQL (Index)	0,00105	0,00414	0,01771	0,03453	0,06930	0,40047
PostgreSQL (No index)	0,00202	0,01508	0,07139	0,19038	0,47505	1,19950

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

Для экспериментов с операцией выборки 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.

Далее 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 ($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

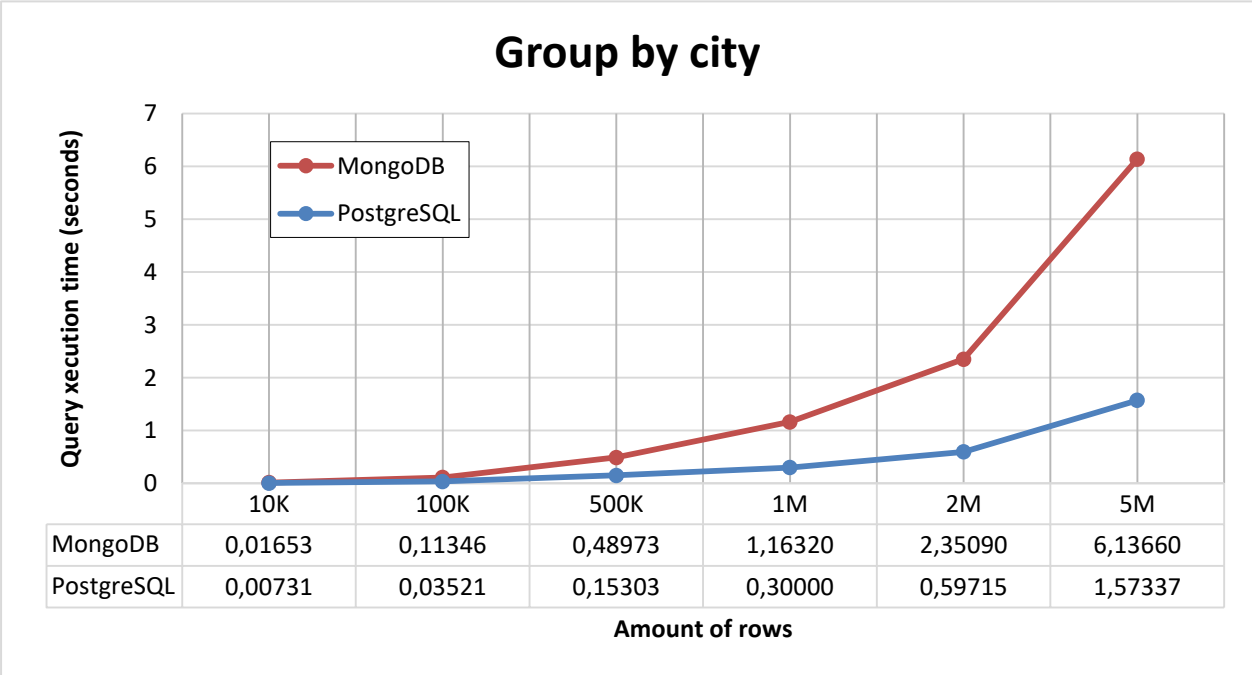
Как уже было упомянуто выше, на сегодняшний день под словом NoSQL понимают не те СУБД, управление которыми осуществляется при помощи языка, не принадлежащего стандарту SQL, а скорее те, которые не являются реляционными. Тем удивительнее видеть, что в версии MongoDB – 3.2 – компания-разработчик предоставила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



	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

Данный эксперимент 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 глава связан с поиском максимального значения поля item_price в таблице t_item.

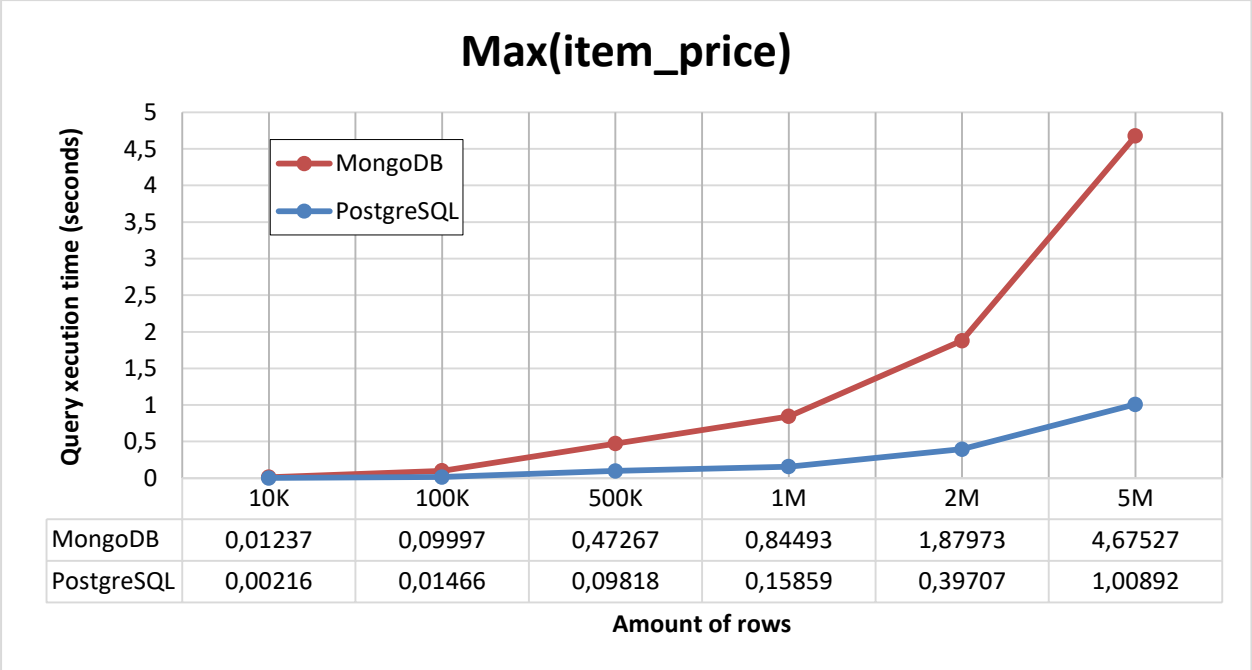


Рис. 7. Сравнение времени выполнения операций поиска максимального значения

Таким образом, эксперименты показывают преимущество PostgreSQL во всех задачах кроме индексированного поиска. 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.

ГЛАВА 3. 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.

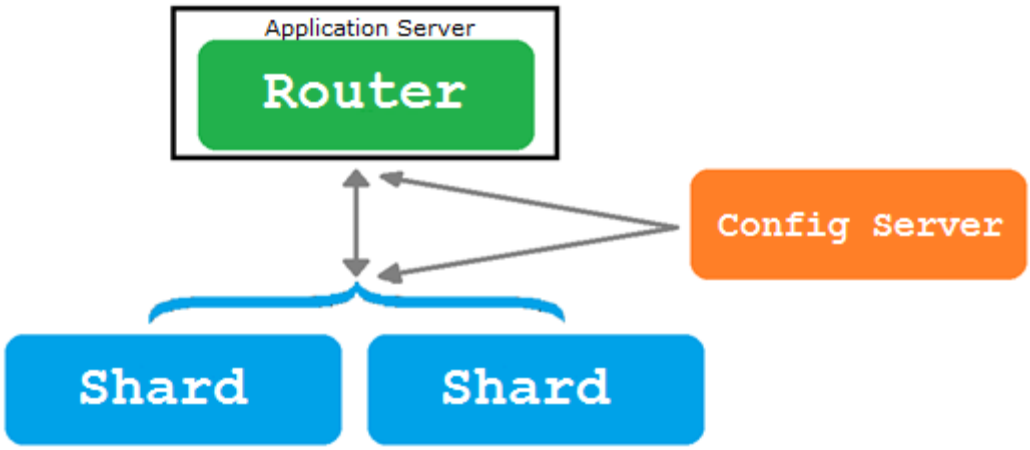


Рис. 1. Пример топологии распределенной сети для MongoDB

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 представить 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 относится. 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, т.е. разбиение таблиц на логические части по выбранным критериям в рамках одной машины, и репликации, заключающейся в построении распределенной системы с рабочими узлами,

хранящими идентичную информацию и обеспечивающими, таким образом, горизонтальное масштабирование операций чтения.

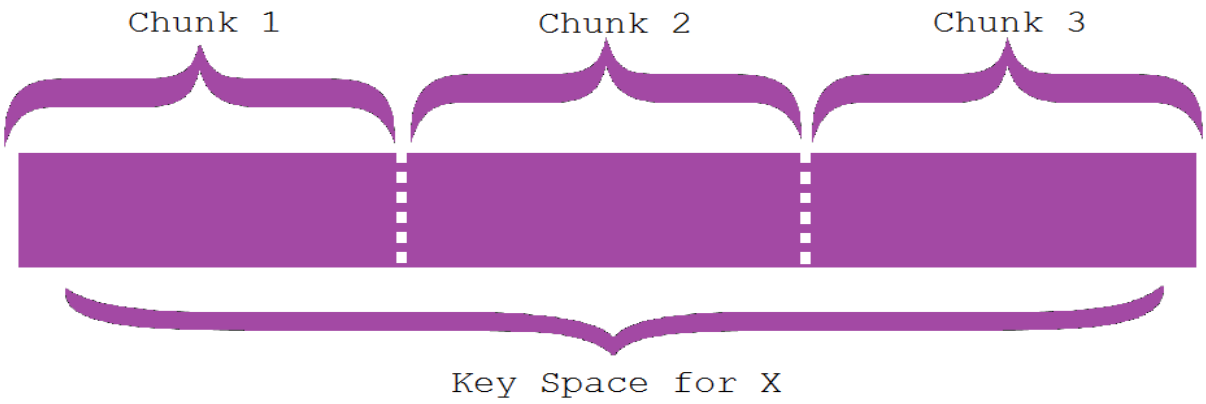


Рис. 2. Иллюстрация механизма получения порций данных из диапазона значений ключа шардирования в MongoDB

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].

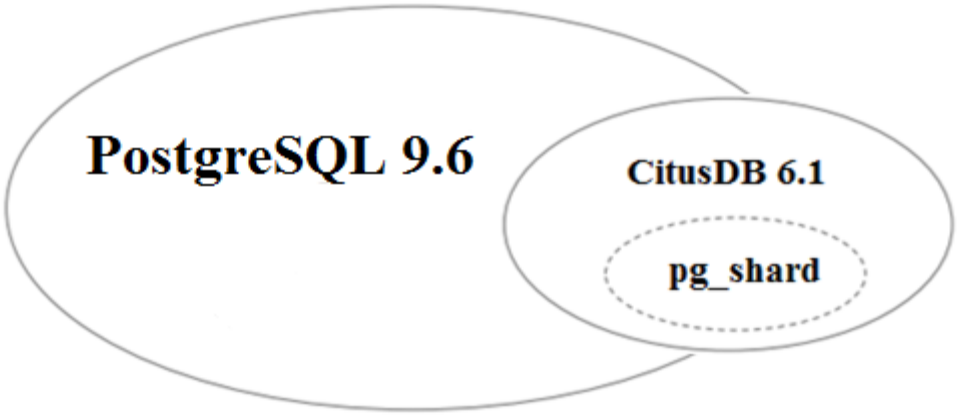


Рис. 3. Диаграмма отношений pg_shard, CitusDB и PostgreSQL

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.

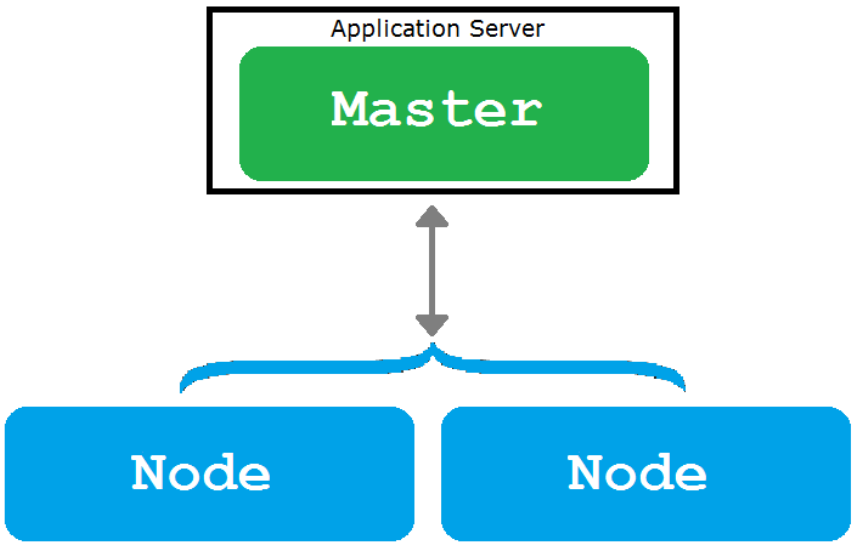


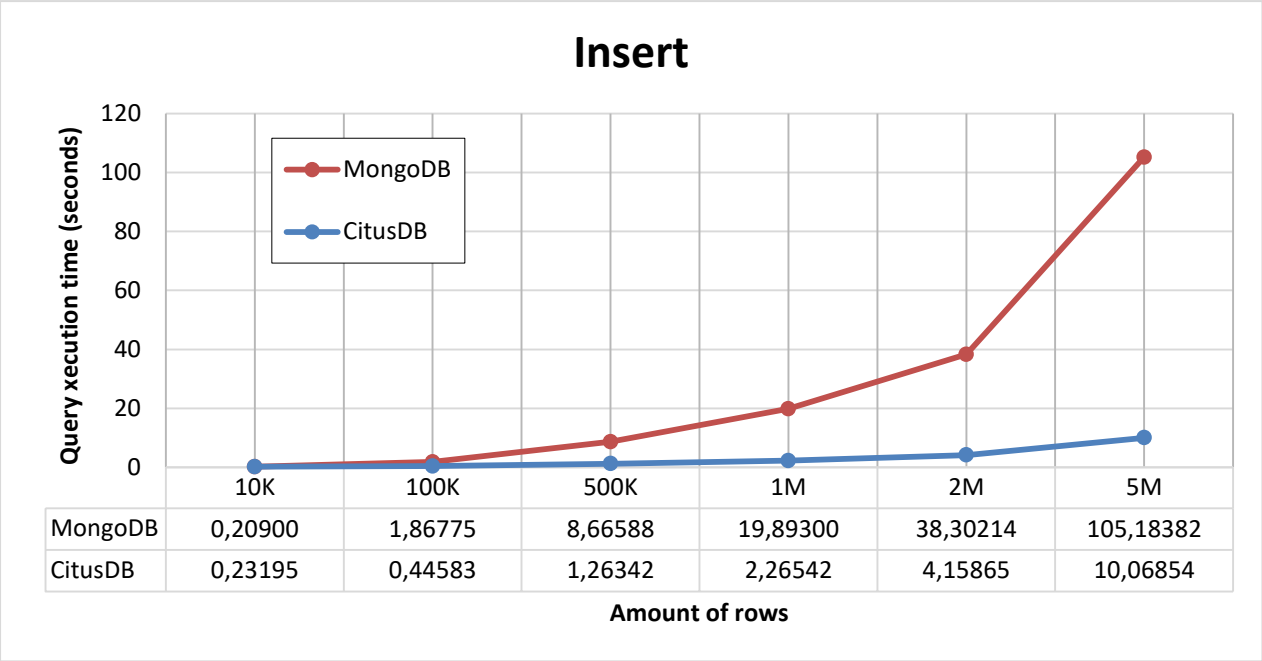
Рис. 4. Пример топологии распределенной сети для CitusDB.

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.

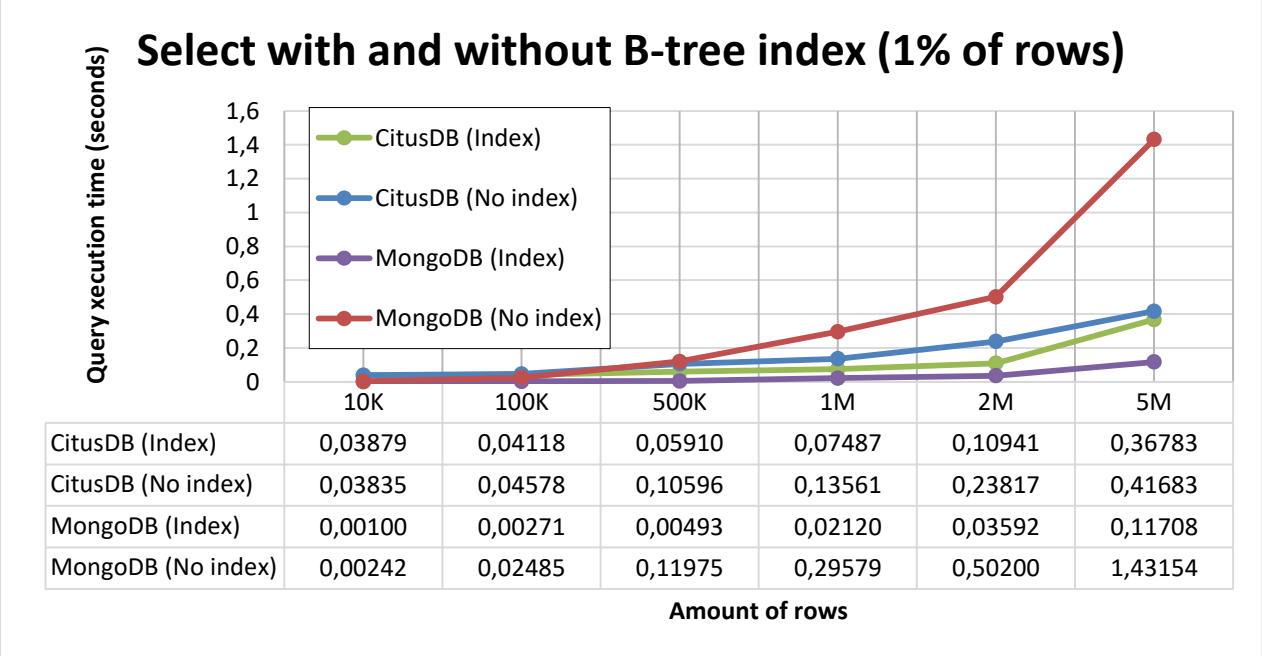
Каждый отдельный узел организованного в рамках проведения экспериментов кластера EC2 представлен инстансом t2.large.

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



	10K	100K	500K	1M	2M	5M
MongoDB	0,20900	1,86775	8,66588	19,89300	38,30214	105,18382
CitusDB	0,23195	0,44583	1,26342	2,26542	4,15865	10,06854

Рис. 5. Сравнение времени выполнения операций вставки данных



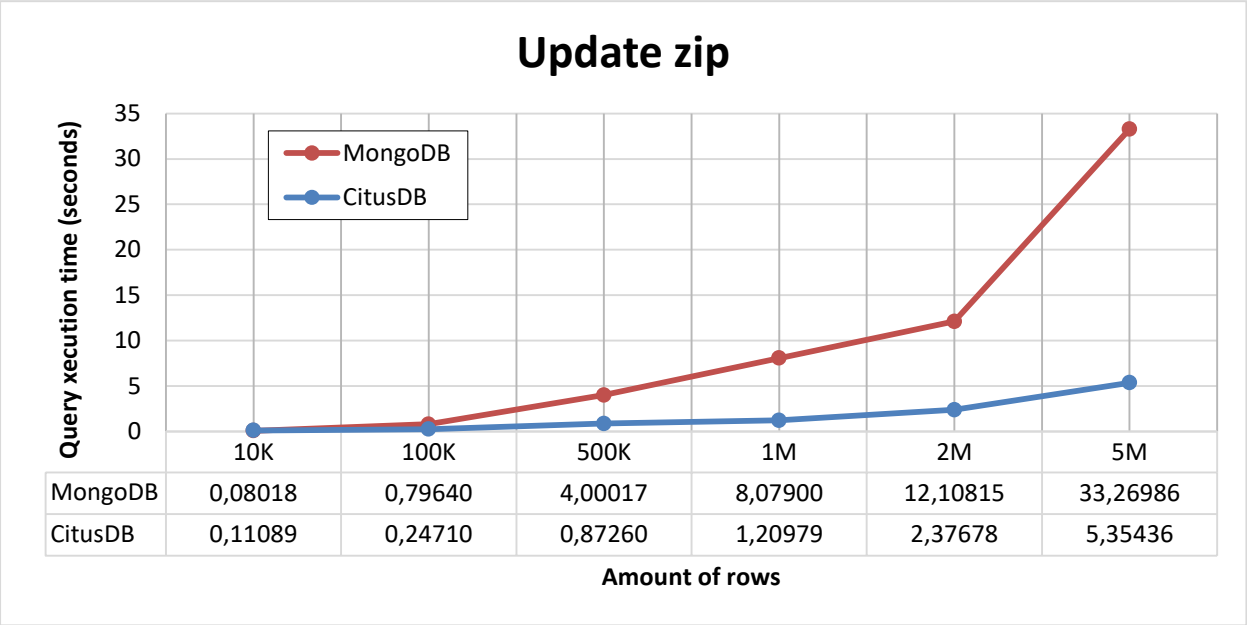
	10K	100K	500K	1M	2M	5M
CitusDB (Index)	0,03879	0,04118	0,05910	0,07487	0,10941	0,36783
CitusDB (No index)	0,03835	0,04578	0,10596	0,13561	0,23817	0,41683
MongoDB (Index)	0,00100	0,00271	0,00493	0,02120	0,03592	0,11708
MongoDB (No index)	0,00242	0,02485	0,11975	0,29579	0,50200	1,43154

Рис. 6. Сравнение времени выполнения операций выборки данных с индексом и без использования индекса

Для экспериментов с индексом применялся следующий сценарий: измерялось время выполнения операций с условием выборки для одних и тех же записей с наличием построенного по соответствующему полю индекса и

без него. Использовалось условие по полю item_price типа float.

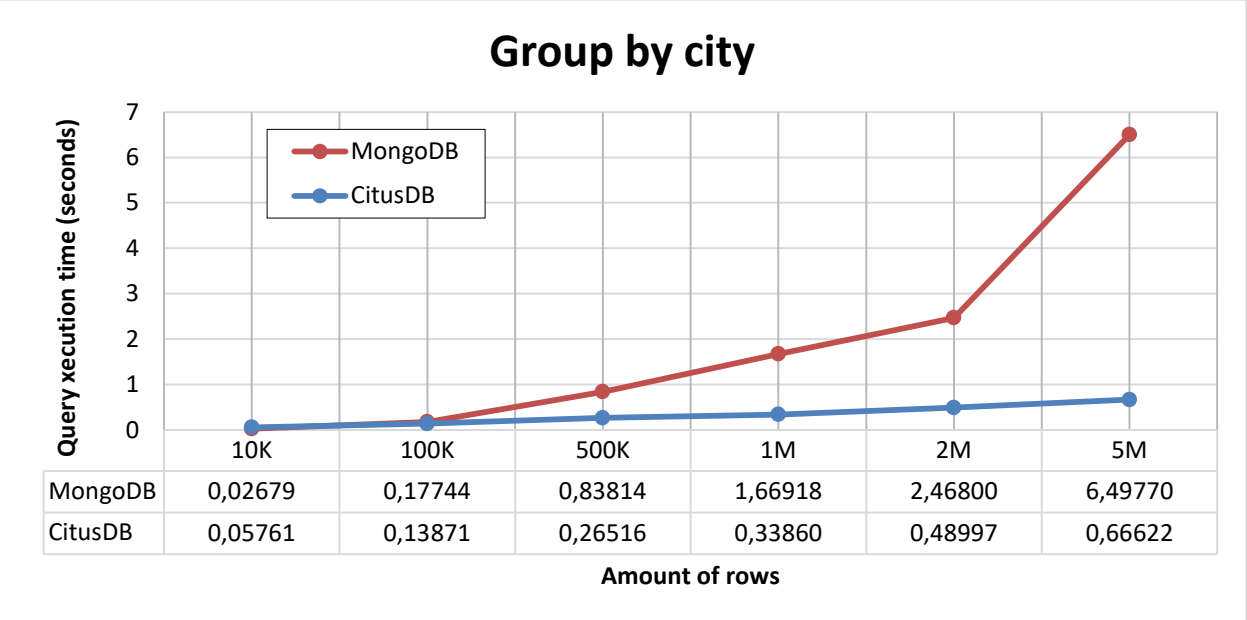
Далее приводится результат экспериментов с операцией выборки с присоединением таблицы (t_user и t_address по полям user_address_id и address_id соответственно).



	10K	100K	500K	1M	2M	5M
MongoDB	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

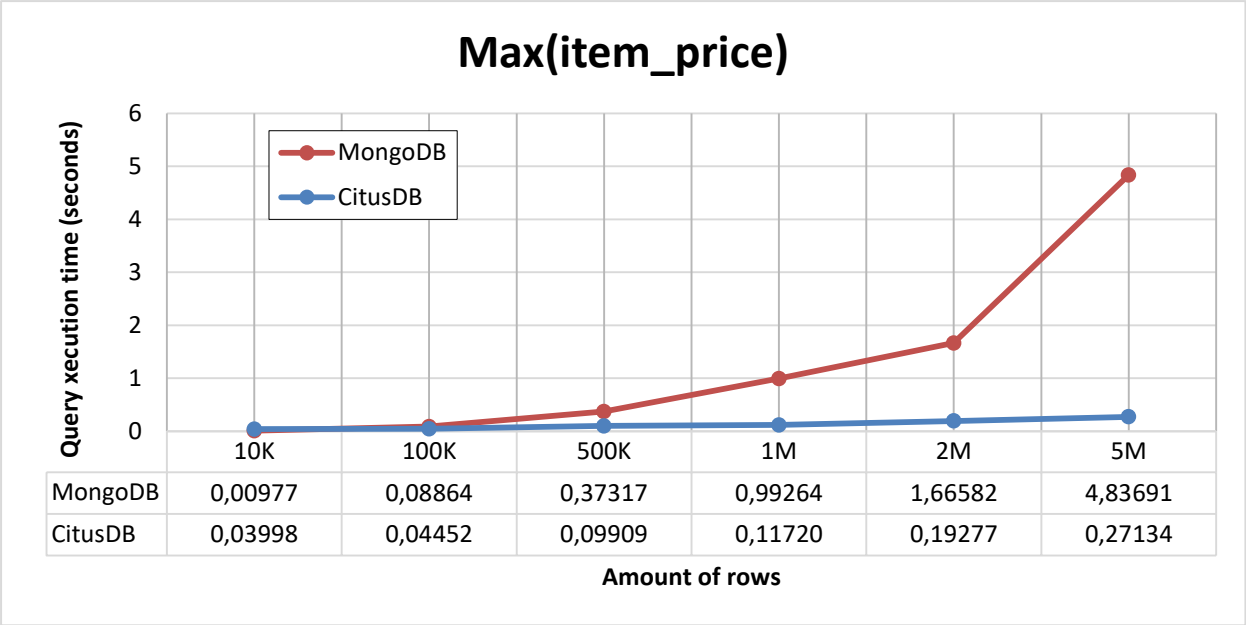
Рис. 7. Сравнение времени выполнения операции обновления данных

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



	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. 8. Сравнение времени выполнения операций выборки с группировкой



	10K	100K	500K	1M	2M	5M
MongoDB	0,00977	0,08864	0,37317	0,99264	1,66582	4,83691
CitusDB	0,03998	0,04452	0,09909	0,11720	0,19277	0,27134

Рис. 9. Сравнение времени выполнения операций поиска максимального значения

С учетом вывода предыдущей главы было принято решение не проводить эксперименты 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.

ГЛАВА 4. КОЛОНОЧНОЕ ХРАНЕНИЕ ДАННЫХ

В течение многих лет реляционные системы управления базами данных являются основой бизнес-приложений в самых разных сферах: финансы, промышленность, доставка товаров, управление кадрами и т.д. Очень часто характерными операциями в подобных системах являются взаимодействия с несколькими атрибутами крайне небольшого количества строк, например, перевод денежных средств с одного счета на другой. Для обеспечения целостности данных средствами используемой СУБД все атомарные операции, обеспечивающие перевод, объединяются в одну неделимую макро-операцию, называемую «транзакцией». Именно поэтому используемые в сфере коммерции информационные системы такого рода называют OLTP-системами (англ. Online Transactional Processing). Данные в них хранятся на диске построчно. Технологии индексирования позволяют очень быстро выбирать отдельные строки, однако время обработки запросов значительно увеличивается с увеличением количества строк. Это объясняет существование систем, которые предназначены для подробного анализа хранимой информации и работают с одним-двумя атрибутами большого строк, имеющихся в таблице. Они называются OLAP-системами (англ. Online Analytical Processing) и впервые появились в семидесятых годах прошлого века. Опуская подробности в виде гиперкубов и предагрегации, стоит сказать, что одна из особенностей их технической организации состоит в том, что столбцы данных, соответствующие атрибутам некоторого отношения, хранятся последовательно и в компактном виде [15]. Благодаря этому аналитические запросы с большим количеством строк и малым количеством атрибутов обрабатываются быстрее. Есть, впрочем, и недостатки, к ним относятся проблемы записи, а также формирование кортежей из отдельных столбцов (например, при осуществлении операций соединения). Критически

важными в OLAP-системах являются технологии сжатия, применение которых обусловлено однородностью данных в рамках одного столбца.

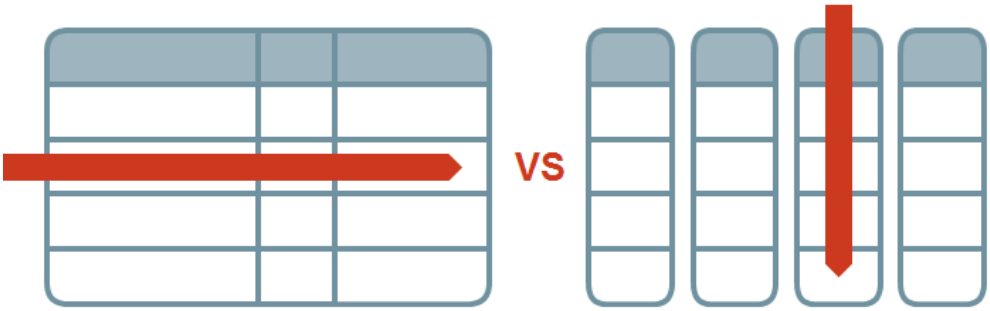


Рис. 1. Иллюстрация построчного и колоночного хранения данных

В производственных условиях специализированные OLAP-хранилища данных довольно часто являются отдельными системами и не зависят от привычных реляционных СУБД. Подобная независимость оборачивается накладными расходами в виде регулярного копирования данных из «обычной» базы данных в «аналитическую». Это обстоятельство для многих профессионалов области является достаточным аргументом для того, чтобы и OLTP-, и OLAP-задачи решать в рамках одной системы.

Обертка сторонних данных (англ. Foreign data wrapper) вошла в стандарт SQL в составе спецификации SQL/MED в 2003 году [16], которая зафиксировала особенности доступа к внешним SQL-хранилищам. В 2011 году, версия PostgreSQL 9.1 позволила читать из любых хранилищ, а версия PostgreSQL 9.3 2013 года отметилась полноценной конфигурацией FDW. Основное предназначение состоит в доступе к данным, хранимым в других источниках (друга реляционная СУБД, не реляционная СУБД или не СУБД вовсее), и манипулирование ими с помощью языка SQL.

Компания CitusData, известная по расширению CitusDB для PostgreSQL, которое обеспечивает горизонтальную масштабируемость соответствующей СУБД, разработала cstore_fdw – специальную обертку сторонних данных, которая реализует колоночное хранилище в рамках PostgreSQL [17]. В основе данного расширения лежит формат данных ORC

(англ. The Optimized Row Columnar), позволяющий снизить размер данных в памяти и на диске в 2-4 раза. Файл соответствующего формата состоит из полос (англ. stripe), в начале и в конце которой содержится вспомогательная информация полоса, служебной информацией заканчивается и весь файл в целом (рис. 2). В качестве ее примера можно привести количество полос в файле, количество строк в полосе и тип каждого из столбцов. Также стоит отметить наличие промежуточных индексов (англ. skip index): для каждой полосы хранятся местный минимум и максимум, что облегчает выполнение запросов такого рода. При разработке cstore_fdw использовался PostgreSQL API для построения оберток сторонних данных, что обеспечило поддержку свыше сорока стандартных типов PostgreSQL, возможность сбора статистики для оптимизатора запросов и простоту установки.

После установки cstore_fdw требуется создать соответствующее расширение SQL-командой, сервер и таблицу, в которой будут храниться данные:

```
CREATE EXTENSION cstore_fdw;

CREATE SERVER cstore_server FOREIGN DATA WRAPPER cstore_fdw;

CREATE FOREIGN TABLE t_item

(

    item_id INTEGER,

    item_name VARCHAR(10),

    item_model VARCHAR(10),

    item_weight INTEGER,

    item_price INTEGER,

    item_description VARCHAR(99),

)

SERVER cstore_server
```

```
OPTIONS(compression 'pglz', stripe_row_count '500000');
```

Опция pglz отвечает за стандартное для PostgreSQL сжатие по алгоритму Лемпеля-Зива [18] (при выполнении запроса разархивируется только столбцы, задействованные в нем, остальная часть таблицы остается неизменной), а stripe_row_count – количество строк таблицы, которое будет соответствовать одной ORC-полосе. Увеличение соответствующего значения увеличивает затраты памяти, но также улучшает производительность. Стоит отметить, что в работе расширения имеются некоторые ограничения: отсутствуют команды удаления и обновления данных, поэтому удалять приходится все отношение в целом.

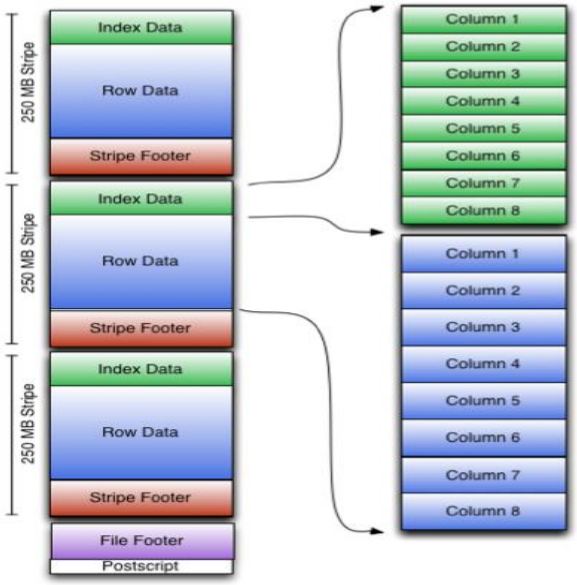
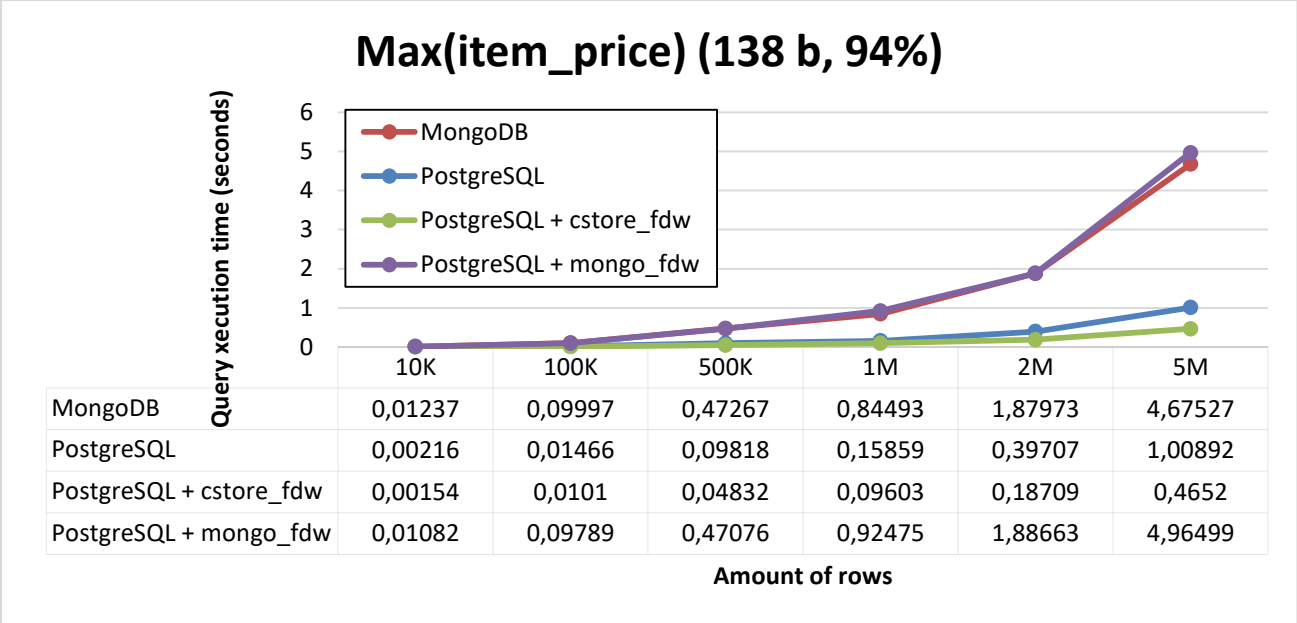


Рис. 2. Устройство формата ORC [19]

В экспериментах данной главы акцент сделан именно на OLAP-направленные задачи. Кроме того, на текущий момент расширение cstore_fdw не поддерживает операции изменения отдельных значений в столбец, именно поэтому из рассмотрения были исключены операции вставки и модификации.

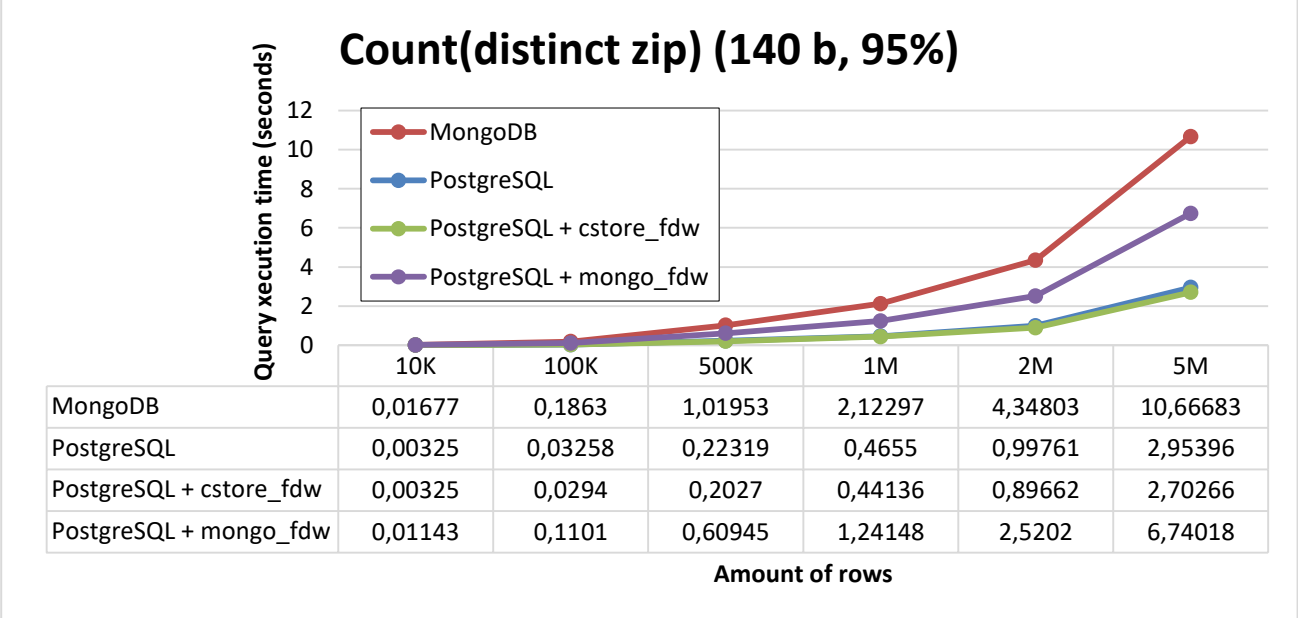
Величины в скобках в названиях диаграмм сообщают количество

информации в байтах и процентах, которые удалось «пропустить» в при работе с модифицированной версией PostgreSQL благодаря колоночной организации хранения информации.



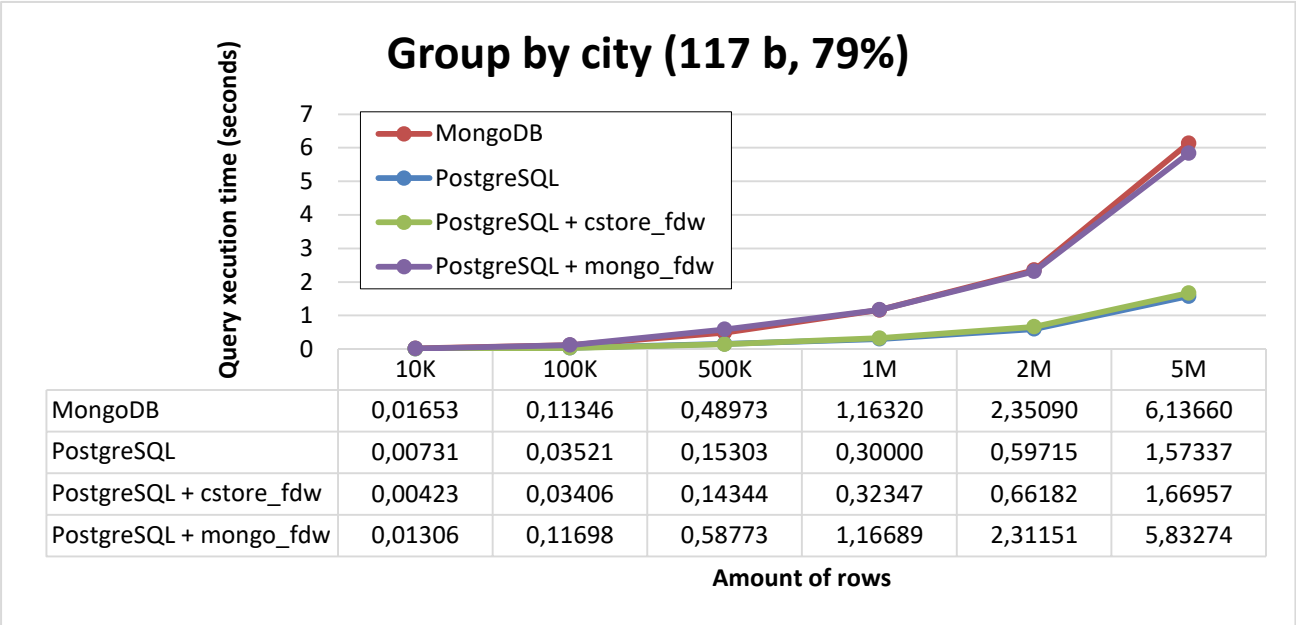
	10K	100K	500K	1M	2M	5M
MongoDB	0,01237	0,09997	0,47267	0,84493	1,87973	4,67527
PostgreSQL	0,00216	0,01466	0,09818	0,15859	0,39707	1,00892
PostgreSQL + cstore_fdw	0,00154	0,0101	0,04832	0,09603	0,18709	0,4652
PostgreSQL + mongo_fdw	0,01082	0,09789	0,47076	0,92475	1,88663	4,96499

Рис. 3. Сравнение времени выполнения нахождения максимального значения



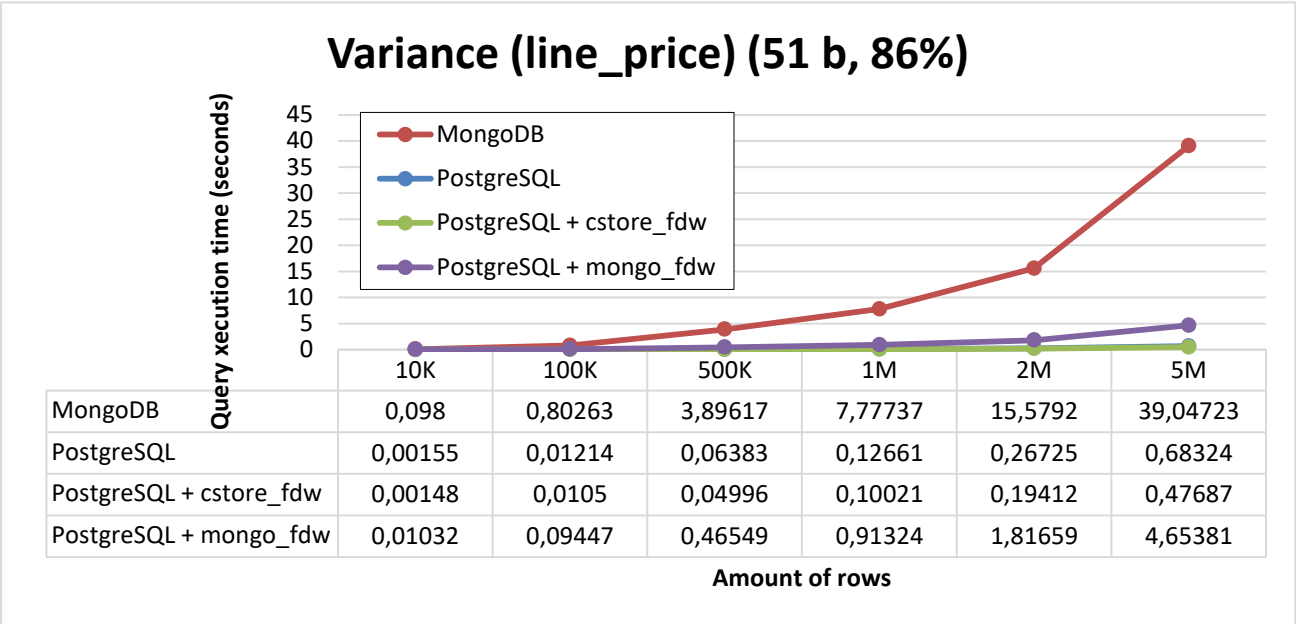
	10K	100K	500K	1M	2M	5M
MongoDB	0,01677	0,1863	1,01953	2,12297	4,34803	10,66683
PostgreSQL	0,00325	0,03258	0,22319	0,4655	0,99761	2,95396
PostgreSQL + cstore_fdw	0,00325	0,0294	0,2027	0,44136	0,89662	2,70266
PostgreSQL + mongo_fdw	0,01143	0,1101	0,60945	1,24148	2,5202	6,74018

Рис. 4. Сравнение времени выполнения нахождения уникальных почтовых индексов



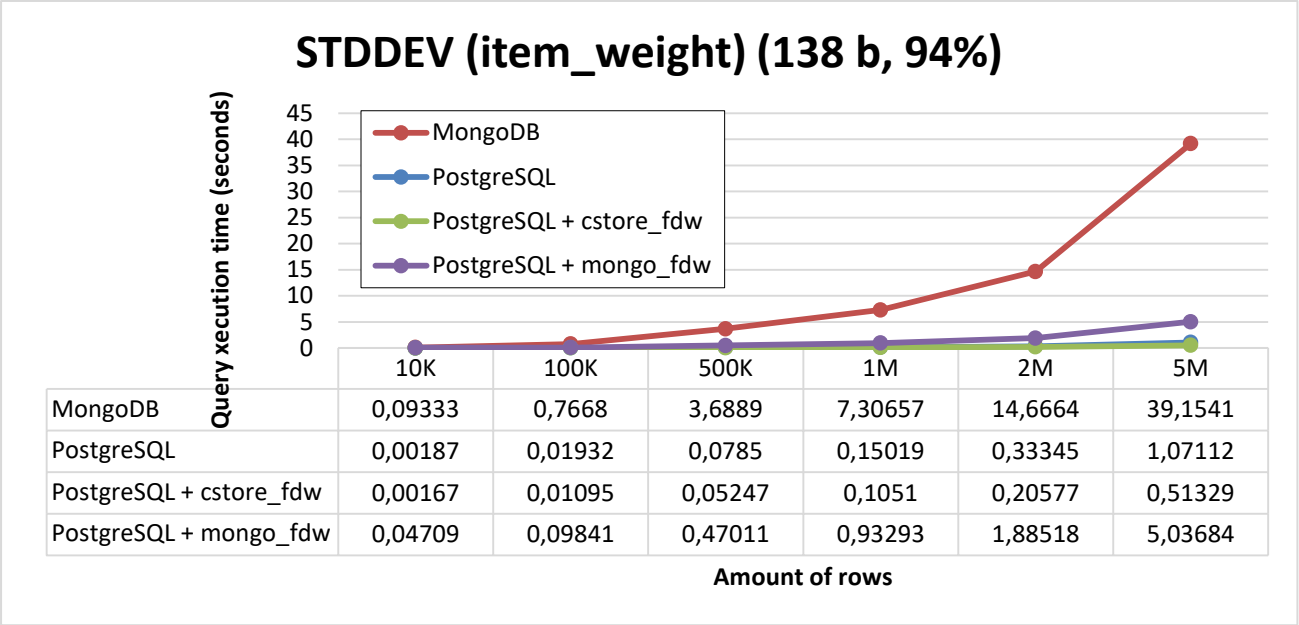
	10K	100K	500K	1M	2M	5M
MongoDB	0,01653	0,11346	0,48973	1,16320	2,35090	6,13660
PostgreSQL	0,00731	0,03521	0,15303	0,30000	0,59715	1,57337
PostgreSQL + cstore_fdw	0,00423	0,03406	0,14344	0,32347	0,66182	1,66957
PostgreSQL + mongo_fdw	0,01306	0,11698	0,58773	1,16689	2,31151	5,83274

Рис. 5. Сравнение времени выполнения операций группировки



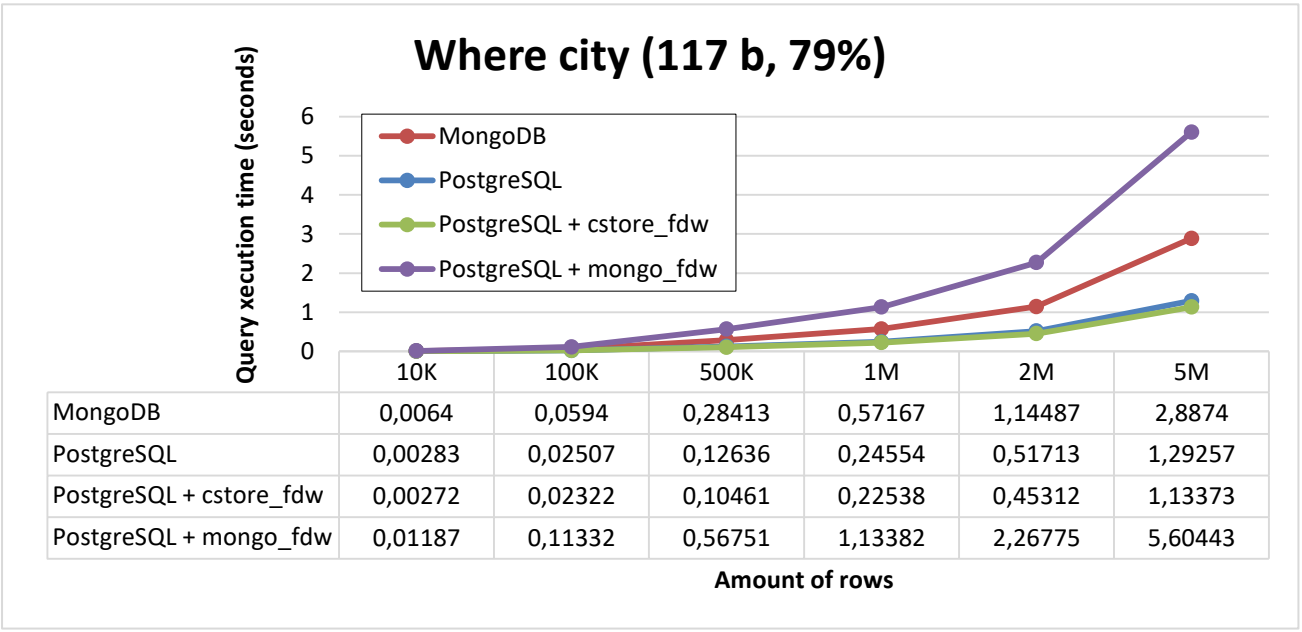
	10K	100K	500K	1M	2M	5M
MongoDB	0,098	0,80263	3,89617	7,77737	15,5792	39,04723
PostgreSQL	0,00155	0,01214	0,06383	0,12661	0,26725	0,68324
PostgreSQL + cstore_fdw	0,00148	0,0105	0,04996	0,10021	0,19412	0,47687
PostgreSQL + mongo_fdw	0,01032	0,09447	0,46549	0,91324	1,81659	4,65381

Рис. 6. Сравнение времени выполнения операций подсчета дисперсии выборки



	10K	100K	500K	1M	2M	5M
MongoDB	0,09333	0,7668	3,6889	7,30657	14,6664	39,1541
PostgreSQL	0,00187	0,01932	0,0785	0,15019	0,33345	1,07112
PostgreSQL + cstore_fdw	0,00167	0,01095	0,05247	0,1051	0,20577	0,51329
PostgreSQL + mongo_fdw	0,04709	0,09841	0,47011	0,93293	1,88518	5,03684

Рис. 7. Сравнение времени выполнения операции подсчета стандартного отклонения выборки



	10K	100K	500K	1M	2M	5M
MongoDB	0,0064	0,0594	0,28413	0,57167	1,14487	2,8874
PostgreSQL	0,00283	0,02507	0,12636	0,24554	0,51713	1,29257
PostgreSQL + cstore_fdw	0,00272	0,02322	0,10461	0,22538	0,45312	1,13373
PostgreSQL + mongo_fdw	0,01187	0,11332	0,56751	1,13382	2,26775	5,60443

Рис. 8. Сравнение времени выполнения операции выборки по шаблону

Практически во всех задачах расширение cstore_fdw позволяет уменьшить время выполнения запросов в PostgreSQL, однако оно не достаточно существенно для отказа от «традиционной» версии данной СУБД с учетом технических ограничений cstore_fdw.

ГЛАВА 5. ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ С JSON-ДОКУМЕНТАМИ

JSON – открытый формат обмена данными, основанный на JavaScript [20]. 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 [21]. Кроме того, на диске BSON-документы представляются в двоичном виде, а не в текстовом, как JSON.

Что касается PostgreSQL, то данная СУБД поддерживает работу с JSON уже в течение нескольких лет [22]. 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 является 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 [23].

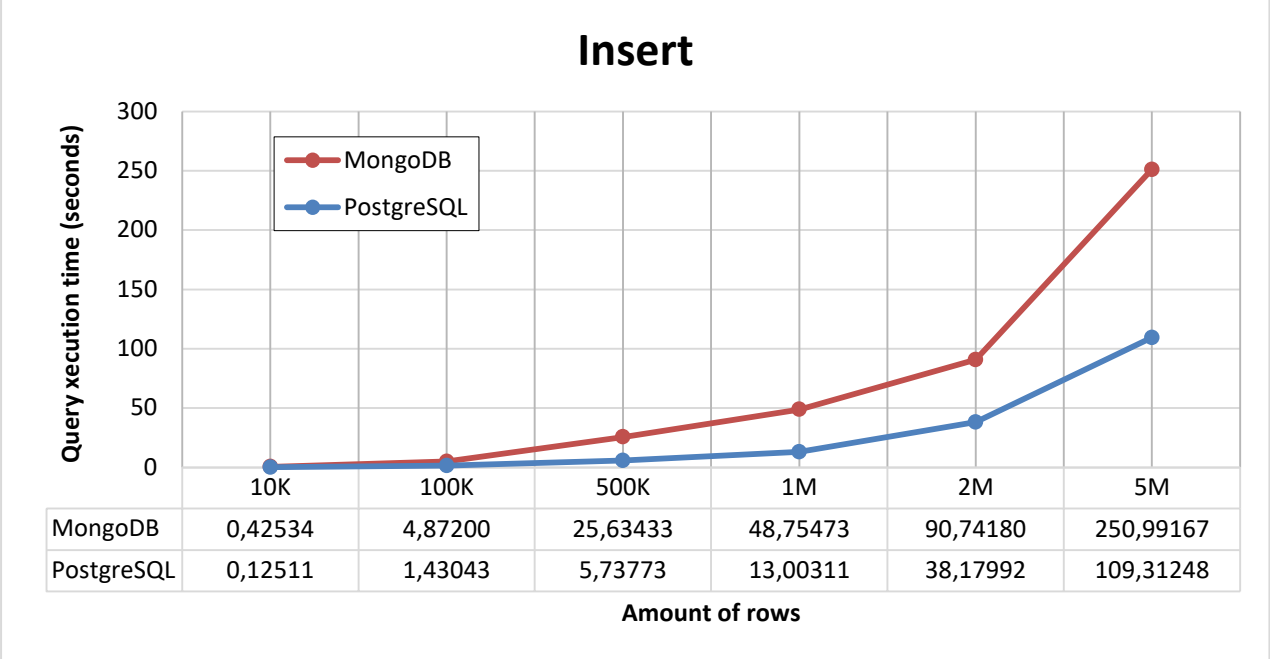


Рис. 1. Сравнение времени выполнения операции вставки данных

Эксперимент проводился на таблице t_item с помощью операций групповой вставки данных COPY и mongoimport в случае PostgreSQL и MongoDB соответственно.

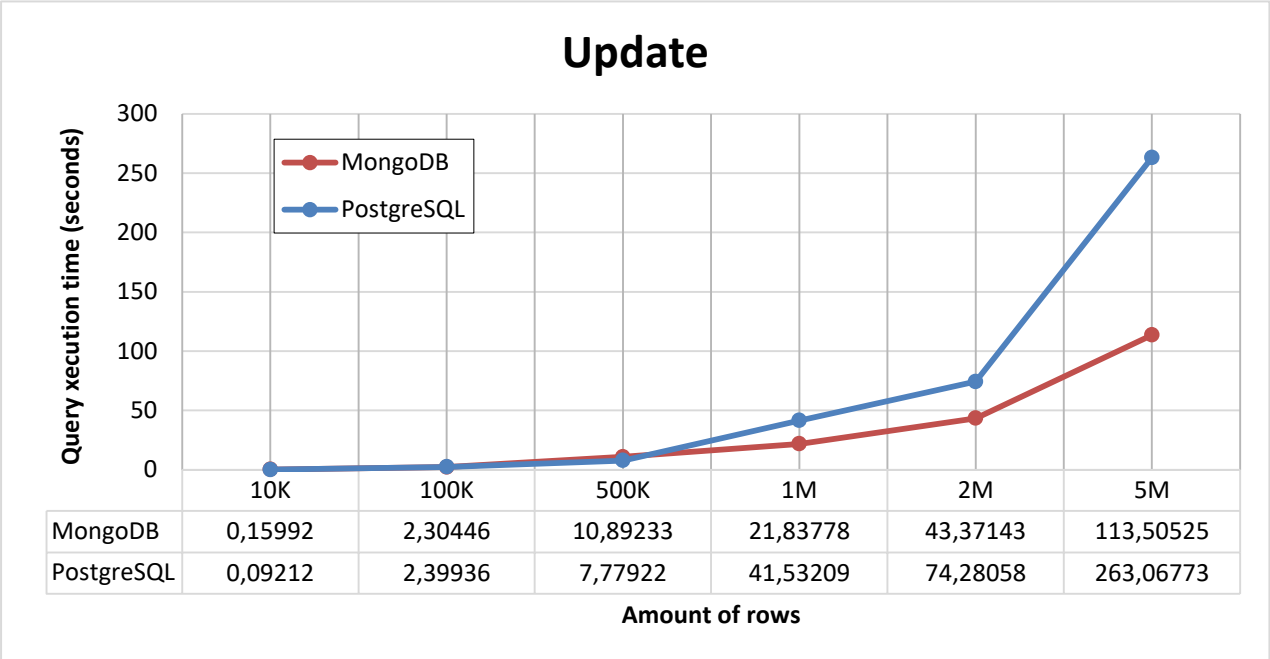
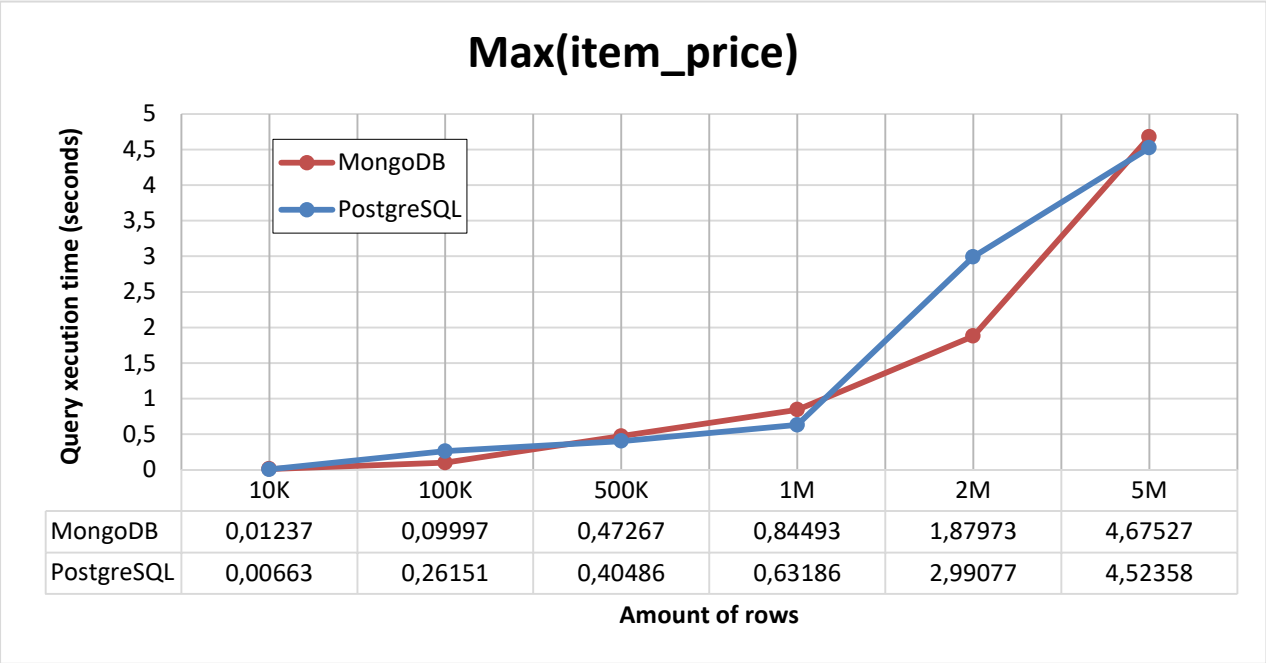


Рис. 2. Сравнение времени выполнения операции обновления данных

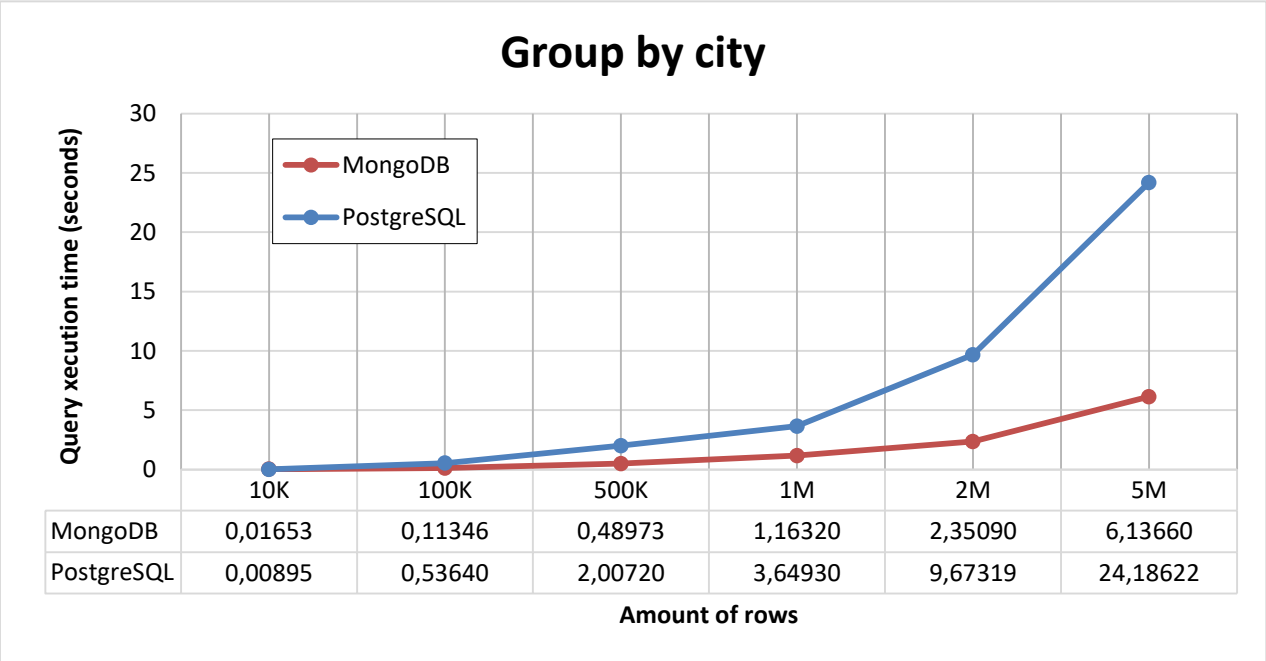
Для обновления данных использовалась та же таблица, изменение касалось почтового индекса.



	10K	100K	500K	1M	2M	5M
MongoDB	0,01237	0,09997	0,47267	0,84493	1,87973	4,67527
PostgreSQL	0,00663	0,26151	0,40486	0,63186	2,99077	4,52358

Рис. 3. Сравнение времени выполнения операции поиска максимума

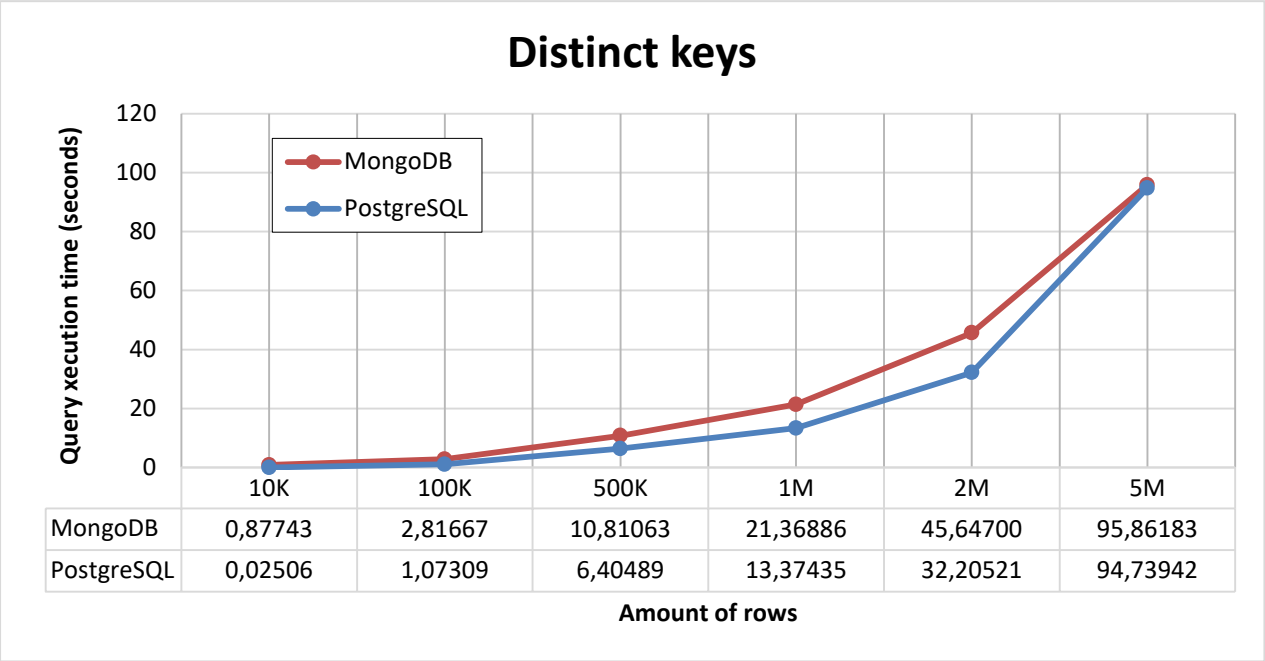
Поиск максимума осуществлялся по полю item_price (целое положительное число) таблицы t_item. Стоимость отметить ухудшение производительности PostgreSQL на нагрузке от 1 млн. до 5 млн. записей. Спрогнозировать дальнейшее развитие «соперничества» двух СУБД в этом запросе довольно трудно.



	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. 4. Сравнение времени выполнения операции группировки данных

Группировка осуществлялась по полю city (строковое значение) таблицы t_address. Итоговый результат отсортирован по убыванию значения агрегирующей функции.



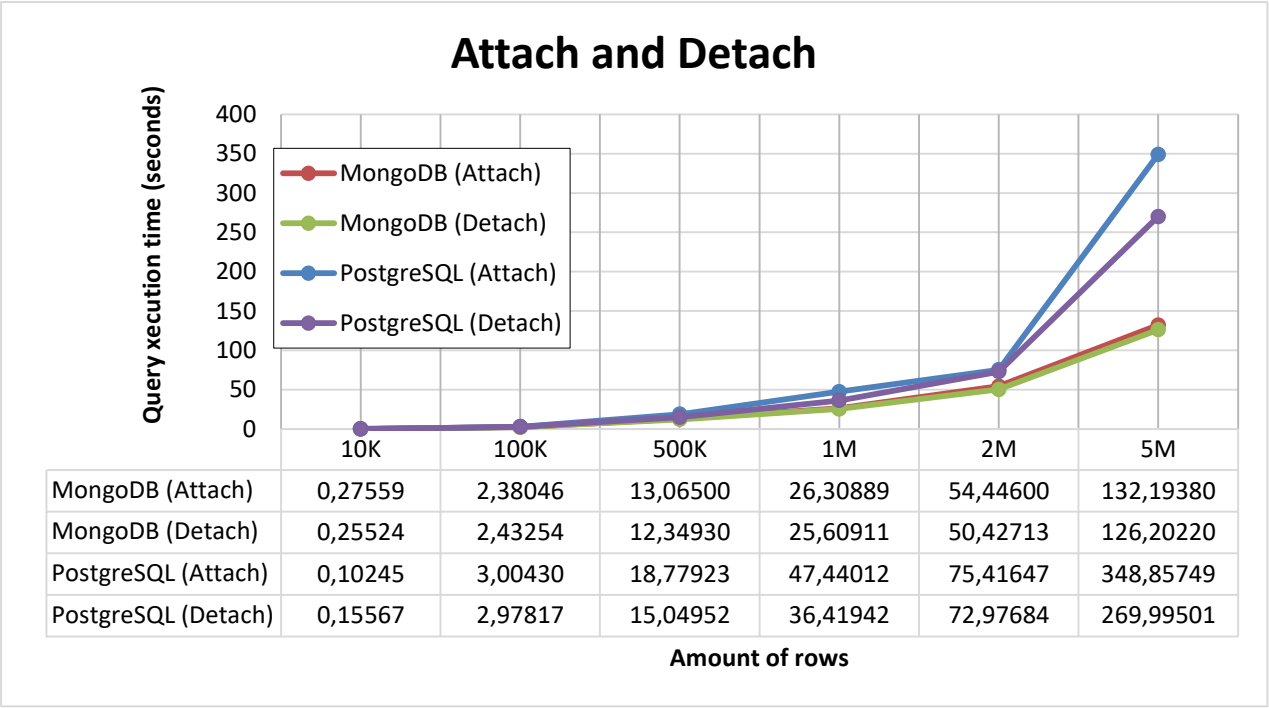
	10K	100K	500K	1M	2M	5M
MongoDB	0,87743	2,81667	10,81063	21,36886	45,64700	95,86183
PostgreSQL	0,02506	1,07309	6,40489	13,37435	32,20521	94,73942

Рис. 5. Сравнение времени выполнения операции поиска различных полей документах формата JSON

В случае PostgreSQL была использована встроенная функция jsonb_object_keys, которая принимает в качества аргумента JSON или JSONB-объект и возвращает набор его ключей, для обеспечения уникальности членов которого используется вспомогательная команда distinct. В то же время, для построения аналогичной функции в MongoDB была использована технология MapReduce:

```
result = db.runCommand({"mapreduce" : "t_address",
      "map" : function() {for (var key in this)
      {emit(key, null);}},
      "reduce" : function(key) {return null;}, "out": "keys"});
```

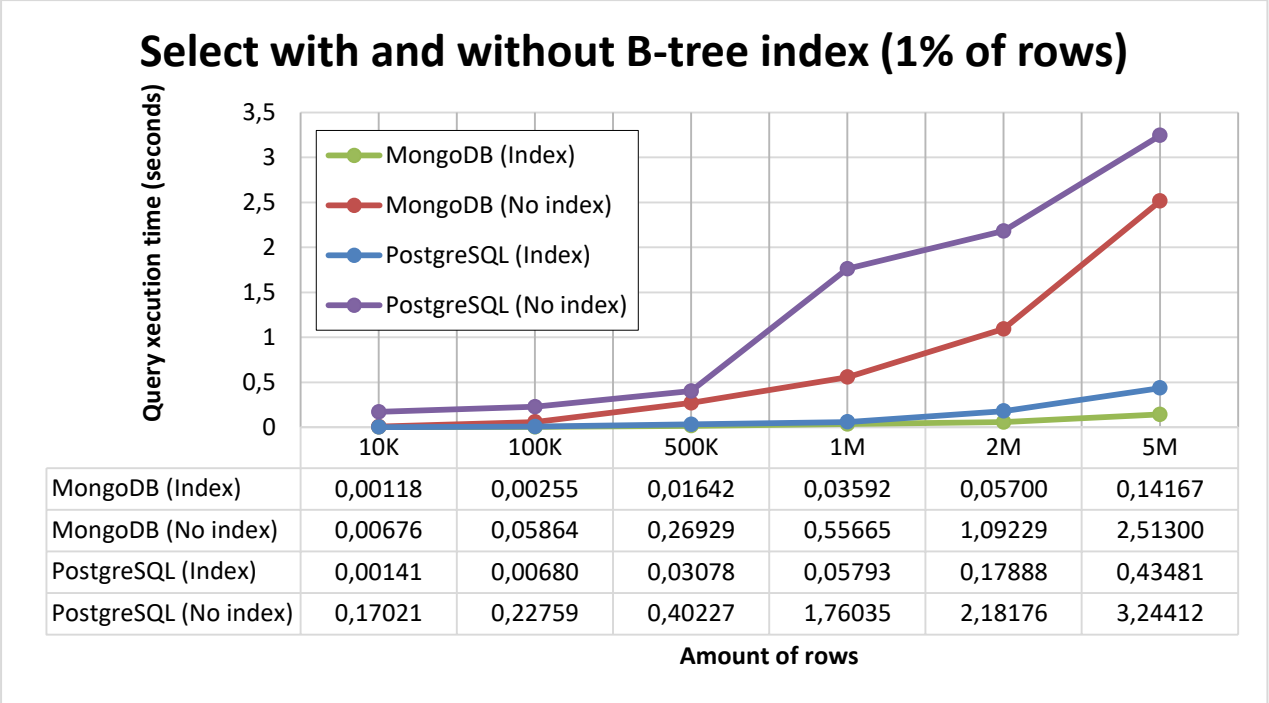
В каждом документе коллекции t_address каждый ключ помечается и перемещается во временную результирующую коллекцию, для обзора которой используется команда find.



	10K	100K	500K	1M	2M	5M
MongoDB (Attach)	0,27559	2,38046	13,06500	26,30889	54,44600	132,19380
MongoDB (Detach)	0,25524	2,43254	12,34930	25,60911	50,42713	126,20220
PostgreSQL (Attach)	0,10245	3,00430	18,77923	47,44012	75,41647	348,85749
PostgreSQL (Detach)	0,15567	2,97817	15,04952	36,41942	72,97684	269,99501

Рис. 6. Сравнение времени выполнения операций удаления и добавления полей в документах формата JSON

Из рисунка 6 можно сделать вполне прогнозируемый вывод: MongoDB справляется с операциями изменения структуры документов быстрее, чем PostgreSQL.



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

В последнем эксперименте в обеих СУБД использовался индекс на основе бинарного дерева поиска по полю item_price таблицы t_item. Условие выборки было выбрано таким образом, чтобы ему соответствовали примерно 1 процент записей, что является оптимальным значением для эффективной работы индекса.

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной магистерской диссертации 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:

- 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. Стоит также отметить выигрыш SQL СУБД в аналитических запросах;
- MongoDB функционирует быстрее PostgreSQL в подавляющем большинстве задач по работе с JSON-документами;
- 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);
- Денормализация данных, как способ превентивного присоединения кортежей в MongoDB, является вполне допустимым решением задачи, что особенно ярко это проявляется на больших объемах данных. Однако стоит отметить, что решение о денормализации нужно принимать исходя из оценок затрат на поддержание согласованности;
- На сегодняшний день рассмотренные в работе расширения над PostgreSQL не могут считаться подходящими решениями в области серьезных производственных задач. По большей части это связано с наличием существенных технических ограничений в отдельных аспектах работы указанных модификаций.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

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: 10 декабря 2015].
2. Бэнкер К., MongoDB в действии. – М.: ДМК Пресс, 2012. – 394 с.
3. MongoDB Production Deployments [Электронный ресурс]. URL: http://www.mongodb.org/about/production-deployments/. [Дата обращения: 10 декабря 2015].
4. Hasso Plattner. 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 (pages 1-2).
5. Parker Z., Poe S., Vrbsky S. Comparing NoSQL MongoDB to an SQL DB, Proceedings of the 51st ACM Southeast Conference. – NY: ACM New York, 2013. – 6 p.
6. 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: 10 декабря 2015].
7. Rick Cattell. Scalable SQL and NoSQL data stores. ACM SIGMOD Record, Volume 39 Issue 4, December 2010. – NY: ACM New York. – p.12-27.
8. Veronika Abramova, Jorge Bernardino, Pedro Furtado. Which NoSQL Database? A Performance, Overview Open Journal of Databases (OJDB), Volume 1, Issue 2, 2014. – Lübeck: RonPub. – p.17-24.
9. Data generator: free tool to generate test data. [Электронный ресурс]. URL: http://www.yandataellan.com/ [Дата обращения: 20 декабря 2016].
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 декабря 2015].

11. Overview | Postgres-XL [Электронный ресурс]. URL: http://www.postgres-xl.org/overview/ [Дата обращения: 10 мая 2016].
12. ToroDB | 8KData [Электронный ресурс]. URL: https://www.8kdata.com/torodb [Дата обращения: 10 мая 2016].
13. Доклады конференции PgConfRussia 2016: CitusDB: расширение для масштабирования PostgreSQL [Электронный ресурс]. URL: http://www.postgres-xl.org/overview/ [Дата обращения: 10 мая 2016].
14. Облачные серверы и хостинг Elastic Compute Cloud (EC2) – AWS [Электронный ресурс]. URL: http://aws.amazon.com/ru/ec2/. [Дата обращения: 10 мая 2016].
15. Daniel J. Abadi, Peter A. Boncz, Stavros Harizopoulos. Column-oriented database systems. Proceedings of the VLDB Endowment, Volume 2 Issue 2, August 2009 (pages 1664-1665).
16. Foreign data wrappers – PostgreSQL Wiki [Электронный ресурс]. URL: https://wiki.postgresql.org/wiki/Foreign_data_wrappers [Дата обращения: 20 декабря 2016].
17. Cstore_fdw [Электронный ресурс]. URL: https://github.com/citusdata/cstore_fdw [Дата обращения: 20 декабря 2016].
18. Lempel-Ziv-Welch (LZW) Algorithm https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node214.html [Дата обращения: 20 декабря 2016].
19. LanguageManual ORC – Apache Hive – Apache Software Foundation [Электронный ресурс]. URL: https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=31818911 [Дата обращения: 20 декабря 2016].
20. Internet Engineering Task Force (IETF). The JavaScript Object Notation (JSON) Data Interchange Format [Электронный ресурс]. URL: https://tools.ietf.org/html/rfc7159. Дата обращения: 20 Марта 2017.

21. JSON and BSON | MongoDB [Электронный ресурс]. URL: https://docs.mongodb.com/v3.0/reference/bson-types/. Дата обращения: 20 Марта 2017.

22. Reuven M. Lerner . At the forge: PostgreSQL, the NoSQL database. Article № 5 / Reuven M. Lerner // Linux Journal. – 2014. – №247. – p. 3-6.

23. PostgreSQL 9.6: Documentation: 9.6: JSON Types [Электронный ресурс]. URL: https://www.postgresql.org/docs/9.6/static/datatype-json.html/. Дата обращения: 20 Марта 2017.

ПРИЛОЖЕНИЕ 1

Обобщенные итоги проведения экспериментов

Табл. 1. Время выполнения запросов к таблицам с 10 тыс. записей (в сек.).

Запросы к данным	Одна ЭВМ			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Group by	0,00731±0,00003		0,01653±0,00030	
Insert	0,03329± 0,00118		0,23400±0,00138	
Join	0,04225±0,00303		0,00244±0,00009/0,37144±0,00094[1]	
Max	0,00216±0,00006		0,01237±0,00018	
Select with and without index	0,00105±0,00004/ 0,00202±0,00002		0,00118±0,00007/ 0,00676±0,00016	
Update	0,03004±0,00129		0,15992±0,00281	
	Распределенное окружение			
	CitusDB 6.1		MongoDB 3.4.2	
Group by	0,05761±0,00091		0,02679±0,00012	
Insert	0,23195±0,00080		0,20900±0,00456	
Max	0,03998±0,00066		0,00977±0,00008	
Select with and without index	0,03879±0,00025/ 0,03835±0,00033		0,00100±0,00001/ 0,00242±0,00009	
Update	0,11089±0,00394		0,08018±0,00011	
	Колоночное хранение и обертки данных			
	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	0,00325±0,00001	0,00325±0,00002	0,01143±0,00002	0,01677±0,00009
Deviance	0,00187±0,00011	0,00167±0,0001	0,04709±0,03481	0,09333±0,00012
Group by	0,00731±0,00003	0,00423±0,00002	0,01306±0,00003	0,01653±0,00030
Max	0,00216±0,00006	0,00154±0,00001	0,01082±0,00033	0,01237±0,00018
Variance	0,00155±0,00002	0,00148±0,00001	0,01032±0,00021	0,09800±0,00013
Where like	0,00283±0,00001	0,00272±0,00001	0,01187±0,00011	0,0064±0,00009
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	

[1] Первое значение относится к выборке данных из денормализованной таблицы, второе – к использованию ключа $lookup функции aggregate.

Продолжение таблицы 1.

Attach and detach	0,10245±0,00868/ 0,15567±0,02631	0,27559±0,00896/ 0,25524±0,01285
Distinct keys	0,02506±0,00015	0,87743±0,00990
Group by	0,00895±0,00008	0,01653±0,00030
Insert	0,12511±0,00370	0,42534±0,01075
Max	0,00663±0,00008	0,01237±0,00018
Select with and without index	0,00141±0,00003/ 0,17021±0,00046	0,00118±0,00007/ 0,00676±0,00016
Update	0,09212±0,00283	0,15992±0,00281

Табл. 2. Время выполнения запросов к таблицам с 100 тыс. записей (в сек.).

Запросы к данным	Одна ЭВМ			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Group by	0,03521±0,00037		0,11346±0,00138	
Insert	1,12710± 0,06001		2,20763±0,04654	
Join	0,43619±0,00303		0,02700±0,00020/3,84392±0,00749	
Max	0,01466±0,00022		0,09997±0,00078	
Select with and without index	0,00414±0,00005/ 0,01508±0,00025		0,00255±0,00009/ 0,05864±0,00044	
Update	0,52360±0,01414		2,30446±0,06587	
	Распределенное окружение			
	CitusDB 6.1		MongoDB 3.4.2	
Group by	0,13871±0,00069		0,17744±0,00062	
Insert	0,44583±0,00731		1,86775±0,01396	
Max	0,04452±0,00035		0,08864±0,00011	
Select with and without index	0,04118±0,00018/ 0,04578±0,00038		0,00271±0,00008/ 0,02485±0,00012	
Update	0,24710±0,00638		0,79640±0,00139	
	Колоночное хранение и обертки данных			
	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	0,03258±0,00006	0,0294±0,00005	0,11010±0,00106	0,18630± 0,00055
Deviance	0,01932±0,00024	0,01095±0,00002	0,09841±0,00019	0,76680± 0,00095
Group by	0,03521±0,00037	0,03406±0,00004	0,11698±0,00012	0,11346±0,00138

Продолжение таблицы 2.

Max	0,01466±0,00022	0,01010±0,00002	0,09789±0,00011	0,09997±0,00078
Variance	0,01214±0,00002	0,01050±0,00003	0,09447±0,00011	0,80263± 0,00106
Where like	0,02507±0,00011	0,02322±0,00001	0,11332±0,00083	0,05940± 0,00016
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Attach and detach	0,10245±0,00868/0,15567±0,02631		2,38046±0,01964/2,43254±0,02943	
Distinct keys	1,07309±0,00210		2,81667±0,02556	
Group by	0,53640±0,00150		0,11346±0,00138	
Insert	1,43043±0,06853		4,87200±0,08109	
Max	0,26151±0,00434		0,09997±0,00078	
Select with and without index	0,00680±0,00004/ 0,22759±0,00024		0,00255±0,00009/ 0,05864±0,00044	
Update	2,39936±0,02926		2,30446±0,06587	

Табл. 3. Время выполнения запросов к таблицам с 500 тыс. записей (в сек.).

Запросы к данным	Одна ЭВМ	
	PostgreSQL 9.6.1	MongoDB 3.4.2
Group by	0,15303±0,00027	0,48973± 0,00287
Insert	4,49183±0,17277	11,60733±0,09828
Join	2,25222±0,04379	0,02700±0,00020/3,84392±0,00749
Max	0,09818±0,00077	0,47267±0,00270
Select with and without index	0,01771±0,00003/ 0,07139±0,00029	0,01642±0,00009/ 0,26929±0,00172
Update	4,48883±0,13284	10,89233±0,14421
	Распределенное окружение	
	CitusDB 6.1	MongoDB 3.4.2
Group by	0,26516±0,00076	0,83814±0,00310
Insert	1,26342±0,00624	8,66588±0,04831
Max	0,09909±0,00092	0,37317±0,00080
Select with and without index	0,05910±0,00054/ 0,15105±0,00066	0,00493±0,00015/ 0,13715±0,00014
Update	0,87260±0,00947	4,00017±0,00714
	Колоночное хранение и обертки данных	

Продолжение таблицы 3.

	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	0,22319±0,00099	0,14344±0,00013	0,60945±0,00111	1,01953±0,00634
Deviance	0,0785±0,00007	0,05247±0,00006	0,47011±0,00045	0,76680± 0,00095
Group by	0,15303±0,00027	0,14344±0,00013	0,58773±0,00055	0,48973± 0,00287
Max	0,09818±0,00077	0,04832±0,00016	0,47076±0,00067	0,47267±0,00270
Variance	0,06383±0,00046	0,04996±0,00001	0,46549±0,00037	3,89617± 0,00364
Where like	0,12636±0,00016	0,10461±0,00007	0,56751±0,00180	0,28413± 0,00030
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Attach and detach	18,77923±0,85997/ 15,04952±0,95553		13,06500±0,19968/ 12,34930±0,17042	
Distinct keys	6,40489±0,01547		10,81063±0,03481	
Group by	2,00720±0,00505		0,48973± 0,00287	
Insert	5,73773±0,09972		25,63433±0,35984	
Max	0,40486±0,00099		0,47267±0,00270	
Select with and without index	0,03078±0,00010/ 0,40227±0,00095		0,01642±0,00009/ 0,26929±0,00172	
Update	7,77922±0,08996		10,89233±0,14421	

Табл. 4. Время выполнения запросов к таблицам с 1 млн. записей (в сек.).

Запросы к данным	Одна ЭВМ	
	PostgreSQL 9.6.1	MongoDB 3.4.2
Group by	0,30000±0,00113	1,16320±0,00341
Insert	12,64069± 0,59800	24,42897±0,47104
Join	4,88880±0,06988	0,22064±0,00051/36,84425±0,03641
Max	0,15859±0,00155	0,84493±0,00257
Select with and without index	0,03453±0,00022/ 0,19038±0,00201	0,00255±0,00009/ 0,55665±0,00174
Update	14,06293±0,58357	21,83778±0,14043
	Распределенное окружение	
	CitusDB 6.1	MongoDB 3.4.2
Group by	0,33860±0,00176	1,66918±0,00422
Insert	2,26542±0,01258	19,89300±0,50364
Max	0,11720±0,00067	0,99264±0,00103

Продолжение таблицы 4.

Select with and without index	0,07487±0,00061/ 0,52476±0,00339		0,02120±0,00010/ 0,34418±0,00043	
Update	1,20979±0,01698		8,07900±0,01807	
	Колоночное хранение и обертки данных			
	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	0,46550±0,00114	0,44136±0,00093	1,24148±0,00275	2,12297± 0,01114
Deviance	0,15019±0,00333	0,10510±0,00017	0,93293±0,00149	7,30657±0,00684
Group by	0,30000±0,00113	0,32347±0,00025	1,16689±0,00125	1,16320±0,00341
Max	0,15859±0,00155	0,09603±0,00022	0,92475±0,00090	0,84493±0,00257
Variance	0,12661±0,00126	0,10021±0,00026	0,91324±0,00115	7,77737± 0,00545
Where like	0,24554±0,00033	0,22538±0,00007	1,13382±0,00336	0,57167± 0,00056
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Attach and detach	47,44012± ,26566/ 36,41942± 1,74296		26,30889±0,31465/ 25,60911±0,50107	
Distinct keys	13,37435± 0,02356		21,36886±0,13119	
Group by	3,64930± 0,00728		1,16320±0,00341	
Insert	13,00311± 0,43203		48,75473±0,88487	
Max	0,63186±0,00160		0,84493±0,00257	
Select with and without index	0,05793±0,00022/ 1,76035±0,00862		0,00255±0,00009/ 0,55665±0,00174	
Update	41,53209±0,73252		21,83778±0,14043	

Табл. 5. Время выполнения запросов к таблицам с 2 млн. записей (в сек.).

Запросы к данным	Одна ЭВМ	
	PostgreSQL 9.6.1	MongoDB 3.4.2
Group by	0,59715± 0,00127	2,35090±0,00695
Insert	26,42558± 0,76868	49,06323±0,62274
Join	11,76832±0,34338	0,49569±0,00375/75,16550±0,24862
Max	0,39707±0,00191	1,87973±0,02807
Select with and without index	0,06930±0,00063/ 0,47505±0,00090	0,05700±0,00053/ 1,09229±0,00103
Update	25,14448±0,16726	43,37143±0,36756

Продолжение таблицы 5.

	Распределенное окружение			
	CitusDB 6.1		MongoDB 3.4.2	
Group by	0,48997±0,00420		2,46800±0,00351	
Insert	4,15865±0,00630		38,30214±0,48357	
Max	0,19277±0,00058		1,66582±0,00739	
Select with and without index	0,10941±0,00056/ 0,80363±0,00569		0,03592±0,00009/ 0,58417±0,00042	
Update	2,37678±0,02964		12,10815±0,02806	
	Колоночное хранение и обертки данных			
	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	0,99761±0,00206	0,89662±0,00059	2,52020±0,00227	4,34803± 0,03519
Deviance	0,33345±0,00022	0,20577±0,00002	1,88518±0,00146	14,66640± 0,01300
Group by	0,68631± 0,00038	0,66182±0,00044	2,31151±0,00128	2,35090±0,00695
Max	0,34072±0,00094	0,18709±0,00018	1,88663±0,00446	1,87973±0,02807
Variance	0,26725±0,00217	0,19412±0,00027	1,81659±0,00283	15,57920± 0,00958
Where like	0,51713±0,00249	0,45312±0,00016	2,26775±0,00511	1,14487± 0,00071
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Attach and detach	75,41647±1,06176/72,97684±0,98472		54,44600±0,18694/50,42713±0,21114	
Distinct keys	32,20521±0,13983		45,64700±0,49979	
Group by	9,67319±0,03595		2,35090±0,00695	
Insert	38,17992±0,70289		90,74180±0,98956	
Max	2,99077±0,01172		1,87973±0,02807	
Select with and without index	0,17888±0,00244/ 3,06131±0,00648		0,05700±0,00053/ 1,09229±0,00103	
Update	74,28058±3,33239		43,37143±0,36756	

Табл. 6. Время выполнения запросов к таблицам с 5 млн. записей (в сек.).

Запросы к данным	Одна ЭВМ	
	PostgreSQL 9.6.1	MongoDB 3.4.2
Group by	1,57337± 0,00116	6,13660±0,01841
Insert	74,21024± 1,44471	149,83365±4,46471
Join	11,76832±0,34338	1,05170±0,00506/228,74725±2,78573

Продолжение таблицы 6.

Max	1,00892±0,00537		4,67527±0,03898	
Select with and without index	0,40047±0,00229/ 1,19950±0,00178		0,14167±0,00268/ 2,51300±0,00799	
Update	65,26734±0,49490		113,50525±0,46552	
	Распределенное окружение			
	CitusDB 6.1		MongoDB 3.4.2	
Group by	0,66622±0,00165		6,49770±0,01515	
Insert	10,06854±0,05450		105,18382±2,58623	
Max	0,27134±0,00140		4,83691±0,05623	
Select with and without index	0,36783±0,00084/ 1,85557±0,00482		0,11708±0,00063/ 1,63193±0,00132	
Update	5,35436±0,03714		33,26986±0,09143	
	Колоночное хранение и обертки данных			
	PostgreSQL 9.6.1	PostgreSQL+ cstrore_fdw	PostgreSQL+ mongofdw	MongoDB 3.4.2
Count	2,95396±0,09967	2,70266±0,00308	2,52020±0,00227	10,66683± 0,06822
Deviance	1,07112±0,18143	0,51329±0,00043	4,65381±0,01354	39,15410± 0,02464
Group by	1,57337± 0,00116	1,66957±0,00160	5,83274±0,00788	6,13660±0,018410
Max	1,00892±0,00537	0,46520±0,00070	4,96499±0,00505	4,67527±0,03898
Variance	0,68324±0,02110	0,47687±0,00053	4,65381±0,01354	39,04723± 0,02353
Where like	1,29257±0,00081	1,13373±0,00073	5,60443±0,01226	2,88740± 0,00323
	Работа с JSON-документами			
	PostgreSQL 9.6.1		MongoDB 3.4.2	
Attach and detach	348,85749±1,06176/269,99501±4,84769		132,19380±0,33935/126,20220±0,56451	
Distinct keys	94,73942±0,51552		95,86183±0,13497	
Group by	24,18622±0,05169		6,13660±0,01841	
Insert	109,31248±4,87215		250,99167±6,30033	
Max	4,52358±0,01372		4,67527±0,03898	
Select with and without index	0,43481±0,00060/ 5,37046±0,01132		0,14167±0,00268/ 2,51300±0,00799	
Update	263,06773±1,86969		113,50525±0,46552	