УДК 004

К.А. Лаврентьев,
старший преподаватель кафедры информационных систем и технологий
Хабаровского государственного университета экономики и права,
серверный программист ООО «Интерсол»

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ WEB-ПРИЛОЖЕНИЯ, ИСПОЛЬЗУЮЩЕГО КЭШИРОВАНИЕ ЗАПРОСОВ

Объем информации, которая передается сейчас по телекоммуникационным каналам связи, огромен, и каждый разработчик стремится снизить количество обращений к серверу и информации, получаемой за каждый запрос. Для этого используется механизм кэширования, но не всегда нужно кэшировать информацию и не каждая таблица базы данных подлежит кэшированию. В данной работе автором будут проведены эксперименты по определению целесообразности использования в web-приложении кэша и возможности использования механизма localStorage в браузере для реализации кэша в клиентском приложении.

Ключевые слова: *кэширование, LocalStorage, объединения, кортеж, реляционная база данных, производительность, автоматизация.*

The information volume that is now transmitted through telecommunication channels is huge, and each developer seeks to reduce the number of calls to the server and the information received for each request. This is implemented by caching mechanism, but it is not always necessary to cache information and not every database table is cached. In this work, the author conducts experiments to determine the feasibility of using the cache in a web application and the possibility of using the Local Storage mechanism in the browser to implement the cache in the client application.

Keywords: *caching, Local Storage, union types, tuple, relational database, performance, automation.*

Степень информатизации современного общества в целом и обеспечения бизнес-процессов в частности сейчас очень высока. Информатизация и обработка данных сильно связаны, а значит, для эффективной информатизации необходимо эффективно обрабатывать данные. Для хранения и обработки данных в современных информационных системах используются базы данных, построенные на основе различных моделей. Самые распространенные это, конечно, реляционная и постреляционная модели. Информационные системы для автоматизации бизнес-процессов обычно представляют собой web-приложение. Причин популярности web-приложений для реализации информационных систем много, и разговор о них является темой полноценной работы, поэтому просто примем за факт, что web-приложения как тип архитектуры для системы автоматизации бизнес-процессов сейчас превалирует над всеми остальными типами архитектуры. Обычно web-приложение представляет собой клиент-серверное приложение, в котором в качестве клиента выступает браузер пользователя, в качестве сервера приложения выступает какое-либо приложение, реализованное по принципу REST API (далее – сервер), также есть третий уровень системы – сервер базы данных. Пользователи при выполнении бизнес-функций взаимодействуют посредством браузера с серве-

ром приложений, а значит, вся бизнес-логика реализуется на нем же. Таким образом, чем эффективнее работает сервер, тем быстрее клиент получит данные [1]. Для проектирования и разработки сервера используют ООП и такие механизмы, как ORM – object relation mapping, которые позволяют представить реляционную базу данных в виде объектов приложения [2]. Производительность АИС зависит от скорости обработки данных в приложении и в большей степени скорость обработки зависит от сервера. Можно, конечно, предположить, что на скорость обработки данных влияет сервер базы данных, но это не так. На сервере базы данных выполняются только стандартные CRUD-операции, пусть и с сортировками и группировками, но если сервер реализован неэффективно и где-то допускает неправильную обработку данных, то вся оптимизация сервера баз данных нивелируется [3]. Основным показателем эффективности работы сервера является время обработки запроса, то есть время чтения, обновления или удаления данных, необходимых для выполнения бизнес-процессов. Время выполнения запроса зависит от времени работы сервера БД над запросом, времени обработки результатов на сервере и времени передачи данных на клиент. Как уже было сказано, на время работы сервера БД над запросом влиять не получается, значит, нужно обратить все внимание на последние два компонента времени выполнения запроса. Самый простой вариант сокращения времени выполнения запроса – это снижение количества запросов, то есть нужно кэшировать информацию на клиенте. Если необходимая для работы информация будет находиться на рабочей станции, где происходит выполнение бизнес-процесса, то не придется каждый раз обращаться к серверу за информацией и тратить время на запрос. Однако при кэшировании тоже возникают различные проблемы: необходимо заботиться об обновлении кэша, следить за размером кэшируемых данных и т.д. [4]. Чтобы понять, в каких случаях нужно кэшировать информацию, а в каких лучше спроектировать систему по принципу «тонкого клиента» автором было решено провести серию экспериментов на реальном бизнес-процессе.

Целью работы является ответ на вопрос о том, что позволяет повысить производительность web-приложения, работающего с базой данных, – централизация данных или же их распределение по узлам системы. Задачами работы являются:

1. Выбор и формализация бизнес-процесса.
2. Проектирование и реализация тестового приложения.
3. Проведение экспериментов, которые позволят ответить на поставленный вопрос.
4. Анализ полученных в результате экспериментов данных.

Для проведения экспериментов был выбран бизнес-процесс поиска объектов недвижимости для сдачи внаем. Данный бизнес-процесс активно эксплуатируется в АИС для автоматизации работы риелторского агентства. Его можно описать следующем образом: клиент приходит в агентство с потребностью снять недвижимость, у него определенные требования к недвижимости. Требованиями яв-

ляются определенные районы города, тип жилья, стоимость. Менеджер агентства производит поиск объектов недвижимости, которые удовлетворяют заданным критериям. В результате бизнес-процесса образуется массив экземпляров сущности *«объект»*, которую можно описать следующей диаграммой (рисунок 1).

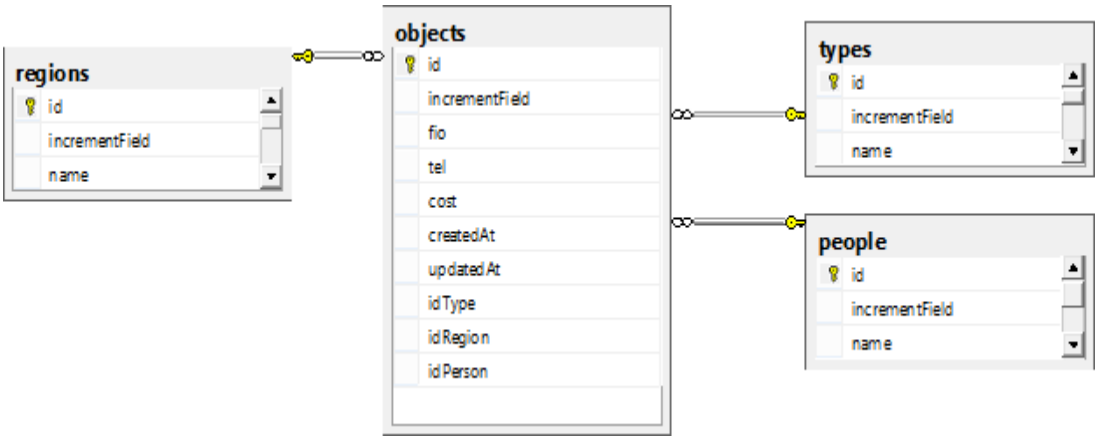

Рисунок 1 – Диаграмма сущности *«объект»*

Для достижения цели работы необходимо ответить на следующие вопросы:

– сколько времени нужно потратить на получение с сервера списка объектов недвижимости без объединения с другими таблицами по внешнему ключу;

– сколько времени нужно потратить на получение с сервера списка объектов недвижимости с объединением с другими таблицами по внешнему ключу (для каждой таблицы отдельно);

– сколько времени нужно потратить на добавление объекта недвижимости при условии, что необходимо запросить с сервера содержимое справочников;

– какой объем информации (в МБайтах) можно хранить на клиенте в кэше.

Для того чтобы ответить на эти вопросы, необходимо провести серию экспериментов и замерить время выполнения запросов при заданных условиях. Для проведения экспериментов автором было разработано клиент-серверное приложение, состоящее из 3 звеньев. Первое звено – уровень сервера – представляет собой web-приложение, написанное на языке javascript с использованием фреймворка Express. В задачи этого звена будет входить получение запросов от клиента, передача их серверу баз данных, обработка результатов и отправка результатов на клиент. Второе звено – уровень клиента – представляет собой web-приложение, написанное на языке javascript с использованием фреймворка AngularJS. В задачи этого звена будет входить представление интерфейса пользователя, отправка запросов на сервер и отображение результатов. Третьим звеном является сервер баз данных, автор в качестве этого звена выбрал СУБД Microsoft SQL Server. В таблице 1 представлена конфигурация тестового стенда (клиентского и серверного уровня).

Таблица 1 – Конфигурация тестового стенда

Характеристика	Клиентский уровень	Серверный уровень
Процессор, название и частота	Intel Core i3 1,8GHz	AMD Opteron 4171 HE, 2,10 Ghz
Объем ОЗУ	4 Гб	1,75 Гб
Качество интернет-соединения	50 Мбит, широкополосный	100 Мбит, широкополосный
Используемый браузер	Google Chrome, Mozilla Firefox	-
Операционная система	Windows 8 Prof	Windows Server 2012 R2 Datacenter

Таблица 2 – Планируемые параметры экспериментов

Номер эксперимента	Параметры
1-й эксперимент	N-join, все доступные записи из таблицы «Объекты», N-клиентов
2-й эксперимент	N-join, все доступные записи из таблицы «Объекты», N-клиентов, 1 условие для любого из связанных справочников
3-й эксперимент	N-join, все доступные записи из таблицы «Объекты», N-клиентов, N-условий для любого из связанных справочников (условия объединяются одним из выбранных логических операторов – AND, OR, IN)

Результаты проведения экспериментов будут храниться в базе данных приложения и экспортироваться в таблицу excel для дальнейшего анализа. В общем виде эксперименты будут похожи между собой и отличаться только количеством связанных справочников, поэтому спланируем параметры экспериментов. Эти параметры представлены в таблице 2.

Всего в рамках данной работы запланировано проведение трех групп экспериментов. Они отличаются количество объединений со связанными справочниками, количеством клиентов (подразумевается эмуляция разного количества клиентов) и таких параметров, как количество условий в секции WHERE для фильтрации значений. Приложение для проведения экспериментов, которое разработано автором, позволяет проводить эксперименты в автоматическом режиме, необходимо только задать параметры.

Первым этапом работы ответим на первые три поставленных вопроса, все они связаны со временем получения или добавления данных. В разработанное приложение были загружены параметры экспериментов и запущено проведение экспериментов. В итоге приложение выдало как агрегированную информацию (средние значения времени выполнения запросов), так и первичную информацию (время выполнения запроса в каждом эксперименте). Агрегированная информация использовалась автором для анализа и решения поставленных вопросов, и результат анализа приведен в таблицах 3–6 и на рисунках 1–4. Первичная же информация использовалась для уточнения анализа усредненных показателей.

Таблица 3 – Результаты экспериментов для одного клиента

Параметры эксперимента	Время выборки (мс.)	Время записи в базу (мс.)
0 join	407	46
1 join	563	47
2 join	686	31
3 join	1156	31

Таблица 4 – Результаты экспериментов для 5 клиентов

Параметры эксперимента	Время выборки (мс.)	Время записи в базу (мс.)
0 join	1 144,2	184,4
1 join	1 896,8	359
2 join	3 311	301,8
3 join	1 281,8	340,4

Таблица 5 – Результаты экспериментов для 20 клиентов

Параметры эксперимента	Время выборки (мс.)	Время записи в базу (мс.)
0 join	863,8	170,4
1 join	1 691,4	305,4
2 join	2 249,4	430,35
3 join	2 877,85	637,6

Таблица 6 – Результаты экспериментов для 100 клиентов

Параметры эксперимента	Время выборки (мс.)	Время записи в базу (мс.)
0 join	1 000,65	168,35
1 join	1 496,24	360,32
2 join	2 047,53	456,79
3 join	2 684,3	565,44

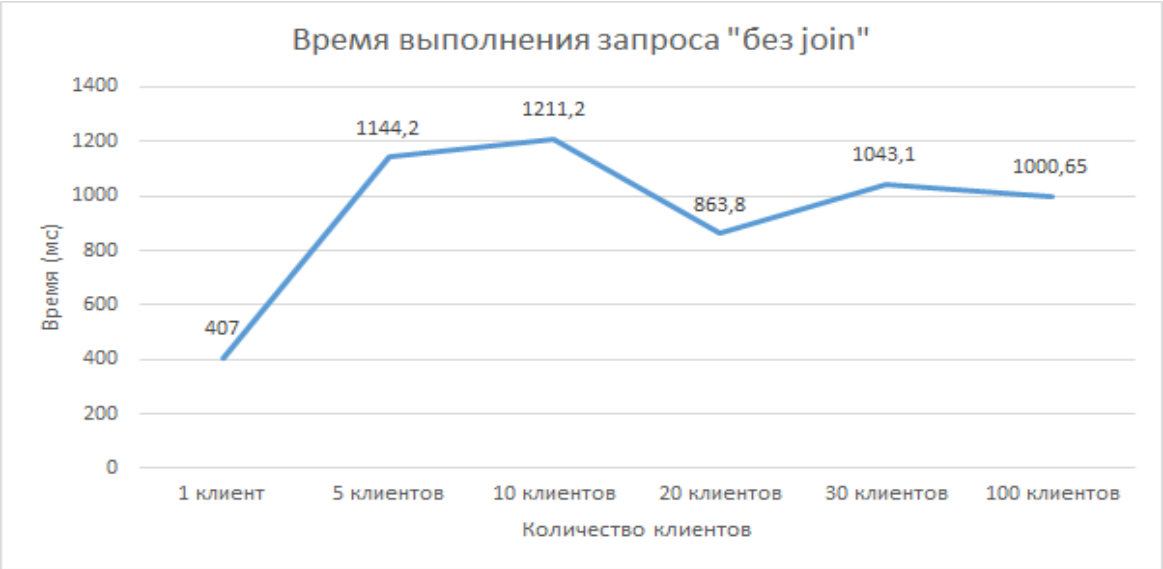

Рисунок 2 – График «Время выполнения эксперимента с 0 join»

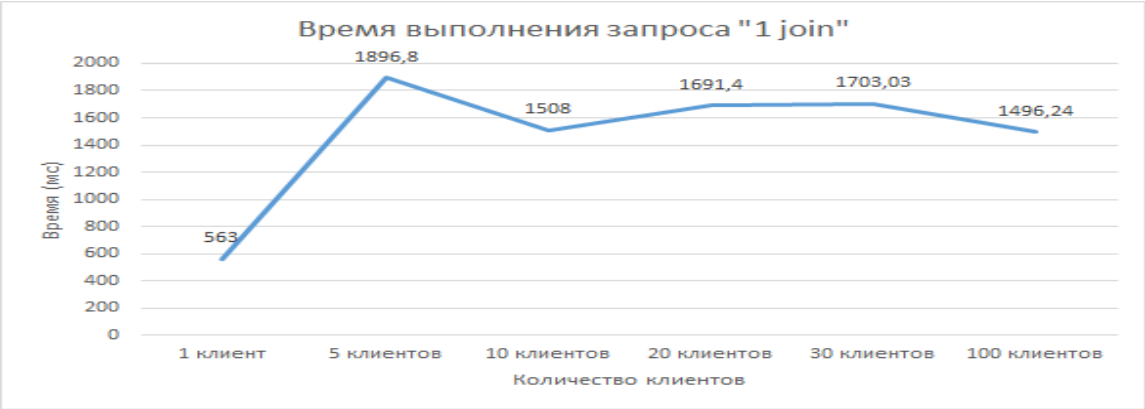

Рисунок 3 – График «Время выполнения эксперимента с 1 join»

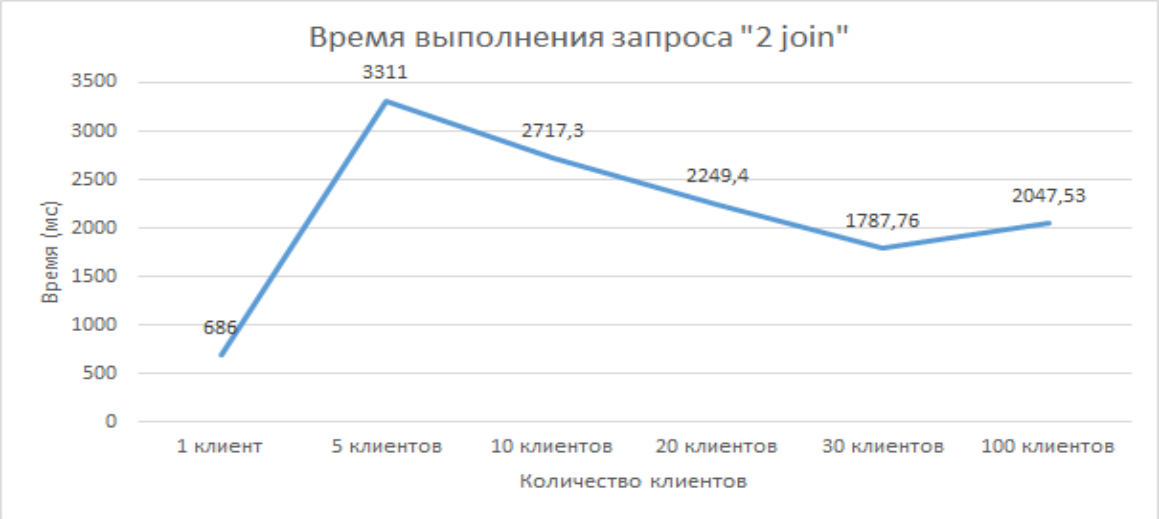

Рисунок 4 – График «Время выполнения эксперимента с 2 join»

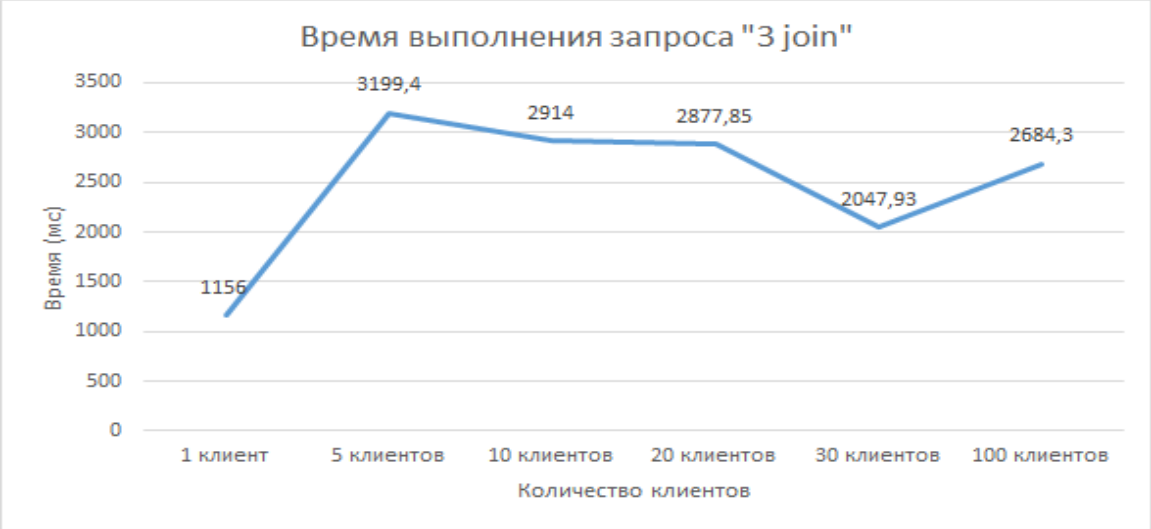

Рисунок 5 – График «Время выполнения эксперимента с 3 join»

При анализе графиков можно заметить парадоксальную на первый взгляд ситуацию: с увеличением количества клиентов время выполнения запроса снижается. Эту ситуацию можно легко объяснить: при разработке приложения для проведения экспериментов использовался фреймворк Express, который работает в среде технологии Node.js, а эта технология является полностью асинхронной. Полная асинхронность означает, что все запросы к бекэнду осуществляются в параллельном режиме, соответственно при обращении к серверу разного количества клиентов (больше 1) все их запросы обрабатывались в параллельном режиме, то есть время выполнения запроса для N-клиентов в самом худшем случае будет равно времени выполнения запроса для одного клиента.

Однако вопрос, почему с увеличением количества клиентов уменьшается время выполнения, остается открытым. Автор предполагает, что это связано с уменьшением накладных расходов на обработку запроса. В web-приложениях следуют помнить, что сервер тратит время не только не получение данных из базы данных, но и на обработку входящих заголовков запроса, формирование ответа и, самое главное, инициализацию пула соединений. Сервер приложений в web-приложении не работает постоянно: если к нему нет запросов, то он «засыпает» и при появлении нового запроса снова включается в работу, значит, чем меньше будет интервал между запросами (чем больше клиентов) тем меньше серверу необходимо будет тратить времени на открытие пула соединений и включение в работу. Таким образом, можно сказать, что никаких отклонений и ошибок во времени выполнения запросов в зависимости от разного количества объединений и разного количества клиентов нет. На рисунке 6 приведен график зависимости времени выполнения запроса от количества объединений.

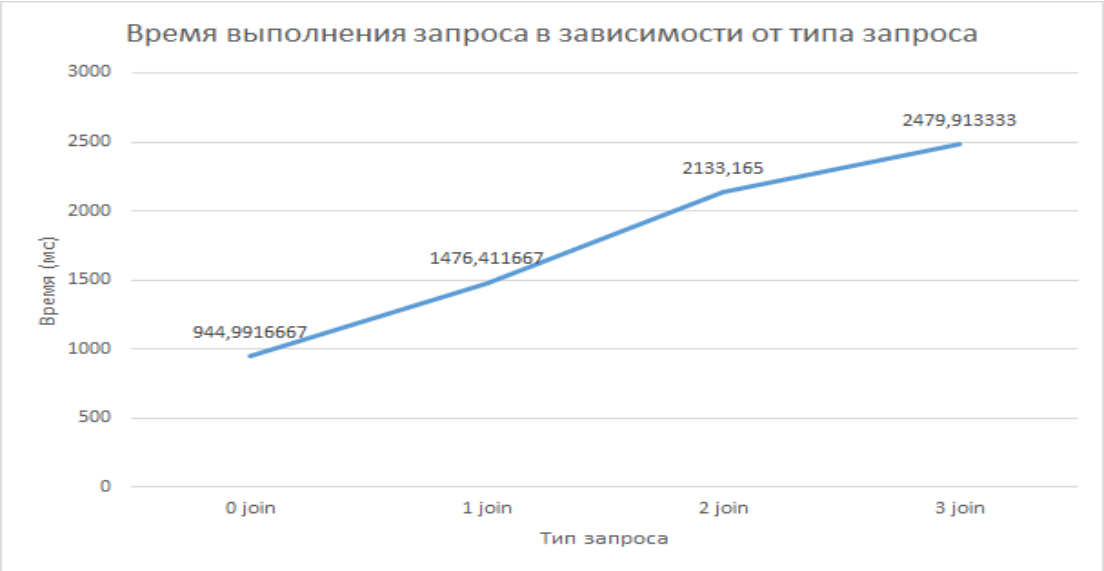

Рисунок 6 – График «Время выполнения запроса
в зависимости от количества объединений»

Как видно из графика, ошибок и отклонений в данных нет, они подчиняются логике: чем больше количество объединений в запросе тем дольше выполняется запрос.

Анализируя результаты экспериментов – агрегированные показатели и первичную информацию, автор учитывал такой показатель, как время скачивания информации. Конечно, значение этого показателя будет зависеть от скорости Интернета в каждом конкретном случае, но можно вывести какую-то среднюю величину для анализируемого случая. В данном случае средняя величина времени скачивания информации с сервера равна 4,974 секунды, что можно округлить до 5 секунд.

Следует помнить, что это значение получено для 3 объединений, если же объединений будет больше, то цифра будет неизменно возрастать. С каждым новым объединением время выполнения запроса будет внушительно изменяться. Можно сказать, что объединения таблиц отрицательно влияют на скорость работы системы и лучше декомпозировать запросы для сокращения количества объединений.

Таким образом, проводя анализ времени выполнения запросов с учетом времени скачивания результатов этих запросов, можно сказать, что выгоднее распределять данные, то есть кэшировать их. Это позволит сократить время ожидания информации пользователем и повысит быстродействие системы.

Для кэширования информации на стороне клиента существует множество способов, и выбор конкретного способа зависит от архитектуры системы. В данной работе рассматривается web-приложение, состоящее из сервера приложений (в работе он называется «бэкенд») и уровня клиента, который представляет собой браузер. Значит, для реализации кэша можно использовать только механизмы, которые предоставляет браузер. В браузере можно использовать либо механизм cookie, либо новую технологию localStorage. Механизм cookie автором рассматриваться не будет, потому что он ограничен, и в нем можно хранить только 2 килобайта информации. Интересным для рассмотрения и анализа является механизм localStorage. Это механизм, построенный на базе cookie, но позволяющий хранить уже 5 мегабайт информации.

По данным справочных систем различных браузеров, размер локального хранилища (localStorage) может колебаться от 5 до 10 мегабайт, а иногда он может быть и безлимитным. Автором работы для уточнения размера локального хранилища и получения знаний о том, сколько кортежей базы данных можно хранить в этом локальном хранилище, был проведен эксперимент. В процессе эксперимента запрашивались данные из базы данных и записывались в локальное хранилище, на каждой итерации записи проверялась успешность записи в хранилище. Если свободное место в локальном хранилище закончится, то, естественно, запись в него закончится неуспешно, именно эта ситуация и проверялась в эксперименте на каждой итерации записи.

Для проведения эксперимента в рассмотренное выше тестовое приложение была добавлена логика по записи и проверке корректности этой записи в локальное хранилище. Так как перед автором

стояла задача не просто выяснить допустимый размер локального хранилища в мегабайтах, а узнать, сколько кортежей можно хранить в этом хранилище, то было проведено несколько экспериментов с различными параметрами. Параметры экспериментов и их результаты приведены в таблице 7.

Таблица 7 – Параметры и результаты экспериментов по определению максимального размера локального хранилища браузера (localStorage)

Параметры эксперимента	**Размер полученной информации (МБайт)**	**Количество кортежей**
Информация об объектах недвижимости без объединений (7 полей)	4,6	13 784
Информация об объектах недвижимости с объединением «Типы жилья» (8 полей)	4,6	9 189
Информация об объектах недвижимости с объединением «Районы» (8 полей)	4,6	9 189
Информация об объектах недвижимости с объединением «Собственники» (8 полей)	4,6	9 189
Информация об объектах, вся информация (10 полей)	3,9	4 594

Как видно из таблицы, размер локального хранилища браузера является достаточным для реализации кэша. Если предположить, что кэшироваться будет вся таблица с объединениями, то в кэше можно будет хранить до 4 600 кортежей, что является вполне нормальным количеством кортежей для работы одного специалиста.

В результате работы автором было исследовано время получения информации об объектах недвижимости в зависимости от различных условий. Для этого были спроектированы и проведены на тестовом стенде эксперименты, задачей которых была выборка данных по заданным условиям. В итоге были получены количественные оценки времени выполнения запроса в зависимости от количества запрашиваемых кортежей и объединений таблиц, проведен анализ данных и сделан вывод. Своей работой автор обосновал целесообразность использования кэша для хранения данных, необходимых для работы специалиста риелторской компании, и показал, как количество объединений таблиц в запросе влияет на производительность системы. Во второй части работы автором был поставлен вопрос о том, какое количество кортежей можно хранить в локальном хранилище браузера и можно ли использовать его для организации кэша. Для ответа на этот вопрос также были спроектированы и проведены эксперименты, которые показали, что локальное хранилище браузера целесообразно использовать для реализации кэша.

Список использованных источников

1 Хританков А. С. Модели и алгоритмы распределения нагрузки / А. С. Хританков // Информационные технологии и вычислительные системы. 2009. № 3. С. 33–48.

2 Halpin T., Morgan T. Information modeling and relational databases. Morgan Kaufmann Publishers, 2008. 970 p.

3 Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт. М. : Вильямс, 2006. 1316 с.

4 Лаврентьев К. А. Проблема интеграции новых баз данных в распределенную информационную систему предприятия / К. А. Лаврентьев, С. И. Смагин, Ю. В. Пономарчук // Высокие технологии, исследования, финансы, промышленность : сб. статей Восемнадцатой международ. науч.-практич. конференции. 4–5 декабря 2014 г. / под науч. ред. А. П. Кудинова, И. А. Кудинова. СПб. : Изд-во Политехн. ун-та, 2014. 200 с.