Научная статья
УДК 004.416.6:004.415.25
doi: 10.17586/2713-1874-2023-1-63-70

ВЫБОР СТРАТЕГИИ КЭШИРОВАНИЯ ДАННЫХ В РЕАЛИЗАЦИИ ПРОЕКТОВ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ РАБОТЫ С КЛИЕНТАМИ

Иван Владимирович Шуст[1]✉, Наталия Николаевна Горлушкина[2]

[1]АО «АЛЬФА-БАНК», Санкт-Петербург, Россия
[1,2]Университет ИТМО, Санкт-Петербург, Россия
[1]vt_shustiv@mail.ru✉
[2]nagor.spb@mail.ru, https://orcid.org/0000-0002-6549-1723
Язык статьи – русский

Аннотация: Для реализации бизнес-процессов взаимодействия с клиентами используются мобильные приложения. Авторами статьи поднята проблема неэффективного, а в некоторых случаях и неправильного использования кэширования в мобильных приложениях. Также указано, с какими сложностями могут столкнуться пользователи и к каким проблемам это может привести разработчиков приложения. Проведен анализ различных стратегий кэширования для мобильных приложений, таких как lazy кэш, synchronized кэш, write-through кэш и lru кэш. Описаны их преимущества и недостатки, выявлены узкие места рассматриваемых стратегий кэширования. Результаты анализа сведены в таблицу. Для каждой из рассматриваемой стратегии кэширования определены подходящие места использования в структуре мобильного приложения и предложены рекомендации по их использованию. На основе этих рекомендаций был составлен план по интеграции кэширования и приведен пример такой интеграции на тестовом приложении с универсальным функционалом. При проведении интеграции был аргументирован и объяснен выбор необходимой стратегии кэширования в каждом из модулей. Кроме того, проведены замеры количества обращений к серверу за данными до и после интеграции. В результате получены коэффициенты отношения количества запросов для каждой из стратегии.

Ключевые слова: кэш, мобильная разработка, стратегии кэширования, lazy кэш, lru кэш, synchronized кэш, write-through кэш

Ссылка для цитирования: Шуст И.В., Горлушкина Н.Н. Выбор стратегии кэширования данных в реализации проектов мобильных приложений для работы с клиентами // Экономика. Право. Инновации. 2023. № 1. С. 63–70. http://dx.doi.org/10.17586/2713-1874-2023-1-63-70.

CHOOSING A DATA CACHING STRATEGY IN THE IMPLEMENTATION OF MOBILE APPLICATION PROJECTS FOR WORKING WITH CLIENTS

Ivan V. Shust[1]✉, Natalia N. Gorlushkina[2]

[1]JSC «ALFA-BANK», Saint Petersburg, Russia
[1,2]ITMO University, Saint Petersburg, Russia
[1]vt_shustiv@mail.ru✉
[2]nagor.spb@mail.ru, https://orcid.org/0000-0002-6549-1723
Article in Russian

Abstract: Mobile applications are used to implement business processes of interaction with customers. The authors of the article raised the problem of inefficient, and in some cases incorrect, use of caching in mobile applications, and also indicated what difficulties users may face and what problems this may lead to application developers. The analysis of various caching strategies for mobile applications, such as lazy cache, synchronized cache, write-through cache and lru cache, is carried out. Their advantages and disadvantages are described, the bottlenecks of the caching strategies under consideration are identified. The results of the analysis are summarized in a table. For each of the caching strategies under consideration suitable places of use in the structure of the mobile application are identified and recommendations for their use are proposed. Based on these recommendations, a plan for caching integration was drawn up and an example of such integration was given on a test application with universal functionality. During the integration, the choice of the necessary caching strategy in each of the modules was reasoned and explained. In addition, measurements of the number of requests to the server for data before and after integration were carried out. As a result, the coefficients of the ratio of the number of requests for each of the strategies are obtained.

Keywords: cache, caching strategies, lazy cache, lru cache, mobile development, synchronized cache, write-through cache

For citation: Shust I.V., Gorlushkina N.N. Choosing a Data Caching Strategy in the Implementation of Mobile Application Projects for Working with Clients. *Ekonomika. Pravo. Innovacii*. 2023. No. 1. pp. 63–70. (In Russ.). http://dx.doi.org/10.17586/2713-1874-2023-1-63-70.

Введение. В информационных системах бизнеса, торговли, банкинга для организации взаимодействия с клиентами используются мобильные приложения. Эти приложения должны быть удобны для потребителей и не вызывать у них отрицательных эмоций при работе.

В любом «правильном» мобильном приложении между долгими запросами в сеть применяются различного рода тробберы, спиннеры, скелетоны, шиммеры и многое другое, что пользователь может распознать как индикаторы выполнения операции, загрузки данных [1]. Так пользователь понимает, что в скором времени он получит желаемый контент. Однако сеть может вести себя крайне непредсказуемо, пакеты могут теряться, токены доступа устаревать или доступ к сети просто пропадать. Все это ведет к ухудшению настроения пользователей и, как следствие, к потере интереса и уходу из мобильного приложения. По мере накопления такого негативного опыта пользователь может оставить плохой отзыв о приложении в маркете, долгое время не заходить туда или вообще удалить продукт со своего мобильного устройства [2]. В некоторых случаях худшим исходом может служить уход клиента к конкурентам, например, в сфере банкинга.

Эту проблему можно решить за счет увеличения пропускной способности своих сервисов или кэширования.

Следствием увеличения пропускной способности сервисов является повышение стоимости обслуживания и возникновение проблем при масштабировании. Кроме того, стоимость для пользователей возрастет при увеличении трафика [3].

Современные исследования утверждают [4–6], что использование кэша в мобильном приложении способствует удержанию клиентов, так как позволяет повысить скорость вывода контента на экран и за счет этого снизить возможную при длительном выводе негативную реакцию пользователя [7]. При отсутствии кэша многократное запрашивание данных из сети может привести к негативному пользовательскому опыту [4, 5].

Наиболее распространенными проблемами организация кэширования являются низкая эффективность для некоторых приложений [8], высокая стоимость и сложность реализации, возможная низкая производительность при управлении бизнес-процессами.

Постановка задачи. Для того чтобы разработчики успешно использовали технологии кеширования, необходимы рекомендации по правильному применению этих технологий.

Целью прикладного исследования является составление рекомендаций для корректного выбора стратегии кэширования данных в реализации проектов мобильных приложений для работы с клиентами на основе сравнительного анализа этих стратегий.

Методика исследования. На начальном этапе исследования требовались следующие методы исследования: поиск, сравнение и анализ данных. Их необходимо применить для определения узких мест каждой из стратегий кэширования, для выявления их сильных и слабых сторон, а также в формировании сравнительной таблицы их характеристик.

После проведения сравнения различных стратегий необходимо составить набор рекомендация по использованию стратегий кэширования, для этих целей хорошо подходит такой метод исследования, как обобщение опыта. Заключительным этапом исследования является проведение интеграций различных стратегий и проведение замеров количества запросов к серверу до и после интеграции. Для этого этапа в качестве метода исследования будет использоваться моделирование.

Полученные результаты. На первом этапе исследования был проведен поиск и анализ существующих стратегий кеширования. Как показал обзор источников и опыт специалистов, наиболее популярные стратегии кэширования для мобильных приложений являются lazy кэш [9], synchronized кэш [10], write-through кэш [11] и lru кэш [12].

Применение стратегий кэширования. Рассмотрим выбранные стратегии кеширования и проведем в дальнейшем их сравнительный анализ.

Lazy кэш – это самый простой вид кэширования, но его нужно использовать осторожно, так как он может отдавать устаревшие данные [9]. Плюсами этой стратегии кэширования является простота реализации, легкость в переиспользовании и отсутствие зависимости от интернет соединения. Однако минусом является отсутствие инвалидации данных.

Такой тип кэширования можно использовать для данных, которые почти никогда не меняются. Другой вариант использования – делать ленивый кэш с небольшим временем устаревания для стабильной работы при всплесках нагрузки. Такой тип кэширования позволит быстрее всех дать ответ.

Lazy стратегию кэширования данных стоит использовать в модулях с возможностью дозагрузки неизменяемых данных.

Synchronized кэш – это наиболее полезный тип кэширования, так как он отдает свежие данные и позволяет реализовать многоуровневый кэш [10]. Плюсами данной стратегии являются наличие ускоренной доставки данных и инвалидация кэша. Однако минусом выступает наличие зависимости от качества интернет соединения.

Такой тип кэширования встроен в протокол HTTP. Сервер передает метку изменения, а клиент кэширует на своей стороне результат, а при повторном запросе передает эту метку. Сервер может дать ответ, что состояние не изменилось и можно использовать закэшированные на клиенте данные. Сервер в свою очередь, получив метку, может переспросить у хранилища, были изменения или нет.

Synchronized стратегию кэширования данных стоит использовать в модулях с изменяемыми данными, однако без возможности повлиять на них.

Write-through кэш – это наиболее сложная в реализации стратегия [11]. Плюсами данной стратегии являются наличие ускоренной доставки данных и полной синхронизации с сервером. Однако минусами выступают зависимость от качества интернет соединения, сложность реализации и тот факт, что при неудачной записи нужно возвращаться в исходное состояние.

При некорректном использовании может вызвать негативный эффект. Собственная реализация синхронизации кэшей между узлами распределенной системы сама по себе очень сложна, потому не стоит заниматься ей в рамках разработки простого приложения. Также не стоит забывать, что системы распределенного кэширования также требуют общения между системами, что может сказываться на быстродействии. Как и в любой распределенной системе необходимо производить синхронизацию, например, с использованием логических часов Лэмпорта.

Write-through стратегию кэширования данных стоит использовать в модулях с изменяемыми данными, при этом имеется возможность повлиять на них. Идеально подойдет для реализации чатов.

LRU – это метод вытеснения из кэша [12]. Однако стратегия кэширования с использованием LRU имеет гибкую настройку инвалидации, имея все преимущества lazy кэша. Так как неправильная настройка инвалидации может сломать все приложение, то важно внимательно относиться к этому. Можно использовать механизм фича тоглов или удаленной настройки. LRU кэш отлично дополнит любую из стратегий.

LRU стратегию кэширования данных стоит использовать в модулях с тяжеловесным контентом, например, в приложениях с множеством документов, картинок или видео.

Узкие места стратегий кэширования. Главным узким местом всех стратегий является их настройка и управление процессом кэширования данных.

При выборе стратегии кэширования необходимо исходить, в первую очередь, из особенностей кэшируемых данных, а также из количества запросов этих данных. Кэширование всех данных может привести к неэффективному расходу ресурсов устройства. Также если процесс кэширования никак не контролируется, могут возникнуть проблемы с устареванием данных и когерентностью кэша. Применять кэширование данных необходимо только в том случае, если возникают проблемы с производительностью. Очевидно, что производить кэширование данных необходимо далеко не на всех экранах приложения.

Используемые в приложении библиотеки с медиа ресурсами или СУБД имеют собственные реализации по работе с кэшем, поэтому не имеет смысла кэшировать результаты запросов [13].

Важно понимать на этапе подключения кэширования, где оно будет использоваться, а где его быть точно не должно. В противном случае может нарушиться логика работы приложения. Также необходимо учитывать риски того, что мобильные устройства клиентов могут иметь малый объем памяти. Возможно возникновение ситуации, когда размер свободного места меньше, чем те данные, которые необходимо сохранить в кэше. Поэтому нужно принимать во внимание, что старые данные могут быть вытеснены. Некоторые стратегии кэширования позволяют очень гибко управлять устареванием, учитывая различные признаки, такие как: приоритет, время устаревания, объем данных и так далее [10].

Из-за некорректной работы с очисткой и обновлением кэша могут возникнуть проблемы, связанные с нарушением согласованности поведения приложения. Один из способов поддержания когерентности данных – это принудительный сброс кэша или его обновление до актуального состояния. По этой причине не всегда хорошей идеей является увеличение памяти для кэша, чтобы он меньше устаревал.

Такую проблему можно решить с помощью настройки поведения кэша. Управление конфигурацией можно осуществлять удаленно через фича тоглы или через консоль.

Сравнительный анализ стратегий. Все оценки в Таблице 1 указаны сугубо для стратегий без каких-либо доработок и улучшений. Часть этих характеристик можно улучшить с помощью вынесения логики стратегии кэширования в отдельный модуль, добавив возможность настройки поведения.

Как видно из Таблицы 1, простотой в реализации и легкостью в переиспользовании обладает только Lazy кэш. Обусловлено это тем, что вся бизнес-логика по кэшированию данных реализуется на клиентской части.

Другие стратегии кэширования также можно переиспользовать, однако, для этого необходимо провести группировку функционала приложения на основании сходства работы с данными. Отсутствие доступа к сети для успешной работы с кэшем в ряде случаев не является проблемой.

Выбор той или иной стратегии кэширования определяется бизнес требованиями. Так, например, в приложении с заполнением форм хранить черновик формы не обязательно на сервере, можно использовать обычный Lazy кэш, однако в случае чатов такая стратегия явно не подойдет.

Как уже было упомянуто выше, гибкость кэширования определяется способностью последнего к настройке. Выбор конфигурации можно осуществлять как на основании информации, полученной по сети от сервера, так и на основании данных внутри самого приложения. Например, на медленных устройствах или устройствах, имеющих малый объем свободной памяти, можно выбирать ту или иную стратегию инвалидации кэша.

Таблица 1

Сводная таблица стратегий кэширования

Источник: составлено авторами

	Lazy	Synchronized	Write-through	LRU
Простота реализации	да	нет	нет	нет
Легкость в переиспользовнии	да	нет	нет	нет
Зависимость от соединения	нет	да	да	да
Инвалидация кэша	нет	да	да	да
Синхронизация с сервером	нет	нет	да	нет
Гибкая настройка	нет	нет	нет	да

Рекомендации по интеграции кэширования. Исходя из опыта одного из авторов и проведенного исследования, были сделаны рекомендации. Перед тем как выбрать стратегию кэширования для приложения, необходимо определиться, требуется ли она вообще. Для ряда приложений кэширование данных является недопустимым или частично недопустимым. К таким приложениям можно отнести приложения из банковского сектора, приложения видеохостинга и другие.

Разумеется, что данные, которые будут часто меняться или они каждый раз будут новыми, кэширование не имеет смысла. Также в ряде случаев может быть недопустимым кэшировать персональные данные. Важно понимать, что если кэширование не было реализовано на этапе проектирования приложения, то в дальнейшем это может вызвать глобальный рефакторинг всего приложения.

Все шаги интеграции кэширования проверялись на тестовом приложении. Само приложение несло в себе собирательный характер и состояло из нескольких основных блоков: экрана покупок, профиля, настроек, главного экрана, чата.

Первым шагом явился выбор стратегии. В одном приложении можно, а зачастую и необходимо, использовать несколько стратегий кэширования. Как уже было сказано, у каждой стратегии есть свои плюсы и минусы, область применимости у них разная. Поэтому необходимо разбивать весь функционал приложения на группы на основании сходства работы с данными. Для каждой из группы выбирать оптимальную стратегию кэширования. На первом этапе не обязательно выбирать размер кэша или скорость его работы, так как эти параметры можно настраивать удаленно через фича тоглы. Лучше всего использовать механизм A/B тестирования для определения характеристик кэширования для каждой из групп, так как он позволяет проанализировать поведение пользователя и проверить различные гипотезы.

Экран покупок является основным экраном приложения, так как именно на нем пользователями совершаются операции, непосредственно приносящие прибыль. Необходимость применения кэширования на экране определяется стратегией ведения бизнеса. Если предполагается, что пользователь будет совершать несколько операций в год, а вся надежда идет на количество пользователей, то применять кэширование нет никакого смысла. Однако
если исходить из того, что пользователь будет совершать множество операций, то кэ-

ширование в данном случае можно использовать. Для этого блока стоит использовать synchronized кэш, так как часть данных, таких как цена или список платежных систем, могли измениться. Для экранов настроек и профиля можно использовать lazy кэш, так как данные для этих блоков нельзя изменить извне. Однако это не распространяется на фото пользователя из профиля, так как это – медиа ресурс, для такой части профиля стоит использовать LRU кэш, хотя его и нельзя изменить извне. Обратная ситуация с чатом, где список сообщений зависит не только от клиентов, но и от оператора, поэтому в этом случае необходимо использовать write-through кэш. Для главного же экрана использовать какой-либо кэш не стоит, так как информация на нем может очень быстро меняться.

Следующим шагом является определение места хранения кэшируемых данных. Такими местами могут быть: база данных, файл, оперативная память или preferences. При выборе места хранения стоит обращать внимание на скорость работы с хранилищем, его допустимый объем и сложность в реализации.

Так как в рассматриваемом приложении предполагается совершение множества покупок, то для подобных данных лучше использовать реляционные базы данных. Это же относится и к модулю чата. Причиной такого выбора является большой объем данных, а также наличие одинаковых по структуре объектов, в первом случае – это товар, во втором – сообщение. Для простых данных, таких как названия чего-либо или числовые значения, экраны профиля и настроек подойдет preferences. А вот для фото пользователя с экрана профиля подойдет внутреннее хранилище приложения, оно там может храниться, как файл.

Основной параметр, который характеризует стратегию кэширования – это процент попаданий запросов в кэш. Этот параметр довольно легко измерить, чтобы понять, насколько выбранная стратегия кэширования эффективна. Если подключить инструмент A/B тестирования, то можно изменять натройки выбранных стратегий кэширования. Получая при этом информацию о проценте попадания, можно подобрать оптимальные настройки для конкретного приложения. Так, например, если часто сбрасывать кэш, то будет происходить редкий запрос данных, а если будет недостаточный объем кэша, то это приведет к пустой трате оперативной памяти. При этом повышение эффективности работы не произойдет. Подобные зависимости уникальны для каждого приложения ввиду различной бизнес-логики, поэтому интеграция процесса кэширования в приложение без наличия проверки гипотез не столь эффективна.

Предварительные замеры, проведенные в приложении до и после интеграции кэширования, представляют из себя следующую картину. Коэффициент отношения количества запросов в сеть до и после интеграции для synchronized кэша равен 2, а для LRU и write through кэшей – 1,5. Эти коэффициенты являются таковыми сугубо для рассматриваемого приложения, так как в других приложениях кэшируемый объем данных будет другой, как и место его хранения. Для lazy кэша коэффициент не был определен, так как он зависит от количества запросов и от объема доступной памяти на устройстве. Так как приложение тестовое, то полученный результат по этой стратегии не отражал бы реального результата. Также были попытки на первом шаге изменить выбранные стратегии на другие, что привело к поломке логики работы приложения и некорректному отображению данных. В случае использования неподходящей стратегии кэширования модулем оплаты в реальном приложении привело бы к финансовым потерям. Также на экранах профиля и настроек было замечено уменьшение скорости отдачи контента, так как при использовании более сложных стратегий кэширования увеличивалось количество действий для получения данных, хотя первоначально там использовалась самая простая стратегия кэширования.

Выводы. В результате исследования подготовлены рекомендации по внедрению каждого из видов кэширования, а также описаны особенности, на которые необходимо обратить внимание при интеграции.

Перед внедрением кэширования нужно вначале понять, необходимо ли оно. В боль-

шинстве случаев на раннем этапе разработки от кэширования можно отказаться. Однако всегда нужно помнить, что в дальнейшем оно может понадобиться, и выстраивать архитектуру приложения соответствующим образом.

В процессе интеграции кэширования не стоит отдавать предпочтение какой-то конкретной стратегии, так как у каждой есть преимущества, недостатки и зона применения. По этой причине в самом начале нужно разобраться, где и какую стратегию целесообразно использовать.

С помощью предложенных рекомендаций можно улучшить производительность и повысить организационные параметры систем, использующих мобильные приложения. Сохранение данных в мобильном приложении ведет к уменьшению количества запросов в сеть, снижая трафик, и к повышению позитивного пользовательского опыта.

Если правильно определиться со стратегией кэширования данных в конкретном модуле, то можно получить заметное повышение скорости предоставления услуг без потери качества.

Список источников

1. Фаулер М. Архитектура корпоративных программных приложений: Пер. с англ. – М.: Издательский дом «Вильямс», 2007.
2. Скрутелев Е.С., Кузнецов А.А.. Особенности разработки мобильных приложений для устройств под управлением ОС iOS с использованием технологии bluetooth low energy // Экономика. Право. Инновации. 2022. № 2. С. 56–62.
3. Hao Jin Dan, Xu Chenglin, Zhao Dong Liang. Information-centric Mobile Caching Network Frameworks and Caching Optimization: a Survey // Eurasip Journal on Wireless Communications and Networking. 2017. (In Eng.).
DOI 10.1186/s13638-017-0806-6
4. Wickham M. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. – Издательство Apress, 2018. – 257 С. (In Eng).
5. Montane R.R., Dawson L. Learning Android Application Development. – Издательство Packt Publishing, 2016. – 320 С. (In Eng.).
6. Для чего нужно кеширование данных в мобильных приложениях [Электронный ресурс] – Режим доступа: https://app72.ru/blog/288-dlya-chego-nuzhno-keshirovanie-dannykh-v-mobilnykh-prilozheniyakh
7. Коэффициент удержания пользователей мобильного приложения [Электронный ресурс]. – Режим доступа: https://asomobile.net/blog/koefficzient-uderzhaniya-polzovatelej-mobilnogo-prilozheniya
8. Бринкли М., Чхабра Д. Взлеты и падения технологии кеширования // Amazon Builders' Library [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/builders-library/caching-challenges-and-strategies/
9. Ranasinghe K. How to Use LazyCache in Your .Net Core Application – Beginner’s Guide [Электронный ресурс]. – Режим доступа: https://medium.com/swlh/how-to-use-lazycache-in-your-net-core-application-beginners-guide-c413724 1591 (In Eng.).

References

1. Fowler M. Architecture of Corporate Software applications. Trans. from English. *Isdatelskiy dom Vil’ams.* 2007. (In Russ.).
2. Skrutelev E.S., Kuznetsov A.A. Features of Developing Mobile Applications for iOS Devices Using Bluetooth Low Energy Technology. *Ekonomika. Pravo. Innovacii.* 2022. No. 2. pp. 56–62. (In Russ.).
3. Hao Jin Dan, Xu Chenglin, Zhao Dong Liang. Information-centric Mobile Caching Network Frameworks and Caching Optimization: a Survey. *Eurasip Journal on Wireless Communications and Networking.* 2017.
DOI 10.1186/s13638-017-0806-6
4. Wickham M. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. *Apress.* 2018. 257 p.
5. Montane R.R., Dawson L. Learning Android Application Development. *Packt Publishing.* 2016. 320 p.
6. Why Do You Need Data Caching in Mobile Applications. Available at: https://app72.ru/blog/288-dlya-chego-nuzhno-keshirovanie-dannykh-v-mobil nykh-prilozheniyakh (In Russ.).
7. Retention Rate of Mobile Application Users. Available at: https://asomobile.net/blog/koefficzient-uderzhaniya-polzovatelej-mobilnogo-prilozheniya (In Russ.).
8. Brinkley M., Chhabra D. Ups and Downs of Caching Technology. *Amazon Builders' Library*. Available at: https://aws.amazon.com/ru/builders-library/caching-challenges-and-strategies/ (In Russ.).
9. Ranasinghe K. How to Use LazyCache in Your .Net Core Application – Beginner’s Guide. Available at: https://medium.com/swlh/how-to-use-lazycache-in-your-net-core-application-beginners-guide-c413724 1591

10. Стратегия кеширования в приложении [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/168725/
11. Caching Strategies and How to Choose the Right One [Электронный ресурс]. – Режим доступа: https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/ (In Eng.).
12. Хранение изображений с помощью LruCache [Электронный ресурс]. – Режим доступа: https://android-tools.ru/coding/xranenie-izobrazhenij-s-pomoshhyu-lrucache/
13. Иванов В.И., Новиков С.П.. Программное средство исследования различных алгоритмов кэширования в реляционных СУБД // Молодой исследователь Дона. 2018. № 3 (12).

10. Caching Strategy in the Application. Available at: https://habr.com/ru/post/168725/ (In Russ.).

11. Caching Strategies and How to Choose the Right One. Available at: https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/
12. Storing Images Using LruCache. Available at: https://android-tools.ru/coding/xranenie-izobrazhenij-s-pomoshhyu-lrucache/ (In Russ.).

13. Ivanov V.I., Novikov S.P. Software Tool for Researching Various Caching Algorithms in Relational DBMS. *Molodoy issledovatel Dona.* 2018. No. 3 (12). (In Russ.).

Статья поступила в редакцию 04.03.2023; одобрена после рецензирования 07.03.2023; принята к публикации 10.03.2023.
The article was submitted 04.03.2023; approved after reviewing 07.03.2023; accepted for publication 10.03.2023.