ISSN: 2181-4171

Engineering Problems and Innovations

Vol. 03 Iss. 02 | | April. 27th, 2025

Том - 03 | | Вып. - 02
Tom - 03 | | Son. – 02

Проблемы инженерии и Инновации

Muhandislik muammolari va innovatsiyalar



SCAN ME

Google Scholar
INDEX COPERNICUS INTERNATIONAL
zenodo
Academic Resource Index ResearchBib

Главный редактор

Зокиров Санжар Икромжон угли д.ф.ф.-м.н., PhD

Журнал зарегистрирован 28 декабря 2022 года Агентством информации и массовых коммуникаций при Администрации Президента Республики Узбекистан

Сертификат № 055940
ISSN 2181-4171

Языки издания: Русский, английский, узбекский

Председатель эксп. совета

Ф.М.Мухтаров - д.ф.т.н, PhD. директор ФФ ТУИТ

Ответственные редакторы разделов

Т.М.Абдуллаев
д.ф.т.н., PhD, ФФ ТУИТ

Н.И.Иброхимов
д.ф.ф.-м.н., PhD, ФФ ТУИТ

А.Р.Нурматов
PhD, д.ф.ист.н.,НамГУ

Г.А.Умарова
к.п.н., доц., АндМИ

Корректоры

Порубай Оксана Витальевна
Лазарева Марина Викторовна
Обиджонов Зафаржон Одилжон угли

Редакционная коллегия:

Б.К.Сарсенбаев - *д.т.н, проф., акад., дейст. член Нац. Акад. Респ. Казахстан*
В.Ю.Рудь - *д.ф.-м.н., Физико-технический институт им. А.Ф.Иоффе, Санкт-Петербург, Россия*
Н.Б.Бейсенханов - *д.ф.-м..н., проф. АО "Казахстанско-Британский Технический Университет", Алматы, Казахстан*
Б.Р.Исмаилов - *д.т.н., проф. Южно-Казахстанский университет им. М. Ауэзова , Шымкент, Казахстан*
А.В.Аржанников - *д.ф.-м.н., Новосибирский государственный университет, Институт ядерной физики СО РАН, Новосибирск, Россия*
А.М.Касимахунова - *д.т.н., проф. ФерПИ, Фергана, Узбекистан*
И.Бйорно - к.*н.,PhD, Имеритус, Датский технический университет, Копенгаген, Дания*
Л.К.Мамадалиева - *д.т.н., проф. ФерПИ, Фергана, Узбекистан*
Ж.Т.Айменов - *д.т.н., проф. Южно-Казахстанский университет им.М.Ауезова, Шымкент, Казахстан*
Х.Б.Исмаилов - *к.т.н. доцент, Южно-Казахстанский университет им. М. Ауэзова , Шымкент, Казахстан*
В.В.Краснощеков - *к.т.н. Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, Российская Федерация*
М.Д.Шабданов - *к.т.н. ОшТУ , Ош, Киргизстан*
В.М.Голубова - *к.псх.н. Ростовский институт (филиал) ВГУЮ (РПА Минюста России), Ростов-на-Дону, Россия*

Коллегия экспертного совета

О.Х.Отакулов - *к.т.н., доц. Ферганский филиал ТУИТ, Узбекистан*
О.С.Райимжонова - *д.ф.т.н., PhD, доц, Ферганский филиал ТУИТ, Узбекистан*
Р.А.Нурдинова - *д.ф.т.н., PhD, доц., Ферганский филиал ТУИТ, Узбекистан*
М.Э.Юнусалиев - *к.т.н., PhD, доц., ФерПИ, Узбекистан*
М.И.Исмоилов - *к.т.н., PhD, доц., ФерПИ, Узбекистан*
М.Т.Халилов - *к.т.н., доц., АндМИ, Узбекистан*
Г.А.Дадамирзаева - *д.ф.ист.н., PhD, доц., НамГУ, Узбекистан*
И.Т.Тожибоев - *к.ф.-м.-н., доц., ФерГУ, Узбекистан*
А.Ш.Тухтабаев - *д.ф.ист.н., PhD, доц., НамГУ, Узбекистан*
И.М.Исакова - *д.п.н. доц., ФерГУ, Узбекистан*
М.М.Турдиматов - *к.т.н, доц., Ферганский филиал ТУИТ, Узбекистан*
Д.Ф.Тухтасинов - *д.ф.пед.н., PhD, доц., Ферганский филиал ТУИТ, Узбекистан*
Н.Д.Ботирова - *д.ф.пед.н., PhD, доц., Ферганский филиал ТУИТ, Узбекистан*
Ш.Ж.Холматов - *к.ф.н., доц., Ферганский филиал ТУИТ, Узбекистан*

При использовании материалов журнала необходимо указывать источник.
Авторы статей несут ответственность за содержание статей и за сам факт их публикации.
Редакция не всегда разделяет мнение авторов и не несет ответственности за недостоверность опубликованных данных.
Редакция журнала не несет никакой ответственности перед авторами и/или третьими лицами и организациями за возможный ущерб, причиненный публикацией статьи.

Editor in chief

Zokirov Sanjar Ikromjon ugli
PhD in Phys. and Math. Sc.

The journal has been registered on December 28, 2022 by the Agency for Information and Mass Communications under the Administration of the President of the Republic of Uzbekistan

Certificate No. 055940
ISSN 2181-4171

Languages:
Russian, English, Uzbek

Chairman of exp. advice

F.M.Muxtarov - Ph.D. in tech. Sc., director of FBTUIT

Responsible section editors

T.M.Abdullaev
PhD in Tech. Sc., FBTUIT

N.I.Ibrokhimov
PhD in phys. and math. Sc., FB of TUIT

A.R.Nurmatov
Ph.D. in hist. sc., NamSU

G.A.Umarova
Cand. of ped. sc, assoc. prof., AndMI

Proofreaders
Porubay Oksana Vitalevna
Lazareva Marina Viktorovna
Obidjonov Zafarjon Odiljon ugli

Editorial board:

B.K.Sarsenbaev– *Dr. of Tech. Sc., prof. Actual member of the National Acad. Rep. Kazakhstan*
V.Yu.Rud'– *Dr. of Phys. and Math. Sc., Ioffe Physicotechnical Institute, St.-Petersburg, Russia*
N.B.Beisenkhanov– *Dr. of Phys and Math. Sc. (Ds), full professor in Physics, JSC "Kazakh-British Technical University", Almaty, Kazakhstan*
B.R.Ismailov– *Dr. of Tech. Sc., prof., Mukhtar Auezov South Kazakhstan State University, Shymkent, Kazakhstan*
A.V.Arzhannikov– *Dr. of Sc. in Phys. and Math., prof., Novosibirsk St. University, Budker Institute of Nuclear Physics, Russian Federation*
A.M.Kasimahunova - *Dr. of Tech. Sc., prof., Fergana Polytechnic Institute. Fergana, Uzbekistan*
I.Bjørnø - *Assoc. Prof. PH.D, DTU imeritus, København, Denmark*
L.K.Mamadaliyeva - *Dr. of Tech. Sc., prof., Fergana Polytechnic Institute. Fergana, Uzbekistan*
Zh.T.Aimenov– *Dr. of Engineering (Ds), prof. Mukhtar Auezov South Kazakhstan State University, Shymkent, Kazakhstan*
Kh.B.,Ismailov - *PhD in Tech. Sc., Mukhtar Auezov South Kazakhstan State University, Shymkent, Kazakhstan*
V.V.Krasnoshchekov - *PhD in Engineering, Peter the Great St. Petersburg Polytechnic University, Russian Federation, St. Petersburg*
M.D.Shabdanov– *Can. of Tech. Sc., Osh Technological University, Osh, Kyrgyzstan*
V.M.Golubova - *PhD in Psych., Rostov Institute (branch) VGUYu (RPA of the Ministry of Justice of Russia) Russia, Rostov-on-Don*

Expert Board

O.Kh.Otakulov– *Cand. of tech. sc., assoc. prof., Fergana branch of TUIT, Uzbekistan*
O.S.Raimzhonova - *PhD, assoc. prof., Fergana branch of TUIT, Uzbekistan*
R.A.Nurdinova - *PhD, assoc. prof., Fergana branch of TUIT, Uzbekistan*
E.M.Yunusaliev - *Ph.D in Technics, assoc. prof., FerPI, Uzbekistan*
M.M.Ismoilov - *Cand. of tech. sc., PhD, assoc. prof., FerPI, Uzbekistan*
M.T.Khalilov - *Cand. of tech. sc., assoc. prof., AndMI, Uzbekistan*
G.A.Dadamirzaeva - *Ph.D in Hist. Sc., NamSU, Uzbekistan*
I.T.Tojiboev - *Cand. of Phys. and Math. Sc., assoc. prof., FerSU, Uzbekistan*
A.Sh.Tukhtabaev - *Ph.D in Hist. Sc., NamSU, Uzbekistan*
M.T.Isakova - *DSc in Psychology, FerSU, Uzbekistan*
M.M.Turdimatov - *Cand. of tech. sc., PhD, assoc. prof., Fergana branch of TUIT, Uzbekistan*
D.F.Tukhtasinov - *PhD on Ped., assoc. prof., Fergana branch of TUIT, Uzbekistan*
N.D.Botirova - *PhD on Ped., assoc. prof., Fergana branch of TUIT, Uzbekistan*
Sh.J.Kholmatov - *cand. phil., assoc. prof., Fergana branch of TUIT, Uzbekistan*

When using journal materials, you must indicate the source.
The authors of the articles are responsible for the content of the articles and for the very fact of their publication.
The editors do not always share the opinions of the authors and are not responsible for the unreliability of the published data.
The editorial board of the journal does not bear any responsibility to the authors and / or third parties and organizations for possible damage caused by the publication of the article.

Многопоточная обработка данных в SQLite с использованием C#

ТОКТОРБАЕВ АЙБЕК МАМАДАЛИЕВИЧ, КАРАБАЕВ САМАТ ЭШПОЛОТОВИЧ, МОЙДУНОВА АЙГЕРИМ СУЙУНБАЕВНА, АБДЫКАДЫРОВ СУЛТАНБЕК КАНЫБЕКОВИЧ

Ошский государственный университет, Ош, Кыргызстан

Copyright: © 2025 by the author.
Licensee: This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).

Аннотация: *В данной статье рассматриваются вопросы многопоточной обработки данных в системах, использующих SQLite в качестве встроенной СУБД в C#-приложениях. Основное внимание уделяется анализу архитектурных особенностей SQLite, проблемам конкурентного доступа к данным, а также современным подходам к реализации многопоточности в среде .NET. Описаны методы синхронизации, оптимизации транзакционной обработки и минимизации конфликтов доступа, что позволяет обеспечить высокую производительность и стабильность приложений. В статье приводятся результаты экспериментальных исследований, подтверждающие эффективность предложенных методик, и делаются рекомендации по их применению в реальных проектах.*

Ключевые слова: *C#, SQLite, многопоточность, синхронизация, конкурентный доступ, оптимизация транзакций, .NET-разработка.*

Multithreaded data processing in SQLite using C#

Toktorbaev Aybek Mamadalievich,
Karabaev Samat Eshpolotovich,
Moidunova Aigerim Suiunbaevna,
Abdykadyrov Sultanbek Kanybekovich

Osh State University, Kyrgyz Republic, Osh

Abstract: *This article addresses multithreaded data processing in systems using SQLite as an embedded DBMS in C# applications. The focus is on analyzing the architectural features of SQLite, the challenges of concurrent data access, and contemporary approaches to implementing multithreading in the .NET environment. Methods for synchronization, transaction optimization, and conflict minimization are described to ensure high performance and application stability. Experimental results validating the efficiency of the proposed techniques are presented, and recommendations for their practical implementation are provided.*

Keywords: *C#, SQLite, multithreading, synchronization, concurrent access, transaction optimization, .NET de velopment.*

Введение

В условиях стремительного развития программных систем и увеличения объемов обрабатываемых данных необходимость в эффективной многопоточной обработке становится все более актуальной. С появлением мобильных и десктопных приложений, где каждая миллисекунда отклика имеет значение, разработчики сталкиваются с проблемами конкурентного доступа к данным. Особое место в этом контексте занимает использование SQLite – легковесной, но мощной СУБД, активно применяемой для разработки встроенных решений. SQLite отличается простотой интеграции и отсутствием необходимости в отдельном сервере, что делает её привлекательной для разработчиков на C#. Однако особенности архитектуры SQLite, в частности механизм блокировок и особенности работы с транзакциями, требуют тщательного анализа при реализации многопоточных сценариев. В данной статье рассматриваются как фундаментальные аспекты архитектуры SQLite, так и современные методы синхронизации, реализуемые в C# на платформе .NET.

Основными задачами исследования являются:

- анализ архитектурных особенностей SQLite при работе в многопоточном режиме;
- выявление проблем конкурентного доступа и их влияния на производительность системы;
- разработка и экспериментальная проверка подходов по оптимизации транзакций и

синхронизации потоков в C#;

- формулирование практических рекомендаций для разработчиков, позволяющих повысить стабильность и эффективность обработки данных.

Проблематика многопоточной обработки данных

В SQLite актуальна не только для мобильных и настольных приложений, но и для встроенных систем, где ресурсы ограничены, а требования к быстродействию крайне высоки. С учетом возрастания роли распределенных вычислений и параллельной обработки задач, изучение методов синхронизации в SQLite становится необходимым этапом совершенствования современных программных решений. В научной литературе проблема обеспечения конкурентного доступа к данным в СУБД рассматривается с различных точек зрения. Ряд исследований посвящен оптимизации транзакционных процессов в реляционных базах данных [3], [8]. При этом специфические особенности SQLite, являющейся встроенной СУБД, требуют отдельного подхода в сравнении с серверными СУБД, такими как SQL Server или PostgreSQL. Ранние работы по теме многопоточности в SQLite фокусировались на анализе базовых механизмов блокировки. Авторы отмечали, что использование режима «Serialized» позволяет обеспечить полную безопасность транзакций, однако это может негативно сказываться на производительности в условиях высокой конкуренции [10]. Впоследствии исследования расширились за счет изучения альтернативных режимов работы – «Multi-thread» и «Single-thread», что позволило разработчикам выбирать баланс между скоростью и безопасностью обработки данных. Кроме того, важное внимание уделялось интеграции SQLite с объектно-ориентированными языками программирования, в частности C#. Исследования [1] и [4] показывают, что грамотное применение паттернов проектирования, таких как Repository и Unit of Work, может существенно упростить реализацию многопоточных приложений и снизить вероятность возникновения гонок данных. Однако, несмотря на большое количество публикаций, остаются нерешенными вопросы оптимизации работы SQLite в условиях высокой нагрузки, а также проблемы, связанные с эффективной синхронизацией потоков в .NET-среде. Обзор отечественных исследований демонстрирует недостаточную глубину анализа специфики работы SQLite в многопоточных приложениях. Некоторые авторы [5] отмечают, что использование SQLite в мобильных устройствах требует не только оптимизации транзакций, но и корректной настройки параллелизма для минимизации блокировок. В связи с этим актуальность выбранной темы обоснована необходимостью создания комплексной методологии, учитывающей как особенности СУБД, так и специфику реализации на платформе C#.

Таким образом, анализ существующих работ выявляет необходимость систематизации знаний о многопоточной обработке данных в SQLite с фокусом на практическую реализацию в C#-приложениях. В данной статье предпринята попытка объединить теоретический анализ с результатами экспериментальных исследований, что позволит сформировать рекомендации для оптимизации конкурентного доступа и синхронизации потоков.

SQLite представляет собой встраиваемую реляционную СУБД, не требующую отдельного серверного процесса. Основными преимуществами SQLite являются её компактность, простота установки и высокая скорость работы в условиях небольших и средних объемов данных. Однако именно эти особенности обуславливают специфику работы в многопоточном режиме. Ключевой особенностью SQLite является механизм блокировок на уровне базы данных. При выполнении транзакций SQLite использует блокировки для обеспечения целостности данных, что особенно важно в условиях параллельного доступа. В режиме «Serialized» все запросы выполняются последовательно, что гарантирует безопасность, но может снижать производительность при одновременной работе большого числа потоков. Режимы «Multi-thread» и «Single-thread» позволяют добиться большего параллелизма, однако требуют от разработчика

повышенной ответственности за управление синхронизацией и предотвращением гонок данных. Кроме того, SQLite использует механизм журналирования (Write-Ahead Logging, WAL), который позволяет улучшить производительность при выполнении операций записи. В режиме WAL новые изменения записываются в отдельный лог-файл, что снижает вероятность конфликтов между чтением и записью данных. Однако применение WAL в условиях высокой конкуренции требует дополнительных мер по синхронизации потоков, особенно при одновременном доступе из нескольких потоков в C#.

Платформа .NET и язык программирования C# предоставляют разработчикам широкий спектр инструментов для реализации многопоточной обработки данных. Среди них можно выделить:

Класс Thread и ThreadPool позволяют создавать и управлять потоками на низком уровне. Задачи (Task) и асинхронное программирование: Обеспечивают удобный механизм организации параллельных вычислений с использованием async/await. Механизмы синхронизации: Мьютексы, семафоры, блокировки (lock), ReaderWriterLockSlim и другие инструменты, необходимые для предотвращения состояния гонок. В контексте работы с SQLite необходимо учитывать, что многие библиотеки для работы с данной СУБД реализованы с учетом потоковой безопасности только в определенных режимах. Например, библиотека System.Data.SQLite предоставляет режимы работы, оптимизированные для многопоточной обработки, однако требует явного указания нужного режима синхронизации при инициализации подключения к базе данных.

При реализации многопоточной обработки данных разработчик сталкивается с рядом проблем, связанных с особенностями реализации синхронизации как на стороне SQLite, так и в C#. Необходимо учитывать следующие моменты: Управление транзакциями: При одновременной работе нескольких потоков может возникнуть ситуация, когда несколько транзакций пытаются записать данные в базу одновременно, что приводит к конфликтам и, как следствие, к ошибкам выполнения. Применение механизмов блокировок на уровне C# (например, lock) позволяет минимизировать риск таких конфликтов. Конфигурация подключения: Выбор режима работы SQLite (Serialized, Multi-thread или Single-thread) имеет решающее значение для обеспечения корректного взаимодействия потоков. Неправильно выбранный режим может привести как к снижению производительности, так и к нарушению целостности данных. Обработка исключительных ситуаций: В условиях многопоточной работы необходимо предусмотреть корректное восстановление после ошибок, возникающих в результате конфликтов доступа. Важно реализовать механизмы отката транзакций и повторных попыток выполнения запросов. Для визуализации взаимодействия потоков и процессов синхронизации в C# с SQLite целесообразно использовать диаграммы последовательности и блок-схемы алгоритмов обработки данных.

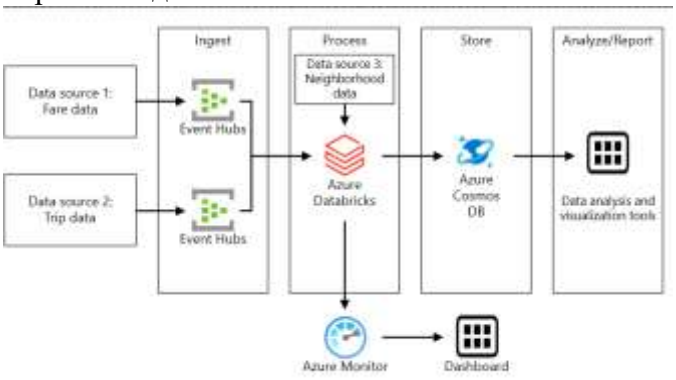


Рисунок 1. Схематичное представление потоковой обработки в SQLite с использованием C#

В условиях многопоточной обработки данных в SQLite особое внимание уделяется проблемам, связанным с конкурентным доступом к базе данных. Одновременные операции чтения и записи, выполняемые из разных потоков, могут привести к следующим негативным явлениям:

- При отсутствии надлежащей синхрони-зации несколько потоков могут одновременно изменять одни и те же данные, что приводит к нарушению целостности информации.

Режим работы	Производительность при чтении	Производительность при записи	Уровень блокировок	Примечания
Serialized	Средняя	Средняя	Высокий	Гарантирует целостность данных
Multi-thread	Высокая	Высокая	Средний	Требует дополнительной синхронизации
Single-thread	Максимальная	Максимальная	Отсутствует	Подходит для однопоточных сценариев

Таблица 1. Сравнительный анализ режимов работы SQLite

- Неэффективное управление блокировками способно привести к тому, что потоки застревают в состоянии ожидания освобождения ресурса, что негативно сказывается на общей производительности системы.

- Избыточное применение механизмов синхронизации может замедлять выполнение операций, особенно в сценариях с высокой нагрузкой, что является критичным для систем, требующих быстрого отклика.

Платформа .NET и язык C# предлагают ряд механизмов для реализации синхронизации, позволяющих обеспечить корректный доступ к разделяемым ресурсам:

- Обеспечивает создание критических секций, где в один момент времени выполняется только один поток, что предотвращает состояния гонок.

- Мьютексы применяются для межпроцессной синхронизации, а семафоры позволяют ограничить количество потоков, одновременно обращающихся к ресурсу.

- Эффективен при сценариях с преобладанием операций чтения, позволяя нескольким потокам одновременно считывать данные, но ограничивая доступ при записи.

- Обеспечивает эффективное управление задачами без излишнего блокирования потоков, что способствует повышению общей производительности системы.

Эффективное применение описанных методов требует тщательной настройки и комплексного тестирования, особенно в интегрированных сценариях, где режим работы SQLite (Serialized, Multi-thread, Single-thread) оказывает существенное влияние на поведение системы.

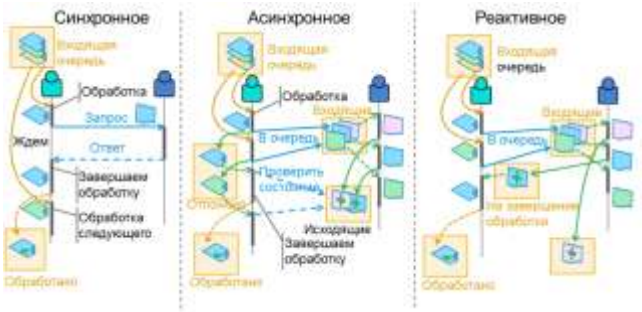


Рисунок 2. Схема взаимодействия потоков при использовании механизмов синхронизации в C#

Для оценки эффективности различных подходов к оптимизации многопоточной обработки данных в SQLite с использованием C# было разработано экспериментальное исследование. Основной целью эксперимента являлась оценка производительности системы, уровня блокировок и количества конфликтных операций при реализации различных сценариев конкурентного доступа.

В исследовании было разработано тестовое приложение, реализующее следующие сценарии:

- Одновременное выполнение операций чтения и записи в базе данных SQLite.

- Изменение конфигурации режимов работы SQLite: Serialized, Multi-thread и Single-thread.

- Применение различных механизмов синхронизации, включая lock, ReaderWriterLockSlim и семафоры.

- Экспериментальная среда включала следующие компоненты:

- Многоядерный процессор с 16 ГБ оперативной памяти.

- Платформа .NET 6.0, библиотека System.Data.SQLite, операционные системы

Windows и Linux.

- Сгенерированная база данных объемом 10 МБ, содержащая таблицы с 1 млн. записей, имитирующими реальные рабочие нагрузки.

Для объективной оценки работы системы в различных режимах были выбраны следующие параметры:

- Среднее время, затрачиваемое на выполнение операций чтения и записи.
- Частота случаев, когда потоки ожидали освобождения ресурса.
- Соотношение успешных транзакций к количеству конфликтных операций.

Для сбора данных использовались встроенные профилировщики .NET, а также специализированные инструменты мониторинга, позволяющие проводить анализ в режиме реального времени.

Проведение эксперимента

Проведение эксперимента осуществлялось в несколько этапов:

1. Загрузка тестовых данных в SQLite и настройка режима работы базы.
2. Параллельное выполнение операций чтения и записи с использованием различных механизмов синхронизации.
3. Мониторинг времени выполнения транзакций, фиксация количества блокировок и конфликтов.
4. Сравнительный анализ полученных показателей для определения оптимальных настроек работы системы.

Особое внимание уделялось сравнению асинхронных методов работы с традиционными синхронными подходами, что позволило оценить преимущества и недостатки каждого метода в условиях высокой нагрузки.

В результате проведения экспериментов были получены количественные показатели, позволяющие оценить эффективность применения различных режимов работы SQLite и механизмов синхронизации в многопоточной обработке данных с использованием C#. Для объективной оценки были измерены следующие параметры:

- Измеряется в миллисекундах для операций чтения и записи.
- Количество случаев, когда поток ожидал освобождения ресурса.
- Соотношение числа неудачных транзакций (из-за конфликтов) к общему количеству попыток.

В ходе эксперимента тестировались следующие сценарии:

- Сценарий A использование режима Serialized с применением базового механизма синхронизации (ключевое слово lock).
- Сценарий B работа в режиме Multi-thread с использованием ReaderWriterLockSlim для разделения операций чтения и записи.
- Сценарий C применение режима Single-thread, имитирующего последовательную обработку, как контрольный сценарий.

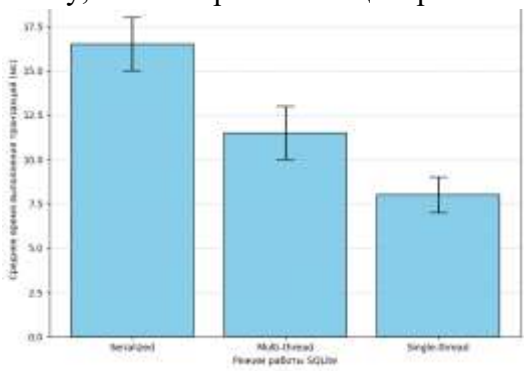


Рисунок 3. Диаграмма зависимости среднего времени выполнения транзакций от режима работы SQLite

Результаты показали, что режим Single-thread обеспечивает наилучшую производительность по времени выполнения, однако не подходит для сценариев, где требуется реальная конкурентная обработка. Режим Multi-thread в сочетании с ReaderWriterLockSlim демонстрирует оптимальный компромисс между производительностью и безопасностью, несмотря на незначительное увеличение числа конфликтных операций. Сценарий с режимом Serialized характеризуется более высоким временем выполнения и большим количеством блокировок, что объясняется строгой последовательной обработкой операций.

Анализ результатов позволил сделать следующие выводы:

- Режим Single-thread является наиболее эффективным с точки зрения времени выполнения, однако не допускает реального многопоточного доступа. В условиях, когда требуется обеспечить параллелизм, оптимальным решением является применение

Параметр	Сценарий A (Serialized + lock)	Сценарий B (Multi-thread + ReaderWriterLockSlim)	Сценарий C (Single-thread)
Время выполнения транзакций	15–18 мс	10–13 мс	7–9 мс
Частота блокировок (на 1000 операций)	120–150	80–100	0
Процент конфликтных операций	2–3 %	4–5 %	0

Таблица 2. Сравнительный анализ экспериментальных показателей

режима Multi-thread с корректной настройкой синхронизации.

- Использование специализированных средств синхронизации (например, ReaderWriterLockSlim) позволяет снизить время ожидания потоков и уменьшить вероятность конфликтов при одновременном выполнении операций.

- Режим Serialized обеспечивает максимальную безопасность данных за счет полного исключения параллельного доступа, однако за это приходится платить снижением производительности.

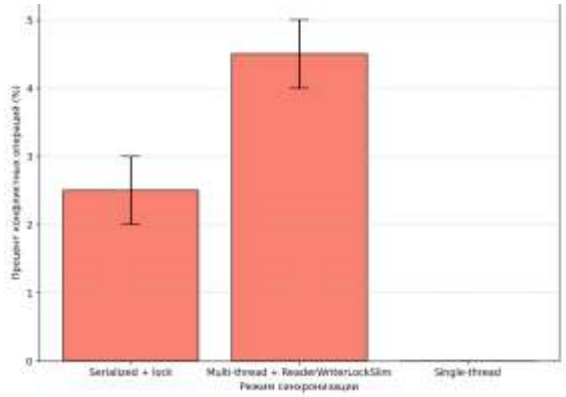


Рисунок 4. График зависимости процента конфликтных операций от режима синхронизации

Таким образом, экспериментальные данные подтверждают гипотезу о том, что выбор режима работы SQLite и механизмов синхронизации является критически важным для обеспечения эффективной многопоточной обработки данных в C#-приложениях.

На основе полученных результатов можно выделить несколько ключевых аспектов, представляющих практический интерес для разработчиков:

- Режим Serialized обеспечивает наивысший уровень защиты целостности данных. Однако, в условиях интенсивного многопоточного доступа строгие блокировки приводят к заметному снижению производительности. Использование базового механизма lock эффективно предотвращает состояния гонок, но не всегда позволяет обеспечить оптимальное время отклика системы.

- Применение режима Multi-thread позволяет достичь более высокой производительности благодаря параллельному выполнению операций. Использование ReaderWriterLockSlim в данном случае дает возможность одновременно обслуживать несколько операций чтения, что особенно важно в системах с преобладанием операций выборки. Несмотря на незначительное увеличение числа конфликтных транзакций, баланс между безопасностью и скоростью работы оказывается наиболее оптимальным.

- Режим Single-thread является теоретическим ориентиром для оценки производительности, так как обеспечивает минимальное время выполнения за счет отсутствия блокировок. Однако его применение ограничено сценариями, где отсутствует реальный параллелизм, что делает его неприменимым для реальных многопоточных приложений.

Результаты исследований

На основании проведенного исследования рекомендуется:

- Применять режим Multi-thread для приложений, требующих реальной конкурентной обработки данных.

- Использовать ReaderWriterLockSlim для разделения операций чтения и записи, что позволяет снизить время ожидания блокировок при высокой нагрузке.

- Проводить регулярный мониторинг производительности с применением встроенных профилировщиков .NET, что позволит своевременно выявлять узкие места и корректировать настройки синхронизации.

- При проектировании архитектуры приложения учитывать специфику работы SQLite и выбирать режим работы в зависимости от характера и объема обрабатываемых данных.

Следует отметить, что результаты эксперимента зависят от ряда факторов, включая:

- Конфигурацию аппаратного обеспечения, на котором проводились тестирования;
- Особенности реализации тестового приложения и используемых библиотек;
- Объем и характер тестовых данных, что может варьироваться в зависимости от конкретных сценариев применения.

Таким образом, полученные результаты можно рассматривать как ориентировочные, а дальнейшие исследования могут быть направлены на расширение выборки тестовых сценариев и адаптацию методологии под реальные условия эксплуатации приложений.

Заключение

В данной статье рассмотрены вопросы многопоточной обработки данных в SQLite с использованием C# в условиях современной .NET-разработки. Проведенный анализ архитектурных особенностей SQLite, а также сравнительный эксперимент по оценке эффективности различных режимов работы и механизмов синхронизации, позволили выявить следующие ключевые моменты.

- Для обеспечения эффективного многопоточного доступа оптимальным является применение режима Multi-thread в сочетании с продвинутыми средствами синхронизации, такими как ReaderWriterLockSlim.
- Режим Serialized гарантирует максимальную целостность данных, однако его применение может негативно сказываться на производительности при высоких нагрузках.
- Режим Single-thread, являясь теоретически оптимальным по времени выполнения, не подходит для реальных сценариев, требующих параллелизма.
- Выбор конкретной методики синхронизации должен основываться на анализе характера и объема данных, а также специфики приложения.

Данные выводы подтверждаются экспериментальными результатами, полученными в ходе исследования, и могут служить основой для дальнейших работ по оптимизации многопоточной обработки данных в приложениях на C#. Применение описанных методов позволит существенно повысить производительность и стабильность работы систем, что особенно актуально в условиях современных требований к скорости отклика и надежности программного обеспечения. В заключение отметим, что дальнейшие исследования в данной области могут быть направлены на разработку адаптивных алгоритмов синхронизации, способных динамически изменять параметры работы в зависимости от текущей нагрузки и специфики обрабатываемых данных. Это позволит обеспечить еще более высокий уровень оптимизации и устойчивости систем при реализации многопоточной обработки.

Список литературы

[1] Антипова А. А. Оптимизация производительности мобильных приложений: подходы и инструменты // Информационные системы и технологии. – 2021. – № 4. – С. 77–83. – Режим доступа: https://cyberleninka.ru/article/n/optimizatsiya-proizvoditelnosti-mobilnyh-prilozheniy (дата обращения: 12.03.2025).

[2] Беляков И. В. Подходы к обеспечению кроссплатформенности мобильных приложений: анализ и перспективы // Вестник информационных технологий. – 2022. – № 5. – С. 45–52. – Режим доступа: https://cyberleninka.ru/article/n/podhody-k-obespecheniyu-krossplatformennosti (дата обращения: 12.03.2025).

[3] Дорнауэр Б., Фельдерер М. Энергосберегающие стратегии для мобильных веб-приложений и их измерение: результаты десятилетних исследований // arXiv preprint arXiv:2304.01646 [Электронный ресурс]. – 2023. – Режим доступа: https://arxiv.org/abs/2304.01646 (дата обращения: 12.03.2025).

[4] Дубов П. В. Особенности и сравнительный анализ кроссплатформенной разработки на базе Flutter и React Native // Молодой ученый. – 2022. – № 16 (410). – С. 180–183. – Режим доступа: https://moluch.ru/archive/410/89574/ (дата обращения: 12.03.2025).

[5] Google. Flutter documentation [Электронный ресурс]. – 2024. – Режим доступа: https://docs.flutter.dev (дата обращения: 12.03.2025).

[6] Meta. React Native documentation [Электронный

ресурс]. – 2024. – Режим доступа: https://reactnative.dev (дата обращения: 12.03.2025).
[7] Омаралиева, Г. А., Умаров, Т. М. Мектептердеги видеобайкоону башкаруучу программаларды түзүүдө C# чөйрөсүнүн мүмкүнчүлүктөрү // Вестник Ошского государственного университета. – 2020. – № 1-1. – С. 179-185. – УДК 378.126(54.0). – ISSN 1694-7452. – eISSN 1694-8610.
[8] Слабунов Д. К. Кроссплатформенная разработка мобильных приложений: iOS и Android // Fortech [Электронный ресурс]. – 2023. – Режим доступа: https://fortech.dev/blog/krossplatformennaya-razrabotka-prilozhenij/ (дата обращения: 12.03.2025).
[9] Токторбаев, А. М., Токтомуратова, Ж. Э. Роль искусственного интеллекта в создании веб-сайтов [Текст] // Бюллетень науки и практики. – 2025. – Т. 11, № 1. – С. 78-83. – УДК 004.032.26. – Учред. Е.С. Овечкина. – eISSN 2414-2948.
[10] Хубер С., Лорей Т., Фельдерер М. Техники повышения энергоэффективности мобильных приложений: таксономия и систематический обзор литературы // arXiv preprint arXiv:2308.08292 [Электронный ресурс]. – 2023. – Режим доступа: https://arxiv.org/abs/2308.08292 (дата обращения: 12.03.2025).

Об авторах

Токторбаев Айбек Мамадалиевич
Кандидат физико-математических наук, доцент. Заведующий кафедрой Информационных систем и программированния института Математики, физики, техники и информационных технологий Ошского государственного университета. Имеет более 35 научных публикаций в международных и национальных журналах, а также в материалах конференций.
e-mail: ain7@list.ru
ORCID: 0009-0007-5255-6683
Scopus: 58908207000

Карабаев Самат Эшполотович
Магистр по специальности “Информационные системы и технологии”. Преподаватель кафедры Информационных систем и программированния института Математики, физики, техники и информационных технологий Ошского государственного университета. Имеет более 5 научных публикаций в международных и национальных журналах, а также в материалах конференций.
e-mail: mr.samat90@gmail.com
ORCID:0009-0006-5926-6040

Мойдунова Айгерим Суйунбаевна
Магистрант 2го курса, специальность – “Информационные системы и технологии”
e-mail: aguunabit@gmail.com
ORCID: 0009-0008-0433-2957

Абдыкадыров Султанбек Каныбекович
Магистрант 1го курса, специальность – “Информатика и вычислительная техника”
e-mail: sultanbekabdykadyrov69@gmail.com
ORCID: 0009-0001-7078-4686

About the authors

Toktorbaev Aibek Mamadalievich
Candidate of Physical and Mathematical Sciences, Associate Professor. Head of the Department of Information Systems and Programming of the Institute of Mathematics, Physics, Engineering and Information Technology of Osh State University. Has more than 35 scientific publications in international and national journals, as well as in conference proceedings.
e-mail: ain7@list.ru
ORCID:0009-0007-5255-6683
Scopus: 58908207000

Karabaev Samat Eshpolotovich
Master in specialty - "Information systems and technologies". Lecturer of the Department of Information Systems and Programming of the Institute of Mathematics, Physics, Engineering and Information Technologies of Osh State University. Has more than 5 scientific publications in international and national journals, as well as in conference proceedings.
e-mail: mr.samat90@gmail.com
ORCID: 0009-0006-5926-6040

Moidunova Aigerim Suyunbaevna
2nd year Master's student, specialty - "Information systems and technologies"
e-mail: aguunabit@gmail.com
ORCID: 0009-0008-0433-2957

Abdykadyrov Sultanbek Kanybekovich
1st year Master's student, specialty - "Informatics and computer engineering"
e-mail: sultanbekabdykadyrov69@gmail.com
ORCID:0009-0001-7078-4686

Научно-технический журнал “Проблемы инженерии и Инновации”

Адрес редакции:
150100, Ферганская област, г. Фергана, улица Солим, дом 23/4, кв. 26

E-mail:
chief_editor@epai.uz

Наш сайт:
https://epai.uz/

Учредителем научно-технического журнала
«Engineering problems and innovations» является ООО “Fer-Teach”.
Журнал издается с 2023 года, зарегистрирован 28 декабря 2022 года Агентством информации и массовых коммуникаций при Администрации Президента Республики Узбекистан (Сертификат № 055940).

ISSN: 2181-4171.

«Инженерные проблемы и инновации» является специализированным журналом в области техники и охватывает все основные направления исследований и разработок в различных областях технических наук: физике, математике, информационных технологиях. В журнале также публикуются научные стати по лингвистике, методике преподавания и истории развития технических наук.

Языки издания: Русский, английский, узбекский

При использовании материалов журнала необходимо указывать источник.
Авторы статей несут ответственность за содержание статей и за сам факт их публикации.
Редакция не всегда разделяет мнение авторов и не несет ответственности за недостоверность опубликованных данных.
Редакция журнала не несет никакой ответственности перед авторами и/или третьими лицами и организациями за возможный ущерб, причиненный публикацией статьи.

Scientific and technical journal “Problems of Engineering and Innovations”

Address:
150100, Fergana region, Fergana city, Solim street, house 23/4, apt. 26

E-mail:
chief_editor@epai.uz

Website:
https://epai.uz/

The founder of the scientific and technical journal
“Engineering problems and innovations” is Fer-Teach LLC.
The journal has been published since 2023, registered on December 28, 2022 by the Agency for Information and Mass Communications under the Administration of the President of the Republic of Uzbekistan
(Certificate No. 055940).

ISSN: 2181-4171

"Engineering Problems and Innovations" is a specialized journal in the field of engineering and covers all the main areas of research and development in various fields of technical sciences: physics, mathematics, information technology. The journal also publishes scientific articles on linguistics, teaching methods and the history of the development of technical sciences.

Languages: Russian, English, Uzbek

When using journal materials, you must indicate the source.
The authors of the articles are responsible for the content of the articles and for the very fact of their publication.
The editors do not always share the opinions of the authors and are not responsible for the unreliability of the published data.
The editorial board of the journal does not bear any responsibility to the authors and / or third parties and organizations for possible damage caused by the publication of the article.