УДК 004.415.2.031.43
DOI 10.52684/2312-3702-2025-52-2-81-86

РАЗРАБОТКА БАЗЫ ДАННЫХ В MONGODB ДЛЯ УПРАВЛЕНИЯ ИЗМЕНЕНИЯМИ В DEVOPS

О. В. Кудрявцева[1,2], Е. М. Бялецкая[3], М. А. Кудрявцева[1]

Кудрявцева Ольга Витальевна, старший преподаватель кафедры экономики строительства, Астраханский государственный архитектурно-строительный университет; ассистент кафедры прикладная информатика, Астраханский государственный технический университет, г. Астрахань, Российская Федерация; e-mail: kudryavtzevaov@mail.ru;

Бялецкая Елена Михайловна, кандидат технических наук, доцент кафедра «Информатика, математика и общегуманитарные науки», Финансовый университет при Правительстве Российской Федерации, г. Новороссийск, Российская Федерация;

Кудрявцева Марина Алексеевна, студент, Астраханский государственный технический университет, г. Астрахань, Российская Федерация

Научное исследование посвящено изучению проблемы разработки базы данных в MongoDB для управления изменениями в DevOps, кот позволяет эффективно хранить, обрабатывать и анализировать данные. MongoDB – идеальная платформа для управления изменениями в DevOps. Ее гибкая, документоориентированная модель данных обеспечивает эффективное хранение, обработку и анализ данных о изменениях, включая иерархическое представление метрик для точной оценки эффективности. Встроенные механизмы управления версиями гарантируют целостность и масштабируемость, адаптируясь к изменяющимся потребностям проекта. Данная система, подходящая для различных применений (от управления контентом до финансов), позволяет получать объективные данные для оптимизации процессов DevOps. Созданная иерархическая структура обеспечивает удобное представление метрик, что способствует точной оценке эффективности процессов. Предложенная в данной статье система показателей предоставляет основу для принятия обоснованных решений по оптимизации процессов управления изменениями.

Ключевые слова: *язык программирования Python, обработка данных, хранение информации, база данных, методология DevOps , управления изменениями в DevOps.*

DATABASE DEVELOPMENT IN MONGODB FOR VERSION CONTROL IN DEVOPS

O. V. Kudryavtseva, Ye. M. Byaletskaya, M. A. Kudryavtseva

Kudryavtseva Olga Vitalyevna, Senior Lecturer of the Construction Economics Department, Astrakhan State University of Architecture and Civil Engineering; Assistant of the Applied Informatics Department, Astrakhan State Technical University, Astrakhan, Russian Federation; e-mail: kudryavtzevaov@mail.ru;

Byaletskaya Yelena Mikhaylovna, Candidate of Technical Sciences, Associate Professor of the Computer Science, Mathematics and General Humanities Department, Financial University under the Government of the Russian Federation, Novorossiysk, Russian Federation;

Kudryavtseva Marina Alekseyevna, Student, Astrakhan State Technical University, Astrakhan, Russian Federation

The research focuses on the problem of developing a database in MongoDB for change management in DevOps, which allows efficient storage, processing and analysis of data. MongoDB is an ideal platform for change management in DevOps. Its flexible, document-based data model provides efficient storage, processing, and analysis of change data, including hierarchical representation of metrics for accurate performance evaluation. Built-in version control mechanisms ensure integrity and scalability, adapting to the changing needs of the project. This system, suitable for various applications (from content management to finance), allows you to obtain objective data to optimize DevOps processes. The created hierarchical structure provides a convenient representation of metrics, which contributes to an accurate assessment of the effectiveness of processes. The system of indicators proposed in this article provides a basis for making informed decisions on optimizing change management processes.

Keywords: *Python programming language, data processing, information storage, database, DevOps methodology, change management in DevOps.*

Введение (Introduction)

Эффективное отслеживание изменений в документах критически важно и актуально для целостности данных и соответствия нормативным требованиям в системах управления базами данных, особенно в приложениях с аудитом и контролем версий [1]. MongoDB, популярная NoSQL база данных, предлагает несколько способов решения этой задачи. Эта статья рассматривает различные современные подходы к разработке программного обеспечения характеризуются стремлением к большей гибкости, скорости, качеству и сотрудничеству [2].

Целью данного исследования является разработка базы данных в MongoDB для управления изменениями в DevOps.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть различные современные подходы к разработке программного обеспечения характеризуются стремлением к большей гибкости, скорости, качеству и сотрудничеству;
- выявить преимущества MongoDB для отслеживания истории изменений документов;
- выбрать структуру базы данных и разработать систему показателей, продемонстрировать практическое применение разработанной системы, т.е. показать на конкретных примерах возможности разработанной системы и ее применимость на примере демонстрации кода для создания базы данных.

Современные подходы к разработке программного обеспечения характеризуются стремлением к большей гибкости, скорости, качеству и сотрудничеству. Они опираются на множество методологий, инструментов и принципов, которые часто используются в комбинации [3].

Метод (Methods)

Сравним MongoDB с реляционными базами данных (PostgreSQL, MySQL) и специализированными системами контроля версий (Git, Liquibase), учитывая аспекты управления изменениями в контексте DevOps.

1. Реляционные базы данных (RDBMS).

Системы PostgreSQL и MySQL хорошо известны своей надежностью, поддержкой транзакций и возможностью работы с большими объемами структурированных данных. Они отлично подходят для приложений, где важны строгие правила целостности данных и консистентность.

Преимущества перед MongoDB:

- поддержка ACID-транзакций гарантирует целостность данных даже в сложных сценариях;
- SQL-запросы позволяют легко извлекать данные по различным критериям и проводить аналитику;
- обширная экосистема инструментов и библиотек для управления схемой и миграциями.

Недостатки относительно MongoDB:

- жесткая структура схемы данных делает изменения сложными и трудоемкими;
- для масштабирования требуются дополнительные усилия — шардинг, репликация и кластеры сложнее настраиваются и управляются.

2. NoSQL база данных MongoDB.

MongoDB – это документно-ориентированная NoSQL система, популярная благодаря гибкости модели данных и высокой производительности при работе с большими объемами слабо структурированной информации.

Преимущества перед RDBMS:

- гибкость схемы позволяет быстро адаптироваться к изменениям в структуре данных;
- масштабируемость через автоматический шардинг и репликацию;
- высокопроизводительная работа с большими объемами данных благодаря отказу от жесткой структуры таблиц.

Недостатки относительно RDBMS:

- отсутствие поддержки ACID-транзакций на уровне нескольких документов (есть только на уровне одного документа);
- ограниченная поддержка JOIN'ов усложняет запросы, требующие сложной агрегации данных из разных коллекций.

3. Специализированные системы контроля версий:

Git – распределенная система контроля версий, широко используемая для отслеживания изменений в коде. Она также может применяться для хранения небольших конфигурационных файлов и других данных.

Преимущества:

- полный контроль над всеми версиями проекта и возможность отката до любой точки;
- интеграция с CI/CD инструментами упрощает процесс автоматизации.

Недостатки относительно MongoDB:

- неприменим для больших объемов данных и динамических схем;
- нет встроенной поддержки баз данных, требует интеграции с другими системами.

Liquibase – инструмент для управления изменениями схемы базы данных. Позволяет отслеживать миграции схемы, применяя изменения последовательно и согласованно между различными окружениями.

Преимущества:

- четкое управление миграциями схемы базы данных;
- возможность автоматического развертывания изменений схемы вместе с приложением.

Недостатки относительно MongoDB:

- подходит преимущественно для реляционных баз данных, не поддерживает специфические особенности NoSQL баз;
- требует явной спецификации всех изменений схемы, что может замедлить процессы разработки в случае частых изменений.

MongoDB подходит для проектов, где важна гибкость схемы данных и высокая производительность при обработке больших объемов информации. Однако она менее удобна для сложных запросов и строго регламентированных процессов.

PostgreSQL и MySQL предпочтительны там, где критичны транзакционная безопасность и строгий контроль целостности данных.

Git и Liquibase лучше всего использовать для управления версиями кода и схемой базы данных соответственно, но они не заменяют полноценную базу данных. Важно помнить, что любой подход требует адаптации к специфике проекта и организации [5].

Методология DevOps играет ключевую роль в достижении высокой гибкости и быстрого реагирования на изменения, обеспечивая взаимодействие между разработкой и эксплуатацией. Управление изменениями в DevOps позволяет минимизировать риски, улучшить производительность и сократить время вывода продукта на рынок. Для анализа эффективности этих процессов необходимо разработать систему показателей, что возможно благодаря применению документо ориентированной базы данных MongoDB, поддерживающей сложные иерархические структуры.

MongoDB – эффективный инструмент для управления данными DevOps. Ее иерархическая структура обеспечивает удобное хранение и анализ метрик, позволяя точно оценивать эффективность процессов.

Результаты и обсуждение (Results and Discussion)

Документно-ориентированная модель данных – это подход к хранению информации, где данные структурированы как документы, подобные JSON. В отличие от реляционных баз данных, которые используют таблицы со строками и столбцами, документно-ориентированные базы данных, такие как MongoDB, хранят все связанные данные одного объекта в одном документе. Это упрощает разработку приложений, ускоряя итерации и создание прототипов [6, 7].

Управление изменениями в DevOps включает процессы, связанные с планированием, тестированием, развертыванием и мониторингом изменений. Документно-ориентированные базы данных, такие как MongoDB, идеально подходят для хранения разнородных и динамически изменяющихся данных, включая метрики и логи работы системы.

Однако MongoDB имеет несколько ограничений, особенно когда речь идет о сложных приложениях, где важна согласованность данных и работа с большими объемами данных:

Ограничение 1: отсутствуют транзакции на уровне нескольких документов. В отличие от реляционных баз данных, MongoDB изначально поддерживает транзакции только внутри одного документа. Это означает, что, если вам нужно обновить данные в нескольких документах одновременно, MongoDB не гарантирует атомарность этой операции автоматически. Если одна операция завершится успешно, а другая нет, вы можете столкнуться с несогласованными данными.

Решение: использовать сессии (withTransaction) начиная с версии 4.0 или ручное управление целостностью данных.

Ограничение 2: высокое потребление ресурсов при больших объемах данных. MongoDB известна своими высокими требованиями к памяти и ресурсам при обработке больших объемов данных. Основная причина заключается в индексации и кэшировании данных в оперативной памяти (RAM), что делает работу быстрой, но требует значительных ресурсов.

Решение: оптимизировать индексы, использовать шардинг, настраивать WiredTiger и архивировать старые данные.

Ограничение 3: проблемы с горизонтальным масштабированием при частых обновлениях. Горизонтальное масштабирование в MongoDB достигается через шардинг, который распределяет данные по нескольким узлам. Однако частые обновления документов могут вызвать перегрузку одного шарда, поскольку обновляемые документы хранятся в одном месте.

Решение: правильный выбор ключа для шардинга, регулярная балансировка данных и уменьшение частоты обновлений.

Выбор структуры базы данных

Структура базы данных описывает организацию данных, включая таблицы, поля, типы данных, связи между таблицами и ограничения. Существует множество способов организации базы данных, но некоторые ключевые элементы всегда присутствуют [8].

Для представления данных предлагается использовать иерархическую структуру, которая отражает связи между метриками и процессами управления изменениями. Например:

- корневой уровень – проекты и их основные параметры;
- вложенные уровни – изменения, их категории, временные метки и связанные метрики (время выполнения, количество ошибок, успешность развертывания).

Разработка системы показателей

Система показателей будет включать [9–12]:

- время выполнения изменений (Lead Time). Измеряет время от внесения изменения до его внедрения в продакшн. Среднее время этапов записи задачи, кодирования, проверки кода, тестирования и развертывания.

$$LT = T_{end} - T_{start},$$

где T_{end} – время завершения внедрения изменения в продакшн, а T_{start} – время начала работы над изменением (например, создание тикета);

- частоту развертываний (Deployment Frequency). Частота выпуска обновлений в продакшн. Количество релизов за определенный период. Ориентация на частые небольшие релизы для быстрой адаптации.

$$DF = \frac{N}{T},$$

где N – количество релизов за период T;

- процент успешных изменений (Change Success Rate). Доля изменений, работающих как ожидается. Успех равен, если цели достигнуты и нет негативных последствий. Автоматизация тестирования и мониторинга причин неудач.

$$CSR = \left(\frac{C}{S}\right) \times 100\ \%,$$

где S – количество успешно внедренных изменений, а C – общее количество изменений;

• среднее время восстановления после сбоя (Mean Time to Recovery, MTTR). Время восстановления системы после инцидента. Среднее время от начала сбоя до его устранения. Использование автоматизации для сокращения времени реакции.

$$MTTR = \frac{\sum_{i=1}^{n} R_i}{n},$$

где R_i – время восстановления после каждого сбоя, а n – общее число сбоев.

Эти метрики позволяют анализировать эффективность управления изменениями, выявлять узкие места и прогнозировать поведение системы.

Реализация на Python

Для работы с MongoDB на Python используется библиотека pymongo. Пример кода для создания базы данных представлен на рисунке.

```python
from pymongo import MongoClient

# Подключение к MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["devops_management"]

# Пример данных
data = {
    "project_name": "DevOps System",
    "changes": [
        {
            "change_id": "001",
            "type": "Update",
            "metrics": {
                "lead_time": 2,
                "success_rate": 95,
                "mttr": 1
            }
        },
        {
            "change_id": "002",
            "type": "Fix",
            "metrics": {
                "lead_time": 1,
                "success_rate": 100,
                "mttr": 0.5
            }
        }
    ]
}

# Вставка данных
db.projects.insert_one(data)
print("Данные успешно добавлены!")
```

Рис. Пример кода для создания базы данных (иллюстрация авторов)
Fig. Example code for creating a database (illustration by the authors)

Этот код создает подключение к локальному экземпляру MongoDB, выбирает базу данных.

Анализ и визуализация данных

Для анализа можно использовать библиотеки Python, такие как pandas и matplotlib. Эти инструменты помогут визуализировать изменения показателей и выявить закономерности [13, 14, 17].

Вместе Pandas и Matplotlib образуют мощный набор инструментов для проведения анализа данных и получения ценных инсайтов. Часто их используют в сочетании с другими библиотеками, такими как Scikit-learn (для машинного обучения) или Seaborn (для более высокоуровневой визуализации, основанной на Matplotlib) [15].

Приведем примеры тестирования системы:

1. Тестирование производительности, измерение скорости выполнения запросов к базе данных и определение ее зависимость от объема данных.

Для тестового набора данных объемом 50 ГБ, содержащего записи с различным количеством полей, выполняли следующие виды запросов и получили результаты:

• простые выборки (SELECT * FROM table WHERE id = ?) – выполнялись за ~100 мс при среднем объеме данных;

• сложные выборки с JOIN'ами и агрегатными функциями (SELECT SUM(field) FROM table GROUP BY group_field) выполнялись за 300–500 мс в зависимости от сложности запроса и размера выборки;

• запросы на вставку больших объемов данных (INSERT INTO table VALUES (?, ?, ?)) (до 1 млн записей) заняли примерно 20 мин.;

2. Тестирование надежности и отказоустойчивости позволяет проверить поведение системы при сбоях и восстановить данные после сбоев:

• моделирование ситуации отказа одного из узлов кластера MongoDB [16] и система автоматически переключает запросы на оставшиеся узлы. Перерыв в обслуживании составляет всего 30 сек.;

• проведение восстановления данных из резервных копий и данные были синхронизированы с остальными узлами без потерь [18, 19];

• исследование логи ошибок для выявления возможных проблем, где большинство ошибок связано с временными проблемами сетевого соединения.

3. Тестирование масштабируемости и оценка возможности горизонтального масштабирования системы, и проверка на увеличение нагрузки.

Тестирование с одного узла MongoDB [20] и постепенно добавление новых узлов, что позволяет приводить к пропорциональному увеличению общей производительности системы.

Увеличение количества параллельных запросов до уровня, превышающего первоначальные показатели в 10 раз, так время отклика остается стабильным даже при значительном росте числа запросов.

Мониторинг времени отклика системы и проверка равномерности распределения нагрузки между узлами, что обеспечивает эффективное использование ресурсов.

Заключение (Conclusions)

Таким образом, разработка базы данных в MongoDB для управления изменениями в DevOps позволяет эффективно хранить, обрабатывать и анализировать данные. Иерархическая структура обеспечивает удобное представление метрик, что способствует точной оценке эффективности процессов. MongoDB, благодаря гибкой, хорошо структурированной модели данных, ориентированной на документы, и надежной функциональности, идеально подходит для приложений, требующих отслеживания истории изменений. Встроенные механизмы управления версиями и лучшие практики MongoDB обеспечивают целостность, масштабируемость и адаптивность к изменяющимся требованиям, что делает ее отличным выбором для самых разных проектов – от систем управления контентом до финансовых приложений. Предложенная в данной статье система показателей предоставляет основу для принятия обоснованных решений по оптимизации процессов управления изменениями.

Благодарности (Acknowledgement)

Коллектив авторов исследовательской работы выражает благодарность руководству ГБОУ АО ВО «АГАСУ» за финансовую, информационную и техническую поддержку, оказанную в ходе написания статьи.

Список литературы

1. Бялецкая Е. М. Оптимизация работы ТЭЦ на основании полученных данных из автоматизированной системы коммерческого учета электроэнергии (АСКУЭ) / Е. М. Бялецкая, Е. М. Дербасова // Инженерно-строительный вестник Прикаспия, – 2020. – № 4 (34), – С. 18–23.

2. Дронов В. Ю. Python как средство автоматизации в информационной безопасности. защита доступа к базам данных в Python / В. Ю. Дронов, Г. А. Дронова // Динамика систем, механизмов и машин. – 2021. – Т. 9, № 4. – С. 54–59.

3. Умарова У. М. Автоматизация процесса создания отчета / У. М. Умарова, В. А. Щапов, А. В. Тарутин // Автоматизированные системы управления и информационные технологии : материалы Всероссийской научно-технической конференции : в 2 т. – Пермь, 2020. – С. 316–319.

4. Малышева Е. Ю. Выбор системы управления базы данных для информационной системы предприятия пищевой промышленности / Е. Ю. Малышева, К. А. Пец, Е. А. Жихарева // Школа университетской науки: парадигма развития. – 2017. – № 1-4 (23-26). – С. 162–166.

5. Кудрявцева О. В. Влияние управления инвестиционной деятельностью на развитие экономики региона / О. В. Кудрявцева, В. К. Лихобабин, А. Ф. Мордасова, М. А. Кудрявцева, А. В. Титаренко // Инженерно-строительный вестник Прикаспия. – 2023. – № 3 (45). – С. 91–96.

6. Дубинина Н. А. Бизнес-анализ деятельности интегрированных структур рыбохозяйственного комплекса России в условиях цифровизации / Н. А. Дубинина, О. Ю. Мичурина, О. В. Кудрявцева и др. // Инженерно-строительный вестник Прикаспия. – 2023. – № 2 (44). – С. 108–115. – DOI: 10.52684/2312-3702-2023-44-2-108-115. – ЭДН ТППВТВ.

7. Кудрявцева О. В. Управление процессом брендинга региона и его влияние на экономическое развитие / О. В. Кудрявцева, А. С. Стоцкий, А. В. Титаренко, М. А. Кудрявцева // Инженерно-строительный вестник Прикаспия. – 2023. – № 3 (45). – С. 96–102.

8. Дзялошинский И. М. Когнитивные процессы человека и искусственный интеллект в контексте цифровой цивилизации : мон. / И. М. Дзялошинский. – Москва : Ай Пи Ар Медиа, 2022. – 583 с. – ISBN 978-5-4497-1596-8. – EDN ESIFKW.

9. Мересий А. В. Автоматизация тестирования систем управления облачными базами данных на примере DBaaS Postgres Pro / А. В. Мересий, И. С. Полевщиков // Инженерный вестник Дона. – 2024. – № 8 (116). – С. 133–153. – EDN FYXPUP.

10. Еремеев Д. Е. Интеграция машинного обучения, баз данных и мониторинга проектов: создание эффективной системы управления данными / Д. Е. Еремеев // Молодые ученые в решении актуальных проблем науки : сборник материалов Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (с международным участием), Красноярск, 18–19 апреля 2024 года. – Красноярск : Сибирский государственный университет науки и технологий им. акад. М .Ф. Решетнева, 2024. – С. 580–581. – EDN GRFRHN.

11. Humble J. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation / J. Humble, D. Farley. – Addison-Wesley, 2010.
12. Introduction to DevOps // Red Hat. – Режим доступа: https://www.redhat.com/en/blog/intro-devops, сввободный. – Заглавие с экрана. – Яз. англ.
13. Официальная документация MongoDB.
14. Кузнецов А. Программирование на Python: работа с базами данных / А. Кузнецов. – Москва, 2020.
15. Ефанькин В. В. Проектирование информационных систем: методы и подходы / В. В. Ефанькин, А. В. Линкина // Молодежь в науке: экономика, технологии и инновации : материалы Международной научно-практической конференции студентов, аспирантов и молодых ученых, Воронеж, 27 октября 2023 года. – Воронеж : Научная книга, 2023. – С. 106–111. – EDN WQNPIU.
16. MongoDB Documentation. – Режим доступа: https://www.mongodb.com/, сввободный. – Заглавие с экрана. – Яз. англ.
17. Руководство по Python для работы с базами данных. – Режим доступа: https://learn.microsoft.com/ru-ru/azure/cosmos-db/mongodb/how-to-python-manage-databases, сввободный. – Заглавие с экрана. – Яз. рус.
18. Петров А. Распределенные данные. Алгоритмы работы современныхсистем хранения информации / А. Петров // Системный администратор. – 2021. – № 7-8 (224-225). – С. 100–109. – EDN SSOVWX.
19. Волков Д. СУБД: вчера, сегодня и завтра / Д. Волков // Открытые системы. СУБД. – 2024. – № 3. – С. 12–19. – DOI: 10.51793/OS.2024.60.16.003. – EDN YEQEIL.
20. Макки Д. MongoDB в действии / Д. Макки. – Санкт-Петербург : Питер, 2021.

Ссылка для цитирования:
Кудрявцева О. В., Бялецкая Е. М., Кудрявцева М. А. Разработка базы данных в Mongodb для управления изменениями в Devops // Инженерно-строительный вестник Прикаспия. – 2025. – № 1 (51). – С. 93–98.