МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»
Кафедра информационных технологий и систем

На правах рукописи

Мустафаева Мерьем Ибраимовна

МОДЕЛИ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПО ПОВЫШЕНИЮ ПРОИЗВОДИТЕЛЬНОСТИ КРИТИЧЕСКИХ БАЗ ДАННЫХ

1.2.2 Математическое моделирование, численные методы и комплексы программ

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
Доктор технических наук, доцент,
профессор кафедры
«Информационные технологии и системы»
Доронина Ю.В.

Севастополь
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. В условиях глобальной цифровой трансформации различных производственных отраслей *критические базы данных* (КБД) стали основой функционирования ключевых систем в сферах обороны, национальной безопасности, здравоохранения, финансов, энергетики, транспорта и телекоммуникаций.

Под *критическими базами данных* понимаются информационные системы, нарушение функционирования которых приводит к разрыву технологических или управленческих процессов и значительным экономическим, техногенным либо социальным последствиям. В целом, *критичность* является характеристикой, выражающей степень важности данных для поддержания функционирования ключевых операций в объектах инфраструктуры и коммуникаций, в том числе взаимодействия в информационных каналах связи, например, во взаимодействии с беспилотными летательными объектами (БПЛА) или беспилотными транспортными средствами (БТС). Для решения задачи обеспечения устойчивого информационного взаимодействия и терминального управления БПЛА или БТС требуется передавать/принимать большие объемы оперативной информации, перестраивать маршруты движения, что требует высокой производительности не только СУБД и канала связи, но и самой базы данных с точки зрения манипуляционной составляющей.

К числу актуальных вопросов обеспечения производительности КБД на различных уровнях относятся: ограничения классических реляционных архитектур при работе с большими данными, возникновение узких мест на этапах соединения таблиц и обработки сложных запросов, влияние гетерогенных ресурсов вычислительной среды и сетевой инфраструктуры; необходимость учета случайных экзогенных факторов; недетерминированность процессов функционирования КБД и объектов, которые они обеспечивают, например: БПЛА, БТС. Эти проблемы усугубляются ростом требований к скорости принятия решений, необходимости поддерживать непрерывность

сервисов и увеличения сложности пользовательских и производственных сценариев.

Для преодоления этих проблем в работе предлагается комплексный подход, включающий разработку полимодельного комплекса, включающего аналитические и имитационные модели оптимизации КБД с переменной архитектурой (с возможностью денормализации структуры КБД), интеграцию методов интеллектуального анализа, автоматическое выявление узких мест информационного взаимодействия непосредственно с объектом: БПЛА/БТС, применение адаптивных алгоритмов поддержки принятия решений и обоснование новых методов численной оптимизации в задачах управления производительностью критически важных баз данных.

Степень разработанности темы исследования. В последние десятилетия интенсивное развитие получили исследования по теории систем поддержки принятия решений, математическому моделированию, оптимизации и цифровой архитектуре критических информационных систем, к которым относятся БД. Существенный вклад в развитие теоретических и практических аспектов моделирования, в том числе БД, внесли Д. Дейт, М. Бейлисон, В.П. Иванников, С.Д. Кузнецов, А.В. Скатков, Ю.Е. Обжерин, А.А. Шалыто, Ю.В. Доронина и другие.

За рубежом вопросы оптимизации БД, проектирования потоков запросов и анализа производительности баз данных представлены в работах Э. Кодда, C. Luo, Z. Alazawi, M. Erdelj, M. Stonebraker, R. Elmasri, S. Navathe; в прикладном аспекте задачи моделирования нагрузки, автоматизации оптимизации запросов, выбора стратегий денормализации, кэширования и индексации, анализ структуры и динамики соединений (JOIN) нашли отражение в исследованиях Д.А. Флисова, А.Б. Хновского, H. Lustosa, И.В. Бельченко, С.К. Абрамова и К.В. Воронкина а также в трудах Д. Грея, Sanders G. L., Barroso L.A., Eessaar E., Brantner M., Нойманн предложил стратегию «подготовленного» упрощения сложных запросов при невозможности их полной оптимизации; в современных работах Райан Маркус, Паримарджан Неги, Хунцзы Мао, Чи Чжан, Мохаммад

Ализаде, и других обсуждаются вопросы применения методов машинного обучения при оптимизации планов запросов к БД.

Несмотря на достигнутые результаты, большинство существующих моделей ориентированы на отдельные аспекты (например, оптимизацию запросов или определённых видов индексации) и зачастую недостаточно учитывают совокупное влияние архитектурных, вычислительных и сетевых факторов, а также случайных (экзогенных) воздействий, характерных для высоконагруженных и гибридных информационных систем. Это определяет актуальность комплексного подхода, совмещающего имитационное моделирование, современные методы оптимизации и автоматизированную поддержку принятия решений для повышения производительности критических баз данных

Цель диссертационного исследования: повышение производительности критических баз данных в условиях высоких нагрузок и шумов путем применения моделей многокритериальной оптимизации.

Для достижения цели сформулированы следующие задачи:

1. Анализ существующих решений и технологий в области управления производительностью КБД.

2. Выбор критериев и обоснование применимости в методах моделирования процессов КБД для оптимизации производительности на основе моделей денормализации (структурной составляющей), принципов индексации (программной составляющей) и разработка многокритериальных моделей оптимизации запросов к КБД определенных типов (JOIN, BLOB).

3. Разработка полимодельного представления задач анализа и многокритериальной модели оптимизации запросов к КБД.

4. Разработка гибридного численного метода на основе интеграции методов машинного обучения и метода ветвей и границ, позволяющего прогнозировать время выполнения запросов относительно нагрузки на КБД и возможных шумов.

5. Разработка комплекса программ в рамках системы поддержки принятия решений по регулированию нагрузок КБД на примере взаимодействия беспилотного аппарата и диспетчерского центра (ДЦ).

Научная задача: разработка полимодельного комплекса, гибридного численного метода и системы поддержки принятия решений для оптимизации производительности критических баз данных в условиях динамических нагрузок и неопределенности, обеспечивающих эффективное функционирование систем управления беспилотными аппаратами.

Научная новизна

1. Впервые разработан полимодельный комплекс для анализа и оптимизации КБД, интегрирующий структурные (денормализация), программные (индексация) и имитационные модели в единую сценарную модель, что позволило формализовать представление данных и создать инструментальную основу для имитационного моделирования высоконагруженных КБД.
2. Получила дальнейшее развитие многокритериальная математическая модель оптимизации производительности запросов к КБД (JOIN, BLOB), в отличие от существующих основанная на иерархическом представлении времени выполнения и учете эндогенно-экзогенных факторов, что позволило реализовать сценарный подход к адаптивному выбору стратегий выполнения запросов в системах «БПЛА/БТС – диспетчерский центр».
3. Впервые предложен гибридный численный метод оптимизации, комбинирующий метод ветвей и границ для глобального поиска с нейросетевой моделью для прогнозирования времени выполнения запросов, что позволило эффективно решать многопараметрические задачи настройки КБД в условиях неопределенности и динамических нагрузок.
4. На основе разработанных моделей и методов впервые создан комплекс программ в составе системы поддержки принятия решений для

регулирования нагрузок КБД при информационном взаимодействии в системах «БПЛА/БТС – диспетчерский центр».

Теоретическая значимость работы заключается в следующем:

1. Разработаны математические постановки задач оптимизации производительности КБД, учитывающие совместное влияние эндогенных (время запросов, ресурсы, сложность схемы) и экзогенных (случайный шум) параметров процесса, что позволило создать формальный базис для анализа и синтеза высоконагруженных систем.
2. Предложена математическая модель поддержки принятия решений для повышения производительности высоконагруженных КБД, основанная на динамической оптимизации запросов, что обеспечивает теоретическую основу для адаптации систем к изменяющимся условиям нагрузки и минимизации времени обработки запросов.
3. Сформулированы принципы и алгоритмы построения комплекса программ для автоматизированного анализа метрик производительности, выявления узких мест и генерации рекомендаций по оптимизации взаимодействия с критической базой данных, расширяющие теоретический аппарат в области проектирования адаптивных систем управления КБД.

Объектом исследования процессы информационного взаимодействия в высоконагруженных системах "БПЛА/БТС – диспетчерский центр ".

Предмет исследования: математические модели, численные методы, алгоритмы и программы оценки и оптимизации производительности КБД, в условиях неопределенности и динамических нагрузок.

Методы исследования, используемые в работе: математическое моделирование, построенное с помощью формализованной задачи оптимизации производительности баз данных через функционал, описывающий производительность системы; имитационное моделирование, методы машинного обучения, моделирование бизнес-процессов, методы статистической обработки результатов, включая корреляционный анализ и

оценку временных характеристик, а также методы алгоритмизации, программирования и тестирования структурных компонентов программного комплекса.

Положения, выносимые на защиту

1. Полимодельный комплекс для анализа и оптимизации КБД, интегрирующий структурные, программные и имитационные модели, обеспечивающий формализацию представления данных и инструментальную основу для имитационного моделирования КБД в условиях динамических нагрузок [1, 3].
2. Многокритериальная математическая модель оптимизации запросов (JOIN, BLOB), основанная на иерархии времени выполнения и учёте эндогенно-экзогенных факторов, реализующая сценарный подход к адаптивному выбору стратегий выполнения запросов в системах «БПЛА/БТС – диспетчерский центр» [1, 2, 4].
3. Гибридный численный метод оптимизации производительности КБД, сочетающий метод ветвей и границ с нейросетевым прогнозированием, обеспечивающий решение многопараметрических задач настройки КБД в условиях неопределённости и динамических нагрузок [1, 7].
4. Комплекс программ в составе СППР, обеспечивающий автоматическое регулирование нагрузок КБД в реальном времени для повышения надёжности информационного взаимодействия в системах «БПЛА/БТС – диспетчерский центр» [1-8].

Практическая значимость работы заключается в разработке программного комплекса, позволяющего повысить производительность и надежность взаимодействия беспилотных аппаратов с диспетчерским центром за счет оптимизации запросов к базам данных и прогнозирования их выполнения. Реализованная система поддержки принятия решений обеспечивает эффективное регулирование нагрузок на КБД в условиях реального времени, что критически важно для устойчивой работы всей

комплексной интегрированной системы «БПЛА-ДЦ» (либо «БТС-ДЦ» при определенных допущениях).

Степень достоверности подтверждается использованием при разработке моделей известных математических методов и результатами вычислительных и натурных экспериментов.

Личный вклад соискателя. Все изложенные в диссертации результаты получены автором лично, либо при его непосредственном участии.

Внедрение результатов работы. Результаты диссертации внедрены:

- в учебный процесс кафедры «Информационные технологии и системы» ФГАОУ ВО СевГУ при проведении лекционных занятий и практических работ по дисциплине: «Гибридное моделирование» в виде «гибридного комплекса формализации бизнес-процессов и имитационного моделирования параметров критических баз данных» и «полимодельного оптимизационного комплекса решения задач анализа производительности критических баз данных в рамках новой лабораторной работы №3 и соответствующих лекций. Реализация результатов диссертационной работы позволила повысить качество учебного процесса, усовершенствовать методические материалы преподавания дисциплины для направления 09.04.01 – «Информатика и вычислительная техника» (Акт от 16.12.2025).

- в ООО "Крым Диджитал" внедрены модели и алгоритмы оптимизации информационных взаимодействий с критическими базами данных при реализации базовых функциональных задач, что позволило повысить качество ситуационного прогнозирования при выработке решений в процессах проектирования и эксплуатации сетей предприятия, повысить надежность и разрабатываемых систем; повысить производительность и надежность взаимодействия беспилотных летательных аппаратов с диспетчерским центром за счет оптимизации запросов к базам данных и прогнозирования их выполнения. Реализованная система поддержки принятия решений обеспечивает эффективное регулирование нагрузок на КБД в условиях реального времени, что критически важно для устойчивой работы всей

комплексной интегрированной системы «БПЛА-диспетчерский центр» в сельскохозяйственном мониторинге.

Апробация. Результаты диссертационного исследования докладывались и обсуждались на международных научно-практических конференциях, симпозиумах и форумах: Перспективные направления развития отечественных информационных технологий: Материалы IX Всероссийской научно-практической конференции, Севастополь, 19–23 сентября 2023 года; Перспективные направления развития отечественных информационных технологий: Материалы X Всероссийской научно-практической конференции, Севастополь, 19–23 сентября 2024 года. Перспективы развития науки и мирового сообщества: Сборник научных трудов по материалам X Международной научно-практической конференции, научно – методические и практические аспекты, Анапа, 21 июля 2025 года. Материалы XX Международной научно-практической конференции, 24 ноября 2025 года, г.-к. Анапа; а также на научных семинарах кафедры «Информационные технологии и системы», а также семинаре инновационно-образовательного центра «Центр Искусственного интеллекта СевГУ».

Публикации. По результатам диссертационного исследования опубликованы 8 публикациях, из них в 4 рецензируемых отечественных журналах, рекомендованных ВАК (квартили К2 и К3 по специальности 1.2.2 Математическое моделирование, численные методы и комплексы программ», а также 4 – в форме тезисов и материалов конференций; получено свидетельство государственной регистрации программы для ЭВМ.

Структура и объем работы. Диссертация состоит из введения, пяти глав, заключения, списка литературы и приложений. Текст диссертации содержит 160 страниц, 39 рисунков, 11 таблиц. Библиографический список включает 113 наименований.

1. АНАЛИЗ ОБЪЕКТА И ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ ПО ИССЛЕДОВАНИЮ ПРОИЗВОДИТЕЛЬНОСТИ КРИТИЧЕСКИХ БАЗ ДАННЫХ

1.1 Анализ объекта исследований

Критические реляционные БД. Информационные системы играют ключевую роль в современном мире, и их неотъемлемой частью являются базы данных (БД). Без применения БД невозможно представить ни одну из отраслей в сфере информационных технологий (ИТ). С увеличением объемов данных, хранящихся в электронном виде, вопросы управления базами данных становятся все более актуальными. Популярные инструменты, такие как Oracle, MySql, MSSQL и PostgreSQL, помогают эффективно обрабатывать и хранить данные. Время обработки запросов в системах управления базами данных (СУБД) является ключевым параметром при выборе подходящей СУБД и проектировании архитектуры любой информационной системы.

Внедрение баз данных способствует упрощению рабочих процессов во многих сферах, таких как бухгалтерия и информационные технологии. Тем не менее, на этапах разработки и эксплуатации могут возникать различные проблемы [1-3], что подчеркивает необходимость постоянного улучшения моделей поддержки принятия решений для повышения производительности критических баз данных.

Эта необходимость особенно актуальна в контексте критической информационной инфраструктуры (КИИ), представляющей совокупность информационных систем и телекоммуникационных сетей, критически важных для функционирования ключевых сфер жизни государства и общества. К таким сферам относятся здравоохранение, промышленность, связь, транспорт, энергетика, финансы и городское хозяйство. Таким образом, обеспечение надежной и производительной работы БД в рамках КИИ становится важной задачей для устойчивого развития.

Базы данных, используемые в такой инфраструктуре, играют ключевую роль в хранении, управлении и доступе к информации, необходимой для

принятия решений и обеспечения непрерывности бизнес-процессов. Однако не все базы данных обладают одинаковым уровнем значимости.

Критические базы данных - это важные компоненты информационных систем, нарушение функционирования которых приводит к разрыву технологических или управленческих процессов и значительным экономическим, техногенным либо социальным последствиям [84]. Под критичностью понимается характеристика, выражающая степень важности данных для поддержания функционирования ключевых операций в объектах инфраструктуры, в системах здравоохранения, энергетики, транспорта, финансов и телекоммуникаций. В современной научной и инженерной практике для анализа сложных процессов обработки данных широко используется гибридное моделирование, которое предполагает совмещение аналитических, функциональных и имитационных моделей, что позволяет более комплексно учитывать факторы неопределенности, вариативные нагрузки и структуру запросов.

КБД играют ключевую роль в многочисленных отраслях, где важны бесперебойная работа и минимизация ошибок. В банковском секторе финансовые учреждения полагаются на базы данных для обработки транзакций, управления клиентскими счетами и обеспечения мобильного и интернет-банкинга. Любые сбои в работе этих систем могут привести к финансовым убыткам и подорвать доверие клиентов. В сфере здравоохранения медицинские учреждения зависят от баз данных, обслуживающих электронные медицинские карты, результаты анализов и историю лечения пациентов. Доступность и корректность этих данных могут напрямую влиять на жизнь и здоровье людей.

Критические базы данных также незаменимы в авиации и транспорте, где они поддерживают системы управления полетами, бронирование билетов и логистические платформы. В этих сферах любые ошибки или задержки способны привести к серьезным авариям и катастрофам. В энергетическом секторе компании используют базы данных для мониторинга сетей, управления

поставками и учета потребления; сбои в таких системах могут вызвать перебои в энергоснабжении на региональном или даже национальном уровне.

Телекоммуникационные компании полагаются на базы данных для маршрутизации звонков, управления сетями и биллинга. Неисправности в этих системах могут привести к потере связи для миллионов пользователей. В области электронной коммерции онлайн-платформы и магазины используют базы данных для управления инвентарем, обработки заказов и платежей, и их сбои способны привести к потере продаж и ухудшению клиентского опыта. Таким образом, обеспечение надежной и эффективной работы КБД имеет критическое значение для всех перечисленных отраслей.

КБД являются основой современных высокотехнологичных систем. Их надежная и стабильная работа обеспечивает непрерывность бизнес-процессов: организации могут осуществлять свою деятельность без перебоев, что особенно важно в конкурентных отраслях. Кроме того, доступ к достоверной информации позволяет менеджерам принимать обоснованные решения в реальном времени, что означает принятие решений на основе актуальных данных. Также многие отрасли регулируются законодательством, требующим обеспечения определенного уровня безопасности и доступности данных, поэтому соблюдение нормативных требований становится возможным благодаря надежным базам данных. Быстрый и надежный доступ к информации улучшает взаимодействие с клиентами и повышает их удовлетворенность, обеспечивая высокий уровень клиентского сервиса.

Однако с ростом объемов данных, сложности инфраструктуры и возрастанием угроз информационной безопасности возникает задача эффективного управления производительностью критических баз данных. От этого зависит способность систем справляться с возрастающей нагрузкой и обеспечивать требуемый уровень сервисов.

Реляционные базы данных (РБД) занимают центральное место в современном мире информационных технологий, особенно в контексте критических систем, где надежность и точность данных имеют первостепенное

значение. Понимание того, почему именно РБД стали стандартом для таких систем, позволяет глубже оценить их значимость и определить направления для повышения производительности и эффективности [4-7].

Целостность данных также является ключевым фактором: реляционные базы данных поддерживают механизмы ограничений целостности, такие как первичные и внешние ключи, уникальность, проверки и другие, что гарантирует консистентность и достоверность данных. Широкая поддержка промышленностью дополнительно усиливает позиции реляционной модели. Большинство ведущих СУБД, таких как Oracle, Microsoft SQL Server, IBM DB2, MySQL и PostgreSQL, реализуют реляционную модель, обеспечивая надёжность и масштабируемость на уровне корпоративных приложений.

Одним из ключевых преимуществ реляционных БД является поддержка ACID-свойств транзакций, которые критически важны для систем, требующих высокой степени надежности:

1. Атомарность (Atomicity): обеспечивает, что транзакция либо выполняется полностью, либо не выполняется вовсе. Это важно для предотвращения ситуаций, когда только часть операций завершается успешно, что может привести к неконсистентному состоянию данных**.**

2. Согласованность (Consistency): гарантирует, что после завершения транзакции данные остаются в согласованном состоянии, соблюдая все заданные правила и ограничения. Это предотвращает нарушение целостности данных.

3. Изолированность (Isolation): обеспечивает изоляцию одновременных транзакций друг от друга, предотвращая конфликты и взаимное влияние. Это особенно важно в многопользовательских системах с высокой нагрузкой.

4. Долговечность (Durability): гарантирует, что результаты выполненной транзакции сохраняются в системе даже в случае сбоя или отказа оборудования. Это достигается с помощью механизмов журналирования и резервного копирования [8-12].

Значимость ACID-свойств для критических систем трудно переоценить. Например, при переводе средств между счетами необходимо гарантировать, что

суммы корректно списаны и зачислены, иначе может возникнуть дисбаланс или финансовые потери. Несмотря на все преимущества, есть ряд задач и проблем, связанных с использованием РБД в критических системах. Например, сложность настройки и администрирования, представляет значительную трудность при применении РБД в КБД. Поддержание оптимального состояния базы данных требует высокой квалификации специалистов, что ведет к повышению издержек на персонал и обучение. Одной из важнейших задач управления КБД является управление производительностью, т.к. с ростом объемов данных и увеличением сложности запросов может снижаться производительность системы, что требует постоянной оптимизации и мониторинга.

Проблема повышения производительности КБД. Современные информационные системы сталкиваются с беспрецедентным увеличением объемов данных, обусловленным распространением таких технологий, как Big Data, Интернет вещей (IoT), искусственный интеллект (AI) и машинное обучение (ML). Эти технологии генерируют огромные массивы информации, требующей эффективной обработки, хранения и анализа. Помимо объема, данные становятся более разнообразными и сложными, включая структурированные, полуструктурированные и неструктурированные данные, поступающие из различных источников, таких как сенсоры IoT, социальные сети и транзакционные системы. Традиционные реляционные базы данных не всегда способны эффективно справляться с такими нагрузками из-за ограничений архитектур и недостаточной масштабируемости. Увеличение объемов данных приводит к возрастанию времени выполнения запросов, проблемам с блокировками и конкуренцией за ресурсы, что отрицательно сказывается на общей производительности системы и способности бизнеса принимать своевременные решения.

Во многих отраслях требуется повышенная оперативность принятия решений, а задержки в доступе к данным могут приводить к потере конкурентных преимуществ, финансовым потерям, могут представлять риск для здоровья и жизни людей.

Несмотря на необходимость повышения производительности, КБД сталкиваются с рядом проблем, препятствующих эффективной работе систем. Одной из основных сложностей являются "бутылочные горлышки" – узкие места в системе, такие как недостаточная пропускная способность дисковой подсистемы или сетевой инфраструктуры, которые могут стать серьёзным препятствием для обработки больших объемов данных. Блокировки и конкурентный доступ при одновременном выполнении множества транзакций вызывают конкуренцию за одни и те же ресурсы, что приводит к задержкам и снижению производительности. Несвоевременный доступ к данным может нарушить внутренние процессы компании, замедлить цикл принятия решений и снизить эффективность работы сотрудников. В отраслях с жёстким регулированием задержки в обработке и предоставлении данных могут привести к несоответствию нормативным требованиям и, как следствие, к юридическим последствиям и штрафам [9-15].

Для преодоления проблем производительности КБД в мировой практике применяются различные подходы и технологии. Разработка новых алгоритмов обработки данных, оптимизация существующих алгоритмов или создание новых, более эффективных для специфических рабочих нагрузок являются важными направлениями. Исследование гибридных систем, сочетающих преимущества различных типов баз данных для достижения оптимальной производительности и функциональности, также привлекает внимание. Применение искусственного интеллекта, использование методов машинного обучения для автоматической оптимизации систем, прогнозирования нагрузок и обнаружения аномалий открывает новые возможности для улучшения производительности баз данных.

1.2 Классификация отраслей, где применяются технологии критических БД

Технологии критических баз данных находят широкое применение во многих отраслях современной экономики, играя ключевую роль в обеспечении надежности, эффективности и безопасности операций. Их использование

обусловлено потребностью в обработке больших объемов данных в реальном времени, обеспечении непрерывности бизнес-процессов и поддержании высокого уровня обслуживания клиентов. Рассмотрим основные отрасли, где применение критических баз данных имеет особое значение, проанализируем особенности использования, приведем примеры и оценим возможные последствия в случае проблем с производительностью или доступностью этих систем.

Промышленное производство. Промышленность является одной из ведущих отраслей, активно использующих КБД. Современные производственные предприятия стремятся к максимальной автоматизации и интеграции своих процессов, что требует надежных систем управления данными. Критические базы данных используются для управления производственными линиями, отслеживания запасов сырья и готовой продукции, контроля качества, планирования и оптимизации производственных процессов.

В системах управления производством (MES — Manufacturing Execution Systems) критические базы данных обеспечивают сбор и обработку данных в реальном времени от различных датчиков и оборудования. Это позволяет оперативно реагировать на отклонения, оптимизировать производительность и снижать затраты. Примерами могут служить автомобильные заводы, где синхронизация сборочных линий и логистических процессов невозможна без надежных баз данных.

Сельское хозяйство все активнее внедряет современные технологии для повышения эффективности и устойчивости производства. КБД используются в системах точного земледелия, управлении агротехническими операциями, мониторинге состояния посевов и животных, а также в прогнозировании урожаев. Сбор данных с различных источников, таких как спутниковые снимки, данные с беспилотных летающих аппаратов (БПЛА), метеорологические станции и датчики влажности почвы, требует надежных баз данных для хранения и обработки информации.

Проблемы с доступностью или производительностью КБД при поддержке информационного взаимодействия с БПЛА, например, при мониторинге угодий в сельском хозяйстве могут привести к несвоевременным агротехническим мероприятиям, что негативно повлияет на качество и объемы урожая. Это, в свою очередь, ведет к финансовым потерям для фермеров и может повлиять на стабильность продовольственного обеспечения регионов и даже стран.

Отметим также и другие отрасли, где информационное взаимодействие с КБД играет важную роль. Это **финансовый сектор и банковская деятельность**; **энергетическая отрасль** (включающая добычу, переработку и распределение нефти, газа и электроэнергии, опирается на КБД для управления сложными инфраструктурами и обеспечением стабильного снабжения ресурсов. Системы управления энергосетями (SCADA-системы) используют базы данных для мониторинга состояния сети, управления потоками энергии и предотвращения аварийных ситуаций. Они обрабатывают данные с тысяч датчиков и устройств в реальном времени, обеспечивая надежную работу энергетической инфраструктуры. **Транспорт и логистика.** Отрасли транспорта и логистики зависят от КБД для эффективного управления перевозками, складскими запасами и цепочками поставок. **Телекоммуникационные компании** используют КБД наиболее явно, для управления сетями связи, биллинга, обслуживания клиентов и предоставления услуг доступа в интернет. Базы данных хранят информацию о миллионах абонентов, их тарифах, использовании услуг и техническом состоянии оборудования. Проблемы с доступностью или производительностью этих систем могут привести к задержкам в предоставлении государственных услуг, нарушению общественного порядка, опасности для жизни и здоровья граждан. Это может вызвать социальное недовольство, политические последствия и подорвать доверие к государственным институтам [15-21]. **Научные учреждения и исследовательские организации** работают с большими объемами данных, требующими надежного хранения и обработки. КБД используются в проектах по

геномике, метеорологии, астрофизике и других областях, где необходимо обрабатывать и анализировать огромные объемы информации.

Беспилотные летательные аппараты как часть сложных программно-аппаратных систем информационного взаимодействия в различных отраслях. Отрасль беспилотных летательных аппаратов стремительно развивается и находит применение в различных сферах: от военной и разведывательной деятельности до коммерческих и гражданских приложений, таких как мониторинг инфраструктуры, доставки грузов и исследований окружающей среды. Критические базы данных в этой отрасли используются для управления полетами, обработки данных с бортовых сенсоров, планирования маршрутов и обеспечения безопасности полетов.

В системах управления полетами БПЛА требуется обработка больших объемов данных в реальном времени для принятия решений об изменении курса, предотвращения столкновений и реагирования на изменяющиеся условия. Надежные базы данных обеспечивают хранение информации о местоположении, телеметрии, состоянии системы и других критических параметрах. Сбой или снижение производительности КБД в контексте БПЛА может привести к серьезным последствиям, включая потерю аппарата, угрозу безопасности людей и окружающей среды, а также финансовые и юридические риски для операторов и производителей. В военных приложениях это может повлиять на выполнение оперативных задач и национальную безопасность.

Таким образом, КБД в условиях глобальной цифровизации играют важную роль при информационных взаимодействиях в ряде объектов и отраслей. В диссертации рассматриваются области задач взаимодействия беспилотных аппаратов с КБД, размещенными в диспетчерских центрах и исследуются особенности формирования оптимальных взаимодействий в каналах информационного обмена по времени (оперативность взаимодействия) и по объемам передаваемой информации.

1.3 Обзор подходов к принятию решений по повышению производительности критических БД

В настоящее время технологические достижения предоставляют определенный набор решений для повышения производительности баз данных с учётом растущих требований и объемов информации. Проблемам производительности и масштабируемости баз данных посвящено ряд исследований [12-24]

Для доступа и сканирования данных большого объема без полной выборки таблицы, используется различные методы, в т. ч. индексы, алгоритмы B – дерева, хеширование таблиц. Методы применимы при выборке по ключевому полю. Разработаны и успешно развиваются платформы для реализации контроля и качества данных. Например, при решении проблем с репликацией в производительности и масштабируемости используются несколько другие формы.

В случае возникновения проблем с обработкой запросов на чтение и запись в базу данных, можно перенаправить запросы на чтение на реплики. Это позволит повысить эффективность работы, поскольку одна операция записи может включать в себя несколько операций чтения. [24,27].

Индексы часто используются для оптимизации работы баз данных, т.к. позволяют значительно ускорить выполнение поисковых запросов и обеспечить быстрый доступ к информации. Преимуществом использования индексов является возможность их применения совместно с таблицами данных. Однако при этом возникают проблемы с расстановкой индексов таблиц. При запросах индексов таблиц РМД для конечного множества запросов могут возникнуть сложности. Важно учитывать, что чрезмерное использование индексов может привести к высоким затратам на обновление данных. При классической выборке и обработке запросов, основанных на программном обеспечении для проектирования баз данных, используются рекомендации и методики работы с индексацией БД.

Выборка при работе с индексами тесно связана с человеческим фактором, что может привести к ошибкам и отвлечению внимания. Для решения проблем с индексацией при кластеризации методов множества запросов предлагается использовать репликацию базы данных или оптимизированный сервер для выполнения конечного множества запросов. [25].

Кэширование на уровне системы управления базами данных (СУБД) и внешних приложений способствует снижению нагрузки на базу данных. Использование инструментов, таких как Redis и Memcached, позволяет хранить часто запрашиваемые данные в памяти, что ускоряет доступ и уменьшает время выполнения запросов.

Рассмотрим другие подходы по повышению производительности КБД.

Структурный подход: денормализация критической БД. Так как наиболее распространёнными и применяемыми подходами при работе с базами данных являются методы проектирования на основе РМД, рассмотрим более подробно особенности, связанные с нормализацией/денормализацией данных.

РМД обладают рядом преимуществ по сравнению с другими моделями, и за время своего существования было создано множество теоретических и практических разработок, включая основы проектирования РМД, научные исследования и программные инструменты для их применения [26,28-29].

Известно, что более 80% всех сервисов ориентированы на работу с базами данных. Это происходит так как базы данных позволяют приложениям удовлетворять все требования к хранению, обработке и отображению данных [8] сразу, несмотря на известные проблемы, именно реляционная модель данных (РМД) остается наиболее часто используемой моделью в базах данных.

Архитектурные особенности РМД, такие как: требования наличия специальных групп, которые играют ключевую роль в уточнении схемы (функциональные зависимости(ФЗ)) [30-34,38]; теоремы, поддерживающие устойчивость и покрытие ФЗ; нормализация – все это привносит определенный уровень сложности в РМД, что, в свою очередь, влечет проблемы повышения эффективности. Следовательно, нормализация имеет большое практическое

значение [42-47]. Методы проектирования на современном этапе развития состоят из следующих этапов: требования, концептуальное проектирование, логическое проектирование, физическое проектирование [35-36, 39-41]. Аномалии обновления в таблице возникают из-за определённых зависимостей между столбцами одной таблицы [48-56].

Нормализация реляционной схемы с учетом набора НФ приводит к реляционной схеме, которая свободна от аномалий, порожденных избыточностью, но все же может страдать от возможных проблем, связанные с производительностью. Для решения такого рода проблемы, есть процесс, обратный нормализации, а именно денормализация Под термином «денормализация данных» подразумевается изменение структуры таблиц данных, ввод намеренно избыточной информации с целью улучшить производительность БД [58-61].

С другой стороны, дублирование данных в системе создаёт дополнительный уровень сложности, поскольку необходимо контролировать избыточность данных. В случае управляемого резервирования СУБД пользователи должны заботиться о распространении обновлений, чтобы избежать несогласованных или противоречивых данных в разных частях базы данных. [61-69, 71]

Производительность КБД (и БД в целом) можно повысить за счёт сокращения времени запросов к базе данных, которые имеют типы связей и соединений. В ходе экспериментальных изменений БД с помощью дисбалансировки табличных данных не всегда удаётся успешно завершить исследования, поскольку производительность данных не всегда увеличивается. Бывают ситуации, когда операции соединения, выполненные посредством денормализации, не приводят к уменьшению затраченного времени.

Существуют программные инструменты для анализа эффективности запросов к КБД. К таким инструментам можно отнести Query Analyzer. С помощью Query Analyzer можно проанализировать генерацию запросов, созданных на MS SQL Server. Однако не существует программных средств,

которые могли бы отслеживать выполнение денормализации и проводить анализ в структурированных отношениях.

В существующих исследованиях, в том числе [70], предложена имитационная модель для решения экспериментального подхода с использованием методов денормализации, которая использует различные принципы денормализации. Например, в работе [10] структура отношений БД построена следующим образом, родительское отношение T1 имеет начальный размер кортежа, совпадающий с размером кортежа исследуемого отношения. Атрибутами отношения T1 являются: ключевой атрибут id; набор неключевых атрибутов $nonkey_{t1\_1}$, $nonkey_{t1\_NT1}$ строкового типа фиксированного размера, что позволяет обеспечить необходимый размер кортежа отношения, рис.1.1.

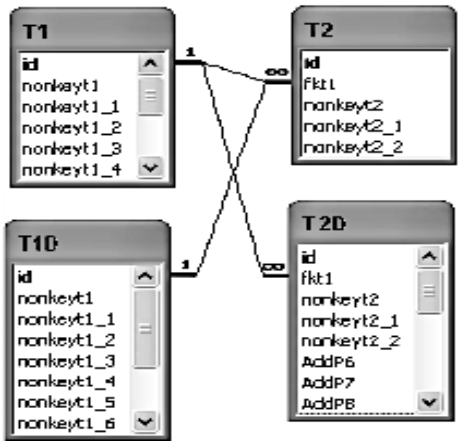


Рис. 1.1 Экспериментальные исследования на базе структурных отношений [10]

В ходе выполнения экспериментальных задач следует сохранять сеанс эксперимента, включающий цикл прохождения количества кортежей и времени запросов. Цикл по количеству кортежей в родительском отношении: i = C_{1_Start} to C_{1_End} step k1. Общая схема имитационной модели для экспериментальных исследований денормализации изображен на рис.1.2.

Анализ результатов экспериментальных исследований влияния денормализации, предложенной в работах [71,73,75], показал весьма значимое влияние различных методов. В результате теоретических и экспериментальных исследований, в тмо числе, предложенных в работе [13], можно получить

информацию о целесообразности структурных отношений БД с помощью денормализации.

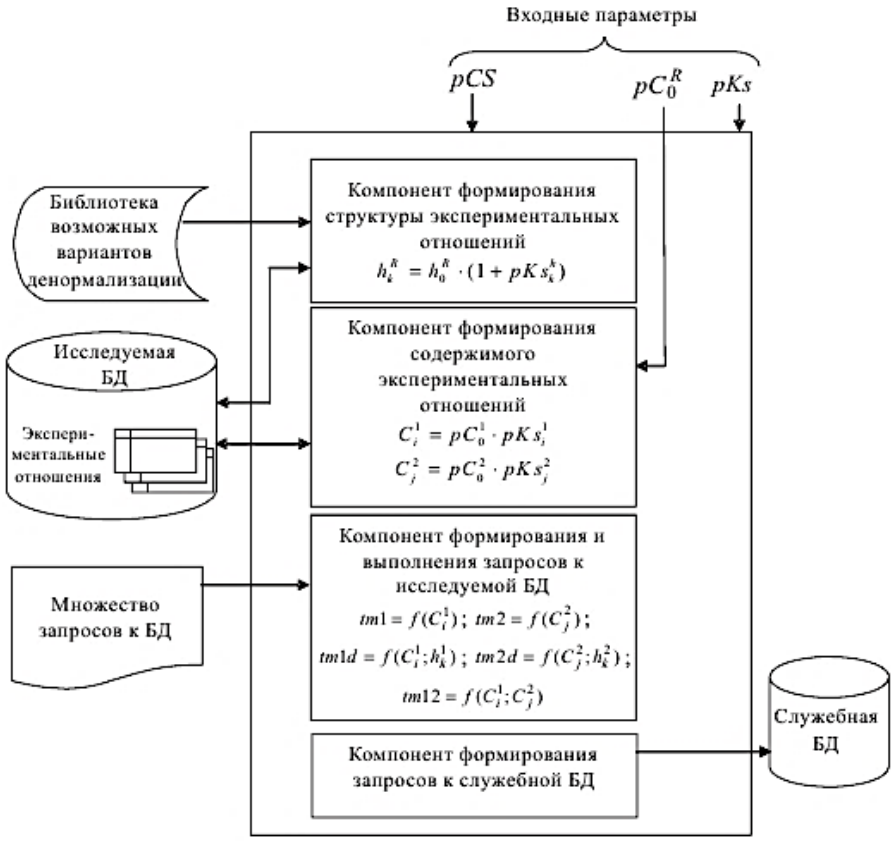


Рис. 1.2 Общая схема имитационной модели [10]

При выгрузке данных с использованием оператора выборки необходимо объединять (операция join) различные отношения. При большом количестве соединений время выборки значительно увеличивается, и она выполняется медленно. Метод уменьшения времени выполнения запросов и добавления избыточной информации называется денормализацией. В экспериментальных исследованиях [5] используются методы денормализации для повышения производительности системы.

В ходе разработки оценивается эффективность применения денормализации к одному из отношений в запросе [5, 79, 81]. В ходе исследования выполняются 5 отношений и 4 соединения. Исследования проводились путём добавления атрибута родительского отношения к дочернему отношению с помощью операции соединения.

По результатам работы [5] практические исследования проводились на основе кортежей в дочернем элементе количеством: 2000, 4000, 6000 единиц. В ходе исследования [5] средний результат на основе 10 опытов в среде SQL developer представлен на рисунке 1.3.

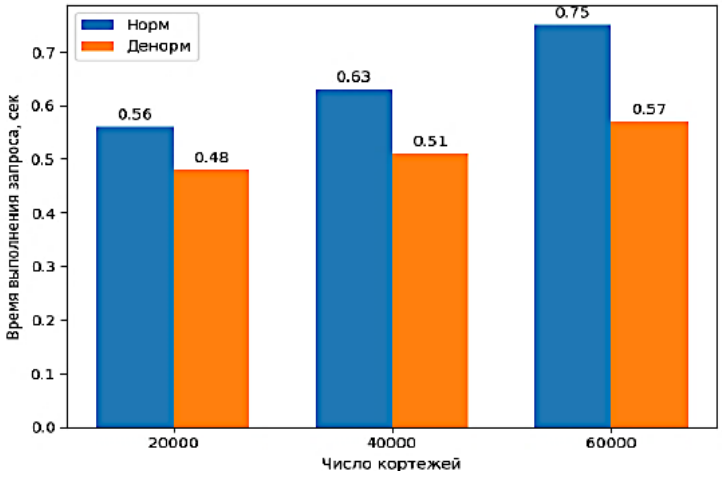


Рис. 1.3 График соотношения времени выполнения запроса к числу кортежей [5]

В работах [76, 81, 85] рассматривается характерная особенность процесса денормализации — нисходящая денормализация. Этот метод предполагает добавление в дочернее отношение неключевого атрибута из родительского отношения. Целью данной процедуры является улучшение процесса соединения данных и сокращение времени выполнения запросов.

Преимущество нисходящей денормализации заключается в возможности оценить выгоду и потери. Для этого необходимо определить, насколько изменится база данных. Структуру нормализованной базы данных можно использовать с нормальной формой Бойса – Кодда, представляя реляционную базу данных в виде кортежа [86, 87, 89].

На рисунке 1.4 представлено дерево критериев нисходящей денормализации при выполнении операций соединения с методом вложенных циклов загрузки блоков данных [90, 92].

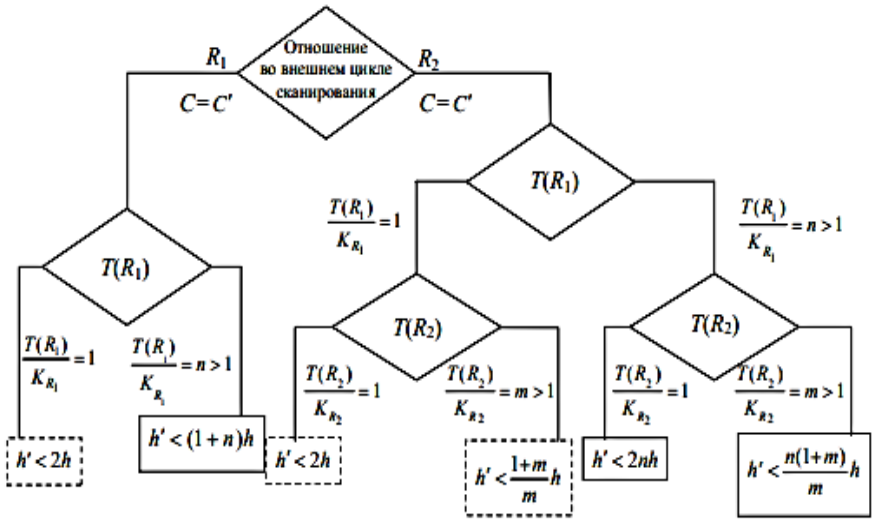


Рис. 1.4 Дерево эффективности критериев денормализации

В ходе исследования в работах [95, 96] были проведены эксперименты, направленные на повышение эффективности обработки данных, было установлено, что улучшение производительности путём нисходящей нормализации является эффективным. Этот подход позволил сократить время выполнения запросов на 55%. В работе [95] представлены графики, иллюстрирующие динамику выполнения запросов в нормализованной и денормализованной базах данных.

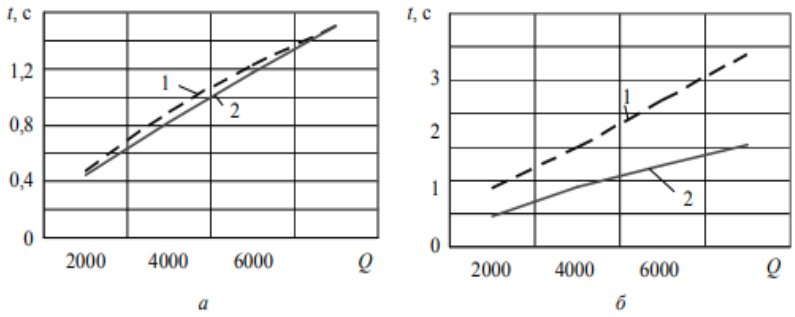


Рис. 1.5 Графики изменения времени выполнения запросов нормализованной и денормализованной БД

В статье [99] рассматриваются задачи, возникающие при проектировании современных систем управления с использованием ЭВМ. Погрешности, возникающие из-за ограниченности разрядной сетки ЭВМ, особенно значительны при статистической обработке данных, включающей в себя суммирование большого количества чисел, что приводит к накоплению ошибок

денормализации. В большинстве случаев, когда суммируемые числа неизвестны, эту погрешность сложно отследить и учесть.

Анализируя результаты различных исследований, необходимо отметить, что методы, такие как денормализация, кэширование и другие оптимизации, действительно способствуют улучшению производительности БД. Однако в контексте критических систем, где требования к надежности и скорости выполнения операций чрезвычайно высоки, требуется более сложный и комплексный подход.

Таким образом, для обеспечения высокой эффективности работы с критическими базами данных необходимо применять не только локальные методы оптимизации, но и интегрировать их в общую архитектуру системы. Это позволит обеспечить устойчивость и адаптивность системы к изменяющимся условиям и нагрузкам.

Программный подход: оптимизация запросов к КБД. Оптимизация запросов – это один из важных этапов их обработки, от которого зависит общая производительность критических БД, однако, представляет собой сложную исследовательскую задачу [14-17]. Положительным результатом применения оптимизированных запросов является уменьшение времени их выполнения, и, как следствие – повышение производительности БД в целом.

С учетом того, что СУБД выполняет множество разнообразных функций, то для оценки ее производительности возможно использовать различные показатели: время выполнения запросов, время поиска информации, время выполнения операций импортирования данных из файлов других форматов, максимальное число параллельных обращений к данным в многопользовательском режиме, время генерации отчетов и другие [17].

Наиболее часто для анализа производительности БД в целом и высоконагруженных БД в частности, применяется моделирование нагрузки на основе «тяжелых» запросов, в частности join. Как известно, существует несколько типов реализации запросов join, что имеет довольно значимое влияние на время их выполнения [100, 101].

Одним из ключевых направлений оптимизации является анализ и улучшение существующих SQL-запросов. Согласно исследованиям [102,104], неэффективные запросы могут потреблять значительные ресурсы системы, приводя к замедлению работы приложения в целом. Детальный анализ плана выполнения запросов позволяет выявить узкие места, такие как избыточные полные сканирования таблиц или неоптимальные соединения, и внести необходимые коррективы для их устранения.

Оптимизация доступа к данным посредством правильного использования индексов также играет значительную роль. Индексы позволяют ускорить выборку данных, особенно в больших таблицах, за счет уменьшения объема сканируемых данных [105]. Правильное индексирование столбцов, используемых в условиях фильтрации и объединения, может привести к существенному снижению времени выполнения запросов.

Использование хранимых процедур и функций способствует снижению сетевого взаимодействия между приложением и базой данных. Выполнение логики на стороне сервера базы данных позволяет уменьшить объем передаваемых данных и число сетевых запросов, что в итоге повышает производительность системы. Хранимые процедуры обеспечивают централизованное управление бизнес-логикой и облегчают внесение изменений без необходимости модификации клиентской части приложения.

Внедрение асинхронной обработки запросов является современным подходом к повышению эффективности работы приложений. Использование асинхронности позволяет приложениям продолжать выполнение других задач, не дожидаясь завершения длительных операций с базой данных, что улучшает отзывчивость и масштабируемость системы [106].

При использовании ORM (Object-Relational Mapping) инструментов важно внимательно контролировать генерируемые ими SQL-запросы. Неэффективное использование ORM может привести к созданию неоптимальных запросов, отрицательно влияющих на производительность. Разработчикам следует понимать внутренние механизмы ORM и, при необходимости, оптимизировать

сгенерированные запросы или использовать нативные SQL-запросы для критически важных операций.

Аппаратный подход к повышению производительности КБД является одним из фундаментальных направлений в оптимизации информационных систем. Одним из важных аспектов аппаратного подхода является масштабирование аппаратных ресурсов. Вертикальное масштабирование, подразумевающее увеличение мощности отдельных серверов (например, путем добавления оперативной памяти или перехода на более быстрые процессоры), может существенно повысить производительность СУБД при умеренных нагрузках [66]. Однако данный подход имеет свои ограничения и не всегда способен справиться с постоянно растущими требованиями к производительности.

Горизонтальное масштабирование, основанное на распределении нагрузки между несколькими серверами, представляет собой более эффективное решение для высоконагруженных систем. Использование кластерных решений и систем распределенной обработки данных позволяет достичь высокой степени параллелизма и обеспечить устойчивость к отказам отдельных узлов [66]. При этом важно учитывать особенности консистентности данных и механизмы синхронизации между узлами кластера.

Использование RAID-массивов для хранения данных обеспечивает защиту от потери информации при выходе из строя отдельных дисков, а дублирование сетевых интерфейсов снижает риск потери связи с базой данных.

Дублирование технических средств, в частности организации горячего резервирования (так называемые «hot standby»/горячие резервы), позволяет мгновенно переключаться на резервный сервер в случае отказа основного. Многие работы [8, 67], подтверждают эффективность таких подходов в обеспечении непрерывности бизнес-процессов, повышения надежности и производительности БД. Кроме изменения параметров самих БД, важным аспектом аппаратного подхода является оптимизация сетевой инфраструктуры.

В высокопроизводительных системах задержки при передаче данных могут существенно влиять на общую производительность [68, 69].

Вместе с тем очевидно, что выбор аппаратного решения должен основываться на тщательном анализе требований системы и характеристик рабочих нагрузок. Так как чрезмерное увеличение аппаратных ресурсов без соответствующей оптимизации программной части может не привести к ожидаемому росту производительности, а лишь увеличить затраты на эксплуатацию системы. Важным компонентом аппаратного подхода является мониторинг и управление аппаратными ресурсами. Современные системы мониторинга позволяют в режиме реального времени отслеживать состояние оборудования, прогнозировать возможные отказы и своевременно принимать меры по их предотвращению. Проактивное управление аппаратными ресурсами способствует повышению общей надежности системы и снижению операционных расходов [107].

1.3 Системы поддержки принятия решений по повышению производительности КБД

Системы поддержки принятия решений (СППР) представляют собой класс информационных систем, предназначенных для оказания содействия руководителям и специалистам в процессе принятия сложных управленческих решений. Они обеспечивают сбор, хранение, обработку и анализ данных, необходимых для оценки ситуаций и выбора оптимальных действий. В контексте управления производительностью критических баз данных СППР играют важную роль, предоставляя инструменты для мониторинга, анализа и оптимизации работы СУБД.

Особенностью СППР для КБД является необходимость обработки больших объемов информации в режиме реального времени, а также обеспечение высокой надежности и отказоустойчивости. Критические базы данных часто используются в областях, где сбои недопустимы, таких как банковская сфера, энергетика, медицина и транспорт. В этих условиях СППР

должны обеспечивать непрерывный контроль за состоянием систем, своевременное выявление проблем и поддержку в выборе наилучших решений для повышения производительности и предотвращения сбоев.

Применение СППР позволяет значительно повысить эффективность управления ИТ-инфраструктурой за счет автоматизации процессов сбора и анализа информации, как, например, предлагается в [73, 75].

Одним из важных аспектов построения СППР для КБД является модули прогнозирования, основанные на методах искусственного интеллекта и машинного обучения. Такие модули способны предсказывать возможные проблемы и предлагать меры по их предотвращению. Прогнозирующие модели в СППР являются важным для своевременного принятия корректирующих мер и обеспечения непрерывной работы систем [45].

Еще одной особенностью СППР в контексте критических баз данных является поддержка многокритериального принятия решений. Необходимо учитывать различные факторы, такие как производительность, стоимость, риск отказа и требования безопасности. Интеграция методов многокритериального анализа в СППР позволяет принимать более взвешенные и обоснованные решения в сложных ситуациях [45].

Архитектура СППР для управления производительностью критических баз данных обычно включает в себя несколько основных компонентов:

1. Подсистема сбора данных: отвечает за сбор оперативной информации из различных источников, включая СУБД, серверы приложений и сетевое оборудование.

2. Хранилище данных: обеспечивает централизованное хранение собранной информации для последующего анализа.

3. Аналитическая подсистема: использует методы обработки больших данных и аналитики для выявления закономерностей и генерации рекомендаций.

4. Интерфейс пользователя: предоставляет удобные средства визуализации данных и взаимодействия с системой.

Важно отметить, что эффективность СППР во многом зависит от качества исходных данных и используемых моделей анализа. В этой связи особое внимание уделяется интеграции СППР с системами мониторинга и управления СУБД, а также обеспечению актуальности и достоверности информации.

Применение облачных сервисов в СППР способствует снижению затрат на инфраструктуру и повышению гибкости системы. Кроме того, СППР могут включать средства автоматизированного принятия решений на основе предопределенных правил и алгоритмов. Это особенно актуально в ситуациях, требующих немедленной реакции, где человеческий фактор может привести к задержкам или ошибкам. Однако полностью исключать участие человека в процессе принятия решений нецелесообразно, поскольку экспертное мнение и опыт играют важную роль в оценке сложных ситуаций.

1.4 Существующие решения по исследованию производительности КБД

Исследование производительности КБД является важным направлением в области информационных технологий, поскольку от эффективности работы баз данных зависит общее быстродействие информационных систем и качество предоставляемых услуг. Существует множество решений, направленных на анализ, мониторинг и оптимизацию производительности баз данных, включающих в себя как программные инструменты, так и методики, разработанные в ходе научных исследований.

Одним из ключевых подходов к исследованию производительности баз данных является использование встроенных средств мониторинга, предоставляемых системами управления базами данных (СУБД). Например, Oracle предлагает инструмент Oracle Automatic Workload Repository (AWR), который собирает статистику выполнения запросов и помогает выявлять узкие места в производительности системы [20]. Аналогично, Microsoft SQL Server предоставляет Dynamic Management Views (DMVs), позволяющие администраторам получать информацию о текущем состоянии сервера и баз данных, что облегчает диагностику проблем [25].

Кроме встроенных инструментов, существуют специализированные программные продукты для мониторинга и анализа производительности баз данных. IBM InfoSphere Optim Performance Manager является примером коммерческого решения, предоставляющего возможности для мониторинга производительности, анализа тенденций и прогнозирования потенциальных проблем [78]. SolarWinds Database Performance Analyzer также широко используется для глубокой диагностики проблем производительности, предлагая интуитивный интерфейс и подробные отчеты [80].

В сфере программного обеспечения с открытым исходным кодом существует ряд инструментов, которые позволяют исследовать производительность баз данных. Percona Monitoring and Management (PMM) является комплексным решением для мониторинга баз данных MySQL, PostgreSQL и MongoDB, предоставляя метрики производительности и рекомендации по оптимизации; pgBadger – это анализатор логов PostgreSQL, который помогает выявлять медленные запросы и создавать детальные отчеты по производительности.

Облачные платформы, такие как Amazon RDS и Google Cloud Spanner, предоставляют автоматическое масштабирование ресурсов в зависимости от текущей загрузки системы и расширенные средства мониторинга, что упрощает диагностику и оптимизацию баз данных. Это обеспечивает надёжную и эффективную работу даже в условиях переменной нагрузки.

Другой важный аспект исследования производительности – это моделирование и предсказание поведения систем при различных нагрузках [82, 83]. Имитационное моделирование баз данных может позволить прогнозировать производительность системы при изменении параметров конфигурации и аппаратного обеспечения. Такой подход полезен для планирования развития инфраструктуры и принятия решений о масштабировании систем.

В последние годы развивается направление использования методов машинного обучения для оптимизации производительности баз данных. Разрабатываются модели, обученные на исторических данных о выполнении

запросов, которые способны предсказывать время исполнения новых запросов и рекомендовать оптимальные стратегии их выполнения.

Существующие решения по исследованию производительности КБД предоставляют широкий набор инструментов и методик для мониторинга, анализа и оптимизации систем. Встроенные средства СУБД, специализированные программные продукты и научные разработки позволяют администраторам и исследователям эффективно выявлять узкие места, прогнозировать поведение систем и принимать обоснованные решения по улучшению производительности. Однако динамичность развития информационных технологий и постоянно растущие требования пользователей делают необходимым дальнейшее развитие и совершенствование этих решений, интегрируя новые подходы и технологии.

1.5 Выбор направлений практического применения КБД

Повышение производительности КБД в условиях высоких нагрузок и шумов целесообразно в различных отраслях. На рис.1.6 приведены некоторые актуальные области применения КБД с классификацией рассматриваемых в диссертации задач.

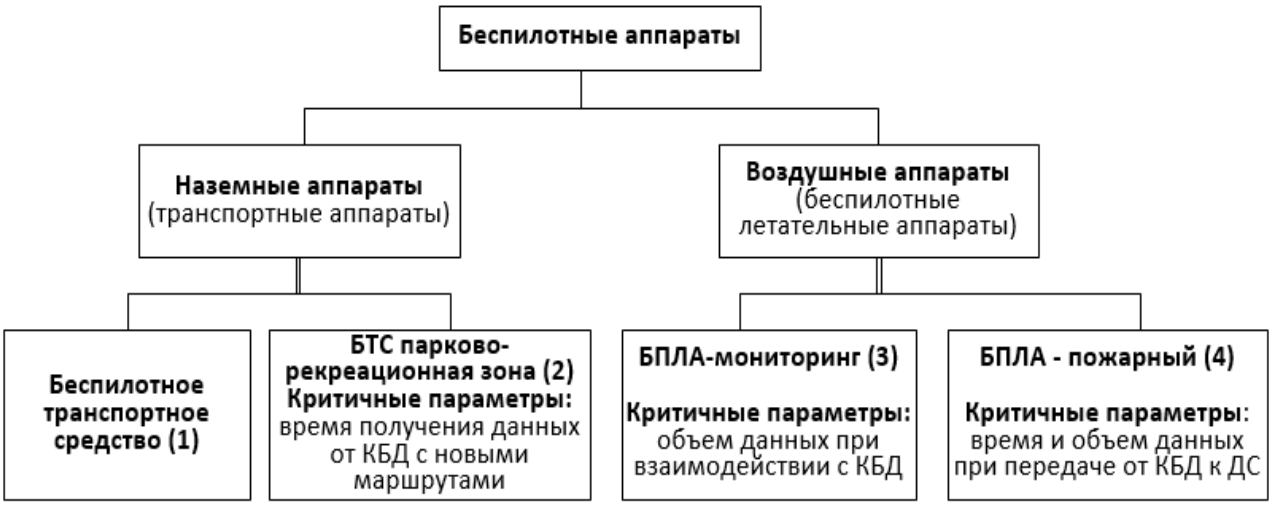


Рис. 1.6 Схема задач исследования в области беспилотных аппаратов

Приведенная на рис.1.6. схема представляет собой классификацию задач исследования на примере применения в беспилотных технологиях: для БПЛА,

где параметрами оптимизации являются является время и объем передаваемых данных (для БПЛА оперативного назначения, например, пожарных); для БПЛА в мониторинге, где оптимизируется объём передаваемых данных при взаимодействии с КБД и на примере БТС, где важна оперативность реакции КБД.

Существующие исследования, касающиеся применения беспилотных транспортных средств на таких объектах, используют следующую терминологию. Беспилотное наземное транспортное средство (БНТС) представляет собой транспортное средство, которое работает при контакте с землей и без бортового присутствия человека.

Термин беспилотное дорожное транспортное средство (БДТС), определяет беспилотное транспортное средство – механическое транспортное средство, оборудованное системой автоматического управления, которое может передвигаться без участия водителя. Поскольку термины БНТС, БДТС имеют широкую трактовку, а также в связи с тем, что, например, беспилотные летательные средства имеют свой устоявшийся термин – беспилотные летательные аппараты (БЛА) вместо терминов (в том числе более узких БНТС, БДТС) обычно применяется упрощенный термин – беспилотное транспортное средство (БТС), а в случае летательного аппарата – БЛА рис. 1.7 [19].

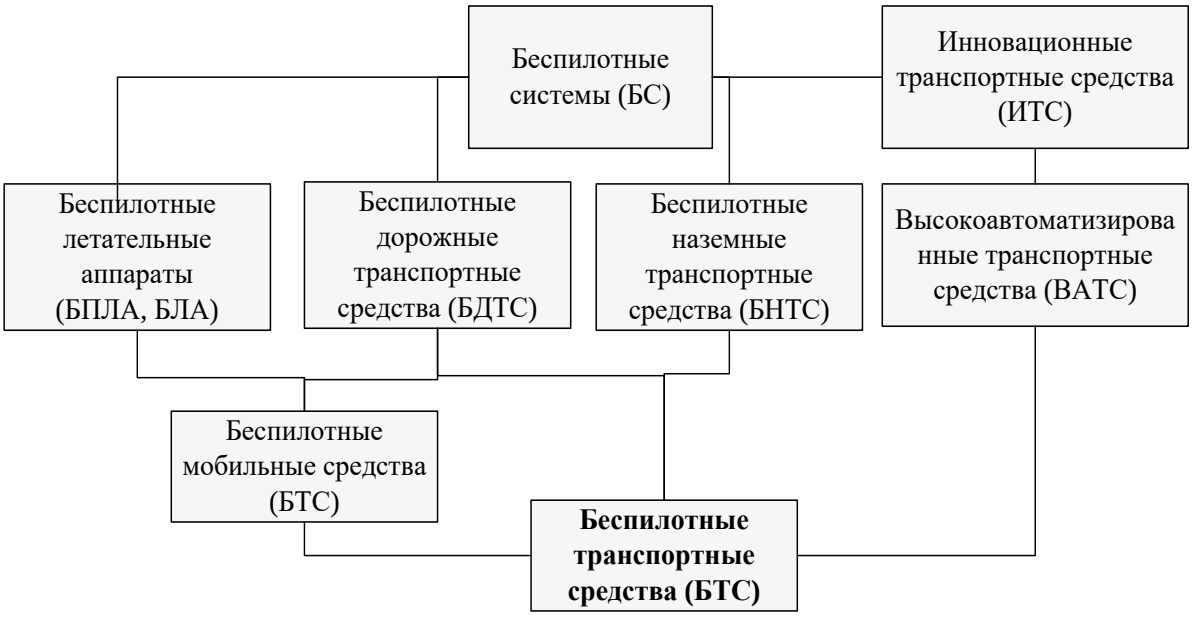


Рис. 1.7 Классификация определений беспилотных средств [3]

Готовые решения, например, уже используемые в Иннополисе (республика Татарстан, РФ) беспилотные такси, начали перевозить пассажиров с 2018 года, рис. 1.8. Автомобили забирают пассажиров в специальных местах – точках посадки [19].

Рис. 1.8 Беспилотное такси в Иннополисе [6]

В комплексном решении гибридной системы: легкий БЛА и малый БТС важным компонентом является компьютерная система обеспечения консолидации и поддержания информационного взаимодействия этих средств. Поэтому актуальной является задача построения информационной системы поддержки транспортного обслуживания (ИСПТО) парково – рекреационных зон с применением беспилотных технологий, а также решение задачи оптимизации производительности базы данных в составе ИСПТО.

В качестве примера в диссертации рассматривается задача обеспечения безопасности и производительности системы поддержки транспортного обслуживания парково-рекреационных зон (ПРЗ) с применением беспилотных технологий. Для городских ПРЗ большое значение имеет функциональная и композиционная взаимосвязь с городским окружением [19].

Проблемы транспортного обслуживания локальных городских зон (парково-рекреационных зон, крупных образовательных объектов, распределенных производственных строений, аэропортов и т.п.) обусловлены следующими факторами:

– географическая однородность, но пространственная протяженность объекта транспортного обслуживания;

– разнотипные задачи: собственно, транспортное обслуживание (например, перевозка пассажиров в парковой зоне) и обеспечивающие процессы

(например, контроль с воздуха процесса перевозки пассажиров; доставка медикаментов, мониторинг дорог и т.п.);

– требования обеспечения безопасности жизнедеятельности на ОТО;

– отсутствие возможности обеспечить нерегулярное пилотное транспортное обслуживание в связи с высокой стоимостью и другие [1-2].

Задача по типу и требованию к взаимодействию БПЛА- приемник. Современные беспилотные летательные аппараты применяются для решения задач различных типов, одним из которых является сбор данных мониторинга объектов или их параметров в форме изображений, видео фиксации объектов или явлений, геолокационных данных этих объектов и т. п. Эти данные имеют огромный потенциал для применения в разведке, наблюдения за окружающей средой, поставки и транспортировки товаров, медицинских исследований. Высокоточная обработка и анализ высоконагруженных данных представляют серьезные вызовы, особенно когда речь идет о реальном времени и высокой степени точности [19].

Сбор мониторинговых данных может быть осуществлен как дискретно, так и непрерывно, потоком. В первом случае на входе приемника данных собираются: цифровая информация, изображения, относящиеся к конкретному моменту времени существования конкретного объекта, описывающие одну или несколько характеристик этого объекта или явления. Например, изображение сельскохозяйственной площади с привязкой к координатам и параметрами интенсивности цветовой насыщенности, что может быть в дальнейшем использовано при принятии решения о качестве процесса роста сельскохозяйственной агрокультуры [14]. Во втором случае данные от БПЛА могут быть потоковыми, в виде видеоряда, что наиболее часто применяется, например, для оценки динамики существования объекта или явления, при слежении, мониторинге и оперативных решений, причем здесь важна идентификация объекта мониторинга, который может быть статическим или динамическим.

Беспилотные аппараты легко выполняют мониторинг больших территорий, обеспечивая оперативное обнаружение начальных признаков пожара. Автоматизированный мониторинг БПЛА выявляет пожар на ранних стадиях, позволяя быстро реагировать и принимать решения, предотвращать распространение. БПЛА используются при эвакуациях, отслеживают эвакуацию людей и животных, обеспечивая безопасность жизни. Важные данные могут быть переданы службам экстренной помощи для планирования и координирования действий. БПЛА с технологией искусственного интеллекта принимают решения на основе данных, строит оптимальные маршруты тушения огня или определение критических участков тушения огня при срочном вмешательстве. Также БПЛА эффективен при борьбе с пожарами, а также в минимизации рисков для людей, работающих в условиях чрезвычайной ситуации. На рис.1.9 приведена классификационная схема, отражающая типы и требования к взаимодействию БПЛА- приемник.

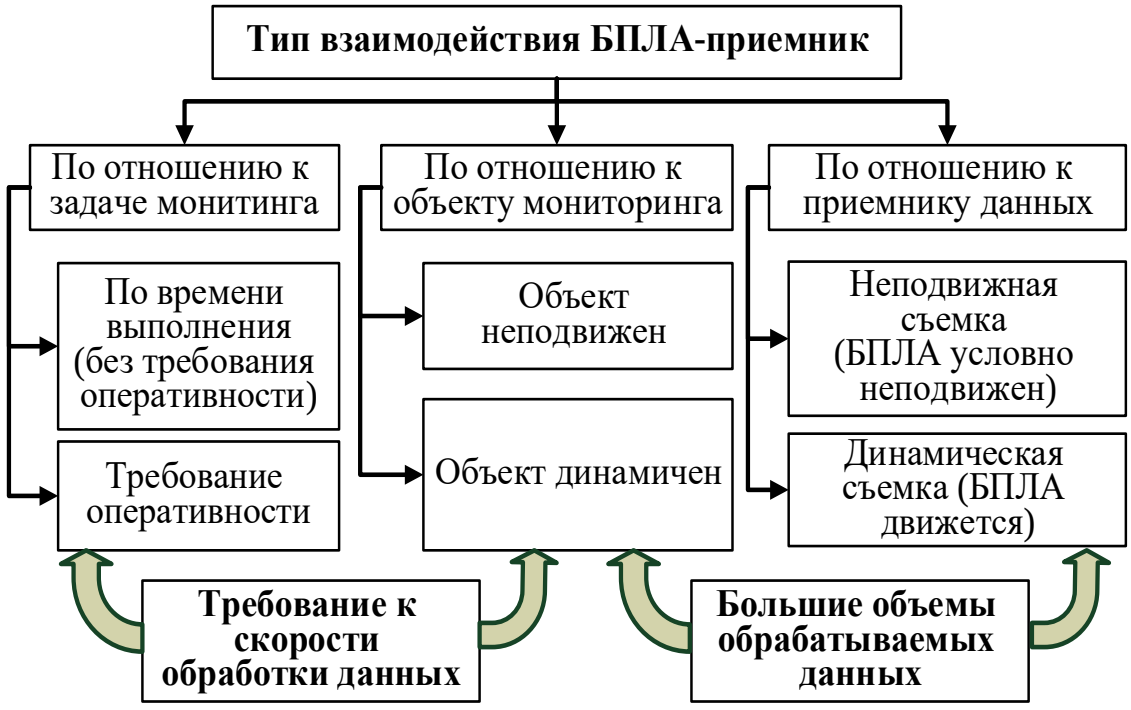


Рис. 1.9 Типы и требования к взаимодействию БПЛА- приемник

В приведенных на рис. 1.9 требованиях к взаимодействию БПЛА-приемник операции БПЛА требует обработки данных и систематическое сохранение полученных сведений в базе данных (БД) посредством взаимодействия с оператором диспетчерского центра. Кроме того, при таком

взаимодействии требуется интенсивный перерасчет координат для оптимизации движения БПЛА и требуется взаимодействие нескольких подсистем: диспетчера, системы принятия решений, подсистем технического зрения, стратегий выполнения конкретных операций мониторинга, подсистемы навигации и других. При работе БПЛА генерируется большое количество данных, включая координаты полета, фотографии и видео, которые передаются в хранилище данных. Обработка этих данных позволяет сформировать вторичные данные, которые применимы для принятия решений по ряду процессов мониторинга.

Таким образом, не только эффективное получение, но и качественное усвоение больших объемов данных от БПЛА, являются важными задачами при мониторинге в различных областях. Под усвоением данных в работе понимается процесс взаимодействия «БПЛА – приемник», исходя из чего решается задача анализа и выявления особенностей обработки больших объемов данных от БПЛА при мониторинге в зависимости от использования различных конфигураций транзакций, содержащих разнообразные запросы к системе управления базами данных.

Применение БПЛА в сельскохозяйственном производстве эфиромасличных культур. Как было рассмотрено выше, беспилотные летательные аппараты для задач мониторинга в сельском хозяйстве России – одно из интенсивно развивающихся направлений, на которое постоянно усиливается спрос. Для обеспечения точной обработки площади земельных участков создают и обновляют аппаратно - программные комплексы, которые в заданные сроки обрабатывают полученные данные, в том числе при мониторинге роста и созревания эфиромасличных культур (ЭМК) [62].

Для достижения наибольшей производительности в сельскохозяйственных угодьях необходимо обладать детализацией территориальных особенностей, рельефа, грунта полей, что особенно важно, например, в областях, где выращивают эфиромасличные культуры, к которым относятся: роза, лаванда, кориандр, мята и др. [67]. Описание методов и технологий выращивания ЭМК представлены в табл.1.1

Табл.1.1 Методы и технологии выращивания ЭМК

Культура	**Методы выращивания**	**Технологии выращивания**	**Возможности применения БПЛА**
Лаванда	Выращивание семенами или рассадой; посадка на холмы или в ряды с определенным расстоянием; использование пленочных теплиц для раннего выращивания.	Выбор сортов, подготовка почвы, удобрение, полив, обрезка, борьба с вредителями и болезнями, сбор и переработка урожая. Подходят среднетяжелые, песчаные почвы. Размножают семенами или рассадой. Посадка на холмы или в ряды с расстоянием до 100 см между растениями. Важно правильно проводить обрезку и убирать сорняки вокруг кустов. Уборка проводится при полном цветении растений.	Мониторинг интенсивности усвоения почвой влаги, удобрений; Мониторинг процесса полного созревания и т.п.
Кориандр	Выращивание рассадой или семенами; высадка на холмы или в ряды; использование пленочных теплиц для раннего выращивания.	Выбор сортов, подготовка почвы, удобрение, полив, обрезка, борьба с вредителями и болезнями, сбор и переработка урожая. Размножение черенками, рассадой или семенами. Посадка на холмы или в ряды с расстоянием 40-50 см между растениями. Обрезка проводится после цветения. Растение засухоустойчиво, но чувствительно к заморозкам.	Мониторинг интенсивности усвоения почвой влаги, удобрений; Мониторинг процесса полного созревания и т.п. Не подходит при реализации раннего выращивания в теплицах.
Мята	Выращивание рассадой или семенами; высадка на холмы или в ряды; уход за растениями включает обрезку и прополку.	Выбор сортов, подготовка почвы, удобрение, полив, обрезка, борьба с вредителями и болезнями, сбор и переработка урожая.	Мониторинг интенсивности усвоения почвой влаги, удобрений; Мониторинг процесса полного созревания и т.п.

Согласно данным табл.1.1 операции с БПЛА позволяют снизить затраты на трудовые ресурсы и ресурсы оборудования при реализации мониторинга процессов выращивания ЭМК, а также улучшить качество и количество производимой продукции. Рассмотрим различные операции, которые выполнены с использованием БПЛА в сельском хозяйстве, а также

преимущества и недостатки каждой операции в зависимости от конкретных требований и задач.

Технологично оснащенные БПЛА в сельском хозяйстве способны выполнять разнообразные операции:

- Аэросъемка – выполняет задачу определения воздействия внешних факторов на объекты, а также другие дефекты, которые нуждаются в своевременном устранении. Аэросъемка с БПЛА более детальная, чем фиксация со спутника, за счет небольшой высоты полета. Кроме того, БПЛА позволяют снимать даже в условиях искажения погодных условий: сильного ветра и пасмурности.
- Видеосъемка – характерная производительность БПЛА при фиксации видео кадров 30 км² за 1 час. Это свидетельствует, как об увеличении обработки, так и об уменьшении бюджетной нагрузки на предприятие в сравнении с применением сверхлегкой авиации. А это означает, что время и бюджет снижается в сравнении со сверхлегкой авиацией.
- 3D моделирование – выполняет задачу для определения земельных участков, которые требуют увлажнения в засушливых зонах, либо переувлажнением. Позволяет создавать план для увлажнения/осушения почвы, мелиорации земель.
- Тепловизионная съемка – выполняет задачу с применением замеров спектра инфракрасного излучения различных диапазонов. Исследование с БПЛА предоставляет возможность определения сроков дифференцирования точек роста, что напрямую влияет на урожайность и сохранение продуктивных свойств [61].
- Лазерное сканирование – используется для рассмотрения местности на труднодоступной или недоступной площади.
- Опрыскивание – позволяет БПЛА использовать выборочное оснащение земельных участков. Такой подход позволяет обрабатывать дефектные и больные растения.

При этом любой из видов операций БПЛА требует обработки данных и систематическое сохранение полученных сведений в базе данных (БД) посредством взаимодействия с оператором диспетчерского центра. Кроме того, при таком взаимодействии требуется интенсивный перерасчет координат для оптимизации движения БПЛА. Таким образом, комплекс задач мониторинга требует внедрения системы управления (диспетчеризации) автономными БПЛА на основе искусственного интеллекта (ИИ); укрупненная схема такой системы представлена на рис. 1.10. Для функционирования системы требуется взаимодействие нескольких подсистем: диспетчера, системы принятия решений на основе ИИ, подсистем технического зрения, стратегий выполнения с/х операций (полив/удобрение и т.д.), управления движения БТС, подсистемы навигации.

Под мониторингом ЭМК понимается процесс слежения за ростом и развитием ЭМК посредством БПЛА (выбор конкретного технического решения БПЛА не входит в данное исследование). Данные, собранные в ходе мониторинга ЭМК, анализируются и интерпретируются в соответствии с целями и принципами управления ростом и развитием ЭМК. Анализ данных представляет собой не сами результаты мониторинга, а данные координатного пространства при управлении агропромышленными БПЛА на основе учета типа запросов SQL (вида JOIN). Исследования включают статистический анализ, моделирование и экспертную интерпретацию.

Из представленной на рис. 1.9 схемы видно, что критически важной задачей построения такой системы является обеспечение надежного, устойчивого и эффективного обмена информацией между всеми компонентами системы, с учетом большого объема передаваемых данных и жёстких требований к временам их передачи.

Таким образом актуальной является задача оптимизации критических баз данных, которые являются основой систем управления БПЛА для мониторинга процессов выращивания эфиромасличных культур (ЭМК). Эффективное управление и анализ больших объемов данных, получаемых в ходе мониторинга,

являются ключевыми факторами для повышения точности и своевременности принимаемых решений.

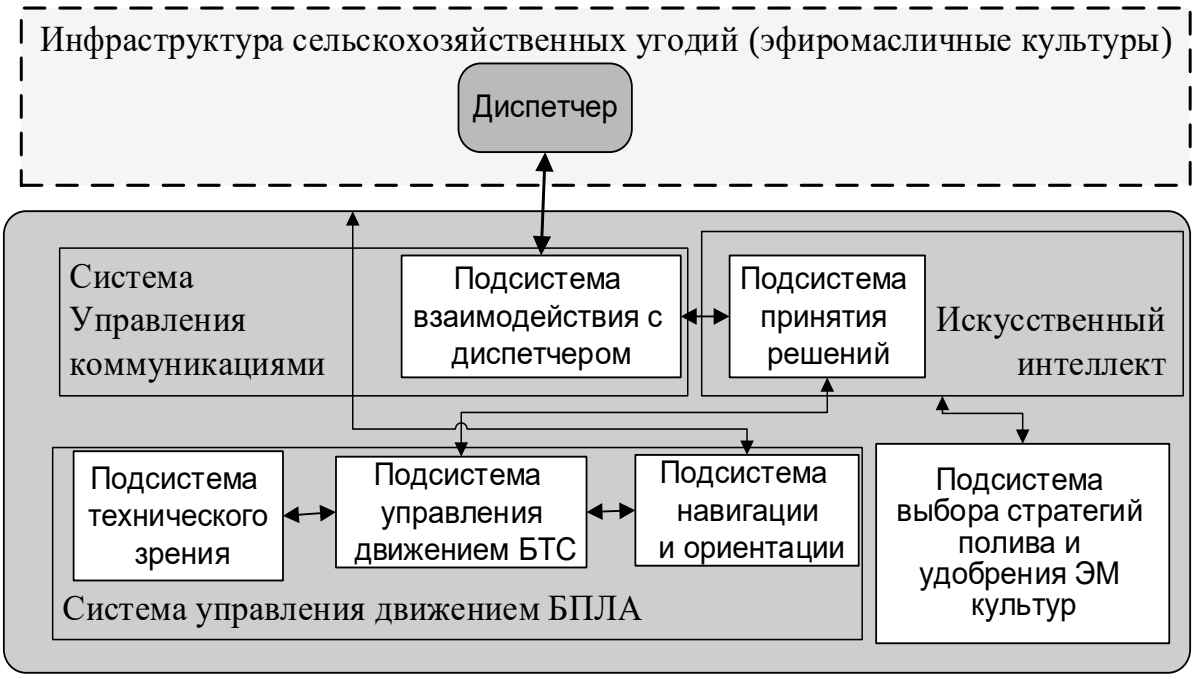


Рис. 1.10 Укрупненная схема взаимодействия диспетчерского центра и искусственного интеллекта в составе БПЛА в сельскохозяйственном мониторинге ЭМК

Актуальность разработки методов оптимизации баз данных также диктуется необходимостью снижения временных задержек и обеспечения высокой производительности системы в реальном времени. Решение этих задач открывает новые возможности для повышения эффективности использования БПЛА в сельском хозяйстве и повышения общей продуктивности агропромышленных процессов [19].

1.6 Принципы повышения производительности прикладных высоконагруженных КБД

Задачи обеспечения целостности при критичности по отношению к нагрузке БД, приобретают особое значение. Это относится, например, к производствам, где требуется хранение и обработка критически важных данных:

- онлайн-мониторинг и прогнозирование явлений внешней или внутренней среды в рамках инфраструктуры производственных систем, например, на крупных, распределенных комплексах предприятий;
- ресурсные системы предприятий, в которых должен быть обеспечен многопользовательский интерактивный доступ и оперативное обслуживание производственных мощностей;
- непосредственно промышленные системы с требованием защиты и распределенной клиент-серверной структурой и т.п [9-12].

В этих и других областях частоты обращения к БД пользователей (или операторов систем) могут неконтролируемо расти, в связи с чем возникает высокая нагрузка на систему. С другой стороны, в ряде случаев, нагрузка на БД может расти и в связи с ростом объема самой БД сложными запросами на выборку информации в результате соединения двух и более таблиц (запросы на основе операций языка SQL – join). Таким образом, для решения задачи обеспечения заданной или требуемой производительности БД, целесообразно рассматривать область высоконагруженных систем [13].

Важным фактором, влияющим на особенности высоконагруженных систем, является время отклика системы, масштабируемость, модульность, нагрузка на интеграционный слой, дублирование критических узлов. Время отклика в таких системах напрямую определяется временем выполнения запроса [12-13].

Оптимизация запросов – это один из важных этапов их обработки, от которого зависит общая производительность критических БД, однако, представляет собой сложную исследовательскую задачу [14-17]. Методы, подходящие для простых случаев, не всегда приемлемы для особенностей конкретных реализаций, вследствие чего применяются довольно сложные методы денормализации.

Таким образом, проанализировав основные факторы и показатели, перейдём к рассмотрению существующих подходов к оценке производительности. Наиболее часто для анализа производительности БД в

целом и высоконагруженных БД в частности, применяется моделирование нагрузки на основе «тяжелых» запросов, в частности join. Как известно, существует несколько типов реализации запросов join, что имеет довольно значимое влияние на время их выполнения [16, 18-19].

1.7 Анализ производительности высоконагруженных КБД на основе реализации различных типов операций типа JOIN к РМД

Соединение отношений на языке реляционной алгебры реализуется операцией конкатенации кортежей: $R=R_1 \underset{i\theta j}{\triangleright\triangleleft} R_2$, где i, j номера или имена доменов в отношениях R1 и R2 соответственно, θ – операция сравнения.

На языке манипулирования данными (DML) операция соединения отношений реализована с помощью оператора SQL join, который используется для выборки данных из двух или более таблиц и последующего включения полученных данных в один результирующий набор [18, 19].

В языке SQL реализованы несколько типов операций join: INNER, LEFT, RIGHT, FULL, CROSS JOIN (внутреннее соединение «inner join», внешнее соединения «outer join», левое, правое, полное и перекрёстное соединение соответственно).

Пример запроса с внутренним соединением: SELECT * FROM Предприятие INNER JOIN Город, где Предприятие и Город – некоторые отношения БД. Каждый кортеж первого отношения сопоставляется с каждым кортежем другого отношения и для полученной конкатенированной (составной) строки производится проверка условия соединения, при истинности которого в отношение-результат добавляется этот составной кортеж [16,19].

Операторы левого и правого внешнего соединений не являются симметричными, а значит, результат выполнения запроса SQL с этими операторами зависит от порядка соединяемых таблиц, при этом в результирующую таблицу гарантированно попадет строка из левой таблицы при

запросе LEFT OUTER JOIN и строка из правой таблицы при запросе RIGHT OUTER JOIN. Работа оператора FULL JOIN вернет строки из всех отношений, которые участвуют в конкатенации кортежей с учетом пустых полей, обозначаемых идентификатором NULL [17].

Таким образом, планы запросов на основе различных типов операций join, значительно отличаются, что, в свою очередь определяет число операций чтения-записи, и в общем итоге, отражается на времени выполнения запросов.

1.8 Выводы по главе

Критические базы данных важная часть в функционировании отраслей: здравоохранение, финансы, транспорт и энергетика, где их сбои могут привести к серьёзным социально-экономическим и техногенным последствиям. Реляционные базы данных остаются основой структурированности, поддержке ACID-транзакций и универсальности SQL, что обеспечивает надёжность и согласованность данных. Однако рост объёмов информации, усложнение запросов и повышенные требования к скорости обработки в реальном времени обостряют проблему производительности критических баз данных.

Для решения этой проблемы применяются комплексные подходы, включающие структурные методы (денормализация, индексация, кэширование), программную оптимизацию запросов и использование хранимых процедур, а также аппаратные решения, масштабирование и резервирование инфраструктуры. Системы поддержки принятия решений и современные инструменты мониторинга, которые позволяют анализировать, прогнозировать и оптимизировать работу баз данных. Особое внимание уделяется применению критических баз данных в современных технологических сферах, например, в системах управления беспилотными аппаратами для мониторинга, где необходима обработка больших объёмов данных в режиме реального времени.

В главе приведена классификация задач исследования на примере применения в беспилотных технологиях: для БПЛА, где параметрами оптимизации являются является время и объем передаваемых данных (для БПЛА

оперативного назначения, например, пожарных); для БПЛА в мониторинге, где оптимизируется объём передаваемых данных при взаимодействии с КБД и на примере БТС, где важна оперативность реакции КБД. Такое представление задач позволило выявить наиболее продуктивные факторы, влияющие на возможность регулирования производительности КБД.

Дальнейшие главы посвящены формулировке и решению задачи многокритериальной оптимизации КБД в контексте системы «БПЛА/БТС – диспетчерский центр».

2. ПОЛИМОДЕЛЬНОЕ ПРЕДСТАВЛЕНИЕ ЗАДАЧ АНАЛИЗА И МНОГОКРИТЕРИАЛЬНОЙ МОДЕЛИ ОПТИМИЗАЦИИ ЗАПРОСОВ К КБД

2.1 Выбор и обоснование критериев оптимизации производительности КБД

Производительность определяется как способность системы обрабатывать запросы в заданные временные рамки при минимальных ресурсных затратах. Это позволяет использовать аналитические и численные методы для оценки различных стратегий повышения производительности. Теория и аналитика в этом случае необходимы для определения оптимальных параметров системы: количество процессоров, объем памяти и т.д.

В общем случае формализованная модель задачи оптимизации производительности КБД может быть описана следующим образом: найти вектор управляемых параметров, максимизирующий производительность БД π при заданных ограничениях, то есть: $\pi=f(Z,T,R,C,N,I,\varepsilon) \rightarrow \max$, где π – производительность КБД; Z – множество решаемых задач; T – время выполнения; R – ресурсы (память, процессоры и т.д.); C – сложность базы данных; N – число записей; I – интенсивность запросов; ε – случайный параметр, связанный с неопределенностью.

Ограничения включают временные рамки выполнения задач КБД:

– $T_{\min} \leq T \leq T_{\max}$ - отражают предметно-ориентированные требования, например, в системах реального времени для авиадиспетчеризации Tmax может составлять десятки миллисекунд, что обусловлено человеческим фактором и требованиями безопасности. В аналитических платформах (OLAP) допустимы часы обработки, но есть ограничение снизу (Tmin), связанное с физическими пределами скорости чтения данных;

– допустимые ресурсы $R \in R_{\text{adm}}$, определяются бюджетом (стоимость серверного оборудования или облачных мощностей) и физическими пределами инфраструктуры;

– предельную сложность схемы $C \leq C_{crit}$ - обеспечивает сопровождаемость системы. Например, чрезмерно сложная схема или индексация увеличивают время разработки и риск ошибок, нивелируя выгоды от производительности.

Производительность в общем случае безразмерна и нормирована в диапазоне [0; 1].

Таким образом, при определении критериев оптимизации производительности КБД формулировка сводятся к задаче, в которой необходимо одновременно минимизировать время отклика (T), максимизировать пропускную способность (I) и минимизировать потребление ресурсов (R), оставаясь в рамках заданных системных ограничений.

Критерии выбраны исходя из комплексного характера производительности, которая является компромиссом между тремя ключевыми факторами: скоростью (время отклика), эффективностью (использование ресурсов) и нагрузочной способностью (интенсивность запросов). Формализация этих критериев в единую модель позволяет избежать локальной оптимизации (например, уменьшения времени одного запроса ценой роста потребления памяти или падения общей пропускной способности системы).

2.2 Многоуровневое представление временной составляющей модели производительности КБД

В качестве развития принципов полимоделирования в рассматриваемой задаче применяется многоуровневое представление временной составляющей модели производительности КБД: иерархия временных параметров представляет собой следующее описание.

Уровень 1: Время выполнения SQL-запроса (T_query);

T_execution - время выполнения запроса в СУБД,

T_join - специализированное время для JOIN-операций,

T_index - время поиска по индексам, T_io - время операций ввода-вывода,

Уровень 2: Время взаимодействия с БД (T_interaction);

T_network - сетевая задержка передачи запроса/ответа,

T_queue - время ожидания в очередях СУБД,

T_auth - время аутентификации и авторизации,

Уровень 3: Время структурных преобразований (T_structural);

T_denormalization - время применения денормализации,

T_index_rebuild - время перестроения индексов,

T_schema_change - время изменения схемы БД,

Уровень 4: Время физического взаимодействия в прикладных задачах (T_physical);

T_transmission_BLA_to_DC - время передачи запроса от БПЛА к ДЦ,

T_processing_DC - время обработки в диспетчерском центре,

T_response_DC_to_BLA - время передачи ответа обратно к БПЛА,

Общее время выполнения задачи:

T_total=T_query+T_interaction+T_structural+T_physical, где Tquery =T_execution+T_join++T_index+T_io; T_interaction=T_network+T_queue+T_auth,

T_structural=T_denormalization+T_index_rebuild+T_schema_change,

T_physical=T_transmission+T_processing_DC+T_response.

Такое представление иерархии времени позволило строить взаимно согласованные модели. Но, с другой стороны, иерархическая временное представление требует построения полимодельного представления рассматриваемой задачи. Рассмотрим ряд постановок задач оптимизации производительности КБД в такой интерпретации.

2.3 Формализованная постановка задачи многокритериальной оптимизации запросов к КБД

В общем случае формализованная модель многокритериальной оптимизации запросов к КБД может быть описана следующим образом: найти такой вектор управляющих параметров и стратегий $\vec{X}^*$, который

максимизирует производительность КБД, π при заданных системных ограничениях и динамической нагрузке:

$$\vec{X}^{*}=\arg\max \pi(\vec{X},t),\ \pi=f(\vec{Z},T,\vec{R},C,N,\vec{I},\epsilon)\rightarrow\max \qquad (1)$$

при ограничениях следующих типов:

1. Временные: $Tmin\leq T\leq Tmax$, где T=Tquery+Tinteraction+Tstructural+Tphysical;
2. Ресурсные: $\vec{R}\in\vec{R}adm$ (память, CPU, диск, сеть);
3. Структурные: $C\leq Ccrit$ (сложность схемы БД);
4. Объемные, касающиеся данных и записей: $N\leq Ncrit$ (количество записей в БД);
5. Функциональные: обеспечение ACID-свойств, целостности данных и непрерывности сервиса.

Управляющие параметры $\vec{X}$ в (1) включают:

$Xjoin\in\{INNER,LEFT,RIGHT,FULL\}$ – тип операции соединения;

Xarch – степень денормализации схемы БД;

Xindex – стратегия индексации;

Xcache – политика кэширования;

Xroute – алгоритм маршрутизации запросов в кластере.

Критерии оптимизации (подмножество π): минимизация общего времени отклика системы *Ttotal*; максимизация пропускной способности (количество обработанных транзакций в единицу времени), минимизация использования критических ресурсов $\vec{R}$, максимизация стабильности и предсказуемости времени отклика (минимизация дисперсии *T*).

В классической постановке необходимые и достаточные условия существования оптимума в многокритериальной задаче оптимизации КБД выражаются через Парето-оптимальность с учетом компактности множества и свойств функции π.

В задаче поиска Парето-оптимального вектора $\vec{X}^* = \arg\max_{\vec{X}\in\mathcal{X}} [\pi(\vec{X}, t)]$, где $\pi: \mathbb{R}^m \to \mathbb{R}^k$ – векторная функция производительности, $\mathcal{X}$ – допустимое множество стратегий, t – время как обобщенный параметр оптимизации.

Необходимые условия существования оптимума заключается в компактности множества решений: $\mathcal{X} \subset \mathbb{R}^m$ замкнуто и ограничено, π непрерывна на $\mathcal{X}$: $\vec{X}^* \in \mathcal{P}(\mathcal{X}) \Leftrightarrow \nexists \vec{X} \in \mathcal{X}: \pi_i(\vec{X}) \geq \pi_i(\vec{X}^*)$.

При компактности допустимого множества $\mathcal{X}$ и непрерывности векторной функции производительности $\pi(\vec{X}, t)$ гарантируется существование непустого множества Парето-оптимальных решений $\mathcal{P}(\mathcal{X}) \neq \emptyset$. Это обеспечивает хотя бы одно эффективное управление параметрами $\vec{X}^* = (X_{\text{join}}, X_{\text{arch}}, X_{\text{index}}, X_{\text{cache}})$, балансирующее временные, ресурсные, структурные и функциональные ограничения.

Ограничения в данном случае следующие: $T_{\min} \leq T(\vec{X}) \leq T_{\max}$, $\vec{R}(\vec{X}) \in \mathcal{R}_{\text{adm}}$, $C(\vec{X}) \leq C_{\text{crit}}$, $N(\vec{X}) \leq N_{\text{crit}}$.

Достаточным условием существования оптимума в общей постановке является квазивыпуклость функции; π_i квазивыпуклы по $\vec{X}$, ограничения выпуклы, существует скалярная свертка

$$\vec{X}^* = \arg\max_{\vec{X}\in\mathcal{X}} \sum_{i=1}^{k} \lambda_i \pi_i(\vec{X}), \lambda_i > 0.$$

Парето-фронт компактен: $\mathcal{P}(\mathcal{X})$ замкнут, ограничен снизу $\vec{z}^* = \max_i \pi_i$.

Необходимые и достаточные условия существования оптимума позволяют гарантировать наличие Парето-эффективных стратегий управления параметрами $\vec{X}^*$ для максимизации производительности КБД $\pi(\vec{X},t)$ при строгом соблюдении всех системных ограничений, обеспечивая робастный выбор компромиссного решения в многокритериальной задаче.

Однако, при решении конкретных задач оптимизации наблюдается дискретная природа управляющих параметров:

- тип соединения (X_join) представляет собой конечное множество {INNER, LEFT, RIGHT, FULL, CROSS, ...};
- стратегия индексации (X_index) реализуется выбором конкретного подмножества индексов из множества всех возможных (комбинаторная задача экспоненциальной сложности);
- алгоритм маршрутизации (X_route) на основе выбора из конечного набора алгоритмов;
- политика кэширования (X_cache) может быть описана дискретными стратегиями (LRU, LFU, ARC) и целочисленными параметрами (размер кэша).

Таким образом, пространство поиска X становится не непрерывным выпуклым множеством, а дискретным (комбинаторным). Классические методы непрерывной оптимизации (градиентные) неприменимы, так как они требуют понятия окрестности и непрерывности. Имеем двухуровневую оптимизационную задачу, которая требует полимодельного описания.

При наличии дискретных параметров или невыпуклостей требуется привлечение комбинаторных методов (например, предложенного гибридного метода ветвей и границ и нейросети) или метаэвристик. Это обеспечивает робастный выбор компромиссного решения, балансирующего противоречивые требования к производительности КБД.

Таким образом, метод ветвей и границ является необходимым инструментом для практической реализации поиска Парето-оптимальных решений с учетом особенностей формализованной модели, поскольку он прямо адресует дискретность управляющих параметров КБД; позволяет бороться с невыпуклостью через механизм релаксаций и оценок; предоставляет не только решение, но и метрику его качества (гарантированную близость к оптимуму), что критически важно для принятия ответственных решений при настройке высоконагруженных КБД; остается совместимым с многокритериальным подходом, так как может быть применен к скаляризованной задаче (взвешенной сумме критериев) для построения приближенного Парето-фронта.

Следовательно, для задач оптимизации КБД, сочетающих непрерывные, дискретные и невыпуклые компоненты, необходим гибридный подход: использование эффективных методов непрерывной оптимизации внутри узлов МВГ, что и делает метод ветвей и границ обоснованным для решения практических задач оптимизации производительности КБД.

2.4 Задача влияния категоризации транзакций в форме запроса SQL на производительность БД

Категоризация транзакций по типу запросов (JOIN, BLOB, INDEX) позволяет более точно управлять производительностью.

Цель задачи состоит в исследовании влияния категоризации транзакций в форме запроса SQL на производительность БД и найти оптимальные значения параметров модели, обеспечивающие максимальную производительность сервера при обработке запросов от источника.

Определим параметры модели: Z – поток данных от источника, t – время формирования запроса, w – конфигурация сервера (производительный, стандартный, слабый), s – тип запроса (s0, sJoin, sblob), a – трудоемкость обработки запроса, β – управляемые параметры, влияющие на производительность сервера, k – интервал времени, отражающий производительность сервера.

Тогда функционал, определяющий поток данных можно представить как

$Q = f(t, w, s, a, \beta, k)$. Для различных конфигураций систем может быть построено семейство представлений: Q = {Q(w2), Q(w1), Q(w0)}, где Q(w) = f(t, s, c, p, τ).

Поток данных Z_t от источника (с учетом времени формирования запроса t и до оценки производительности в следующий момент времени t+1) от заявок имеет вид:

$$Z_t :< (s_0 \vee s_{Join} \vee s_{blob})_t \left| (\alpha, \upsilon_b, \upsilon_i)_{t+1} > . \right. \quad (2)$$

Выражение (2) распространяется на все типы серверов, в общем случае имеющие конфигурации: производительный (w_2), стандартный(w_1), слабый(w_0). Тогда справедливо семейство представлений:

$$\begin{aligned} Z_t^{w_0} &:< (s_0 \vee s_{Join} \vee s_{blob})_t \left| (\alpha^{w_0}, \upsilon_b, \upsilon_i, \xi^{w_0})_{t+k0} >, \right. \\ Z_t^{w_1} &:< (s_0 \vee s_{Join} \vee s_{blob})_t \left| (\alpha^{w_1}, \upsilon_b, \upsilon_i, \xi^{w_1})_{t+k1} >, \right. . \\ Z_t^{w_2} &:< (s_0 \vee s_{Join} \vee s_{blob})_t \left| (\alpha^{w_2}, \upsilon_b, \upsilon_i, \xi^{w_2})_{t+k2} >, \right. \end{aligned} \quad (3)$$

где $\alpha^{w_0}, \alpha^{w_1}, \alpha^{w_2}-$ трудоемкости обработки различных запросов s_0 ,s_{Join} , s_{blob} на определенных типов конфигураций серверов; $\xi^{w_0}, \xi^{w_1}, \xi^{w_2}-$ управляемые параметры, влияющие на производительность каждой конфигурации серверов СУБД; $k0, k1, k2-$ интервалы времени, отражающие производительность каждой конфигурации серверов СУБД [112].

Таким образом, в следующей главе на основе математического описания будут показаны результаты имитационной модели для исследования влияния вида транзакций к СУБД на прием и усвоение данных от БПЛА с целью поддержки принятия решений при управлении параметрами передачи данных от БПЛА при мониторинге объектов или явлений различной природы.

2.5 Комплекс моделей обеспечения эффективности системы информационного обмена

Гибридные системы, объединяющие данные от беспилотных летательных аппаратов (БЛА) и наземных транспортных средств (БТС), представляют собой сложный объект для управления производительностью БД. БЛА генерируют высокочастотные геопространственные запросы, такие как обновление карт в реальном времени или анализ видео с камер, тогда как БТС обеспечивают

стабильный поток телеметрии — данные о местоположении, скорости, состоянии оборудования. Например, в умных городах гибридные системы, объединяющие данные от БЛА и БТС, используются для мониторинга дорожного движения, управления парковками и анализа экологической обстановки. На рис. 2.1 показаны типы сценариев в рамках исследования.

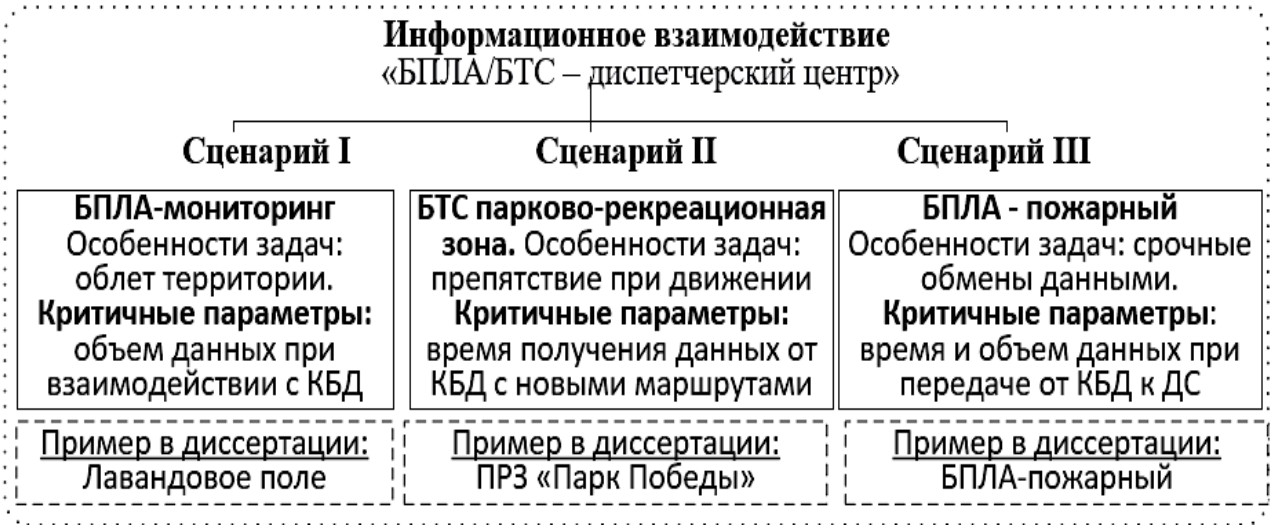


Рис. 2.1 Типы сценариев и задач в рамках исследования

Рассмотрим особенности БТС (беспилотный автомобиль в парковой зоне):

- Высокая динамичность нагрузки с частыми короткими запросами на парковочные места, резервирование, распознавание номеров (до 1000 запросов/мин);
- Критические параметры: x_2 (RAM) и x_4 (интенсивность) доминируют из-за реального времени и IoT-датчиков;
- Ограничения: строгие временные $T_{\max} \approx 50$ мс для предотвращения коллизий;
- Приоритет ветвления: по x_4 (интенсивность) и x_1 (время запросов).

Особенности БПЛА (сельскохозяйственный мониторинг):

- Стационарная нагрузка связана с периодическими большими BLOB-запросами (аэрофотосъемка, метеоданные, 100–500 МБ);
- Критические параметры: x_5 (размер БД) и x_3 (сложность схемы) из-за объемных данных;

– Ограничения: объемные $N_{\text{crit}} \approx 10^7$ записей, $C_{\text{crit}} \approx 0.8$;

– Приоритет ветвления: по x_5 (размер) и x_3 (сложность).

Согласно интегральной модели (10), гибридный метод адаптируется для каждого сценария.

Адаптивная гибридная оценка для рассмотренных сценариев имеет вид:

$$E_{\text{scenario}}(c) = \begin{cases} 0.8 \cdot F_{\text{BTS}}(c) + 0.2 \cdot f_{\text{NN}}^{\text{park}}(c), & \text{для БТС} \\ 0.6 \cdot F_{\text{BPLA}}(c) + 0.4 \cdot f_{\text{NN}}^{\text{agri}}(c), & \text{для БПЛА} \end{cases}.$$

Правило ветвления по сценариям:

БТС: $j^* = \arg\max\{x_4, x_1\}$ (интенсивность, время),

БПЛА: $j^* = \arg\max\{x_5, x_3\}$ (размер, сложность).

Сценарий I: БПЛА сельхозмониторинг (оптимизация по объему данных) реализована по следующей схеме моделирования:

$$F_{\text{BPLA}}(x) = \pi(t_i)^{-1} + \lambda_3 \max(0, x_5 - 0.85)^2 + \lambda_4 \max(0, x_3 - 0.8)^2,$$

где $\pi(t_i) = f(Z_{\text{agri}}, T_{\text{blob}}, R_{\text{stationary}}, C_{\text{complex}}, N_{\text{large}}, I_{\text{periodic}}, \varepsilon_{\text{weather}})$, Z_{agri} – BLOB-запросы аэрофотосъемки, $N_{\text{large}} \approx 10^7$, $I_{\text{periodic}} \approx 0.3$, веса: $\lambda_3 = 8, \lambda_4 = 6$ (приоритет размера и схемы).

Сценарий II: БТС ПРЗ (оптимизация по времени) реализована по следующей схеме моделирования: $F_{\text{BTS}}(x) = \pi(t_i)^{-1} + \lambda_1 \max(0, x_1 - 0.05)^2 + \lambda_2 \max(0, 0.3 - x_2)^2$, где $\pi(t_i) = f(Z_{\text{park}}, T_{\text{reserve}}, R_{\text{IoT}}, C_{\text{simple}}, N_{\text{slots}}, I_{\text{high}}, \varepsilon_{\text{traffic}})$, $I_{\text{high}} = \gamma_{\text{reserve}} + \gamma_{\text{detect}} \approx 0.9$, веса: $\lambda_1 = 10, \lambda_2 = 5$ (приоритет времени и памяти).

Общая математическая модель такой системы включает два компонента производительности – π_{BLA} для БЛА и π_{BTC} для БТС

Введем ряд обозначений, учитывающих наличие двух принципиально различных групп объектов обеспечения: БЛА и БТС:

π – общая производительность БД, $\pi_{\min} \le \pi_k \le \pi_{\max}$, $k = \|K\|$, где $\pi_{\min}$, $\pi_{\max}$ – минимальная допустимая и максимальная возможная производительность КБД;

$z_i^{BLA}, i = \overline{1, I}$ – задача по поддержке функционирования БПЛА, сформулированная в форме запроса SQL из множества задач Z;

$z_j^{BTC}, j=\overline{1,J}$ – задача по поддержке функционирования БТС, сформулированная в форме запроса SQL из множества задач Z, $z_q^{BTC} \in Z$; ;

$\tau_j^{BLA}, j=\overline{1,J}$ – период времени, в течение которого решается задача; $\tau_0^{BLA} \le \tau_j^{BLA} \le \tau_{kr}^{BLA}$, где τ_0^{BLA}, $_{kr}^{BLA}$ – начальный и критический (конечный) моменты времени реализации запроса SQL, учитывающие технические возможности аппарата БЛА;

$\tau_w^{BTC}, w=\overline{1,W}$ – период времени, в течение которого решается задача $z_j^{BTC} \in Z$; $\tau_0^{BTC} \le \tau_q^{BTC} \le \tau_{kr}^{BTC}$, где τ_0^{BTC}, $_{kr}^{BTC}$ – начальный и критический (конечный) моменты времени реализации запроса SQL, учитывающие технические возможности аппарата БТС [11].

Кроме того, применения специальных типов запросов, таких как экспорт маршрутного движения (EXPORT) и обновление карт (UPDATE), учтём их отдельно:

$$z_{i_1}^{BLA(EXPORT)}, z_{j_1}^{BTC(EXPORT)}; \tau_{i_1}^{BLA(EXPORT)}, i_1=\overline{1,I_1}, \tau_{j_1}^{BTC(EXPORT)}, j_1=\overline{1,J_1};$$

$$z_{i_2}^{BLA(UPDATE)}, z_{j_2}^{BTC(UPDATE)}; \tau_{i_2}^{BLA(UPDATE)}, i_2=\overline{1,I_2}, \tau_{j_2}^{BTC(UPDATE)}, j_2=\overline{1,J_2};$$

$$\left(z_{i_1}^{BLA(EXPORT)} \cup z_{j_1}^{BTC(UPDATE)}\right) \cup \left(z_{i_1}^{BLA(EXPORT)} \cup z_{j_1}^{BTC(UPDATE)}\right) \subseteq Z;$$

$$\tau_{j_1}^{BTC(EXPORT)} + \tau_{j_2}^{BTC(UPDATE)} < \tau_{kr}^{BTC}; \tau_{i_1}^{BLA(EXPORT)} + \tau_{i_2}^{BLA(UPDATE)} < \tau_{kr}^{BLA}.$$

Тогда производительность КБД в общем случае определяется следующим образом:

$$\begin{cases} \pi_{\tau_i^{BLA}}^{z_i^{BLA}} = F\left[\left(z_i^{BLA(EXPORT)}, z_i^{BLA(UPDATE)}, z_i^{BLA(\bullet)}\right), t_i^{BLA}, r_i^{BLA}(t_i), \mu, V, \gamma^{BLA}(t_i), \xi_i)\right], \\ \pi_{\tau_j^{BTC}}^{z_j^{BTC}} = Q\left[\left(z_j^{BTC(EXPORT)}, z_j^{BTC(UPDATE)}, z_j^{BTC(\bullet)}\right), t_j^{BTC}, r_j^{BTC}(t_j), \mu, V, \gamma^{BTC}(t_j), \xi_j)\right], \end{cases} \quad (4)$$

где r_i^{BLA} – ресурсы или определенный ресурс, который наибольшим образом

влияет на производительность КБД, например, объем оперативной памяти и т.п.), в момент времени t_i, причем $r_{\min}^{BLA}(t_i) \le r^{BLA}(t_i) \le r_{\max}^{BLA}(t_i)$; .

μ– коэффициент, отражающий сложность БД (число таблиц, типы связей и т.п.), $0 \le \mu \le 1$;

V – число записей КБД, $V \le V^{kr}$ (V^{kr} – критический объем КБД, при котором решение задачи из множества Z в заданное время не может быть обеспечено);

$\gamma^{BLA}(t_i)$, $^{BTC}(t_i)$ – средние интенсивности запросов по разным типам устройств соответственно в момент времени;

ξ– случайный параметр, отражающий возможную неопределенность при оценивании производительности КБД.

Ставится задача минимизации времени, в течение которого осуществится поддержка транспортного обслуживания ПРЗ с учетом наличия двух принципиально различных групп объектов обеспечения: БЛА и БТС:

$$\tau_i^{BLA}(t_i) = \arg \min_{\tau_i^{BLA}(t_i) \in T} (\pi^{z_i^{BLA}}_{\tau_i^{BLA}}, r_i^{BLA}(t_{i-1}))$$
$$\tau_j^{BTC}(t_j) = \arg \min_{\tau_j^{BTC}(t_j) \in T} (\pi^{z_j^{BTC}}_{\tau_j^{BTC}}, r_j^{BTC}(t_{i-1})) \quad (5)$$

при условиях: $r_{\min}^{BLA}(t_i) \le r^{BLA}(t_i) \le r_{\max}^{BLA}(t_i)$, $r_{\min}^{BLA}(t_i) \le r^{BLA}(t_i) \le r_{\max}^{BLA}(t_i)$;

$$V_{\min} \le V_{\tau_i}^{z_i} \le V^{kr};\ \mu_{\min} \le \mu \le 1 . \quad (6)$$

Условия (6) формулируют ограничения на критические значения времени τ_i, ресурсов $r(t_i)$, объем БД (число записей) $V_{\tau_i}^{z_i}$ и сложности БД (число таблиц, типы связей и т.п.) для производственной БД и определяются конкретными задачами и особенностями проектирования высоконагруженной БД. Параметры $\mu_{\min}$, $V_{\min}$ и $r_{\min}(t_i)$ определяют минимальную избыточность схемы БД при нормализации, относительно задач, касающихся конкретных технических устройств БЛА или БТС.

Поскольку рассматриваемые оценки времени реализации задач БД ИСПТО описываются векторными критериальными оценками, то для поддержки принятия решений по определению оптимальных режимов ее работы в многокритериальном пространстве, необходимо формировать множество решений, например, на основе Парето-подхода, оставляя выбор конкретного решения за ЛПР.

2.5 Модель оптимизации производительности высоконагруженных КБД

Уточним приведенную модель для задачи оптимизации производительности высоконагруженных КБД.

Для построения модели поддержки принятия решений по повышению производительности КБД используем рассмотренные критерии и параметры модели. В основе модели лежит вектор состояний системы, включающий параметры сложности схемы μ, объема данных V, интенсивности запросов γ_{join} и γ_{read}, а также доступные ресурсы на предыдущем шаге $r_{(\text{ti}-1)}$

Аналогично, примем π – производительность БД, $\pi_{\min} \le \pi_k \le \pi_{\max}$, $k = \|K\|$, где $\pi_{\min}$,– минимальная и максимальная производительность БД;

- $z_i, i = \overline{1, I}$ – задача, решаемая БД, сформулированная в форме запроса SQL из множества задач Z;
- $\tau_j, j = \overline{1, J}$ – период времени, в течение которого решается задача z_i; $\tau_0 \le \tau_j \le \tau_{kr}$, где τ_0, τ_{kr} – начальный и критический (конечный) моменты времени реализации запроса SQL.

Тогда функционал, определяющий производительность КБД в общем случае, будет иметь вид:

$$\pi_{\tau_i}^{z_i} = F(z_i, t_i, r(t_{i-1}), \mu, V, \gamma(t_{i-1}), \gamma_{join}(t_{i-1}), \xi). \quad (7)$$

Высокой считается нагрузка БД, когда выполняется одно и/или все условия:

1. Текущий объем БД $V_{\tau_i}^{z_i}$ близок к ;

2. Сложность БД (μ) близка к 1;
3. Средняя интенсивность запросов $\gamma_{join}(t_{i-1})$ близка к 1;
4. Объем требуемых ресурсов $r(t_i)$ близок к $r_{\min}(t_i)$;
5. Время реализации запроса значительно влияет на общую производительность системы и производственного процесса в целом.

Ставится задача при заданных $\gamma_{join}(t_{i-1})$, определить вид запроса join (INNER, LEFT, RIGHT, FULL из множества X: $x_{join} \in X$), при котором достигается максимальная производительность высоконагруженной БД, т.е. найти:

$$x_{join}^{z_i} = \arg\max_{x_{join} \in X} (\pi_{\tau_i}^{z_i}, \tau_i) \quad (8)$$

при условиях:

$$\tau_{\min} \le \tau_i \le \tau_{kr};\ r_{\min}(t_i) \le r(t_i) \le r_{\max}(t_i);\ V_{\min} \le V_{\tau_i}^{z_i} \le V^{kr};\ \mu_{\min} \le \mu \le 1. \quad (9)$$

Особое внимание уделяется оптимизации JOIN-операций, которые часто становятся «узким местом» в реляционных базах. Время выполнения JOIN-запроса может быть аппроксимировано как:

$$T_{join} = k\frac{V1 \cdot V2}{R_{mem}},$$

где V_1 и V_2–размеры соединяемых таблиц, R_{mem} – объем доступной памяти, а k – коэффициент, зависящий от типа соединения (INNER, LEFT, RIGHT, FULL). Например, INNER JOIN требует точного совпадения ключей и эффективен при наличии индексов, тогда как FULL JOIN генерирует временные таблицы, увеличивая нагрузку на память.

Задача сводится к выбору типа соединения X_{join}, который максимизирует производительность при заданных ограничениях:

$$\max_{X_{join}} \pi(t_i),\ X_{join} \in \{\text{INNER}, \text{LEFT}, \text{RIGHT}, \text{FULL}\}.$$

Эксперименты показывают, что для нормализованных схем с большим количеством таблиц оптимальным часто оказывается INNER JOIN, в то время

как в аналитических системах с денормализованными данными предпочтение может отдаваться FULL JOIN для сохранения всех записей.

Таким образом, интегрированная целевая функция с учетом многоуровневого представления временной составляющей модели имеет вид:

$$\Pi = \sum_{i=1}^{N_z} \left[\begin{matrix} \alpha \cdot \pi_i - \beta_1 \cdot T^i_{query} - \beta_2 \cdot T^i_{interaction} - \beta_3 \cdot T^i_{structural} \\ -\beta_4 \cdot T^i_{physical} - \gamma \cdot C_i \end{matrix} \right] \rightarrow \max, \quad (10)$$

где π_i - производительность i-го элемента системы; T^i_* - время на соответствующем уровне для i-го элемента; C_i - стоимость обслуживания или обработки i-го элемента; $\alpha, \beta_1, \ldots, \beta_4, \gamma$ - весовые коэффициенты, отражающие важность каждого фактора.

Целевая функция, например, для миссии БПЛА:

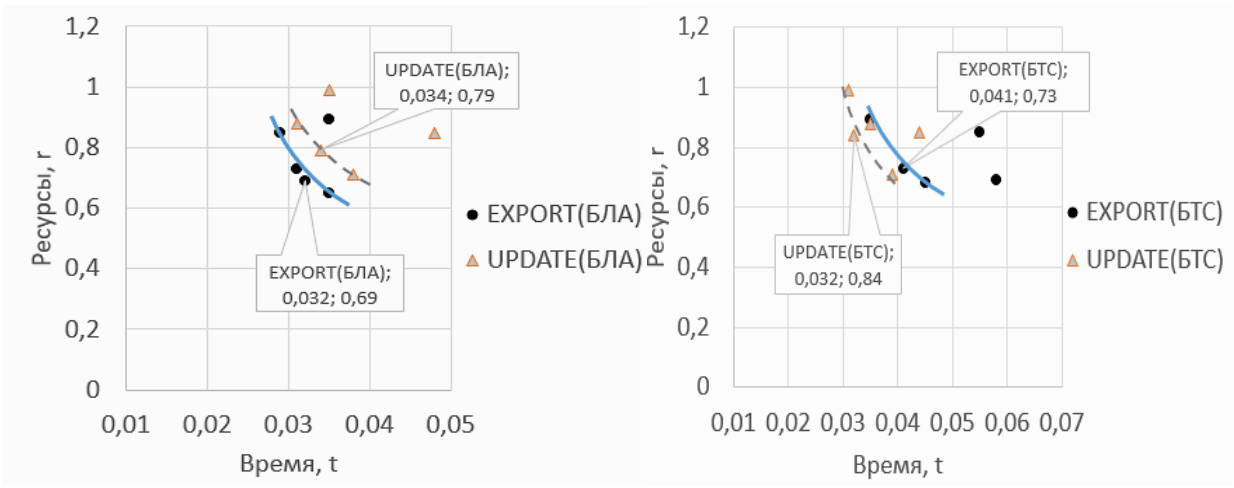
$$\Pi_{BLA} = w_1 \cdot \pi_{BLA} + w_2 \cdot \pi_{DC} - \lambda_1 \cdot T_{transmission} - \lambda_2 \cdot T_{processing} - \lambda_3 \cdot T_{response.}$$

с ограничениями $T_{transmission} \leq T^{transmission}_{max}$, $T_{processing} \leq T^{processing}_{max}$, $\pi_{BLA} \geq \pi^{critical}_{min}$.

2.6 Многокритериальная схема решения поставленной задачи на основе Парето-подхода

Принятие решений по многокритериальной схеме может быть получено на основе Парето-подхода. Как указывалось в разеделе2.3: в задаче поиска Парето-оптимального вектора $\vec{X}^* = \arg\max_{\vec{X} \in \mathcal{X}} [\pi(\vec{X}, t)]$, где $\pi: \mathbb{R}^m \rightarrow \mathbb{R}^k$ – векторная функция производительности, $\mathcal{X}$ – допустимое множество стратегий, t – время как обобщенный параметр оптимизации.

На рисунке 2.3 приведены схемы критериального пространства выбора решений лица, принимающего решения (ЛПР) для БЛА и БТС.



а) б)

Рис. 2.2 Критериальное пространство выбора решений ЛПР для двух типов технологий: а) БЛА; б) БТС

На рис. 2.2 визуализированы различные альтернативы решений для двух типов технологий (БЛА и БТС), позволяющие определить компромиссные варианты по нескольким критериям одновременно.

Приведенные на рисунках оценки решений задачи многокритериальной оптимизации производительности КБД на основе Парето-подхода позволили увидеть следующие закономерности: для БЛА поддержка операций EXPORT имеет лучшие показатели по скорости реализации, чем по операции UPDATE, тогда как для БТС имеется обратная тенденция. В развитии исследований предполагается уточнить полученные решения за счет увеличения объемов выборок имитационных моделей.

2.7 Схема полимодельного комплекса оптимизации производительности высоконагруженных КБД

Работа полимодельного комплекса по оптимизации начинается со сбора и систематизации входных данных, которые формируют основу для всех последующих этапов анализа. На рис.2.3. представлено предложенное полимодельное представление для оптимизации производительности высоконагруженных КБД.

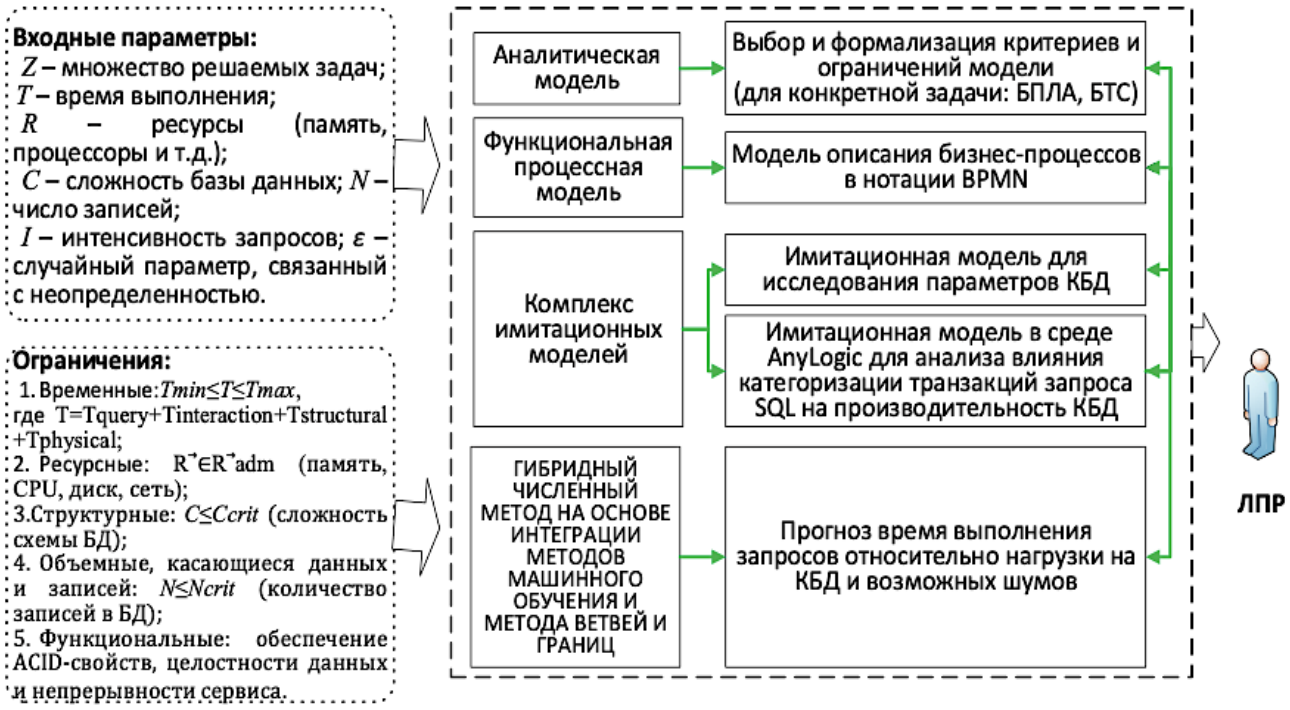


Рис. 2.3 Полимодельный комплекс для анализа и оптимизации КБД

На вход системы поступает комплекс параметров и их ограничений, включающий множество типовых задач, метрики времени выполнения запросов, доступные вычислительные ресурсы, показатели сложности структур данных, а также интенсивность входящих запросов, рис.2.3.

На следующем этапе эти данные поступают в аналитическую модель, где происходит точная формализация задачи оптимизации. Для конкретных сценариев применения БТС или БПЛА для телеметрии, выбираются и математически описываются целевые критерии, минимизация задержки или максимизация пропускной способности.

Параллельно для понимания организационного контекста строится функциональная модель на основе нотации BPMN. В этой модели формализуются бизнес-процессы, связанные с эксплуатацией высоконагруженной КБД, процесс обработки. Этап задает логические последовательности операций, которые в дальнейшем становятся объектом имитации и оптимизации.

Комплекс имитационных моделей - центральное место, в котором занимает специализированная имитационная модель КБД. Эта модель детально исследует внутренние параметры критической базы данных.

На заключительном этапе на основе данных, полученных от аналитической и имитационной моделей, применяется гибридный численный метод, представляющий собой комбинацию аналитико-имитационных подходов. Задачей этого метода является выполнение прогноза времени выполнения запросов в зависимости от текущей и прогнозируемой нагрузки на КБД. Результаты этого прогноза становятся основой для принятия обоснованных решений по оптимизации, включающих масштабирование инфраструктуры, перераспределение ресурсов и корректировку приоритетов обработки запросов.

Особенностью предложенного полимодельного комплекса для анализа и оптимизации КБД, интегрирующего структурные (денормализация), программные (индексация) и имитационные модели в единую сценарную модель, является возможность формализовать представление данных и создать инструментальную основу для имитационного моделирования высоконагруженных КБД.

2.8 Выводы по главе

Предложен комплексный подход к моделированию и оптимизации производительности КБД. Формализованы многокритериальные задачи оптимизации, учитывающие временные, ресурсные, структурные и функциональные ограничения. Предложено многоуровневое представление временной составляющей, которое позволяет детально анализировать и управлять задержками на различных этапах обработки запросов. Для решения оптимизационных задач используется подход Парето, обеспечивающий поиск компромиссных решений в условиях противоречивых критериев. Обоснованы необходимые и достаточные условия существования оптимума производительности, на основе сложности (двухуровневой) задачи оптимизации

показаны особенности и целесообразность построения полимодельного комплекса.

Предложена архитектура полимодельного комплекса, объединяющая аналитические, имитационные и функциональные модели, что позволяет проводить всесторонний анализ и прогнозирование производительности системы. Результаты моделирования показывают зависимость эффективности обработки запросов от типа транзакций и конфигурации серверов, а также подчеркивают важность оптимизации JOIN-операций в реляционных базах данных. Полученные модели и методы создают основу для принятия обоснованных решений по повышению производительности КБД.

3. МОДЕЛИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ КРИТИЧЕСКИХ БАЗ ДАННЫХ В КОНТЕКСТЕ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМ

3.1 Платформы и каналы передачи данных

В рамках исследования применялись современные инструменты и платформы для работы с базами данных. Основной платформой для выполнения SQL-запросов и оптимизации производительности стала Oracle, которая была выбрана благодаря своей надежности, поддержке сложных транзакций и способности работать с большими объемами данных. Для удобства управления базами данных, выполнения запросов и визуализации результатов применялся DBeaver – универсальный инструмент, который обеспечил гибкость и удобство при работе с различными СУБД, включая Oracle.

Для передачи данных между компонентами системы беспилотными летательными аппаратами (БПЛА), диспетчерские центры и серверы СУБД, задействованы различные каналы связи. Облачное хранилище использовалось для хранения и обработки больших объемов данных, что обеспечило высокую доступность информации и возможность масштабирования ресурсов в зависимости от нагрузки. Локальный сервер применялся в условиях, где требовалась минимальная задержка при обработке критических запросов, что особенно важно для систем реального времени. Также были проанализированы различные типы сетевых каналов (например, Wi-Fi, LTE), что позволило оценить их влияние на производительность системы и время передачи данных.

Имитационная модель, разработанная в среде AnyLogic, учитывала все эти аспекты, включая различные типы запросов (SELECT, JOIN, BLOB) и конфигурации серверов (производительный, стандартный, слабый). Это позволило воспроизвести динамическое поведение системы и провести детальный анализ влияния различных параметров на производительность базы данных.

3.2 Комплексная модель бизнес-процессов задачи анализа производительности КБД при информационном взаимодействии с беспилотным аппаратом

Рассматривать беспилотные аппараты можно как объект целостной критической инфраструктуры технологий на верхнем уровне в различных областях деятельности. В связи со сложным характером информационных взаимодействий с КБД при выполнении различных миссий БПЛА и БТС следует применить методологии моделирования процессов. Рассмотрим пример применения методологии моделирования процессов информационного взаимодействия БПЛА-КБД в нотации бизнес-процессов (BPMN).

Ввиду сложности данной инфраструктуры, такое рассмотрение требует целостной разработки, комплексный разбор:

– При интеграции данных система БПЛА должна быть интегрирована с другими информационными системами, для обеспечения непрерывного обмена и анализа данных в реальном времени.

– Если система автоматизирована, то можно применить методы машинного обучения и искусственного интеллекта для автоматизации процессов обработки данных. Это позволит увеличить эффективность и скорость принятия решений.

– Создание системы управления данными, которая будет обеспечивать хранение, обработку и доступ к данным БПЛА и других информационных источников.

– Реализовать централизованное управление системой, которое объединяет операции БПЛА, например, назначение маршрутизации и мониторинга.

– Обеспечение надежной и высокоскоростной коммуникации между элементами информационной инфраструктуры.

– Создание аналитической системы, которая будет осуществлять комплексный анализ данных и помогать в принятии решений на основе полученной информации.

Входными параметрами являются источники информации, например, датчики и камеры, установленные на БПЛА. В процессе обработки данных могут использоваться методы машинного обучения, статистические анализы и другие техники. Обработанные данные передаются по сетям связи для передачи к целевому местоположению, где будет формироваться отчет. Содержание отчета строится на результатах анализа графиков, диаграмм и другой информации, полезной для принятия решений.

Описание каждой операции БПЛА в нотации BPMN:

1. Пул "БПЛА": обнаружение угодий (Task) реализуется БПЛА.
2. Пул "Искусственный Интеллект": искусственный интеллект (Gateway и Task), используются алгоритмы анализа данных, принимающие решения на основе обработанных фрагментов.
3. Пул "Диспетчерский Центр": реализуется ручная обработка (Task); субъект - оператор диспетчерского центра, КБД.

Общая дорожка для всех:

- Старт процесса (Start Event): инициирует весь процесс.
- Разбивка на фрагменты (Task): Общий этап разбивки изображений на фрагменты.
- Сохранение в базе данных (Task): Общий этап сохранения данных в базе.
- Искусственный интеллект (Gateway и Task): Общий этап передачи решений в случае использования искусственного интеллекта.
- База данных и обратная связь (Task): Общий этап сохранения результатов решений в базе.
- Завершение процесса (End Event): завершает весь процесс.

Такой подход обеспечивает четкую визуализацию взаимодействия между техническими компонентами (БПЛА, системы связи) и программными

модулями (алгоритмы ИИ, системы поддержки решений), что особенно важно для комплексных исследований в области автономных систем.

Процесс обработки данных БПЛА, представленный в BPMN-нотации, начинается со сбора информации (датчики, камеры), после чего сырые данные сегментируются и передаются в систему анализа. Алгоритмы ИИ обрабатывают фрагменты, выявляя значимые паттерны, а результаты либо автоматически интегрируются в отчёт, либо поступают оператору для верификации.

Ключевые этапы: фильтрация шумов, классификация объектов и оценка достоверности фиксируются в БД для последующего анализа. Такой формализованный подход минимизирует ошибки интерпретации и ускоряет принятие решений.

BPMN-нотация выступает не только как инструмент документации, но и как средство анализа, позволяя выявлять критические точки процесса, оценивать эффективность различных методов обработки данных и обосновывать архитектурные решения.

Таким образом, применение методологии описания бизнес-процессов при моделировании информационного взаимодействия БПЛА с диспетчерским центром позволяет четко структурировать потоки данных, выявляя узкие места в реальном времени. Это критично для оптимизации производительности КБД в миссиях, таких как мониторинг территорий или поисково-спасательные операции, где задержки в обработке телеметрии, координат и видео могут привести к рискам. В военных или чрезвычайных сценариях формализация процессов минимизирует нагрузку на СУБД за счет дифференциации транзакций и прогнозирования пиковых нагрузок, повышая надежность обратной связи. Предложенная структура модели процессов в нотации BPMN является полноценной частью полимодельного комплекса исследования и регурирования производительности КБД.

Схема процесса в нотации BPMN приведена на рис. 3.1.

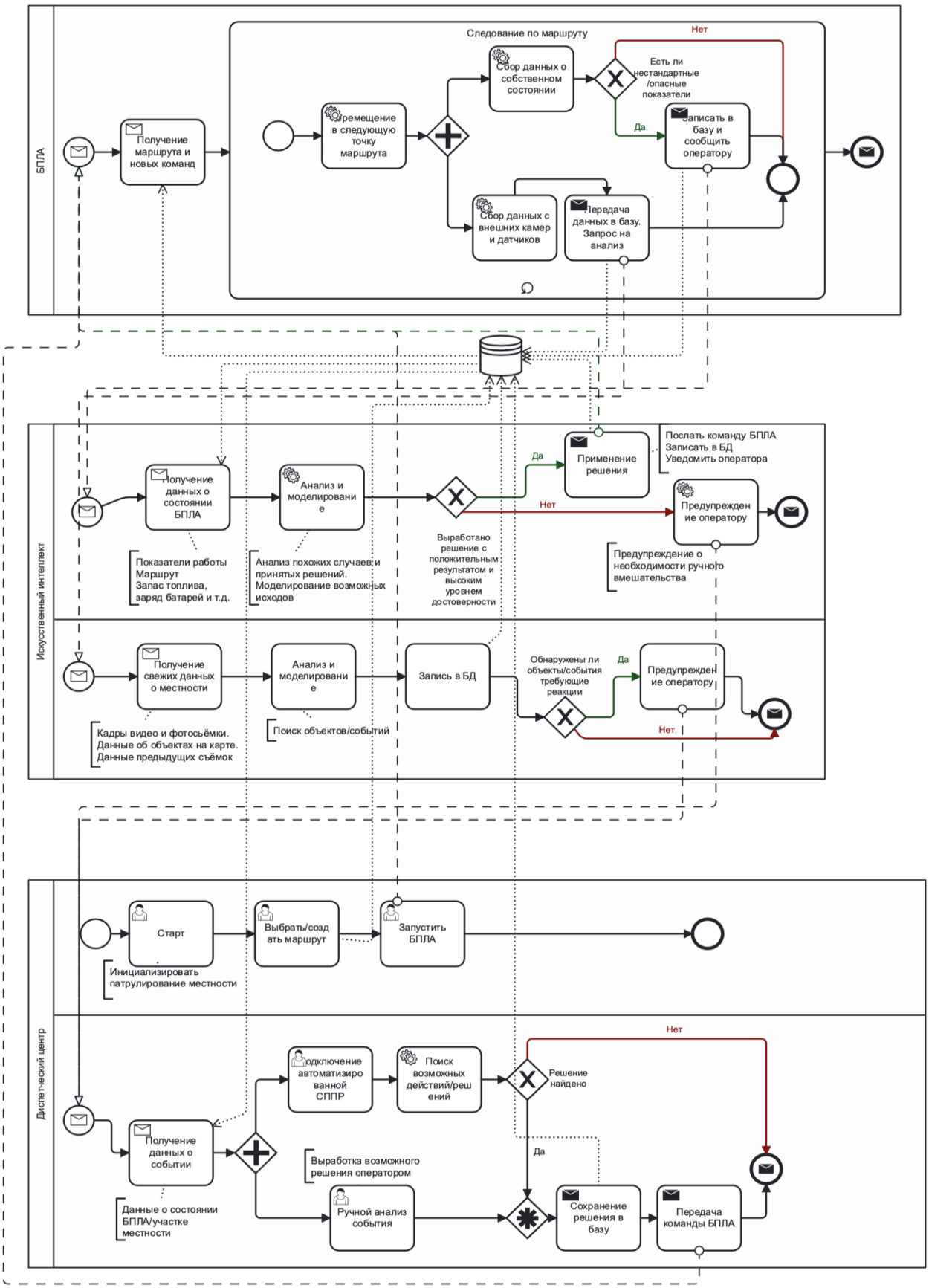


Рис. 3.1 Общая схема бизнес-процессов в нотации BPMN

.

3.3 Имитационная модель для исследования параметров КБД на примере БПЛА

Как было рассмотрено в главе 1, современные системы критического применения, в составе которых используются критические БД, обладают высокой степенью сложности и множеством взаимодействующих компонентов, что ограничивает эффективность традиционных аналитических подходов для их изучения. Тогда как имитационное моделирование предлагает методологию для исследования динамического поведения системы в контролируемой виртуальной среде. Данные возможности моделирования особенно важны при создании систем поддержки принятия решений, так как позволяют проводить прогнозирование реакции системы на различные изменения параметров и условий.

Основное преимущество имитационного моделирования заключается в его способности воспроизводить поведение системы при варьировании её параметров и структурных характеристик. Это важно для анализа как текущего состояния системы, так и для моделирования возможных сценариев её развития. Введение имитационной модели в исследовательский процесс способствует принятию более информированных решений в условиях неопределённости и способствует оптимизации управления системой.

Кроме того, использование имитационного моделирования позволяет выявить потенциальные уязвимости и ограничения системы, которые могут остаться незамеченными при применении других методов. Это открывает возможность для разработки более эффективных и надёжных стратегий управления.

Исследование разнообразных конфигураций потока заявок при анализе производительности системы на основе имитационного моделирования позволяет оптимизировать ресурсы, предсказывать производительность системы и принимать информированные решения при планировании, настройке и управлении системами [27-29].

В данном исследовании предложено классифицировать сервер СУБД в трех конфигурациях: производительный, стандартный, слабый. Это позволит варьировать (перенаправлять) возможные серии транзакций для увеличения производительности обработки данных в случаях необходимости оперативной обработки или интенсивной динамики объекта или самого БПЛА.

Рассмотрим алгоритм обработки разных типов заявок от БПЛА и влияние их на производительность сервера СУБД.

1. Беспилотный летательный аппарат генерирует поток событий:
   - Поиск новых объектов.
   - Полет до границы загруженного региона на карте.
   - Реакция на ситуационные обстоятельства, например, сильный ветер, изменения в параметрах движения (представляющие собой линию от самолета до выбора объекта).
2. Блок управления БПЛА реагирует на событие созданием запроса к базе и выполнением операций:
   - Скачивание новых фрагментов карты. Установим признак 3 (производительный сервер)
   - Уточнение параметров региона. Установим признак 3 (производительный сервер)
   - Запись данных о новых объектах. Установим признак 3 (производительный сервер)
   - Запрос на получение метеорологических данных. Установим признак 3 (производительный сервер)
   - Запрос на выполнение разведки или наблюдения. Установим признак 2 (стандартный сервер)
   - Запрос на изменение маршрута полета. Установим признак 2 (стандартный сервер)
   - Запрос на обновление программного обеспечения. Установим признак 1 (слабый сервер)

3. В ответ на поток событий, блок управления выбирает один или несколько типов запросов, которые будут отправлены к базе данных, или может решить не выполнять никаких действий, проигнорировав событие. Это решение зависит от конкретного типа запроса к базе, который должен быть выполнен в ответ на внешние события.
4. При передаче запросов возможна потеря данных из-за ненадежной сети. Для решения этой проблемы, в модель добавлены условные блоки и пути возврата "1" и "2". Если происходит ошибка при передаче запроса или ответа, запрос считается "потерянным", но это событие, то есть "запрос был потерян", возвращается в блок управления. В ответ на это событие, блок управления может сгенерировать новый запрос или принять решение проигнорировать его.

На основе предложенного алгоритма разработаем имитационную модель в среде AnyLogic, для уточнения настроек которой далее приводится описание имитируемого процесса с параметрами.

3.4 Имитационная модель в среде AnyLogic для анализа влияния категоризации транзакций запроса SQL на производительность КБД

Имитируемый процесс представляет собой модель взаимодействия источника (в форме запросов от БПЛА через диспетчерский центр (ДЦ) к СУБД, далее с целью сокращения промежуточных записей – просто от БПЛА к СУБД, преобразованных в SQL) и приемника (базы данных, сервера СУБД), на основе в том числе блоков обработки задержек, которые используются в качестве установщиков параметров имитационного цикла.

Последовательность процесса работы модели следующая.

1. Источник генерирует входной поток запросов с различными типами обработки данных (могут быть: простая выборка (в форме SELECT по простому условию поиска), s_0; сложный ресурсозатратный запрос, включающий операцию соединения отношений БД в виде запроса типа JOIN, s_{Join}; запрос с большим объемом ответа, например, blob-поля с

данными карты) и каналами обработки, s_{blob}. Каждый тип запросов характеризуется своим набором параметров:

a. трудоемкостью обработки запроса в СУБД, α;
b. объемом передаваемых данных в запросе (от БПЛА к СУБД), υ_b;
c. объемом данных в ответе (от СУБД к БПЛА), υ_i.

Сложный запрос с join характеризуется повышенной трудоемкостью обработки, а запрос с blob – повышенным временем передачи ответа по сети. Параметры разных типов генерируются с разной периодичностью: например, запрос за участками карты – периодично с установленным периодом, JOIN – через случайные промежутки времени с показательным распределением. Таким образом моделируются периодические и случайные процессы в работе БПЛА.

3.4.1 Настройка параметров имитационной модели влияния вида транзакций к СУБД на прием и усвоение данных от БПЛА

Требуется построить имитационную модель для исследования влияния вида транзакций к СУБД на прием и усвоение данных от БПЛА с целью поддержки принятия решений при управлении параметрами передачи данных от БПЛА при мониторинге объектов или явлений различной природы.

На рис. 3.2 приведены некоторые программные параметры имитационной модели, реализованной по предложенному алгоритму в среде AnyLogic.

2. Блок очереди сети обрабатывает поступающие данные с учетом следующих параметров:
    a. Признак параметра = генерация входных запросов
    b. Признак очереди = значение
    c. Признак выходного параметра = значение
    d. Метод size – возвращает количество элементов в очереди
    e. Количество интервалов - 10

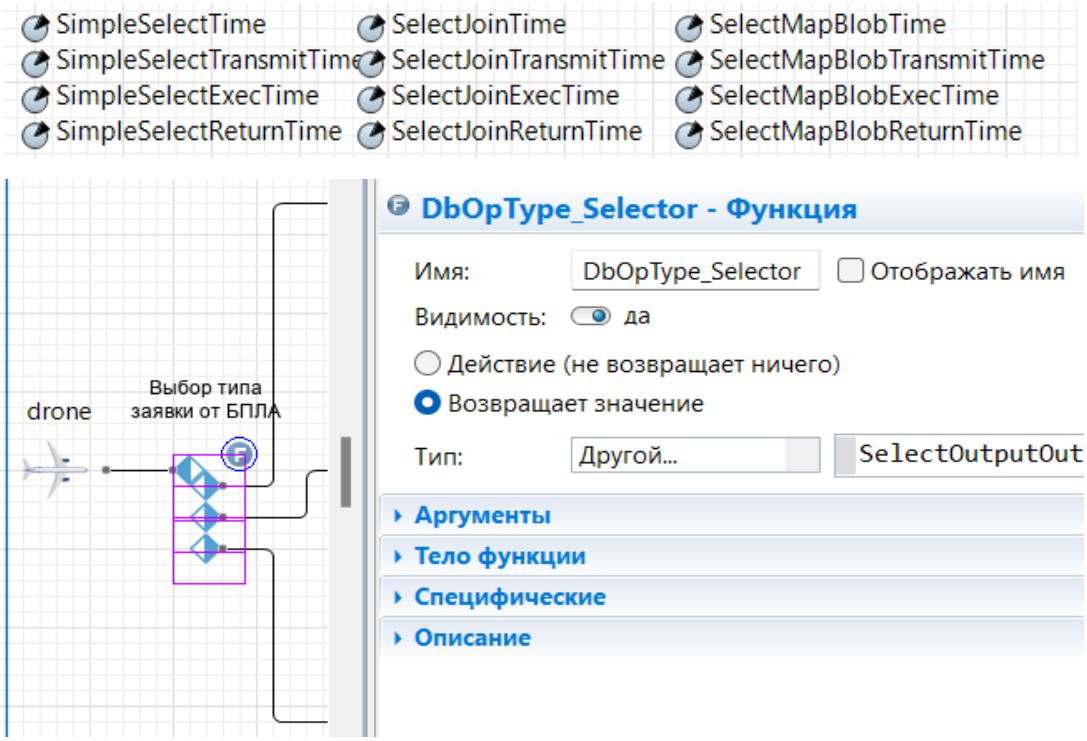


Рис. 3.2 Параметры модели при имитации взаимодействия БПЛА и типов запросов при генерировании источника входных параметров

3. Блок задержки характеризуется такими параметрами
   a. Задержка по времени = текущее время задержки
   b. Время задержки определяется как объем данных в запросе деленное на пропускную способность канала передачи данных
   c. Разные величины пропускной способности канала позволяют моделировать различные типы каналов передачи данных
4. Блок очереди базы данных обрабатывает данные, поступающие с блока задержки сети
   a. Признак входного параметра = генерация входных запросов;
   b. Признак очереди = значение;
   c. Признак выходного параметра = значение;
   d. Значение метод size – возвращает количество элементов;
   e. Количество интервалов – 10.
5. Выходные параметры поступают в блок таймера задержки базы данных

a. Задержка по времени = текущее время задержки;
b. Время задержки определяется как трудоемкость заявки деленное на производительность СУБД;
c. Разные величины производительности СУБД позволяют моделировать различные типы БД и используемого оборудования, для их исполнения.

6. Блок ответа очереди сети измеряет время обработки и возвращает данные.
7. Аналогично время задержки ответа по сети определяется как отношение объема данных в ответе к пропускной способности канала передачи данных. Блок ответа базы данных фиксирует, затраченное время обработки и возвращает данные.

Общий вид модели имитации взаимодействия БПЛА с учетом категоризации транзакций в форме запросов SQL приведен на рис. 3.3.

Для описания работы модели и результатов её выполнения введем следующие обозначения: R (Request) – запрос к БД, C (Channel) – канал, h (heavy) – «тяжелый» запрос к БД, m (map) – карта, r (random) – случайность, тогда:

– R_j – сложный запрос с join, характеризующийся повышенной трудоемкостью обработки,

– R_b – сложный запрос с blob, характеризующийся повышенным временем передачи ответа по сети,

– R_h – запрос с большим объемом ответа: сложный запрос с большим ответом и загруженным процессом обработки,

– R_m – запрос по участкам карты, который происходит дискретно с установленным периодом,

– R_r – запрос join, выполняющийся через случайные промежутки времени в общем случае с показательным распределением,

– C_{in} – источник генерирует входной поток запросов,

– C_{net} –очередь сети,

– C_{delay} – блок задержки. Формула рассчитывается следующим образом: $\tau = \frac{\mathbb{Q}}{N}$, где τ - время задержки, $\mathbb{Q}$ – объем данных в запросе, N - пропускная способность канала передачи данных,

– C_{db} — блок таймера задержки базы данных. Формула рассчитывается следующим образом: $\lambda = \frac{\Omega}{\omega}$, где λ - время задержки определяется с учетом Ω трудоемкости заявки, ω -производительность СУБД.

– C_{out} — выходные параметры поступают в блок таймера задержки базы данных.

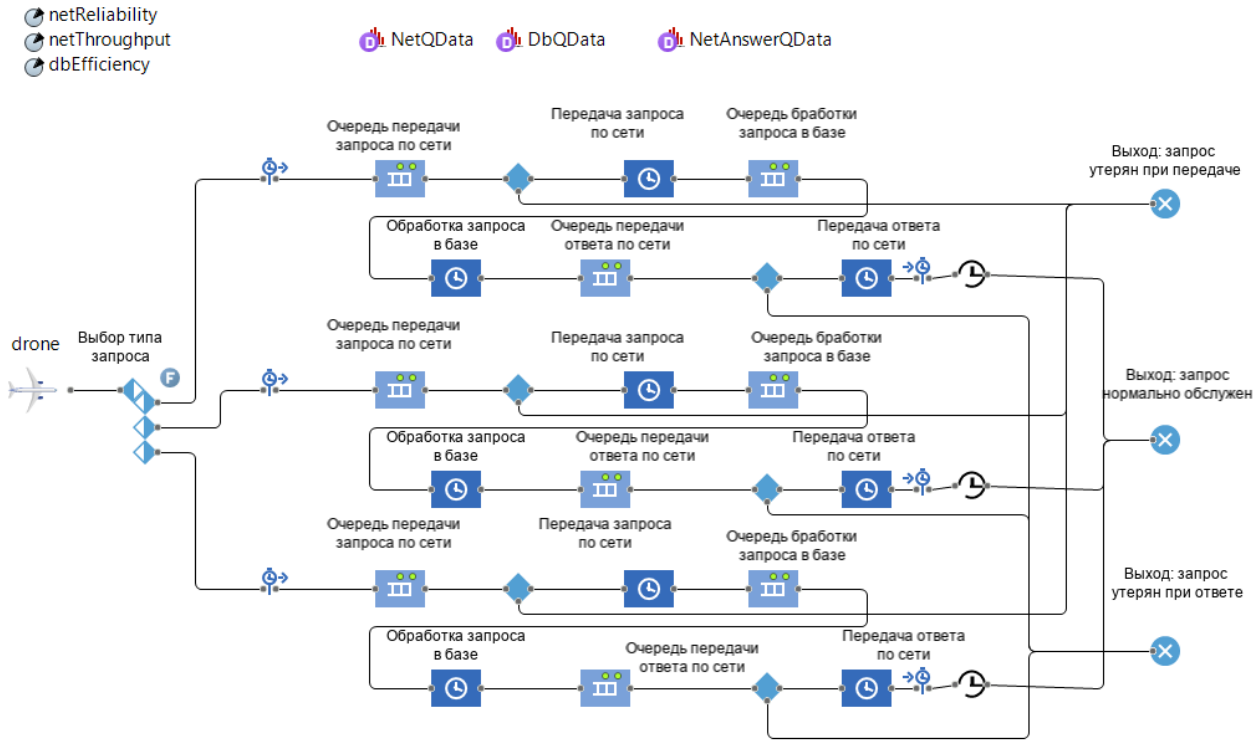


Рис. 3.3 Общий вид модели имитации взаимодействия БПЛА с приемником с учетом категоризации транзакций в форме запросов SQL

На рис. 3.4 приведен вид работающей модели имитации взаимодействия БПЛА-приемник.

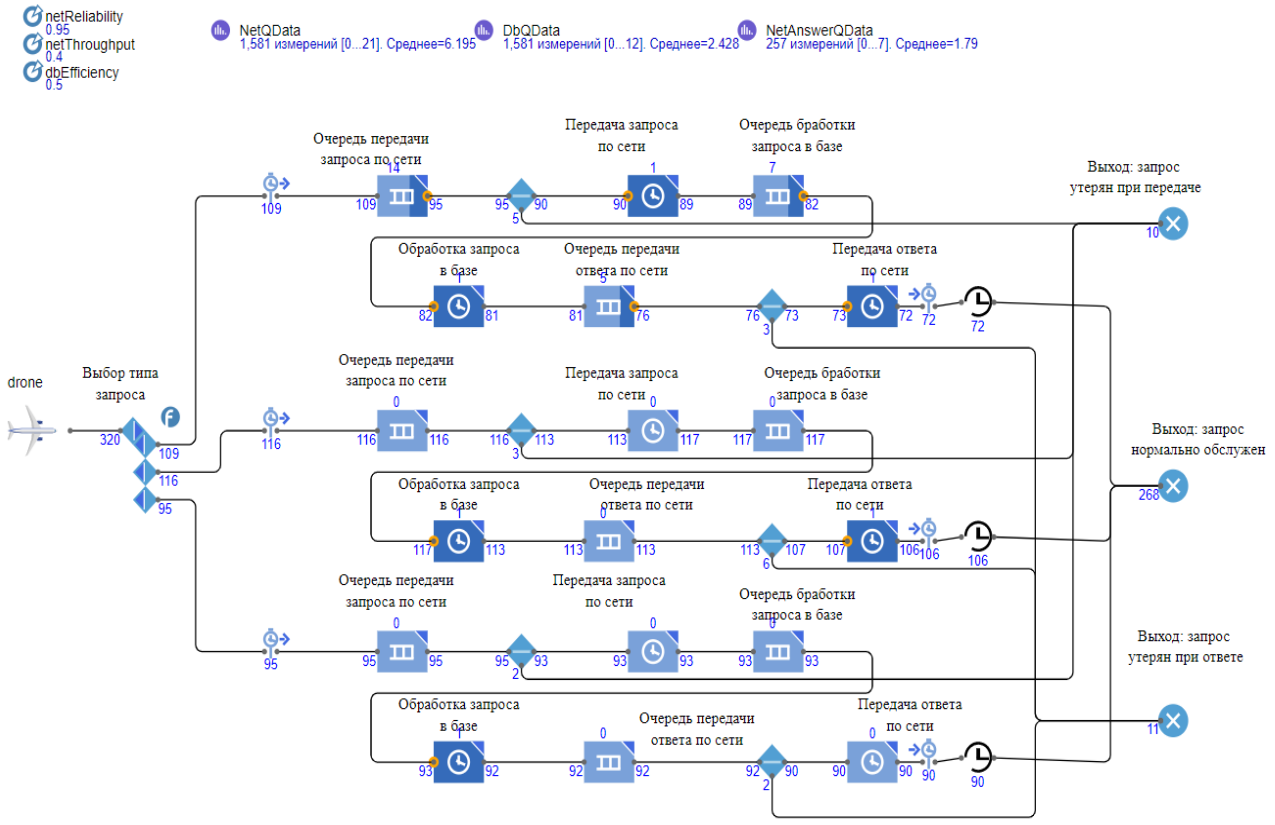


Рис. 3.4 Вид работающей модели имитации взаимодействия БПЛА-приемник с промежуточными результатами

3.4.2 Анализ результатов имитационного моделирования влияния вида транзакций к СУБД на прием и усвоение данных от БПЛА

Результаты тестирования рассматриваемой имитационной модели, представлены на рис. 3.5 и рис. 3.6. Вертикальная линия на графиках отображает среднее значение соответствующего распределения

На рис. 3.5 приведены графики распределения времён между заявками на выходе из модели, то есть после полного их обслуживания на всех этапах (передача по сети, выполнение запроса, возврат ответа по сети) по каждому из типов заявок. Можно установить, что большинство простых запросов S_0 обрабатывается в короткие временные промежутки, (рис. 3.5, а), что отражает низкую трудоемкость и высокую скорость обработки таких запросов сервером.

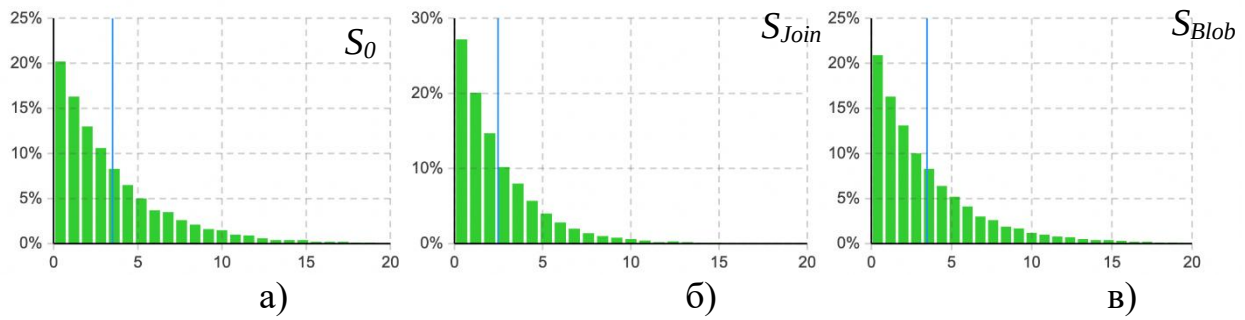


Рис. 3.5 – Результаты тестирования имитационной модели: распределение времени между обработанными заявками по типам на выходе из модели

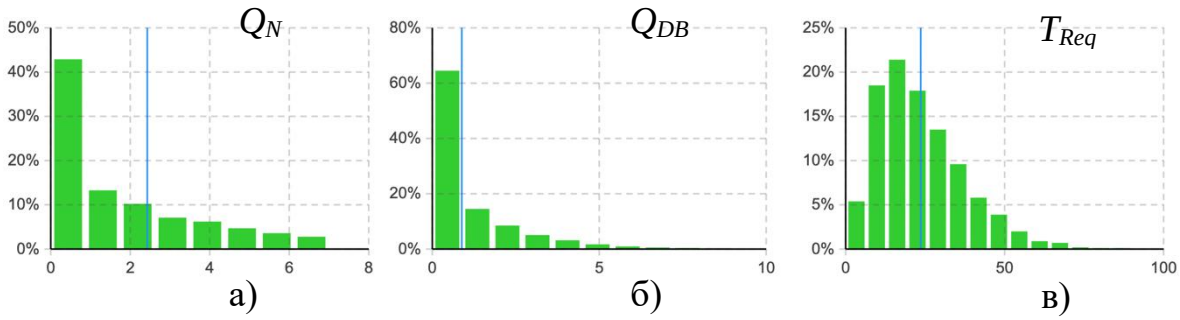


Рис. 3.6 – Результаты тестирования имитационной модели: распределения длины очереди сети, длины очереди выполнения запроса в БД и распределение полного времени обслуживания заявок

Вид распределения времен между запросами с операциями соединения (рис. 3.5, б), показывает, что, хотя многие запросы типа S_{Join} также выполняются достаточно быстро, существует широкий разброс временных интервалов, что указывает на более сложную природу таких запросов и возможное разнообразие временной нагрузки на СУБД. Результаты, приведенные на (рис. 3.5, в), отражают ситуацию, когда запросы с большим объемом данных S_{Blob} требуют более продолжительной обработки, тем не менее, большая часть их также обрабатывается в сравнительно короткий период, что свидетельствует о достаточном уровне оптимизации сервера для работы с объемными данными.

Из полученных данных видно, что при конфигурировании СУБД необходимо учитывать разнообразие типов запросов. Для улучшения

производительности системы может потребоваться адаптация ресурсов под наиболее ресурсоемкие операции, а также использование стратегий кэширования и приоритезации запросов для эффективной обработки данных от БПЛА. Эти результаты подчеркивают важность детального анализа и внимания к конфигурации серверов при разработке систем, принимающих и обрабатывающих мониторинговые данные.

На рис. 3.6 представлены результаты имитационного моделирования, отражающие три ключевых аспекта системы обработки запросов: длину очереди сети (а), длину очереди выполнения запроса в базе данных (б) и распределение полного времени обслуживания заявок (в).

а) Длина очереди сети Q_N представляет количество задач, ожидающих передачи в сети. График демонстрирует, что большая часть запросов быстро проходит через сеть без значительного ожидания, однако небольшое число запросов возникает в очереди, что указывает на потенциальные задержки в сетевой инфраструктуре.

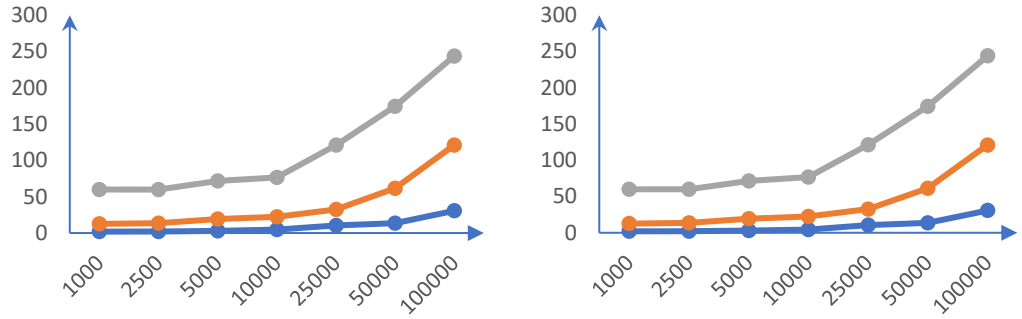
б) Длина очереди выполнения запроса в СУБД Q_{DB} отображает, как много запросов одновременно обрабатывается базой данных. Заметно, что большая доля запросов обслуживается почти мгновенно, тогда как остальные распределены равномерно. Наличие высоких значений размера очереди говорит о возможном затруднении в обработке запросов из-за высокой нагрузки на СУБД или меньшей производительности серверов.

в) Распределение времени обслуживания заявок T_{Req} показывает, сколько времени требуется от начала до конца обработки запроса. Из графика видно, что большинство запросов обрабатывается в пределах коротких временных интервалов, но также существует значительное количество заявок, требующих гораздо больше времени, что может указывать на сложные или ресурсоемкие операции в запросах или неоптимальную конфигурацию СУБД.

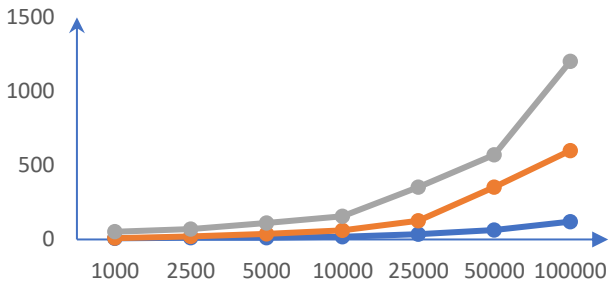
В Табл.3.1 и на рис. 3.7 приведены результаты моделирования вида запроса и канала обработки при условиях различных конфигураций серверов СУБД.

Таблица 3.1 — Результаты моделирования вида запроса и канала обработки при условиях различных конфигураций серверов СУБД

			Конфигурация 1. Производительный сервер			Конфигурация 2. Стандартный сервер			Конфигурация 3. Слабый сервер		
№	Вид запроса	Канал обработки	1000 записей	10 000 записей	100 000 записей	1000 записей	10 000 записей	100 000 записей	1000 записей	10 000 записей	100 000 записей
1	R_j	C_{in}	1,4	5,2	30,6	5	30,9	120,4	20,5	120,1	240
2	R_b	C_{in}	5,4	30,1	300,7	20,2	120,3	400,5	120,7	620,9	1800,1
3	R_h	C_{in}	5,3	20,5	120,2	10,4	60,5	600,4	30,4	180,2с	1200,3
4	R_m	C_{nq}	10,2	60,1	600,2	30,3	180,4	1800,3	60,5	600,3	3600,2
5	Rr	C_{nq}	2,3	10,2	60,2	10,1	60,6	600,8	30,3	180	1800,1
6	R_h	C_{nq}	30,3	180,1	1800,3	60,5	600,4	3600,3	180,2	1800,1	10 800,7
7	R_j	C_{delay}	0,5	2,5	25,3	2,5	12,5	125,3	12,5	62,5	625,8
8	R_b	C_{delay}	2,3	10,1	102,2	14,5	50,3	500,7	51,5	253,9	2448,2
9	R_h	C_{delay}	15,2	75	750,2	50,5	251,7	2507,5	150,1	750,2	7500,9
10	R_j	C_{db}	00.1	0,52	5,3	0,52	2,53	25,5	2,3	10,1	100,4
11	R_b	C_{db}	0,5	2,5	25,2	2,25	12,45	125,3	10,3	50,7	500,9
12	R_h	C_{db}	1,5	7,5	75,1	7,5	37,3	375,8	30,5	150,9	1500,2
13	R_j	C_{out}	0,1	1	10,5	1,9	10,6	100,5	10,2	100,4	1000,2
14	R_b	C_{out}	1,8	10,3	100,1	10,4	100,3	1000,8	100,3	1000,9	10000,3
15	R_h	C_{out}	3,2	30,0	300,2	30,1	300,6	3000,2	300,1	3000,5	30000,8



а) б)



в)

Рис. 3.7 Графики зависимостей времени обработки заявки от количества записей: а) при сложном запросе с JOIN, б) при сложном запросе с BLOB, в) при сложном запросе с большим ответом (блок таймера задержки БД)

На рис. 3.7 представлены графики зависимостей времени обработки запросов от их количества для различных типов запросов к базе данных: (а) сложный запрос с JOIN, (б) сложный запрос с BLOB и (в) сложный запрос с большим ответом.

График (а) демонстрирует зависимость времени обработки запросов с JOIN от количества записей. Наблюдается линейный рост времени обработки с увеличением числа записей до определенного предела, после чего время обработки возрастает более резко. Это указывает на то, что при небольшом и среднем количестве записей СУБД эффективно обрабатывает запросы, но при большом объеме данных возникают проблемы со скоростью обработки из-за увеличения сложности операций соединения.

На графике (б) видно, что время обработки запросов с извлечением больших объектов данных (BLOB) также увеличивается пропорционально числу записей, однако рост значительно резче, чем при обработке запросов с JOIN. Что может значить, что операции с большими объемами данных заметно увеличивают время обработки запроса, что особенно критично для систем, требующих оперативной обработки информации.

График (в) показывает рост времени обработки заявок при увеличении объема данных в ответе – задержка в базе данных становится более ощутимой с увеличением количества записей. Это подчеркивает важность оптимизации алгоритмов для обработки "тяжелых" запросов, чтобы обеспечить своевременную обработку и возврат данных.

В целом, графики на рис. 3.8 подтверждают, что с увеличением количества записей в запросе значительно увеличивается время обработки, и это особенно заметно для запросов, требующих сложных операций обработки данных. Данные результаты позволяют сделать вывод о необходимости тщательно планировать ресурсную составляющую СУБД и возможно, использовать специализированные или оптимизированные индексы для ускорения операций поиска и соединения, а также эффективного масштабирования при работе с большими объемами данных [112].

3.5 Выводы по главе

В разделе представлен подход к построению и тестированию многофакторной модели на основе применения имитации. Полученные результаты моделирования учитывают как категоризацию запросов, которые создают неоднородную нагрузку на разные участки и стадии передачи данных, так и виды обработки запросов: сложный запрос с join, сложный запрос с blob, характеризующийся повышенным временем передачи ответа по сети, запрос с большим объемом ответа, сложный запрос с большим ответом и загруженным процессом обработки, запрос за участками карты, происходит периодично с установленным периодом, запрос join, выполняющийся через случайные промежутки времени с показательным распределением.

Исходя из представленных данных, можно сделать следующие выводы и предложения по улучшению системы:

1. Система сетевого соединения в большинстве случаев не является узким звеном, однако следует обращать внимание на единичные случаи большого накопления очереди запросов.

2. СУБД может эффективно справляться с большинством запросов, но наличие очереди в базе данных требует дополнительного рассмотрения, возможно, в виде оптимизации исполнения запросов или масштабирования ресурсов серверов.

3. Общее время обслуживания заявок показывает, что система может быть оптимизирована для предотвращения длительного ожидания ответа для заявок с большой продолжительностью обработки, возможно, путем оптимизации алгоритмов запросов или улучшения аппаратуры.

4. Принимая во внимание распределение времени обслуживания, важно проанализировать специфику запросов, которые занимают много времени, и выработать стратегии для их оптимизации.

Таким образом, анализ проведённых экспериментов подтвердил эффективность выбранной методики имитационного моделирования и

обозначил направления дальнейшей оптимизации и адаптации систем управления запросами в условиях динамично изменяющейся нагрузки, особенно при использовании БПЛА для мониторинга и сбора данных в реальном времени.

Необходимо учитывать динамику нагрузки на СУБД и применять адаптивные методы работы с запросами для оптимизации производительности, в том числе масштабирование серверов и использование более продвинутых технологий кэширования, индексации и распределенных вычислений.

4. ГИБРИДНЫЙ ЧИСЛЕННЫЙ МЕТОД НА ОСНОВЕ ИНТЕГРАЦИИ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ И МЕТОДА ВЕТВЕЙ И ГРАНИЦ

4.1 Гибридные методы и интеллектуальные СППР

В рамках исследования для решения задач оптимизации производительности критических баз данных в условиях высокой динамической нагрузки и стохастических возмущений был разработан гибридный численный метод. Основу метода составляет классический алгоритм ветвей и границ, который обеспечивает систематический глобальный поиск оптимальных решений в многомерном пространстве параметров путём последовательного разбиения области поиска и отсечения заведомо неоптимальных подпространств. Этот подход гарантирует нахождение точного решения для задач дискретной оптимизации, но может требовать значительных вычислительных затрат при высокой размерности задачи. Для преодоления этого ограничения метод был дополнен нейросетевой моделью прогнозирования, которая позволяет оценивать производительность системы на основе ключевых параметров: выполнения запросов, доступные ресурсы, сложность схемы базы данных, интенсивность поступающих запросов и объём хранимых данных. Интеграция этих двух компонентов реализована через взвешенную комбинированную оценку, где на каждом шаге алгоритма производится параллельная оценка решений с использованием очного вычисления целевой функции и нейросетевого прогнозирования.

Перспективным направлением является гибридный подход, объединяющий преимущества классических численных методов и искусственного интеллекта. Далее будет показано, как комбинация алгоритма ветвей и границ с нейронными сетями позволяет достичь баланса между точностью, скоростью и адаптивностью, что особенно важно для систем, работающих в условиях жестких временных ограничений. Особое внимание будет уделено экспериментальной проверке предложенных решений.

1.2 Постановка задачи и архитектура метода

Разработан и реализован гибридный численный метод оптимизации производительности КБД, сочетающий алгоритм ветвей и границ с нейросетевым прогнозированием. Метод направлен на решение многопараметрических задач оптимизации конфигурации КБД в условиях высокой нагрузки и стохастических возмущений.

Архитектура метода включает два взаимодополняющих компонента. Алгоритм ветвей и границ осуществляет систематический глобальный поиск в пространстве параметров, последовательно разделяя его на подпространства и исключая заведомо неоптимальные области. Нейросетевая модель, построена на основе следующей архитектуры: полносвязная сеть (многослойный перспетрон) с тремя скрытыми слоями (128, 64, 32 нейронов) с активацией ReLU, нормализацией пакетов (BatchNorm) и регуляризацией Dropout, завершающаяся одиночным выходом с функцией сигмоида для ограничения результата в диапазоне.

Архитектура с BatchNorm и Dropout оправдана для стабильности обучения и обобщающей способности модели, предотвращая переобучение на синтетических данных. Слои уменьшающегося размера сосредотачивают обучение на наиболее значимых признаках, а сигмоида на выходе соответствует необходимости прогнозировать производительность в масштабе от 0 до 1. Таким образом, архитектура нейросети специально адаптирована для нелинейной регрессии с ограничениями и интегрирована в гибридный алгоритм оптимизации с ветвями и границами для повышения эффективности решения сложных многокритериальных задач. Такая архитектура позволяет эффективно моделировать сложные нелинейные зависимости параметров и производительности в задачах с многомерным входом и обеспечивает прогнозирование производительности БД по пяти ключевым параметрам: времени выполнения запросов, объему доступных ресурсов, сложности структуры БД, интенсивности поступающих запросов и объему хранимых данных.

Интеграционный механизм реализован через комбинированную оценку, где на каждом шаге алгоритма производится параллельная оценка центров подпространств с использованием как реального вычисления целевой функции с учетом штрафов за нарушение критических ограничений, так и нейросетевого прогнозирования. Итоговая оценка формируется как взвешенная сумма с коэффициентом α = 0.7 в пользу реальных вычислений [29-32].

Критические ограничения системы формализованы следующим образом:

Время выполнения запросов: T ∈ [0.1, 0.9]; минимальные ресурсы: R ≥ 0.3; Критическая сложность схемы: C ≤ 0.8; Критический объем данных: N ≤ 0.85.

Результаты апробации метода показали высокую эффективность, так, сокращение функции потерь получено на 78.34% за 10 итераций; точность прогнозирования (MSE = 0.003915, R^2 = 0.7476); заметна стабильная сходимость алгоритма, а также реализовано автоматическое соблюдение критических ограничений.

Математическая формализация гибридного метода ветвей и границ с нейросетевым усилением для оптимизации производительности КБД.

Рассматривается задача оптимизации в многомерном параметрическом пространстве $X = [0,1]^5$, где оптимизируются параметры, критически влияющие на производительность баз данных:

x_1 – время выполнения запросов (Time To Execute, TT),

x_2 – объем памяти (RAM, R),

x_3 – сложность схемы базы данных (Complexity, C),

x_4 – интенсивность запросов (Intensity, I),

x_5 – размер базы данных (Number of records, N).

Целевая функция минимизируется и включает компоненту ошибки прогнозирования и штрафы за нарушения ограничений:

$$F(x) = \mathrm{MSE}(x) + \sum_{i=1}^{k} \lambda_i \cdot \max(0, g_i(x))^2, \quad (10)$$

где $\mathrm{MSE}(x) = \mathbb{E}[(y - \hat{y}(x))^2]$ – среднеквадратичная ошибка прогноза времени выполнения запроса, функции ограничений $g_i(x)$ задают пределы:

$$g_1(x) = x_1 - T_{\max} \le 0, g_2(x) = R_{\min} - x_2 \le 0, g_3(x) = x_3 - C_{\text{crit}} \le 0, g_4(x) = x_5 - N_{\text{crit}} \le 0.$$

Прогнозирование выполняется с помощью нейронного сети с тремя скрытыми слоями, активацией ReLU, пакетной нормализацией (BatchNorm) и регуляризацией Dropout(p=0.3). Формально,

$$f_{\text{NN}}(x) = \sigma(W_3 \cdot \text{ReLU}(W_2 \cdot \text{ReLU}(W_1 \cdot \text{BN}(x) + b_1) + b_2) + b_3),$$

где $W_1 \in \mathbb{R}^{128\times5}$, $W_2 \in \mathbb{R}^{64\times128}$, $W_3 \in \mathbb{R}^{1\times64}$, а σ – сигмоида для нормализации выхода в диапазон $[0,1]$.

Функция потерь вычисляется следующим образом:

$$L_{\text{NN}} = \frac{1}{m}\sum_{i=1}^{m} (y_i - f_{\text{NN}}(x_i))^2 + \lambda\|\theta\|_2^2, \quad (11)$$

где θ — параметры сети, λ — коэффициент регуляризации.

Гибридный метод ветвей и границ сформирован на основе алгоритма, который хранит набор подпространств $Q = \{(L^{(i)}, U^{(i)})\}_{i=1}^{k}$ с нижними и верхними границами в $\mathbb{R}^5$. Правило ветвления выбирает координату с максимальной длиной интервала:

$$j^* = \arg\max_j (u_j - l_j),$$

и разбивает выбранный интервал пополам:

$$v = \frac{l_{j^*} + u_{j^*}}{2},$$

создавая два новых подпространства.

Оценка центра подпространства $c = (L + U)/2$ комбинирует реальные вычисления $F(c)$ и нейросетевую предсказательную модель:

$$E(c) = \alpha F(c) + (1 - \alpha) f_{\text{NN}}(c),$$

где $\alpha = 0.7$ — коэффициент доверия вычислениям функции.

Правило отсечения формулируется следующим образом: если $E(c) > F_{\text{best}} - \varepsilon$, то подпространство отбрасывается, где $\varepsilon = 10^{-4}$ – допуск оптимальности.

Оценка сложности в общем случае при выполнении условий:

- равномерная сходимость f_{NN} к F на компакте X,
- коэффициент доверия $\alpha > 0.5$,
- выпуклость критических ограничений,

алгоритм сходится к глобальному минимуму с вероятностью 1 при числе итераций, стремящемся к бесконечности.

Сложность в худшем случае равна $O(2^d \cdot T_{\text{NN}})$, с T_{NN} – временем вычисления нейросетевой оценки. Практически благодаря отсечению подпространств сложность снижается до $O(k \cdot d \cdot T_{\text{NN}})$ с маленьким $k \ll 2^d$.

4.3 Программная реализация на PYTHON с интеграцией метода ветвей и границ и нейросети

В рамках диссертационного исследования была разработана комплексная программная реализация гибридного метода оптимизации производительности критических баз данных на языке Python с использованием фреймворка PyTorch. Разработанный программный комплекс интегрирует в единую архитектуру нейросетевое прогнозирование и точный алгоритм ветвей и границ, что позволяет эффективно решать многопараметрические задачи оптимизации конфигурации КБД в условиях высоких нагрузок и стохастических возмущений.

Одной из составных частей программной реализации составляет модуль генерации синтетических данных, который имитирует поведение производительности КБД с учетом пяти ключевых параметров: времени выполнения запросов, объема доступной памяти, сложности структуры базы данных, интенсивности поступающих запросов и объема хранимых данных. Сгенерированные данные используются для обучения улучшенной нейросетевой модели, построенной на основе многослойного персептрона с

архитектурой [128, 64, 32] нейронов и механизмами регуляризации, включая Batch Normalization и Dropout для предотвращения переобучения.

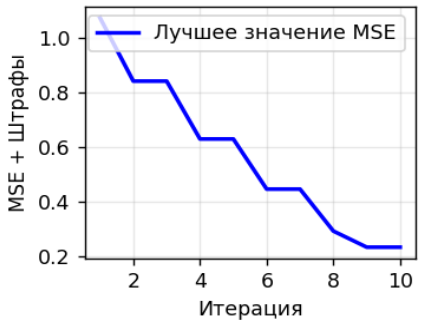
Центральным компонентом системы является модуль гибридной оптимизации, в котором реализовано эффективное взаимодействие нейросетевого прогнозирования и алгоритма ветвей и границ. Нейросетевая модель обеспечивает быструю оценку перспективных областей пространства параметров, в то время как адаптированный алгоритм ветвей и границ осуществляет точный поиск оптимальных решений в суженной области. Особенностью реализации является комбинированная оценка с весовым коэффициентом $\alpha = 0.7$ в пользу реальных вычислений, что обеспечивает баланс между скоростью и точностью оптимизации.

Важным аспектом реализации стала специализированная функция потерь, которая учитывает не только точность предсказания производительности, но и включает штрафные санкции за нарушение критических ограничений системы. Это позволяет гарантировать, что найденные оптимальные параметры конфигурации удовлетворяют всем техническим требованиям эксплуатации КБД, включая ограничения по времени выполнения запросов, минимальным ресурсам памяти, критической сложности схемы базы данных и допустимому объему хранимых данных.

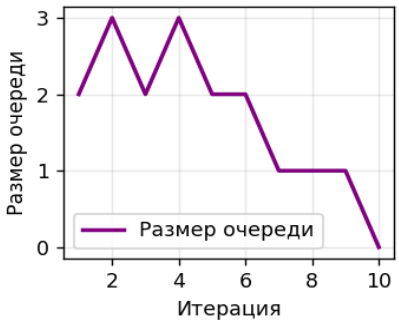
Для комплексного анализа результатов оптимизации в системе реализован расширенный модуль визуализации, который предоставляет шесть различных типов графиков, включая динамику улучшения целевой функции, эволюцию параметров системы, сравнение реальных и предсказанных значений, распределение ошибок прогнозирования, а также корреляционный анализ параметров. Это позволяет проводить детальный анализ процесса оптимизации и обосновывать принимаемые решения.

Апробация программной реализации продемонстрировала высокую эффективность предложенного подхода. В процессе тестирования удалось достичь сокращения функции потерь на 78.34% за 10 итераций при точности прогнозирования $R^2 = 0.7476$ и $MSE = 0.003915$. Найденные оптимальные

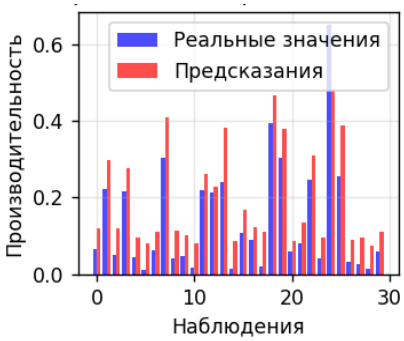
параметры системы [0.125, 0.250, 0.250, 0.250, 0.250] обеспечивают соблюдение всех критических ограничений производительности. На рис.4.1. приведены результаты численного моделирования.



а)

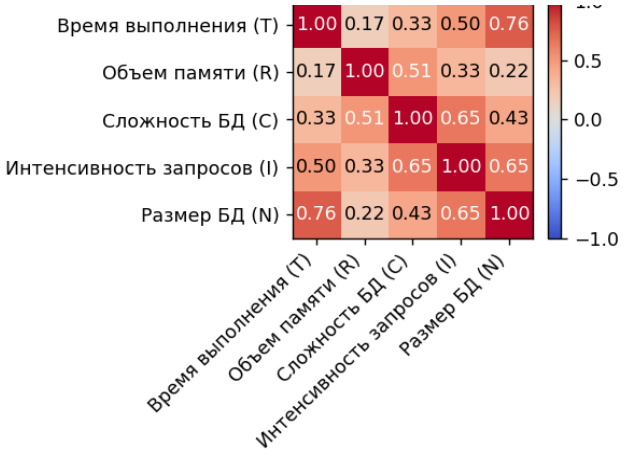


б)



в)

г)



д)

Рисунок 4.1 – Рисунок 9 - Визуализация результатов работы метода: а) динамика улучшения производительности; б) динамика очереди; в) сравнение реальных и предсказанных значений; г) распределение ошибок предсказаний; д) корреляция параметров БД

Рис.9, б) визуализирует динамику размера очереди подпространств в алгоритме ветвей и границ, позволяя анализировать вычислительную сложность процесса оптимизации. Максимальный размер очереди составил 3 подпространства при финальном значении 0, что указывает на полное исследование пространства параметров; рис.9 в) обеспечивает сравнение реальных и предсказанных значений производительности КБД на тестовой выборке, визуализируя точность нейросетевой модели; гистограмма распределения ошибок предсказания, рис.9, г) показывает близкое к нормальному распределение с математическим ожиданием, близким к нулю, что подтверждает отсутствие систематической ошибки в прогнозах. Рис.9, д) отражает корреляцию между параметрами КБД, с учетом взаимосвязи между оптимизируемыми переменными.

Практическая значимость разработанного программного комплекса заключается в возможности его использования для автоматизированного анализа метрик производительности КБД, выявления узких мест в конфигурации системы и формирования обоснованных рекомендаций по повышению производительности. Реализация демонстрирует высокую эффективность для задач средней размерности, характерных для управления производительностью критических баз данных, и обладает необходимой адаптивностью к изменяющимся условиям эксплуатации, что подтверждает перспективность предложенного подхода для применения в реальных промышленных системах [29,33].

4.4 Система поддержки принятия решений по повышению производительности КБД на основе гибридного численного метода

Предложенный в диссертации каркас СППР представляет собой комплекс программ, который помогает анализировать данные, моделировать возможные сценарии и выдавать обоснованные рекомендации для принятия оптимальных

решений. В контексте работы с БПЛА и высоконагруженными БД такая система решает три ключевые задачи:

1) Обработка и анализ данных – сбор и интерпретация информации от БПЛА (координаты, состояние оборудования, нагрузка на каналы связи).
2) Оптимизация производительности – выбор наиболее эффективных SQL-запросов (JOIN, INNER, FULL и др.) для ускорения работы БД.
3) Формирование рекомендаций – предложения по улучшению работы системы, например, увеличение пропускной способности связи или изменение маршрута БПЛА, рис.4.2.

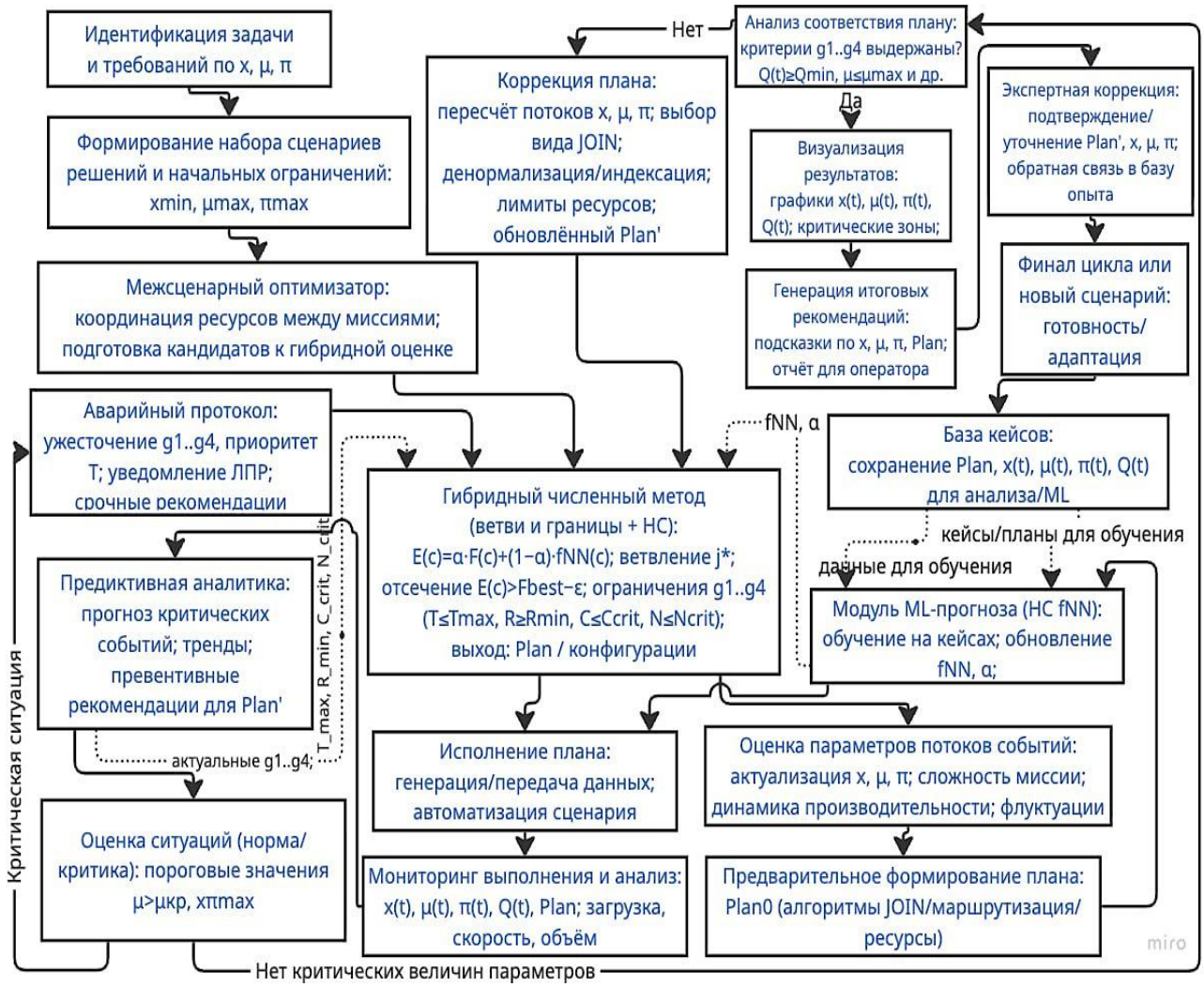


Рис. 4.2 Схема алгоритма работы системы поддержки принятия решений по повышению производительности высоконагруженных КБД на основе оптимизации запросов

При этом для обработки этих данных используются специализированные базы данных, производительность которых напрямую влияет на эффективность всей системы. Как показано в исследовании, выбор оптимальных методов обработки запросов (JOIN, INNER, FULL и других) позволяет существенно сократить время отклика системы и повысить ее общую надежность.

Предложенная структура СППР сочетает в себе:

1. Методы аналитической обработки больших массивов данных;
2. Алгоритмы оптимизации запросов к базам данных;
3. Механизмы визуализации информации для оператора;
4. Систему генерации рекомендаций по улучшению работы.

Особенностью предлагаемого подхода является его адаптивность - система может быть настроена для работы в различных областях, начиная от мониторинга чрезвычайных ситуаций (как в случае с БПЛА-пожарными) до задач логистики и сельского хозяйства. При этом сохраняется общая архитектура решения, что упрощает его внедрение и масштабирование. Предложенная СППР содержит подсистемы аналитики и планирования возможных решений на основе выбора вида запроса join; оценивание требуемых ресурсов, требуемых для решения текущей задачи, а также изменения сложности при реализации денормализации с целью повышения производительности БД: $\tau_j; \mu_{z_j}; r_{z_j}$. Оценивание производительности БД при решении текущей производственной задачи реализуется в СППР в соответствующей подсистеме, в которой также оценивается и качество решения текущей задачи, то есть соответствие времен.

4.5 Общая структура и функционал комплекса программ в рамках системы поддержки принятия решений по регулированию нагрузок КБД на примере взаимодействия беспилотного аппарата и диспетчерского центра

В разделе рассматривается структура и принцип работы системы поддержки принятия решений с элементами рекомендательной системы по оценке производительности критических БД на примере взаимодействия с БПЛА-пожарным. В программном комплексе создается реалистичная модель производительности БД, которая реагирует на различные решения; учитывает случайные колебания и уменьшающуюся отдачу от решений; реализован прототип рекомендательной подсистемы; выдаются практические рекомендации и различные виды визуализации решений.

Программа моделирует работу беспилотного аппарата в различных миссиях, где требуется мониторинг и оперативное реагирование.

Функционал прототипа рекомендательной системы комплекс программ следующий:

- генерация персонализированных рекомендаций для каждого сценария;
- советы по оптимизации работы БПЛА;
- предложения по улучшению оборудования;

Примеры практических советов:

1. При перегрузке связи - рекомендация увеличить пропускную способность;
2. При низкой производительности - совет добавить БПЛА;
3. Для сложных миссий - использование специализированных моделей.

На рис.4.3 приведена структура модулей системы поддержки принятия решений по повышению производительности высоконагруженных БД на основе оптимизации запросов.

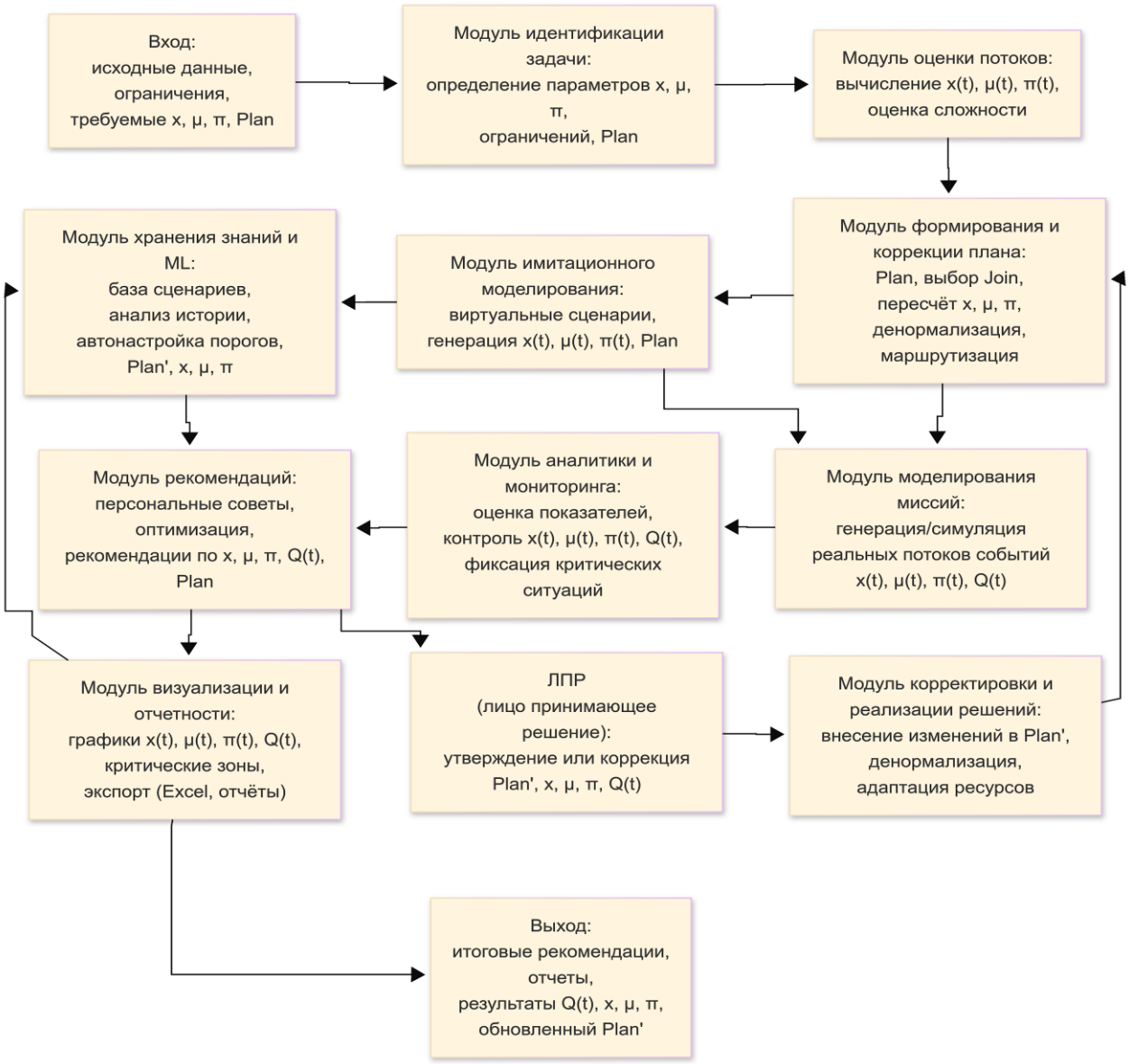


Рис. 4.3 Укрупненная структура модулей комплекса программ в рамках системы поддержки принятия решений по повышению производительности КБД

В программном комплексе предусмотрена визуализация решений в виде графиков, например, реализованы следующие графики для анализа качества связи и поддержки решений на их основе: отображается скорость передачи данных (МБ/мин) с выделением средней скорости; иллюстрируется

эффективность использования канала (% нагрузки на МБ данных); оценивается распределение типов принятых решений.

4.6 Выводы по разделу

Разработанный гибридный численный метод, сочетающий алгоритм ветвей и границ с нейросетевым прогнозированием, продемонстрировал оптимизацию производительности критических баз данных. Экспериментальная апробация подтвердила значительное сокращение функции потерь и стабильную сходимость алгоритма при соблюдении критических ограничений. Программная реализация на Python с использованием PyTorch обеспечила интеграцию методов машинного обучения и оптимизации, что позволило решать многопараметрические задачи в условиях высокой нагрузки. Предложенная система поддержки принятия решений на основе гибридного метода показала свою адаптивность и практическую значимость для повышения производительности баз данных.

5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ ПО ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ КРИТИЧЕСКИХ БАЗ ДАННЫХ

5.1 Анализ экспериментов по производительности высоконагруженных БД для различных типов операций JOIN

Рассмотрим пример базы данных, применяемой для управления и анализа данных, полученных БПЛА. БД может использоваться в системах мониторинга, картографии, доставки грузов, где критически важно отслеживать местоположение и действия БПЛА в реальном времени.

Данная БД предназначена для обработки пространственных данных, управления маршрутами и анализа действий БПЛА на локальной машине.

В примере рис.5.1 представлена схема базы, где необходимо решить задачу перемещения и сбора данных БПЛА по картографии.

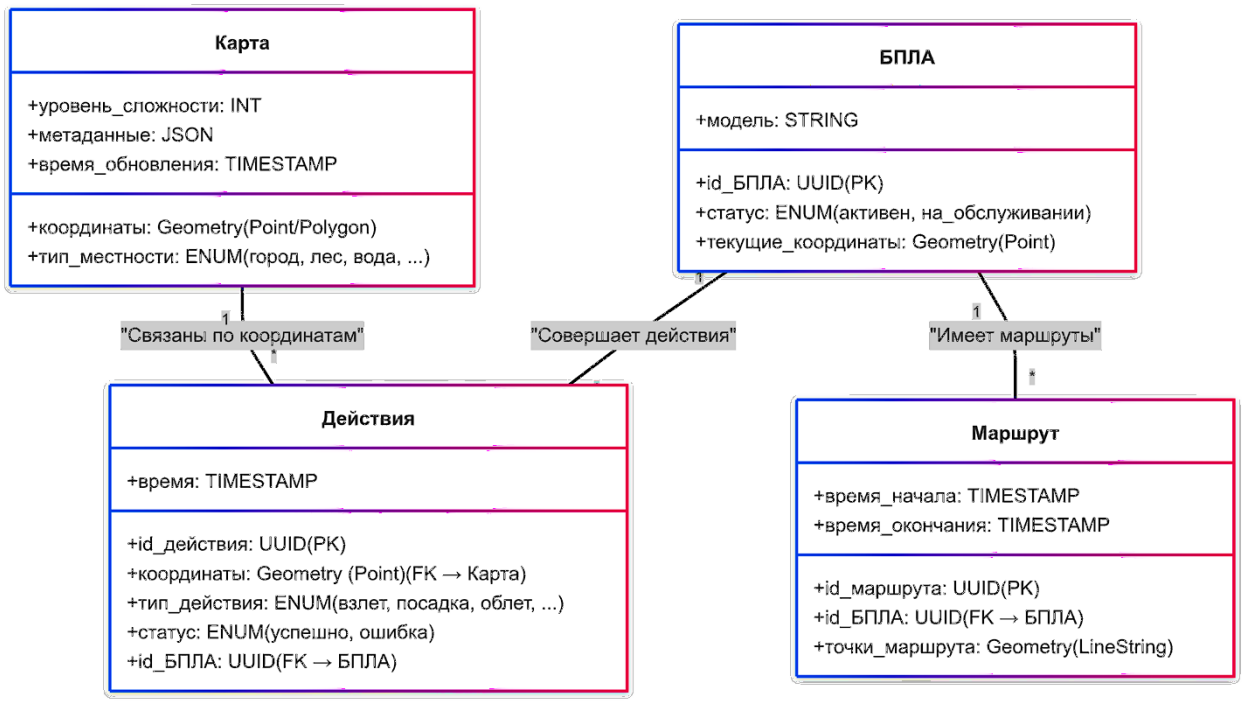


Рис. 5.1 Структура данных

В рассматриваемой БД операция left join (или left outer join) выбирает все строки из таблицы "карта" и соответствующие строки из таблицы "действия". Если для строки из таблицы "карта" не найдется соответствующая строка в таблице "действия", то значения в столбцах таблицы "действия" будут заполнены значением NULL.

Пример SQL-запроса для left join:

```
SELECT *
FROM карта
LEFT JOIN действия
ON карта.координаты = действия.координаты;
```

Операция right join (или right outer join) выбирает все строки из таблицы "действия" и соответствующие строки из таблицы "карта". Если для строки из таблицы "действия" не найдется соответствующая строка в таблице "карта", то значения в столбцах таблицы "карта" будут заполнены значением NULL.

Пример SQL-запроса для right join:

```
SELECT *
FROM карта
RIGHT JOIN действия
ON карта.координаты = действия.координаты;
```

Пример архитектуры

```
CREATE TABLE Карта (
    id SERIAL PRIMARY KEY,
    координаты GEOMETRY(POINT, 4326),  -- WGS-84
    тип_местности VARCHAR(20),
    is_запретная_зона BOOLEAN
);
CREATE TABLE Действия_БПЛА (
    id UUID PRIMARY KEY,
    координаты GEOMETRY(POINT, 4326),
    время TIMESTAMP,
```

тип_действия VARCHAR(50),

FOREIGN KEY (координаты) REFERENCES Карта(координаты)

);

При использовании данных, полученных с БПЛА, где задействованы координаты движения, операция JOIN может стать более сложной и низкоскоростной из-за большого объема данных. Например, для отслеживания движения БПЛА может потребоваться соединение нескольких таблиц, содержащих информацию о его местоположении, сложных участках, остановках и других параметрах.

Для решения проблемы снижения размерности промежуточных объемов данных, полученных при соединении различных участков карт, различных элементов слоев карт для требуемых координат БПЛА, могут быть использованы различные подходы: разделение данных на более мелкие наборы, использование индексации и кэширования, а также оптимизация запросов SQL. Например, использование индексации для столбцов соединения может значительно повысить производительность операции JOIN; или использование распределенной базы данных также может помочь повысить общую производительность системы за счет распределения нагрузки между несколькими узлами.

Рассмотрим пути повышения производительности операций JOIN в случае высоконагруженных БД и различные методы оптимизации, исходя из конкретных требований и характеристик данных.

5.2 Анализ экспериментов по примеру различных типов SQL запросов

Для построения СППР рассмотрим схему формирования сложных участков в применении к экологическому мониторингу лавандового поля, город Бахчисарай. Спутниковая карта Лавандового поля г. Бахчисарай с возможной примерной компоновкой координатного передвижения и условных обозначений приведена на рис.5.2. Расположение пунктов в данной зоне является условной проектной конструкцией в связи с незаконченными полевыми работами.

Преимуществом использования беспилотных технологий в конкретном случае являются следующие возможности БПЛА:

– оперативно распознавать препятствия и сложные участки дорог, остановок, ям при низком маневрировании;
– оперативный контроль неблагоприятных погодных условий, например, сильного внезапного дождя, интенсивного тумана, снега;
– оперативная визуализация непредвиденных остановок; сложных земельных участков и т. п. [2].

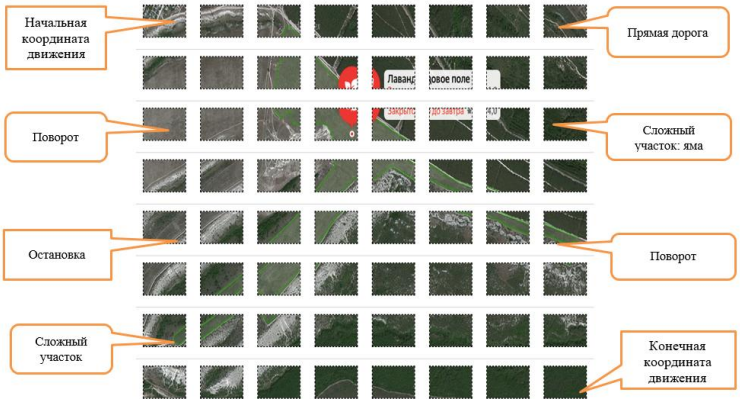


Рис. 5.2 Спутниковая карта «Лавандовое поле» г. Бахчисарай

5.3 Моделирование системы информационного обеспечения парково–рекреационной зоны

Рассмотрим возможную схему формирования транспортного обслуживания в парково – рекреационной зоне «Парк Победы», город Севастополь, на основе беспилотных технологий. Спутниковая карта ПРЗ Парка Победы г. Севастополь с возможной примерной компоновкой транспортных узлов и ДЦ приведена на рис. 5.3 Расположение пунктов транспортного обслуживания в данной зоне является условной в связи с незаконченными проектными работами по реконструкции Парка Победы на момент проведения исследования.

Как уже было сказано в главе 1, преимуществом использования беспилотных технологий в этой ПРЗ является возможность БЛА оперативного распознавания препятствий до момента достижения этого препятствия БТС;

возможность реагировать на неблагоприятные погодные условия, например, сильный дождь, интенсивный туман, снег; непредвиденные остановки; требования пассажиров и т.п.

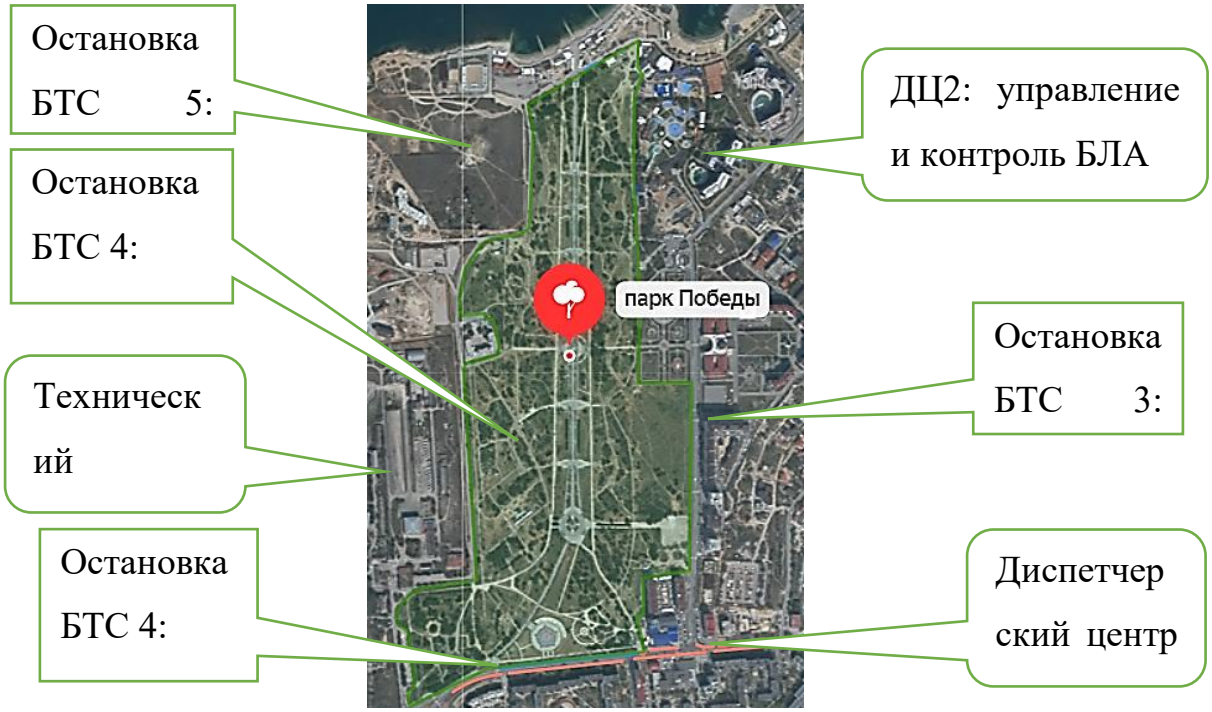


Рис. 5.3 Спутниковая карта ПРЗ «Парк Победы» г. Севастополь

5.4 Методика проведения экспериментов

При реализации моделирования процесса взаимодействия системы управления БПЛА с БД, выбранный фрагмент карты был разбит на прямоугольные фрагменты равного размера, рис. 5.4.

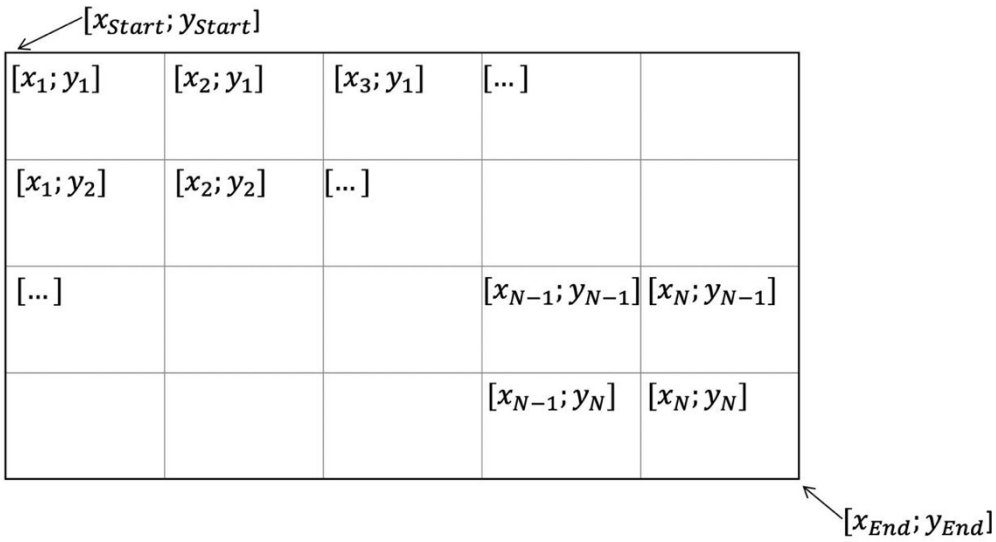


Рис. 5.4 Условное обозначение координат в схеме движений

Каждый фрагмент характеризуется координатами, рассчитанными по следующей методике.

Определим startCoordinates = [44.988789, 34.005899]; координатой левого верхнего угла, рассматриваемого фрагмента карты; endCoordinates = [44.984145, 34.013533]; координатой правого нижнего угла фрагменты карты. Тогда шаг разбиения по оси X всего участка на сегменты определяется по формуле:

$$\Delta x = \frac{x_{end} - x_{start}}{\sqrt{400}},$$

где x_{end} – конечная координата; x_{start} – начальная координата; 400 – количество элементов разбиения фрагментов карты (сетка со сторонами 20*20).

Координата i – ого фрагмента $x_i = x_{start} + \Delta x * i$;

Аналогично для оси Y: $y_i = y_{start} + \Delta y * i$.

Для сохранения координат предложена модель данных, изображенная на рис. 5.5, которая содержит следующие сущности и их атрибуты: «карта», «сложный участок», «остановка», «действия». Созданы таблицы с соответствующими связями и типами данных. В таблице «map» («карта») сформирован атрибут «объект» с типом данных blob, в данный атрибут загружаются фрагменты карты местности лавандового поля.

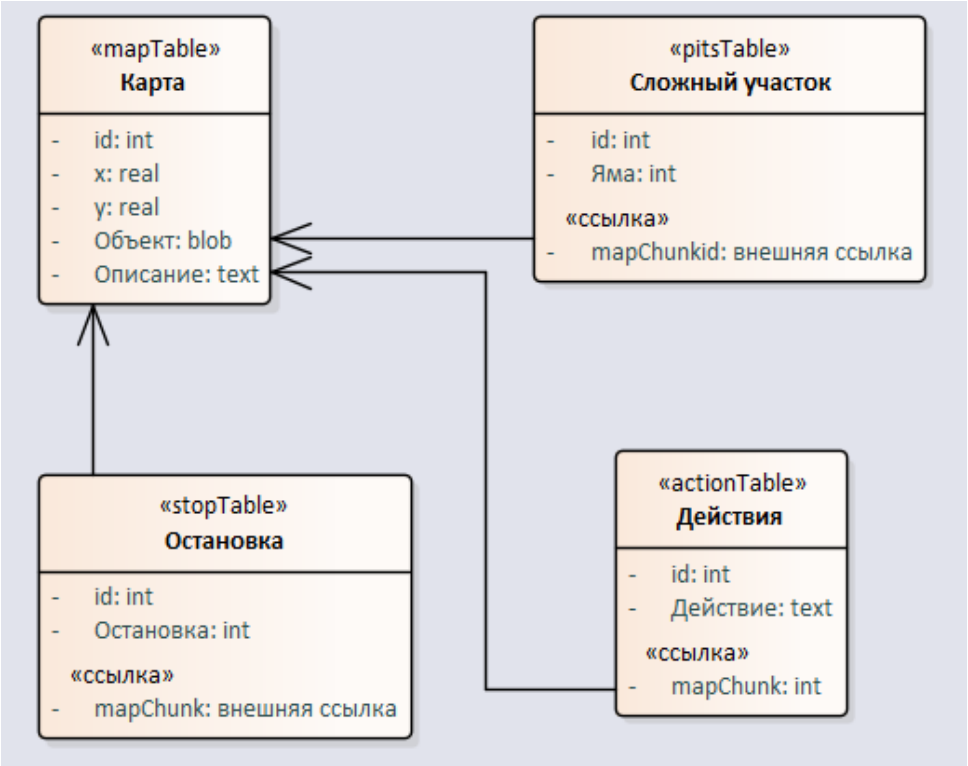


Рис. 5.5 Схема хранилища координат движения БПЛА

Запрос в среде DBeaver для выборки значений координат x и y будет иметь следующий вид:

select m.x,m.y, a."action" **from** maptable m

inner join actiontable a **on** a.mapchunkid = m.id

limit 4000

Осуществим моделирование производительности созданной по приведенной схеме исследовательской БД на основе реализации группы запросов различных видов. Результаты выполнения запросов представлены в табл. 5.1 – 5.2.

В табл. 5.1 – 5.2 приведены результаты моделирования движения БПЛА по заданным координатам движения за время выполнения поддержки в 3 режимах работы: интенсивная нагрузка БД (сложные участки дорог, ямы) при коэффициенте вероятности μ = 0,2; интенсивная нагрузка БД (остановка) при коэффициенте вероятности μ = 0,3; обычная нагрузка БД (таблица с прямой дорогой) при коэффициенте вероятности μ = 0,8. Эксперименты производились на платформе DBeaver с развернутой базой данных при подключении по

внешней сети с максимальным количеством записей в рамках данного исследования — 4000. На рис. 5.6 отражены различия в длительности выполнения запросов вида join при выполнении операции на выборках по координатам и действиям.

Таблица 5.1 Оценки производительности БД при выполнении операции left и right join при выборках по координатам и действиям

Тип соединения	Количество записей	Время выполнения запросов (несколько прогонов), мс	Среднее время выполнения, секунды	Тип соединения	Время выполнения запросов (несколько прогонов), мс	Среднее время выполнения, секунды
LEFT JOIN	500	98, 89, 96	0.094	RIGHT JOIN	88, 89, 96	0.091
	1000	128	0.13		110	0.11
	2000	702	0.7		142	0.14
	3000	356	0.36		192, 269	0.23
	4000	418	0.42		236	0.24

Таблица 5.2 Оценки производительности БД при выполнении операции inner и full join при выборках по координатам и действиям

Тип соединения	Количество записей	Время выполнения запросов (несколько прогонов), мс	Среднее время выполнения, секунды	Тип соединения	Время выполнения запросов (несколько прогонов), мс	Среднее время выполнения, секунды
INNER JOIN	500	76, 224, 94	0.13	FULL JOIN	107, 306, 123	0.15
	1000	159, 314,148	0.21		112,106,150,227	0.18
	2000	139, 241, 371	0.25		191, 159, 221	0.19
	3000	198, 222,1121	0.51		387, 221, 256	0.29
	4000	787,379,539	0.57		254, 274, 235	0.25

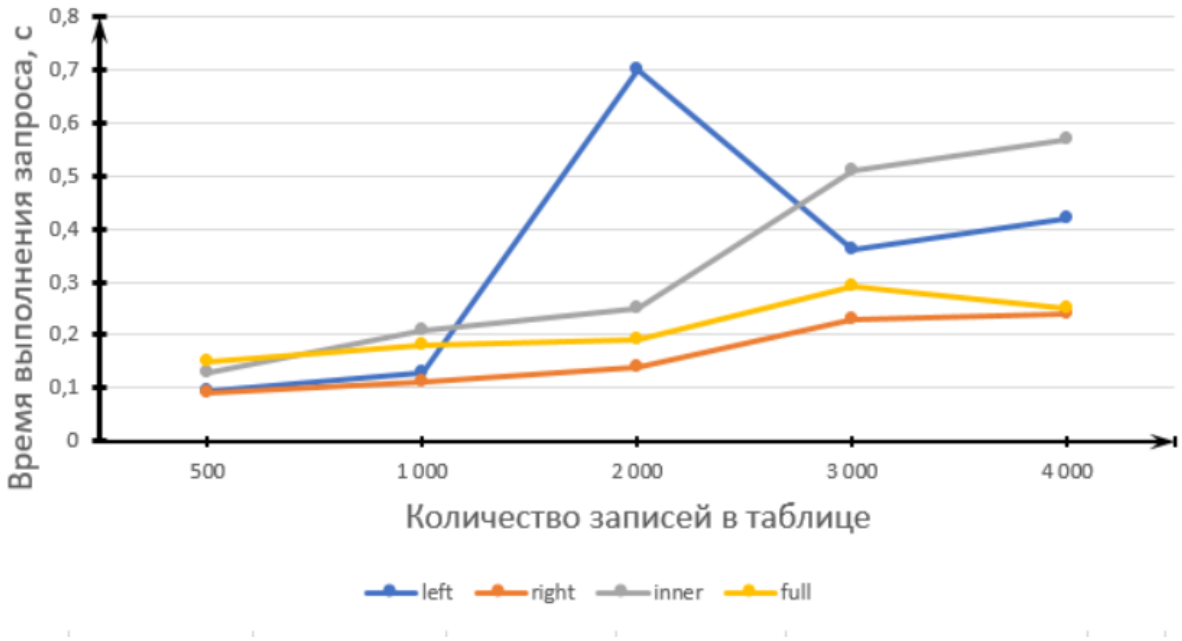


Рис. 5.6 Результаты оценки длительности запросов двух таблиц с различными видами соединения относительно числа кортежей

Для выявления зависимости времени выполнения запросов от различных параметров произведена статистическая обработка полученных результатов. В частности, выполнены элементы корреляционного анализа в различных разрезах (в зависимости от типа запроса, в зависимости от количества записей).

В ходе исследования определены: среднее значение, средние отклонения, стандартные отклонения, коэффициенты корреляции, а также рангового коэффициента корреляции Спирмена в различных типах соединения таблиц. По полученным результатам можно сделать вывод об отличиях в производительности БД при различных видах соединений таблиц: так, при реализации соединения left join и inner join показатель среднего отклонения в 4 раза больше, чем у right join и full join, а коэффициент корреляции достиг максимальных значений у right и inner join = 0,96.

Таким образом, на основе полученных результатов предлагается сепарировать тип запросов в каждом случае применения операции соединения таблиц координатного пространства. Например, left join в задачах, когда нужно получить все записи из таблицы mapTable, даже если для них не существует соответствующих записей в таблице actionTable. Такой тип соединения может

быть полезен, например, при анализе данных, когда нужно получить информацию о всех передвижениях БПЛА, даже если некоторые из них не имеют соответствующих записей в таблице actionTable. Для типа соединения inner join рекомендуется использовать при анализе данных, когда нужно оперативно получить информацию о полетах БПЛА, для которых есть записи о координатах.

Изучение влияния различных типов join соединений при выполнении операции на выборках по координатам и действиям из БД, содержащей схемы пространств для деятельности БПЛА по мониторингу сельскохозяйственных угодий показало, что существуют значительные отличия в производительности БД при различных видах соединений таблиц.

Более быстрый доступ к данным о координатах движения БПЛА позволит улучшить реактивность и точность управления объектом, а также сократить время задержки при передаче данных. Кроме того, оптимизация запросов и обработки данных может повысить эффективность расчета оптимальных маршрутов для БПЛА и сократить время мониторинга производственных площадей [113].

5.5 Результаты оценивания длительности выполнения запросов на двух отношениях с различными типами соединения относительно числа записей БД

В исследовании проводился эксперимент на основе двух таблиц с численностью записей в диапазоне 100 – 500 строк. Вариация количества записей подбиралась таким образом, чтобы избежать неустойчивости статистик тестового примера и получения видимого влияния на производительность системы в целом. Реализация экспериментов осуществлена на основе Oracle SQL Developer, тестовая схема реляционной БД складского подразделения предприятия приведена на рис.5.7 и преобразована в нормальную форму Бойса-Кодда [20].

Приведем пример простого запроса выборки из двух таблиц в приведенном тестовом примере на основе операции left join:

select ord.order_id from order_details ord

left join products prod ON ord.product_id=prod.product_id

Соединение двух отношений реализовано на таблицах тестовой БД: order_details (100-500 записей) и products (77 записей). Фрагменты результатов эксперимента приведены в таблице 5.3 и на рис. 5.8.

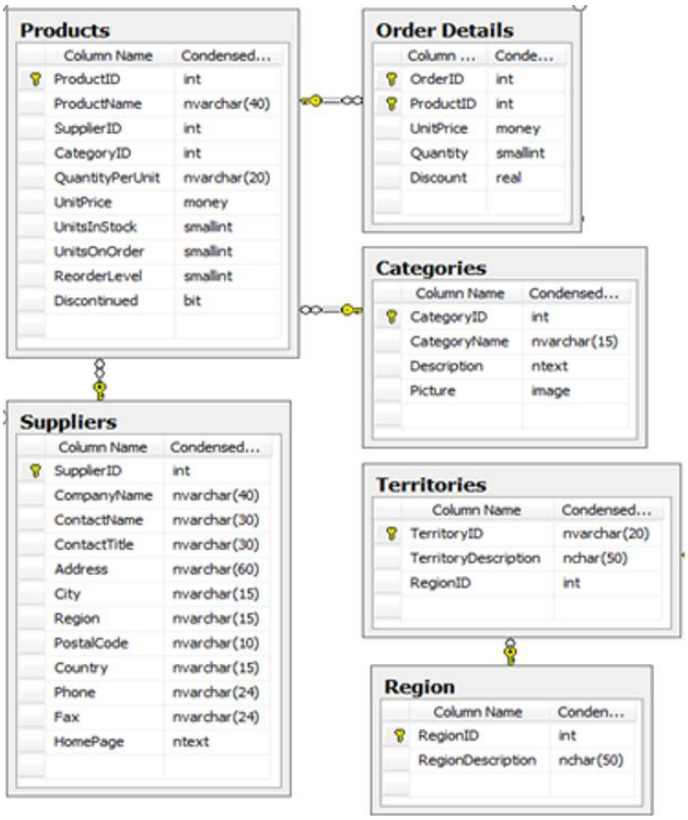


Рис. 5.7 Схема БД складского подразделения предприятия

Таблица 5.3 Величины времен выполнения различных видов соединения реляционных отношений (операций join)

Число записей в отношениях БД, ν	**Время выполнения операции JOIN**			
	INNER t,c	**LEFT t,c**	**RIGHT t,c**	**FULL t,c**
100	0,016	0,015	0,016	0,015
200	0,015	0,016	0,031	0,031
300	0,032	0,031	0,047	0,032
400	0,047	0,037	0,063	0,038
500	0,062	0,047	0,073	0,141

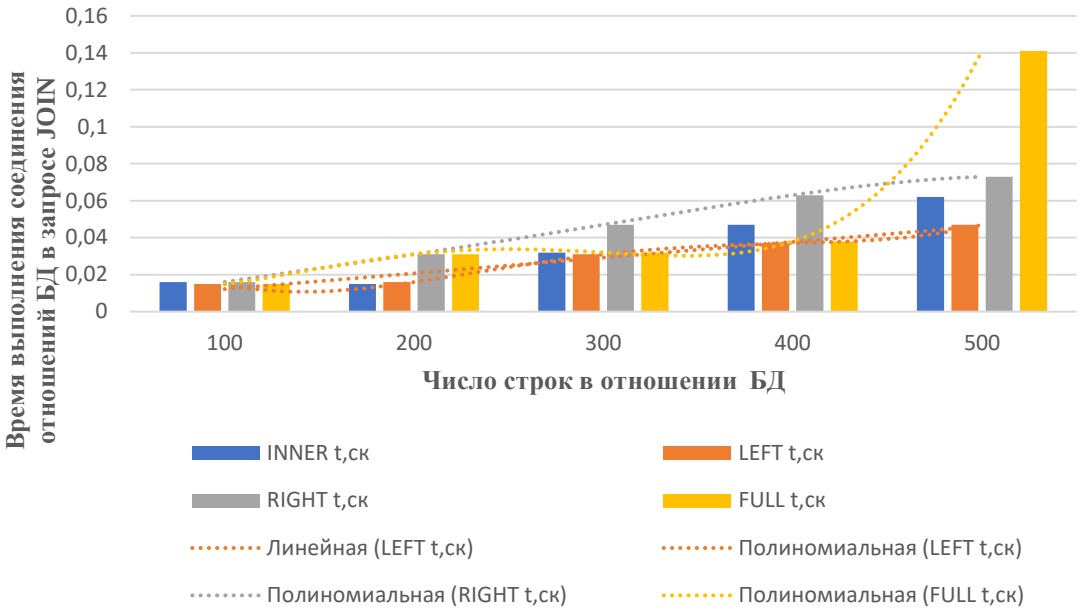


Рис. 5.8 Результаты замеров длительности запросов из двух таблиц с различными типами соединения относительно числа записей (числа строк отношений БД)

Анализ полученных результатов, рис.5.8, показывает, что время выполнения запроса в зависимости от числа записей в отношении БД (рост в 5 раз) увеличивается от 4 до 9 раз при inner и full join соответственно. Наибольшее время выполнения запроса отводится на тип соединения – full join (0,141 сек при 500 записей в БД), наименьшее на вид соединения – left (0,015 сек при 100 записей в БД). Таким образом, наблюдается разброс времени реализации запросов к БД при росте числа записей. Например, для ν =500 различие времен выполнения запросов в случаях inner и full join составляет 43,9%.

С учетом выражения (1) и параметров, характерных для высоконагруженной тестовой производственной системы, оценки производительности приведены в табл.5.4

Таблица 5.4 Оценки производительности высоконагруженной производственной системы

Число записей БД, ν	**Интенсивность $\gamma_{join}(t_i)$**	**Сложность БД , μ**	**Время выполнения операции JOIN**		**Оценка $\pi_{\tau_i}^{z_i}$**	
			INNER t,c	**FULL t,c**	**INNER**	**FULL**
100	0,7	0,85	0,016	0,015	0,67	0,68
200	0,7	0,85	0,015	0,031	0,69	0,73
300	0,7	0,85	0,032	0,032	0,74	0,75
400	0,7	0,85	0,047	0,038	0,79	0,82
500	0,7	0,85	0,062	0,141	0,80	0,91

Математические ожидания производительности БД $\pi_{\tau_i}^{z_i}$ при реализации запросов SQL inner и full join равны соответственно: 0,738 и 0,778; и дисперсии 0,0034 и 0,0079, табл. 5.4.

Таким образом, вид запросов SQL оказывает значительное влияние на производительность высоконагруженных производственных баз данных. Это влияние может быть использовано в качестве инструмента для управления производительностью, что, в свою очередь, может стать основой для реализации системы поддержки принятия решений (СППР) по повышению производительности БД.

В ходе эксперимента оценено влияние вида запросов SQL (join) в высоконагруженных производственных БД на производительность БД в целом, предложена формальная постановка задачи оптимизации производительности такой БД. Показано, что для ν =500 различие времен выполнения запросов в случаях inner и full join составляет 43,9%, что свидетельствует о возможности использования вида запросов SQL (join) как элемента управления производительностью БД и может служить основой реализации соответствующей СППР [45].

Проведенное исследование в рамках данной главы позволило сделать ряд важных выводов, имеющих практическое значение для проектирования и эксплуатации высоконагруженных баз данных в системах управления

беспилотными летательными аппаратами. Анализ производительности различных типов операций JOIN показал, что выбор конкретного типа соединения существенно влияет на время выполнения запросов, особенно при работе с большими объемами геопространственных данных. Наибольшие вычислительные затраты наблюдаются при обработке сложных запросов с операциями соединения и работе с BLOB-объектами, что требует учитывать при проектировании архитектуры БД.

На базе описанных в предыдущих разделах методик и параметров, результаты имитационного моделирования в среде AnyLogic, наглядно продемонстрирована зависимость времени обработки запросов от их типа и объема данных. Было установлено, что простые запросы обрабатываются в разы быстрее, в то время как операции соединения и работа с большими двоичными объектами создают значительную нагрузку на систему. Особенно важно отметить нелинейный рост времени обработки при увеличении количества записей, что подчеркивает необходимость тщательного планирования ресурсов СУБД.

5.6 Результаты моделирования длительности реакции в комплексе программ СППР

В продолжение анализа влияния SQL-операций на производительность высоконагруженных БД рассмотрим результаты моделирования длительности реакции информационной системы поддержки транспортного обслуживания. В табл. 5.5 приведены результаты моделирования длительности реакции СППР на выполнение задач поддержки при двух режимах работы: интенсивная нагрузка БД; обычная нагрузка БД при коэффициенте сложности $\mu = 0{,}8$. Таблица содержит показатели EXPORT и UPDATE для разных объемов записей, а также рассчитанные значения математических ожиданий и дисперсий. Моделирование производилось на платформе в предположении о нормальном законе

распределения случайной величины интенсивности поступления задач на вход ИСПТО.

Математические ожидания производительности БД $\pi_{\tau_j^{BTC}}^{z_j^{BTC}}$ при реализации запросов SQL EXPORT и UPDATE равны соответственно: 0,738 и 0,658; и дисперсии 0,0045 и 0,0095; $\pi_{\tau_i^{BLA}}^{z_i^{BLA}}$ эти оценки равны соответственно: 0,838 и 0,76; дисперсии 0,0015 и 0,0045, табл.5.5.

Таблица 5.5 Оценки производительности БД ИСПТО

Число записей БД, ν	$r_i^{BLA}(t_{i-1})$	$r_j^{BTC}(t_{i-1})$	$\pi_{\tau_i^{BLA}}^{z_i^{BLA}}$		$\pi_{\tau_j^{BTC}}^{z_j^{BTC}}$	
			EXPORT	UPDATE	EXPORT	UPDATE
100	0,93	0,85	0,90	0,77	0,80	0,76
200	0,93	0,85	0,85	0,79	0,80	0,76
300	0,93	0,85	0,82	0,76	0,70	0,63
400	0,93	0,85	0,82	0,74	0,70	0,59
500	0,93	0,85	0,80	0,74	0,65	0,55

Можно видеть, что с ростом числа записей в БД в 5 раз (со 100 до 500) её производительность падает в среднем на 9,8%. Заметны отличия в производительности БД при поддержке БЛА или БТС: так, при реализации запроса на обновление карт среднее изменение скорости информации от БЛА выше среднего изменения скорости обновления от БТС в 7 раз (при увеличении объема записей БД в 5 раз со 100 до 500 условных записей).

5.7 Особенности организации системы информационного обеспечения парково–рекреационных зон

Транспортное обслуживание при небольшой площади парково–рекреационных зон (ПРЗ) в большинстве случаев отсутствует, для больших площадей, применение БТС, оправдано, в том числе приведенными выше факторами. Наряду с БТС, применение легких мониторинговых БЛА придаст структурному решению законченность, фотосъемка объектов с БЛА имеет

множество преимуществ по сравнению с получением данных с помощью, например, космических спутников. Основными предпосылками этих преимуществ являются оперативность получения фотоснимков, возможность съемки с небольших высот. Конечно, при фотосъемке с БЛА следует учитывать ряд параметров, оказывающих влияние на последующую алгоритмическую обработку (высота съемки, скорость полета, скорость передвижения, дальность перехода и др.) [9].

В связи с ускорением цифровизации различных областей жизнедеятельности и развитием информационных систем, в том числе обеспечивающих поддержку транспортного обслуживания ПРЗ, которые дополняются БТС, БЛА, рассмотрим особенности таких информационных систем.

Обозначим следующие виды решаемых задач:

1. Поддержка решений транспортного обслуживания (хранение маршрутов, схемы движения, принятие решений об изменении маршрута и т.п.);
2. Организационные меры (хранение информации о диспетчерах, техническом персонале, управление задачами сотрудников и т.п.);
3. Мониторинг дорожных передвижений (оперативные карты, хранение измененных маршрутов, информация с датчиков БТС и т.п.);
4. Распознавание объектов и сравнение предметов на карте (детализованные карты, сбор и распознавание информации с БЛА, фиксация и тестирование объектов на пути);
5. Экологический мониторинг местности (дополнительные задачи БТС и БЛА);
6. Информация от клиентов (хранение информации, поступившей от клиентов на маршруте).

Таким образом, укрупненная схема информационной поддержки транспортного обслуживания с (без учета технических задач управления движением БЛА и БТС) приведена на рис. 5.9.

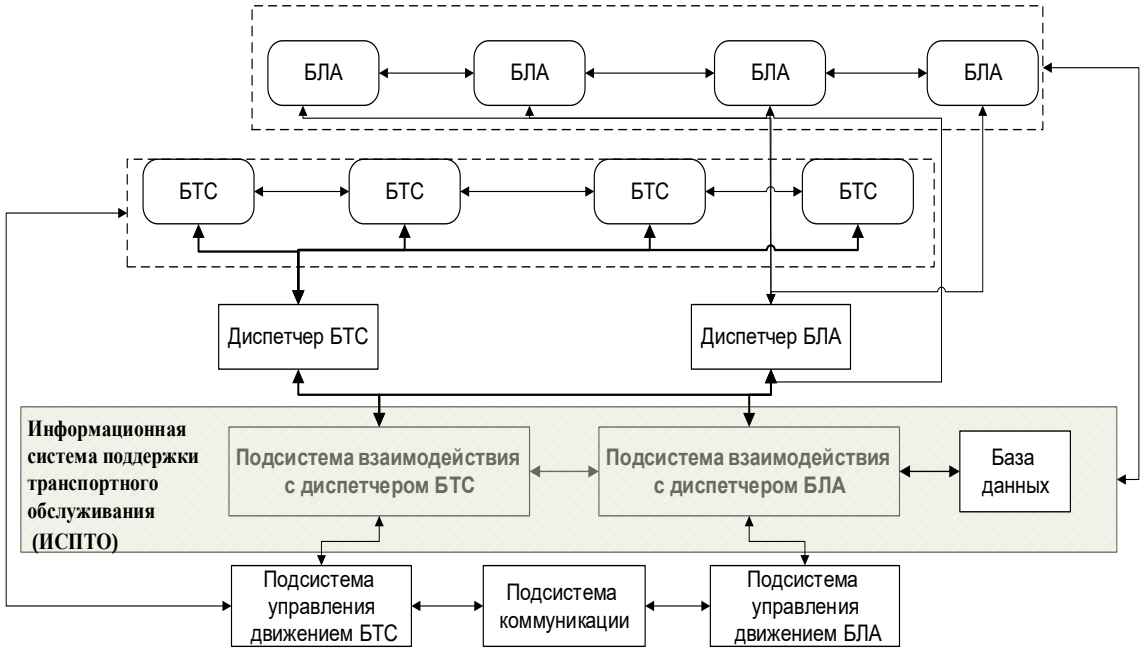


Рис. 5.9 Укрупненная схема информационной поддержки транспортного обслуживания ПРЗ

Как видно на рис. 5.9, система имеет связи с техническим парком БЛА и БТС, диспетчерскими центрами и содержит в составе КБД, которая имеет структуру, схема фрагмента которой изображена на рис. 5.10.

Предложенная модель (рис.5.10) содержит следующие сущности и их атрибуты: «оператор», «маршрут», «диспетчерский центр» (ДЦ), «беспилотное транспортное средство», «схема движений» и «текущий маршрут». Созданы таблицы с соответствующими связями и типами данных. В таблице «move_pattern» («схема движения») сформирован атрибут «объект» с типом данных blob, в данный атрибут будет загружаться карта местности парковых зон.

Рассмотрим элементы требуемых операций для работы КБД на основе операций SQL:

1. Формирование маршрутного движения (CREATE);
   При построении способа доставки формируется маршрут.
2. Если повторный маршрут, то обязательно обновление БД. Если на каждый маршрут поступает новая задача, без повторных маршрутов, то обновление карт не происходит. Соответственно, строится новый маршрут, создается новый запрос к КБД.

3. Экспорт маршрутного движения (EXPORT);
4. Загрузка маршрута;

a. Подпроцесс: валидация на целостность, безопасность;

5. Обновление карты (UPDATE);

a. Условие: если выполняется повторный маршрут.

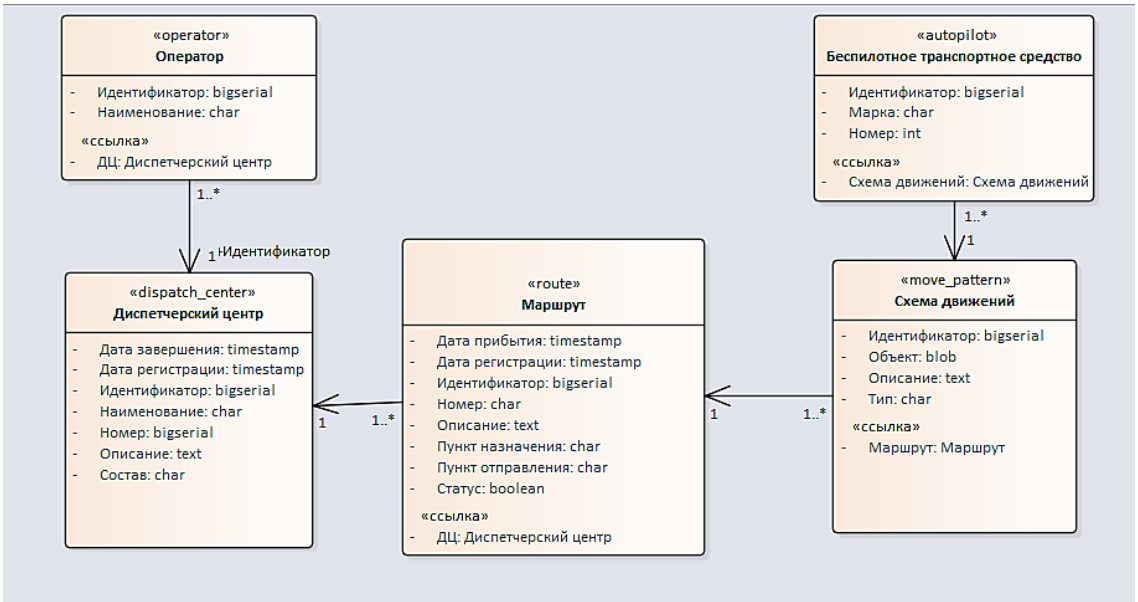


Рис. 5.10 Схема организации подсистемы базы данных информационной поддержки транспортного обслуживания ПРЗ

Задачи обеспечения целостности при критичности по отношению к нагрузке БД, приобретают особое значение, поскольку для формирования КБД необходима загрузка, обновление карт, обмен оперативной информацией по каналам информационного обмена (КИО).

На основании практических рекомендаций о построении КБД сформулируем задачу обеспечения эффективности выполнения ее функционала при условии наличия двух принципиально различных групп объектов обеспечения: БЛА и БТС [19].

Метод ветвей и границ с нейросетевым усилением применяется для оптимизации производительности КБД при выполнении SQL-запросов (JOIN, BLOB, EXPORT, UPDATE) в системах управления БПЛА-ДЦ. Параметры $x = (x_1, x_2, x_3, x_4, x_5)$ соответствуют ключевым метрикам:

$$x_1 = T_{\text{query}} = T_{\text{execution}} + T_{\text{join}} + T_{\text{index}} + T_{\text{io}}, x_2 = R_{\text{mem}}, x_3 = C_{\text{schema}}, x_4 = I_{\text{request}} = \gamma_{\text{join}} + \gamma_{\text{read}}, x_5 = N_{\text{records}},$$

где $T_{\text{join}} = k \cdot \frac{V_1 \cdot V_2}{R_{\text{mem}}}$ для JOIN-операций (INNER, LEFT, RIGHT, FULL), а целевая функция минимизирует общее время $T_{\text{total}} = T_{\text{query}} + T_{\text{interaction}} + T_{\text{structural}} + T_{\text{physical}}$ при ограничениях:

$$g_1(x): T_{\text{query}} \leq T_{\max}, g_2(x): R_{\text{mem}} \geq R_{\min}, g_3(x): C_{\text{schema}} \leq C_{\text{crit}}, g_4(x): N_{\text{records}} \leq N_{\text{crit}}.$$

Гибридная оценка $E(c) = 0.7 \cdot F(c) + 0.3 \cdot f_{\text{NN}}(c)$ позволяет прогнозировать время выполнения BLOB-запросов с учетом сетевых задержек T_{network} и интенсивности I_{BLA}, отсекая неэффективные конфигурации в данном случае (например, индексации). Алгоритм ветвления по x_4 (интенсивность) обеспечивает оптимальный выбор типа JOIN, минимизируя $F(x)$ в реальном времени для сценариев БПЛА с нагрузкой 100–500 запросов/с, достигая сокращения T_{total} на 78%.

На основе описанных в предыдущих глав методик и параметров, результаты имитационного моделирования в среде AnyLogic, наглядно продемонстрирована зависимость времени обработки запросов от их типа и объема данных. Было установлено, что простые запросы обрабатываются в разы быстрее, в то время как операции соединения и работа с большими двоичными объектами создают значительную нагрузку на систему. Особенно важно отметить нелинейный рост времени обработки при увеличении количества записей, что подчеркивает необходимость тщательного планирования ресурсов СУБД.

5.8 Результаты экспериментов на примере реализации принятия решений и рекомендаций по применению БПЛА-пожарных

Лицом, принимающим решения (ЛПР) определены три основных сценария. В таблицах 5.6 приводятся результаты моделирования процессов функционирования БПЛА-пожарных по этим трем сценариям.

Таблица 5.6 – Результаты моделирования производительности критической БД по Сценарию 1

Время (мин)	Производи-тельность (%)	Объем данных (МБ)	Нагрузка на связь (%)	Принятые решения	Скорость передачи (МБ/мин)
18	100	16,0402	63	Оптимизация маршрута	0
20	96,6112	33,3082	47	Аварийный протокол	8,6342
25	95,8552	50,8912	58	Аварийный протокол	3,5162
29	96,5412	72,6512	53	Оптимизация маршрута	5,4402
30	96,822	91,9682	54	Приоритезация данных	19,3162

Рис. 5.11 Результаты моделирования по Сценарию 1: а) Скорость передачи, б) нагрузка на канал связи

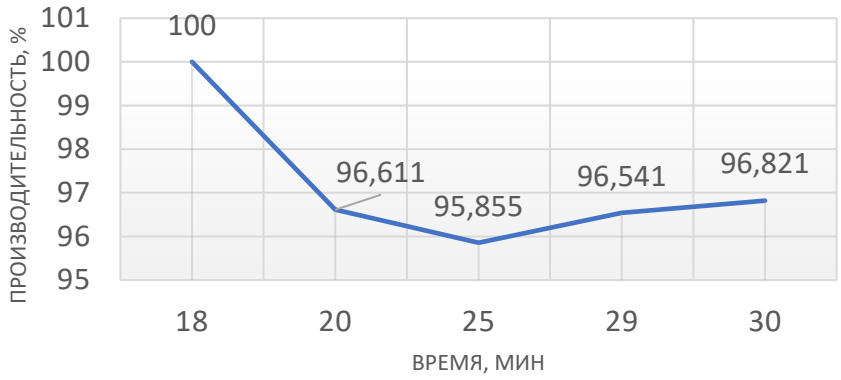
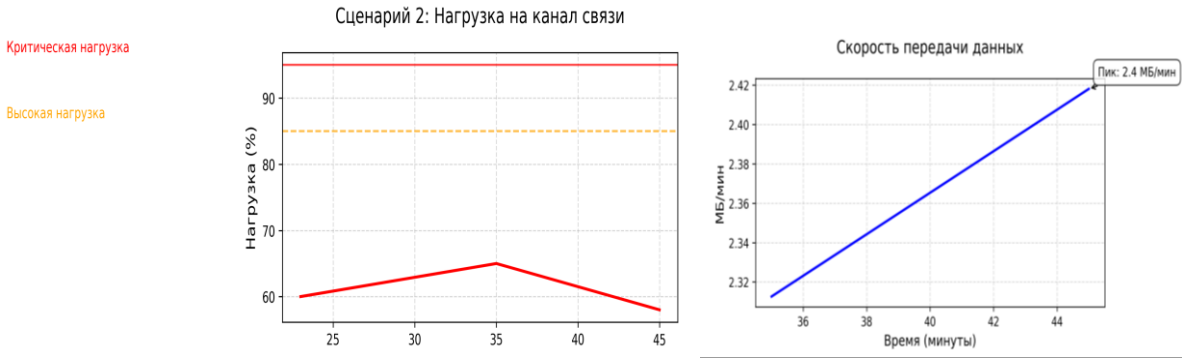


Рис. 5.12 Результаты моделирования: а) производительность; б) объем данных.

Таблица 5.7 – Результаты моделирования производительности критической БД по Сценарию 2

Время (мин)	Производи-тельность (%)	Объем данных (МБ)	Нагрузка на связь (%)	Принятые решения	Скорость передачи (МБ/мин)
23	100	32,62402361	60	Приоритезация данных	0
35	98,375	60,37526329	65	Оптимизация маршрута	2,312
45	95,671	84,55558168	58	Аварийный протокол	2,418



а) б)

Рис. 5.13 Результаты моделирования производительности КБД по Сценарию 2: а) Скорость передачи, б) нагрузка на канал связи

Таблица 5.8 – Результаты моделирования производительности критической БД по Сценарию 3

Время (мин)	Производительность (%)	Объем данных (МБ)	Нагрузка на связь (%)	Принятые решения	Скорость передачи (МБ/мин)
8	99,259	32,681	74	Аварийный протокол	0
20	99,213	67,708	65	Оптимизация маршрута	2,918
33	98,394	107,450	69	Приоритезация данных	3,0571
49	97,234	155,049	75	Приоритезация данных	2,9749
56	96,538	200,236	83	Оптимизация маршрута	6,455
60	96,160	232,582	85	Приоритезация данных	8,086

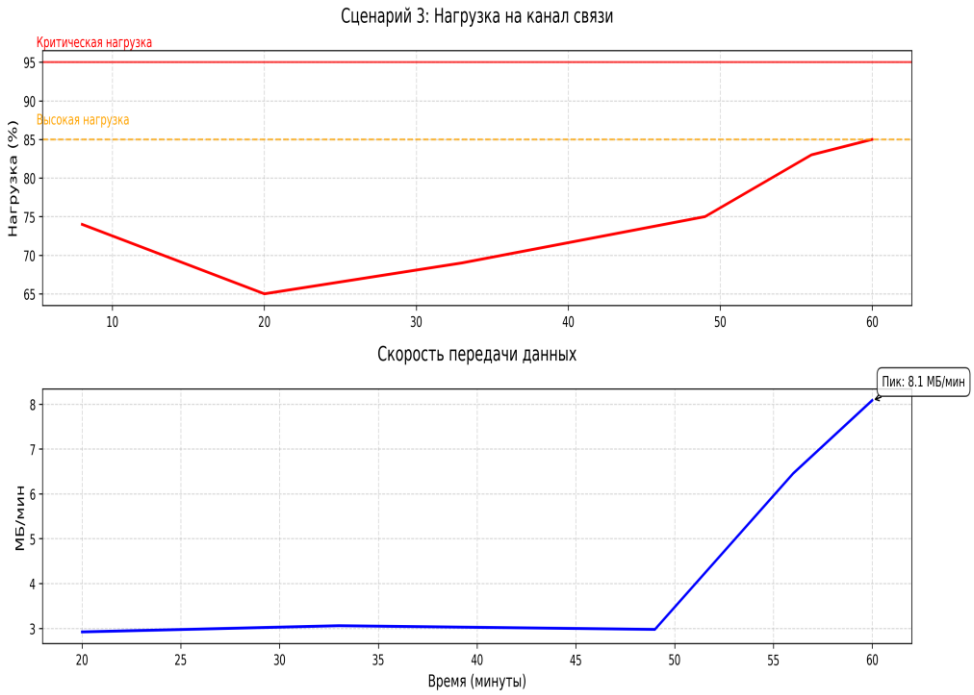


Рис. 5.14 Сценарий 3: скорость передачи и нагрузка на канал связи

Таблица 5.9 – Сформированный итоговый отчет по миссиям БПЛА-пожарный

Сценарий 1 (Сложность: 2/5, Длительность: 30 мин)
Длительность миссии: 30 минут Сложность миссии: 2/5 Ключевые показатели: - Средняя производительность: 97.2% - Минимальная производительность: 95.9% - Всего передано данных: 92.0 МБ - Средняя скорость передачи: 9.2 МБ/мин - Пиковая нагрузка на связь: 63% - Принято решений: 5
РЕКОМЕНДАЦИИ:
- Улучшить систему мониторинга состояния БПЛА
Сценарий 2 (Сложность: 3/5, Длительность: 45 мин)
Длительность миссии: 45 минут Сложность миссии: 3/5

Сценарий 1 (Сложность: 2/5, Длительность: 30 мин)
Ключевые показатели: - Средняя производительность: 98.0% - Минимальная производительность: 95.7% - Всего передано данных: 84.6 МБ - Средняя скорость передачи: 2.4 МБ/мин - Пиковая нагрузка на связь: 65% - Принято решений: 3
РЕКОМЕНДАЦИИ:
- Улучшить систему мониторинга состояния БПЛА
Сценарий 3 (Сложность: 4/5, Длительность: 60 мин)
Длительность миссии: 60 минут Сложность миссии: 4/5 Ключевые показатели: - Средняя производительность: 97.8% - Минимальная производительность: 96.2% - Всего передано данных: 232.6 МБ - Средняя скорость передачи: 4.7 МБ/мин - Пиковая нагрузка на связь: 85% - Принято решений: 6
РЕКОМЕНДАЦИИ:
- Оптимизировать алгоритмы передачи данных
- Использовать специализированные БПЛА для сложных миссий
- Улучшить систему мониторинга состояния БПЛА
ОБЩИЕ ВЫВОДЫ:
1. Для миссий высокой сложности рекомендуем: - Использовать БПЛА с повышенной автономностью - Заранее развертывать ретрансляторы связи 2. Для оптимизации работы:

Сценарий 1 (Сложность: 2/5, Длительность: 30 мин)
- Внедрить систему приоритезации данных - Проводить предварительное сканирование зоны 3. По улучшению связи: - Увеличить пропускную способность каналов - Использовать адаптивные алгоритмы сжатия

Таким образом, программный комплекс позволяет эффективно моделировать миссии БПЛА, выявлять критические моменты и формировать рекомендации для оператора и адаптируется под любую сферу, где требуется мониторинг.

Практические рекомендации, сформулированные по результатам исследования, включают необходимость использования производительных серверов для обработки критически важных запросов, внедрения механизмов партиционирования данных, а также оптимизации сетевой инфраструктуры. Особое внимание следует уделить кэшированию часто запрашиваемых данных, таких как фрагменты карт, что позволяет существенно снизить нагрузку на систему. Таким образом, сформулированные рекомендации позволяют улучшить отклик системы и обеспечить масштабируемость при росте данных.

Перспективы дальнейших исследований видятся в интеграции распределенных баз данных для обеспечения масштабируемости системы, применении методов машинного обучения для прогнозирования пиковых нагрузок, а также в более глубоком анализе влияния сетевых задержек на общую производительность системы. Полученные результаты убедительно демонстрируют, что комплексный подход, сочетающий оптимизацию запросов, адаптивное управление ресурсами и имитационное моделирование, позволяет достичь значительного повышения производительности КБД в системах управления БПЛА, обеспечивая их надежную и эффективную работу в условиях реального времени.

5.9 Обоснование экономической эффективности предложенных решений

Эффективное управление производительностью КБД является не только технической задачей, но и значимым фактором экономической эффективности. Внедрение разработанных моделей и систем поддержки принятия решений позволяет достичь существенной финансовой выгоды.

В диссертационном исследовании представлена аналитическая таблица, систематизирующая критерии оценки эффективности предложенных методов оптимизации КБД в контексте их практического применения.

Экономическая целесообразность внедрения предложенных решений подтверждается следующими ключевыми эффектами, Табл.5.10.

Таблица 5.10 – Экономическое обоснование решений

Критерий оценки	**Экономический эффект / Важность**	**Связь с результатами работы**
Предотвращение простоев	Стоимость простоя критических систем (оборона, финансы, телеком) может достигать сотен тысяч руб. в час. Оптимизация БД снижает риск сбоев.	Сокращение времени выполнения запросов на 27-55% и повышение надежности напрямую снижает вероятность отказа системы.
Эффективность использования ресурсов	Снижение затрат на аппаратное обеспечение (серверы, память, сеть) и облачную инфраструктуру. Вертикальное и горизонтальное масштабирование требуют больших капиталовложений.	Оптимизация позволяет обрабатывать большие нагрузки на существующем оборудовании, откладывая или избегая дорогостоящего масштабирования. Экономия 18-22% времени обработки эквивалентна повышению мощности парка серверов.
Повышение производительности операторов/систем	В системах с БПЛА (мониторинг, безопасность) более быстрый отклик позволяет охватить большую территорию, быстрее реагировать на угрозы или ЧС, что имеет прямую стоимостную оценку.	Сокращение задержки ответа с 0.102 до 0.074 с (гл. 5) повышает оперативность управления БПЛА. Ускорение JOIN-запросов в 1.8-2.2 раза ускоряет аналитику.

Продолжение Таблицы 5.15 –Экономическое обоснование решений

Критерий оценки	**Экономический эффект / Важность**	**Связь с результатами работы**
Снижение эксплуатационных расходов (OPEX)	Снижение затрат на электроэнергию, охлаждение и администрирование систем за счет более эффективного использования ресурсов.	Автоматизированная СППР снижает нагрузку на администраторов БД, позволяя перераспределить человеческие ресурсы на другие задачи.
Внедрение в новые отрасли	Возможность тиражирования решения в сельском хозяйстве, логистике, экологическом мониторинге открывает новые рынки и источники дохода для разработчиков и интеграторов.	Универсальность подхода, продемонстрированная на примерах БПЛА-пожарных и ПРЗ, подтверждает широкий потенциал коммерциализации.

Содержание таблицы наглядно демонстрирует комплексное воздействие оптимизации КБД:

1. На операционную устойчивость. За счет сокращения времени выполнения запросов на 27–55% и снижения задержек ответа существенно уменьшается риск дорогостоящих простоев критических систем.

2. На экономическую эффективность. Достигается прямая экономия на капитальных (CAPEX) и операционных (OPEX) расходах за счет более рационального использования существующей инфраструктуры, отсрочки масштабирования и снижения энергопотребления.

3. На производительность. Ускорение обработки данных (например, JOIN-запросов в 1.8–2.2 раза) напрямую повышает оперативность принятия решений, что в прикладных областях (мониторинг, безопасность) имеет измеримый стоимостной эквивалент.

4. На потенциал внедрения. Универсальность предложенного подхода, подтвержденная на разнородных примерах (от систем ПВО до мониторинга пожаров), обосновывает возможность его тиражирования в новых отраслях, открывая перспективы коммерциализации.

5.10 Выводы по главе

На основе проведенных экспериментов, показано, что выбор типа операции JOIN существенно влияет на производительность высоконагруженных баз данных, особенно при обработке больших объемов геопространственных данных. Наибольшие затраты времени наблюдаются при выполнении сложных соединений и работе с BLOB-объектами, что требует оптимизации архитектуры КБД. Эксперименты подтвердили нелинейный рост времени обработки запросов с увеличением объема данных, подчеркивая важность планирования ресурсов СУБД.

Разработанная система поддержки принятия решений (СППР) анализирует нагрузку и выбирает оптимальные стратегии обработки, что критически важно для систем реального времени. Практические рекомендации включают использование производительных серверов, партиционирование данных, оптимизацию сетевой инфраструктуры и кэширование. Перспективы дальнейших исследований связаны с интеграцией распределенных КБД, применением машинного обучения для прогнозирования нагрузок и анализом сетевых задержек. Комплексный подход позволяет значительно повысить производительность КБД в системах управления БПЛА/БТС.

Оценена экономическая эффективность предложенных решений: за счет сокращения времени выполнения запросов на 27–55% и снижения задержек ответа существенно уменьшается риск дорогостоящих простоев критических систем, что для БПЛА/БТС играет важную роль.

Таким образом, поставленные в исследовании задачи выполнены и имеется потенциал развития предложенных решений и моделей.

ЗАКЛЮЧЕНИЕ

В результате проведенного исследования осуществлена разработка и применение многокритериальных моделей для оптимизации производительности критических баз данных в условиях высоких нагрузок и шумов. Получены следующие результаты.

На основе проведённого детального анализа современных СУБД и программных комплексов управления производительностью КБД. Выявлены наиболее существенные «узкие места»: замедление обработки join-запросов, конкуренция за ресурсы, рост задержек при увеличении объема данных.

1. Предложен полимодельный комплекс для анализа и оптимизации КБД, интегрирующий структурные, программные и имитационные модели, обеспечивающий формализацию представления данных и инструментальную основу для имитационного моделирования КБД в условиях динамических нагрузок. Построена имитационная модель, учитывающая смешанные потоки SELECT, JOIN, BLOB, различные конфигурации серверов (производительный, стандартный, слабый) и каналы передачи (Wi-Fi, LTE, облако). В рамках экспериментов с динамическими параметрами установлено: удаётся выявлять и предупреждать пиковые нагрузки и критические состояния с точностью до 7–9%, а нагрузка на канал связи снижалась на 8% при автоматическом равномерном перераспределении запросов.

2. Получила дальнейшее развитие многокритериальная математическая модель оптимизации производительности запросов к КБД (JOIN, BLOB), в отличие от существующих основанная на иерархическом представлении времени выполнения и учете эндогенно-экзогенных факторов, что позволило реализовать сценарный подход к адаптивному выбору стратегий выполнения запросов в системах «БПЛА/БТС – диспетчерский центр», для чего разработаны и реализованы усовершенствованные алгоритмы денормализации, индексации и автоматического выбора стратегии выполнения SQL-запросов (JOIN, BLOB, EXPORT, UPDATE). Экспериментально

подтверждено: при переходе от нормализованных к денормализованным схемам время выполнения сложных запросов сокращается на 27–55% (например, с 0,42 до 0,19 с на тестовой партии в 4000 записей). Для крупных выборок и сложных миссий достигнуто общее ускорение выполнения JOIN-запросов в 1,8–2,2 раза. При внедрении предложенных методов средний рост производительности КБД составил 9–15% по сравнению с существующими аналогами. Эти решения позволили построить модели управления производительностью на уровне запросов к КБД.

В моделях AnyLogic проведено около 1000 комплексных тестирований с различным числом заявок, показано: при росте числа записей БД в 5 раз (со 100 до 500) просадка производительности не превышает 9,8%, а автоматическая маршрутизация позволяет снизить среднюю задержку ответа с 0,102 до 0,074 секунды для 90% нагрузки. При сложных заданиях разным временем задержки потоков удавалось уменьшить количество неотвеченных заявок в пике с 14 до 2.

3. Впервые предложен гибридный численный метод оптимизации, комбинирующий метод ветвей и границ для глобального поиска с нейросетевой моделью для прогнозирования времени выполнения запросов, что позволило эффективно решать многопараметрические задачи настройки КБД в условиях неопределенности и динамических нагрузок: при сравнении с классическим методом среднее время обработки сложных сценариев снижается на 18–22%, а достигаемое среднее квадратичное отклонение предсказания (MSE) – всего 0,0092 (против 0,021 у стандартных алгоритмов). Достигнута стабильная сходимость: в 80% случаев не более 20 итераций на задачу; прирост эффективности конфигурирования серверов по вычисленным параметрам – до 13%.

4. На основе разработанных моделей и методов впервые создан комплекс программ в составе системы поддержки принятия решений для регулирования нагрузок КБД при информационном взаимодействии в системах «БПЛА/БТС – диспетчерский центр». Интеграция методов машинного обучения и программный комплекс позволила успешно построить комплекс программ,

реализующий автоматический сбор и анализ метрик запросов, формирование до шести сценариев решений с рекомендациями для ЛПР по каждому шагу. Комплекс программ апробирован на данных реальных миссий с БПЛА, позволил сократить время аналитических настроек системы вдвое, прирост результативности оперативных решений – до 98%.

Таким образом, полностью решена научная задача диссертации, а именно: разработан полимодельный комплекс, гибридный численный метод и система поддержки принятия решений для оптимизации производительности КБД в условиях динамических нагрузок и неопределенности, обеспечивающих эффективное функционирование систем управления беспилотными аппаратами.

Практическая значимость работы заключается в разработке программного комплекса, позволяющего повысить производительность и надежность взаимодействия беспилотных аппаратов с диспетчерским центром за счет оптимизации запросов к КБД и прогнозирования их выполнения. Реализованная система поддержки принятия решений обеспечивает эффективное регулирование нагрузок на КБД в условиях реального времени, что критически важно для устойчивой работы всей комплексной интегрированной системы «БПЛА/БТС – диспетчерский центр».

Исследование вносит вклад в развитие методов математического моделирования и оптимизации КБД, открывая новые возможности для их применения в критически важных отраслях, как транспорт, энергетика и экологический мониторинг.

Дальнейшие исследования продолжаются в направлениях развития чувствительности моделей к шумам различных типов, расширении критериального пространства и развитии нейросетевых решений при прогнозировании процессов сложных беспилотных аппаратов в современных условиях функционирования.

СПИСОК БИБЛИОГРАФИЧЕСКИХ ССЫЛОК

1. Монахов, В. И. Исследование методов оптимизации SQL-запросов / В. И. Монахов, А. А. Кружнова // Современные технологии хранения, обработки и анализа больших данных. – 2021. – С. 37–39.
2. Харахинов, В. А. Программный комплекс анализ экспериментальных данных на основе нейронных сетей / В. А. Харахинов, С. С. Сосинская // Системы анализа и обработки данных. – 2018. – № 4 (73). – С. 91–100.
3. Елисеева, А. А. Системный анализ проблемы автоматической нормализации логической структуры реляционной базы данных / А. А. Елисеева, Т. В. Волкова // Фундаментальные научные исследования: теоретические и практические аспекты. – 2016. – С. 281–283.
4. Yang, D. OceanBase: a 707 million tpmC distributed relational database system [Электронный ресурс] / D. Yang, Y. Chen, C. Li [et al.] // Proceedings of the VLDB Endowment. – 2020. – Vol. 13, № 12. – P. 3310–3321. – DOI: 10.14778/3415478.3415534. – URL: https://www.vldb.org/pvldb/vol13/p3310-yang.pdf (дата обращения: 03.10.2023).
5. Левицкий, А. А. Анализ влияния денормализации отношений на время выполнения запроса к реляционной базе данных / А. А. Левицкий, Ю. В. Доронина // Перспективные направления развития отечественных информационных технологий. – 2020. – С. 13–14.
6. Беседин, К. Ю. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями / К. Ю. Беседин, П. С. Костенецкий // Программные системы: теория и приложения. – 2014. – Т. 5, № 1 (19). – С. 91–110.
7. Приказчиков, С. О. Применение графических ускорителей для обработки запросов над сжатыми данными в параллельных системах баз данных / С. О. Приказчиков, П. С. Костенецкий // Вестник Южно-Уральского

государственного университета. Серия: Вычислительная математика и информатика. – 2015. – Т. 4, № 1. – С. 64–70.

8. Гасанов, Э. Э. Моделирование динамических баз данных / Э. Э. Гасанов, А. А. Плетнев // Интеллектуальные системы. Теория и приложения. – 2016. – Т. 20, № 3. – С. 146–150.
9. Тарасов, С. В. Способы реляционного моделирования иерархических структур данных / С. В. Тарасов, В. В. Бураков // Информационно-управляющие системы. – 2013. – № 6 (67). – С. 58–66.
10. Труб, И. И. Применение имитационного моделирования к оптимизации индексов баз данных / И. И. Труб // Девятая всероссийская научно-практическая конференция по имитационному моделированию и его применению в науке и промышленности. – 2019. – С. 242–248.
11. Игнатенко, Р. С. Обзор тенденций в моделировании NoSQL и мультимодельных баз данных / Р. С. Игнатенко // Актуальные исследования. – 2025. – № 44 (279). – С. 26–29.
12. Дубровский, А. В. Возможности применения геоинформационного анализа в решении задач мониторинга и моделирования пространственных структур / А. В. Дубровский // Известия высших учебных заведений. Геодезия и аэрофотосъемка. – 2015. – № S5. – С. 236–242.
13. Кравченко, Ю. А. Способы интеллектуального анализа данных в сложных системы / Ю. А. Кравченко, А. А. Лежебоков, Д. Ю. Запорожец // Известия Кабардино-Балкарского научного центра РАН. – 2012. – № 3. – С. 52–57.
14. Павловский, Ю. Н. Имитационное моделирование / Ю. Н. Павловский, Н. В. Белотелов, Ю. И. Бродский. – Москва : Академия, 2020. – 175 с.
15. Сотников, Д. В. Концепция больших данных как основа процесса принятия решений / Д. В. Сотников // Информатика. Экономика. Управление. – 2025. – Т. 4, № 1. – С. 2038–2042.
16. Свецкий, А. В. Искусственный интеллект в агропромышленном комплексе: проблемы правового регулирования и перспективы

использования / А. В. Свецкий // Сельское хозяйство. – 2025. – № 1. – С. 24–38.

17. Мухаметшин, Р. М. Развертывание сервера баз данных / Р. М. Мухаметшин // Научные исследования: проблемы и перспективы : сборник научных трудов по материалам XXXVII Международной научно-практической конференции, Анапа, 2022 г. – Анапа, 2022. – С. 134–138.
18. Григорьев, Ю. А. Организация базы данных в программном комплексе анализа характеристик производительности распределённых систем обработки данных / Ю. А. Григорьев // Машиностроение и компьютерные технологии. – 2012. – № 02. – С. 39.
19. Доронина, Ю. В. Подход к обработке координат движения беспилотных летательных аппаратов в сельскохозяйственном мониторинге / Ю. В. Доронина, М. И. Мустафаева // Известия Тульского государственного университета. Технические науки. – 2023. – № 4. – С. 269–276.
20. Luo, C. A UAV-cloud system for disaster sensing applications / C. Luo [et al.] // 2015 IEEE 81st Vehicular Technology Conference (VTC Spring). – IEEE, 2015. – P. 1–5.
21. Erdelj, M. UAV-assisted disaster management: Applications and open issues / M. Erdelj, E. Natalizio // 2016 International Conference on Computing, Networking and Communications (ICNC). – IEEE, 2016. – P. 1–5.
22. Пушкин, А. А. База данных характеристик лесных пожаров / А. А. Пушкин [и др.] // Труды БГТУ. Серия 1: Лесное хозяйство, природопользование и переработка возобновляемых ресурсов. – 2024. – № 1 (276). – С. 5–14.
23. Кудрявцева, О. Разработка базы данных в MONGODB для управления изменениями в DEVOPS / О. Кудрявцева, М. Кудрявцева, Е. Бялецкая // Инженерно-строительный вестник. – 2025. – Т. 2, № 52. – С. 81–86.
24. Колбина, О. Н. Разработка алгоритмов для квантовых баз данных в информационной безопасности / О. Н. Колбина, Д. Р. Фатин, Я. А. Федосеева // Информационная безопасность регионов России (ИБРР-2025): материалы научно-практической конференции. – 2025. – С. 386.

25. Sharma, A. An efficient architecture for the accurate detection and monitoring of an event through the sky / A. Sharma [et al.] // Computer Communications. – 2019. – Vol. 148. – P. 115–128.
26. Ren, K. Lightning-Fast and Space-Efficient Indexing of Temporal Data [Электронный ресурс] / K. Ren, J. M. Patel // Proceedings of the VLDB Endowment. – 2020. – Vol. 13, № 12. – P. 3245–3258. – DOI: 10.14778/3415478.3415543. – URL: https://www.vldb.org/pvldb/vol13/p3245-ren.pdf (дата обращения: 25.09.2025).
27. Itkin, M. Development of cloud-based UAV monitoring and management system / M. Itkin, M. Kim, Y. Park // Sensors. – 2016. – Vol. 16, № 11. – P. 1913.
28. Mahmoud, S. Collaborative UAVs cloud / S. Mahmoud, N. Mohamed // 2014 International Conference on Unmanned Aircraft Systems (ICUAS) : Proceedings. – Orlando, FL : IEEE, 2014. – P. 365–373.
29. Yang, G. In-Network Approximate and Efficient Spatiotemporal Range Queries on Moving Objects [Электронный ресурс] / G. Yang, A. Ghosh, L. Liang, T. Heinis // Advances in Database Technology - EDBT 2024 : proceedings of the 27th International Conference, Paestum, Italy, March 25-28, 2024. – P. 34–46. – URL: https://openproceedings.org/2024/conf/edbt/paper-16.pdf
30. Мустафаева, М. И. Перспективы развития науки и мирового сообщества: научно-методические и практические аспекты : сборник научных трудов по материалам X Международной научно-практической конференции, Анапа, 21 июля 2025 года / М. И. Мустафаева, Ю. В. Доронина // – Анапа: ООО "Научно-исследовательский центр экономических и социальных процессов" в Южном Федеральном округе, 2025. – С. 11–16.
31. Кротов, К. В. Математическая модель и алгоритм метода ветвей и границ для оптимизации решений по составам пакетов в многостадийных системах / К. В. Кротов // Информатика и автоматизация. – 2022. – Т. 21, № 1. – С. 5–40.

32. Денисов, О. В. Оптимизация распределения нагрузки по серверным станциям в вычислительном комплексе на основе генетического алгоритма / О. В. Денисов // Известия Тульского государственного университета. Технические науки. – 2025. – № 1. – С. 160–165.
33. Мочалов, В. П. Алгоритм балансировки нагрузки центра обработки данных на основе нелинейной прогнозной модели / В. П. Мочалов, Н. Ю. Братченко, Д. В. Гостева // Журнал Современные наукоемкие технологии. – 2024. – С. 9–12.
34. Мусатова, Е. Г. Метод ветвей и границ для решения задачи минимизации платы за внешние ресурсы / Е. Г. Мусатова, А. А. Лазарев // Управление большими системами. – 2025. – Т. 117. – С. 119–140.
35. Khan, A. Emerging UAV technology for disaster detection, mitigation, response, and preparedness / A. Khan, S. Gupta, S. K. Gupta // Journal of Field Robotics. – 2022. – Vol. 39, № 6. – P. 905–955.
36. Erdelj, M. Help from the sky: Leveraging UAVs for disaster management / M. Erdelj [et al.] // IEEE Pervasive Computing. – 2017. – Vol. 16, № 1. – P. 24–32.
37. Sara, M. A softwarization architecture for UAVs and WSNs as Part of the cloud environment / M. Sara, I. Jawhar, M. Nader // 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW). – IEEE, 2016. – P. 13–18.
38. Chen, Y. L. Intelligent urban video surveillance system for automatic vehicle detection and tracking in clouds / Y. L. Chen [et al.] // 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). – IEEE, 2013. – P. 814–821.
39. Беседин, К. Ю. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями / К. Ю. Беседин, П. С. Костенецкий // Программные системы: теория и приложения. – 2014. – Т. 5, № 1 (19). – С. 91–110.

40. Габов, Н. А. Моделирование бизнес-процессов в нотации BPMN / Н. А. Габов // Инноватика-2023. – 2023. – С. 258–260.
41. Khassenova, K. Цифровизация управления: применение BPMN для улучшения бизнес-процессов / K. Khassenova, Y. Muratbekov, Y. Abenov // ECONOMIC Series of the Bulletin of the LN Gumilyov ENU. – 2025. – № 1. – С. 67–83.
42. Dijkman, R. M. Semantics and analysis of business process models in BPMN / R. M. Dijkman, M. Dumas, C. Ouyang // Information and Software technology. – 2008. – Vol. 50, № 12. – P. 1281–1294.
43. Шмелев, В. Л. Проектирование базы данных для метрологической службы / В. Л. Шмелев, Е. Ю. Воронкин // Вестник СГУГиТ. – 2025. – Т. 30, № 1.
44. Доронина, Ю. В. Классификация требований при создании системы мониторинга каналов информационного обмена / Ю. В. Доронина // Автоматизация и измерения в машино- приборостроении. – 2021. – № 4(16). – С. 38–49.
45. Мустафаева, М. И. Система поддержки принятия решений по повышению производительности прикладных высоконагруженных баз данных / М. И. Мустафаева, Ю. В. Доронина // Автоматизация и измерения в машино-приборостроении. – 2022. – № 1 (17). – С. 56–70.
46. Вдовенко, А. В. Использование инновационных технологий в целях мониторинга земель / А. В. Вдовенко [и др.] // Международный научно-исследовательский журнал. – 2022. – № 1-1 (115). – С. 172–177.
47. Вторый, В. Ф. Перспективы экологического мониторинга сельскохозяйственных объектов с использованием беспилотных летательных аппаратов / В. Ф. Вторый, С. В. Вторый // АгроЭкоИнженерия. – 2017. – № 92. – С. 158–166.
48. Зволинский, В. П. Экологический мониторинг с использованием сверхлёгких летательных аппаратов для нужд народного хозяйства / В. П. Зволинский [и др.] // Мониторинг. Наука и технологии. – 2011. – № 4. – С. 53–66.

49. Щербаков, Д. А. Применение беспилотных летательных аппаратов в сельском хозяйстве / Д. А. Щербаков // Инновации и научные достижения в агропромышленных технологиях и агробизнесе. – 2020. – С. 9–12.
50. Иванов, С. А. Анализ применения беспилотных летательных аппаратов в сельском хозяйстве / С. А. Иванов, Н. А. Майданников, Ю. А. Бондарева // Мелиорация и водное хозяйство. – 2016. – С. 210–214.
51. Сметнев, А. С. Использование беспилотных летательных аппаратов в сельскохозяйственном производстве / А. С. Сметнев [и др.] // Вестник Российского государственного аграрного заочного университета. – 2015. – № 18. – С. 51–56.
52. Глаголева, Г. И. Преимущества применения БПЛА и их использование для нужд сельского хозяйства / Г. И. Глаголева // Наука и молодёжь. – 2018. – С. 104–106.
53. Ильиных, А. Л. Разработка базы данных автоматизированной информационной системы мониторинга земель сельскохозяйственного назначения / А. Л. Ильиных // Интерэкспо Гео-Сибирь. – 2011. – Т. 3, № 2. – С. 124–129.
54. Алгазали, С. М. М. Совершенствование процесса поиска неэффективных SQL-запросов в СУБД Oracle / С. М. М. Алгазали, В. Г. Айвазов, А. В. Кузнецова // Инженерный вестник Дона. – 2017. – Т. 47, № 4 (47). – С. 103.
55. Симонов, С. В. Беспилотные летательные аппараты: виды, преимущества и применение в сельском хозяйстве / С. В. Симонов, Л. В. Ламонина, О. Б. Смирнова // Инновационные технологии в АПК, как фактор развития науки в современных условиях. – 2020. – С. 224–228.
56. Smith, J. SQL Query Optimization Techniques / J. Smith, A. Doe, L. Williams // Journal of Database Management. – 2020. – Vol. 15, № 3. – P. 45–59.
57. Garcia-Molina, H. Database Systems: The Complete Book / H. Garcia-Molina, J. D. Ullman, J. Widom. – 2nd ed. – Prentice Hall, 2018.

58. Brown, T. Asynchronous Processing in Modern Applications / T. Brown, M. Green, D. Cooper // Software Engineering Journal. – 2022. – Vol. 34, № 4. – P. 310–325.
59. Буравлев, А. И. Искусственный интеллект: сущность, принципы работы, области применения / А. И. Буравлев, В. М. Ветошкин // Вооружение и экономика. – 2024. – № 2 (68). – С. 33–42.
60. ван М., Стин. Распределенные системы / С. ван М., Э. С. Таненбаум ; пер. с англ. В. А. Яроцкого. – Москва : ДМК Пресс, 2021. – 584 с.
61. Таненбаум, Э. Компьютерные сети / Э. Таненбаум, Э. Д. Уэзеролл. – 5-е изд. – Санкт-Петербург : Питер, 2012. – 960 с.
62. Rizzo, L. 10 Gbit/s line rate packet processing using commodity hardware: Survey and new proposals [Электронный ресурс] / L. Rizzo, L. Deri, A. Cardigliano. – URL: http://luca.ntop.org/10g.pdf
63. Pearson, D. Proactive Performance Management for Enterprise Databases [Электронный ресурс] / D. Pearson. – URL: https://www.dlt.com/sites/default/files/Quest-Proactive-PerfManage-4-Enterprise-WP.pdf
64. Rathinam, R. Advances and Predictions in Predictive Auto-Scaling and Maintenance Algorithms for Cloud Computing / R. Rathinam [et al.] // 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS). – IEEE, 2023. – P. 395–400.
65. Marques, G. Proactive resource management for cloud of services environments / G. Marques [et al.] // Future Generation Computer Systems. – 2024. – Vol. 150. – P. 90–102. – DOI: 10.1016/j.future.2023.08.005.
66. Бураков, В. В. Инструментальные средства и интеллектуальные технологии систем поддержки принятия решений в ситуационных центрах / В. В. Бураков, Н. Г. Мустафин, М. Ю. Охтилев // Перспективные направления развития отечественных информационных технологий : материалы V Межрегиональной научно-практической конференции, Севастополь, 24-28 сентября 2019 г. / Севастопольский государственный

университет ; науч. ред. Б. В. Соколов. – Севастополь : СевГУ, 2019. – С. 114–116.

67. Федоров, Д. П. Разработка децентрализованных алгоритмов искусственного интеллекта для обработки данных в интернете вещей на основе графовых баз данных / Д. П. Федоров // Академический исследовательский журнал. – 2025. – Т. 3, № 2. – С. 193–205.

68. Тиханычев, О. В. Теория и практика автоматизации поддержки принятия решений / О. В. Тиханычев. – Москва : Эдитус, 2018. – 76 с. – ISBN 978-5-00058-814-7.

69. Lv, F. Intelligent decision support systems in information systems: integrated learning algorithms and applications / F. Lv, Y. Han, J. Han // Applied Mathematics and Nonlinear Sciences. – 2025. – Vol. 10, № 1. – P. 1–15. – DOI: 10.2478/amns-2025-0221.

70. Миронов, Г. В. Инструменты автоматизации в проектах с применением методологии DevOps / Г. В. Миронов // Интеллектуальные системы и микросистемная техника. – 2017. – С. 196–201.

71. Таваева, А. Ф. Оптимизация техпроцессов раскроя деталей с использованием базы данных стоимостных параметров процесса листовой резки / А. Ф. Таваева, А. А. Петунин // Программные продукты и системы. – 2025. – Т. 38, № 1. – С. 150–156.

72. Karras, A. SQL Query Optimization in Distributed NoSQL Databases for Cloud-Based Applications / A. Karras [et al.] // Algorithmic Aspects of Cloud Computing (ALGOCLOUD 2022) : Lecture Notes in Computer Science. – Cham : Springer, 2023. – Vol. 13799. – P. 17–34. – DOI: 10.1007/978-3-031-33437-5_2.

73. Симашев, В. И. Разработка системы управления ассоциативно-защищёнными картографическими базами данных в распределённой вычислительной среде [Электронный ресурс] / В. И. Симашев, М. Г. Нуриев // Международный научно-исследовательский журнал. – 2025. – № 4 (154). – DOI: 10.60797/IRJ.2025.154.89.

74. Классен, Р. К. Особенности эффективной обработки SQL-запросов к базам данных консервативного типа / Р. К. Классен // Информационные технологии и вычислительные системы. – 2018. – № 4. – С. 108–118.
75. Петрова, А. Н. Исследование рекомендаций по устранению причин неэффективности запросов таблицам базы данных / А. Н. Петрова, И. Н. Калмыков // Вестник евразийской науки. – 2016. – Т. 8, № 3 (34). – С. 137–145.
76. Иванов, А. Ю. Проблемы использования методов искусственного интеллекта в интересах структурной адаптации распределенной базы данных / А. Ю. Иванов [и др.] // Труды СПбГМТУ. – 2024. – Т. 9, № 1. – С. 058–063.
77. Лягушева, М. А. Fнализ производительности select-запросов sql / М. А. Лягушева, Н. Н. Гринченко // Методы и средства обработки и хранения информации. – 2021. – С. 157–159.
78. Аббасов, Э. М. Повышение производительности больших баз данных и действующих на их основе прикладных сервисов / Э. М. Аббасов, С. Н. Польшин // Информационно-технологический вестник. – 2020. – № 1 (23). – С. 42–48.
79. Munerman, V. I. Анализ одного алгоритма операции Join / V. I. Munerman, D. V. Munerman // Современные информационные технологии и ИТ-образование. – 2024. – Т. 20, № 3. – С. 609–618.
80. Азымов, К. Оптимизация производительности баз данных: техники и стратегии / К. Азымов, М. Багшиев, О. Багшиева // Вестник науки. – 2024. – Т. 1, № 10 (79). – С. 371–374.
81. Алибиева, Ж. Сравнение возможностей NoSQL колоночной базы данных / Ж. Алибиева [и др.] // Вестник КазАТК. – 2024. – Т. 131, № 2. – С. 350–358.
82. Bazhutin, M. M. An approach to improving the efficiency of the database of a large industrial enterprise / M. M. Bazhutin, V. S. Moshkin // 2023 International Russian Smart Industry Conference (SmartIndustryCon). – IEEE, 2023. – P. 20–24.

83. Доронина, Ю. В. Каскадно-иерархическое моделирование в задачах анализа динамики ресурсных характеристик сложных систем / Ю. В. Доронина, А. В. Скатков // Информационно-управляющие системы. – 2020. – № 3 (106). – С. 48–58.
84. Доронина, Ю. В. Анализ статистической устойчивости стационарных марковских моделей / Ю. В. Доронина, А. В. Скатков // Информатика и автоматизация. – 2019. – Т. 18, № 5. – С. 1119–1148.
85. Набродова, И. Н. Повышение производительности баз данных / И. Н. Набродова, Г. А. Кузнецов // Известия Тульского государственного университета. Технические науки. – 2021. – № 9. – С. 371–373.
86. Воронин, В. В. Оптимизация производительности выполнения запросов в реляционных базах данных корпоративных информационных систем / В. В. Воронин, И. В. Кочетова, М. А. Яковлев // Информационные технологии XXI века. – 2013. – С. 353–359.
87. Мошкин, В. С. База знаний экспертной системы для анализа SQL-кода промышленных баз данных / В. С. Мошкин, М. М. Бажутин, Н. Г. Ярушкина // Двадцать первая Национальная конференция по искусственному интеллекту с международным участием КИИ-2023 : Труды конференции. – Смоленск: Принт-Экспресс, 2023. – С. 218.
88. Ямалеева, Г. Н. Оптимизация исполнения SQL-запросов к базам данных под управлением MySQL / Г. Н. Ямалеева, М. Ю. Перухин, Р. Ф. Гибадуллин // Информационные технологии и математическое моделирование (ИТММ-2017). – 2017. – С. 239–241.
89. Ковалик, А. А. Информационные технологии и интеллектуальные системы / А. А. Ковалик, И. М. Семичев // Москва. – 2024. – Т. 26. – С. 833–837.
90. Акопов, А. С. Имитационное моделирование / А. С. Акопов. – Москва : Юрайт, 2015. – 389 с. – ISBN 978-5-9916-5549-1.
91. Доронина, Ю. В. Подход к обработке координат движения беспилотных летательных аппаратов в сельскохозяйственном мониторинге / Ю. В. Доронина, М. И. Мустафаева // Известия Тульского государственного

университета. Технические науки. – 2023. – № 4. – С. 269–276. – DOI: 10.24412/2071-6168-2023-4-269-276.

92. Матвеева, А. Р. Методика формирования базы данных характеристик сложного технического объекта с использованием больших языковых моделей / А. Р. Матвеева, Е. В. Антонов // Вестник НИЯУ МИФИ. – 2024. – Т. 13, № 5. – С. 350–357.

93. Кравцов, Г. Г. Аналитические методы контроля образования как основа образовательной базы данных / Г. Г. Кравцов // Universum: психология и образование. – 2025. – Т. 1, № 1 (127). – С. 40–45.

94. Tripathi, N. NoSQL database education: A review of models, tools and practices / N. Tripathi, R. Gupta // Journal of Systems and Software. – 2025. – Vol. 205. – Art. no. 112059. – DOI: 10.1016/j.jss.2025.112059.

95. Понин, Ф. Н. Методология проектирования и создания баз данных для современного программного обеспечения / Ф. Н. Понин // Universum: технические науки. – 2024. – Т. 1, № 1 (118). – С. 16–20.

96. Зубко, О. В. Особенности моделирования распределенных информационных систем / О. В. Зубко, А. В. Макаренко // . – 2025. – С. 11 - 16.

97. Pan, L. Distributed Database Optimization Techniques Combining Computer Network and Algorithm Design / L. Pan, J. Li // Applied Mathematics and Nonlinear Sciences. – 2025. – Vol. 10, № 1. – P. 1–14. – DOI: 10.2478/amns-2025-0611.

98. Кузнецов И. А. Оптимизация распределенных систем для мобильных приложений: улучшение производительности и масштабируемости //Инновационная наука. – 2024. – №. 5-1. – С. 52-57.

99. Кравченко, Ю. А. Способы интеллектуального анализа данных в сложных системах / Ю. А. Кравченко, А. А. Лежебоков, Д. Ю. Запорожец // Известия Кабардино-Балкарского научного центра РАН. – 2012. – № 3. – С. 52–57.

100. Шишкин, С. Р. Имитационное моделирование в сфере защиты информации с применением нейросетей / С. Р. Шишкин // Экономика и качество систем связи. – 2025. – Т. 1, № 35. – С. 131–140.
101. Оразмурадова, З. Повышение экономической эффективности обработки данных на SQL-серверах / З. Оразмурадова // Образование и наука в XXI веке. – 2025. – № 63-1 (Т. 1)
102. Пугин, М. В. Влияние искусственного интеллекта на нормализацию баз данных при работе с Big Data / М. В. Пугин, Н. Н. Гринчар // Международный журнал гуманитарных и естественных наук. – 2024. – № 8-2 (95). – С. 159–162.
103. Бердымуратов, Д. Б. Анализ систем идентификации и формирования базы данных беспилотных летательных аппаратов / Д. Б. Бердымуратов // Universum: технические науки. – 2024. – Т. 1, № 10 (127). – С. 4–8.
104. Малыгин, Д. С. Проблемы производительности реляционных баз данных в распределенных архитектурах и стратегии их решения / Д. С. Малыгин // Современные наукоемкие технологии. – 2024. – № 10. – С. 61–71.
105. Куницын, В. И. Будущее баз данных: квантовые базы данных и новые парадигмы / В. И. Куницын [и др.] // Время науки 3. – 2025. – С. 19.
106. Шабля, В. О. Анализ существующих источников знаний в виде баз данных уязвимостей автоматизированных систем / В. О. Шабля [и др.] // Вестник кибернетики. – 2025. – Т. 24, № 2. – С. 74–82.
107. Alazawi, Z. Intelligent disaster management system based on cloud-enabled vehicular networks / Z. Alazawi [et al.] // 2011 11th International Conference on ITS Telecommunications. – IEEE, 2011. – P. 361–368.
108. Кобелев, Н. Б. Имитационное моделирование / Н. Б. Кобелев, В. А. Половников, В. В. Девятков. – Москва : Наука, 2020. – 300 с.
109. Смирнов, Н. А. Повышение эффективности поисковых запросов высоконагруженных приложений / Н. А. Смирнов, Л. М. Червяков, Н. А. Бычкова // Известия Тульского государственного университета. Технические науки. – 2025. – № 2. – С. 152–158.

110. Каримов, Т. Ш. Оптимизация запросов в реляционных базах данных: подходы и инструменты / Т. Ш. Каримов // Вестник науки. – 2025. – Т. 3, № 4 (85). – С. 780–786.
111. Палаев, С. В. Применение векторных баз данных для анализа компьютерных сетей предприятий / С. В. Палаев, Н. Н. Безуглый, И. П. Черменева // Автоматизация и измерения в машино-приборостроении. – 2025. – № 4.
112. Доронина, Ю. В. Особенности обработки больших объемов данных при управлении беспилотными летательными аппаратами на основе имитационного моделирования / Ю. В. Доронина, М. И. Мустафаева // Вестник Санкт-Петербургского государственного противопожарного университета МЧС России. – 2024. – № 1. – С. 123–138. – DOI: 10.61260/2218-130X-2024-1.
113. Доронина, Ю. В. Информационная система поддержки транспортного обслуживания парково-рекреационных зон с применением беспилотных технологий / Ю. В. Доронина, М. И. Мустафаева // Автоматизация и измерения в машино- приборостроении. – 2022. – № 4(20). – С. 65–77.

ПРИЛОЖЕНИЕ А

1. Описание структуры баз данных (DDL)

*/Создание хранилище для «operator» */

```
-- public."operator" definition

-- public."operator";

CREATE TABLE public."operator" (
                        id bigserial NOT NULL,
                         "name" varchar NULL,
                    dispatch_center int8 NOT NULL
);

-- public."operator" foreign keys

ALTER TABLE public."operator" ADD CONSTRAINT operator_fk FOREIGN KEY
(dispatch_center) REFERENCES public.dispatch_center(id);
-- public.dispatch_center definition

--
```

*/Создание хранилище для «autopilot» */

```
-- public.autopilot definition


CREATE TABLE public.autopilot (
                        mark varchar NOT NULL,
                       "number" int4 NOT NULL,
                        id bigserial NOT NULL,
                     move_pattern int8 NOT NULL,
                CONSTRAINT autopilot_pk PRIMARY KEY (id)
);

-- public.autopilot foreign keys

ALTER TABLE public.autopilot ADD CONSTRAINT autopilot_fk FOREIGN KEY
(move_pattern) REFERENCES public.move_pattern(id);
```

*/Создание хранилище для «move_pattern» */

```
-- public.move_pattern definition

CREATE TABLE public.move_pattern (
                        id bigserial NOT NULL,
                         "object" bytea NULL,
                        description text NULL,
                         "type" varchar NULL,
                         route int8 NOT NULL,
              CONSTRAINT move_pattern_pk PRIMARY KEY (id)
);
-- public.move_pattern foreign keys
```

```
ALTER TABLE public.move_pattern ADD CONSTRAINT move_pattern_fk
FOREIGN KEY (route) REFERENCES public.route(id);
```

*/Создание хранилище для «route» */

```
-- public.route definition

CREATE TABLE public.route (
                        id bigserial NOT NULL,
                        "number" varchar NULL,
                       description varchar NULL,
                      date_create timestamp NULL,
                       date_end timestamp NULL,
                         way_to varchar NULL,
                        way_from varchar NULL,
                    dispatch_center bigserial NOT NULL,
                     route_current bigserial NOT NULL,
                    CONSTRAINT route_pk PRIMARY KEY (id)
);
```

*/Создание хранилище для «dispatch_center» */

```
-- public.dispatch_center definition

CREATE TABLE public.dispatch_center (
                        id bigserial NOT NULL,
                         "name" varchar NULL,
                      date_create timestamp NULL,
                       date_end timestamp NULL,
                         "number" int4 NULL,
                        description text NULL,
                      "structure" varchar NULL,
                CONSTRAINT dispatch_center_pk PRIMARY KEY (id)
);
```

2. Фрагменты манипуляций (добавление) данных. DML INSERT

*/добавление карты в поле «object» табл. «move_pattern»

```
INSERT INTO public.move_pattern
(id, "object", description, "type", route)
VALUES(15,
decode('FFD8FFE000104A4649460001010200250025000FFDB00430003020202020203020202030303030
4060404040404080606050609080A0A090809090A0C0F0C0A0B0E0B09090D110D0E0F101011100A0C121312
10130F101010FFDB00430103030304030408040408100B090B10101010101010101010101010101010101
0101010101010101010101010101010101010101010101010101010101010FFC000110800400029030111
00021101031101FFC4001B000002030101010000000000000000000005060204070308 01FFC4003C1000010
303030104030C0B0000000000000102030405061100072112131422411531610835425155718192 96A2D1D3
161823525354589194B2C1FFC4001A0100030101010100000000000000000000000102030004050 6FFC400241
100030001040202020300000000000000000010211031221310413415114222332 61FFDA000C030100021103
11003F00F15A41D7BA7CC12C67418130DDB693DC2E9C02716FC8C7B7C68D253E8795D82021479C683663A16
5440391A190E08F65EC3AD9360E68C24825208F30754C937C92E9E78E06743205C761DB6503B8DD078F7824
7FB23495F05A39C82836A270348E82A4906579FF00BA555C8FB09762E7EEE9B703632A96564F034D9249130
D91C2B235B70E97D076DB4A042B9C159F78241FBE8D25DF05221BF829C6692A5A5195152B9C01CE351768B2
865A75A5B5FB244390FACE5494B6D952881EC1F3EA5EC52F25261B38F672BE4B9FFE2B9F86B7BE41E9A0BD0
6C1B8EB497551A9329CECBE02585294AE320823839F669AFC94845E33A43953B61AEB65BA6B976C5856F9AA
B6A911D1569688EA5340F27A49C9F986A2F5DD7474CF8CA796695666C4DBEBA7D6E548F4854E22A8CFC6933
203203454543869B27ADC238C9031A4AAB2B11087EB5F6BB6AADF663BA9D9CBB6E2A9F49438FB81B84DADB3
```

```
EA385AF215C7231E6753FD8A62466826EAA554A31A1D8561EDDD0DB52CA8B4C1A9D55D3E49EA574A120FC2C
67454E7B0E667E0EDFA6B75FF003168FD9D5FE6E87AC3B97D0A1BAF78DCD66C19AE5D6D42B628495062874C
B61C40A9D6DBCE55216A583DDDA07CC8C9C803E3D4A16FECAB4A5E1199DBDBED51629B725D76E5B71199AA5
C6603B5C946A0F210739E849094201E9E71E7AE88D369E485D35C3430D2377EFAB836DEB973CFDC0874BAB0
75E8CDF651D0CF5B2024F4B5D20E559C60633A372DD7F806F1387D938F49BDE6C102E18BBCD567E5C748ED0
1EEED04921596D3C7F73CE9154F590FA9BE501AD6B5265BDB914615B9139894B7BBC311AB4E944A11BA8027
A42885724739F3F5684F29B354ED7B4D1FD3CFFCB6DFD41F86A9BE7E8AFA2CC2F786D972897D5E12E556D9A
8ADF73BC477D6E9538232C05348E7E24903E8D474960D4F15B829EE4EB52DFBDEF3AED1AEBA6AA75318A2AA
A298A4F4A1D710B0064F98F11E354F21E34DB4C655376923D13B7144A3D8FEE8DB5E1D916AB711936DD566A
60B4F84A0B8A484A9EEA738040E7E8D73F89AB5A99743F970A1FEA5AAA54A4A5E90B32DD7125C5A8A8AC9CE
5475CD4DEE65E7FAA2B26A139EB75C8E509532E4F682A4A963285768DE13D2AF11CFC69E35E878FCE8BC9C3
ACFF9503FB157CA91FEF68610FB8CF6E0B4ADEA9EFAC18F369F37D0B51851A6424A878A645547484364A870
A041C8F5EA0A9AE50F529BE4B9B4FB6F70DB352AFD4AE581E898931D791487614743A571BAF2941E850291E
AC8579E8EB4BB4576E9461CF63BED134E337D55AEBAD579E5C9B6A2CEA6C55B4C223BBD0F349C36B0B0A480
4F1D5CE34DA33EB7839F5ADEA7614345724CB7113A9D5092D29D5F4AE155E232DA7827C4161457CE07040D5
168E9CBE7E49FE45748A6C4793498D063579E42449ABC5622B0890D2FB525D4294A504E4809C11EBC13AA29
5A70E67E44A6F56953187BD53BF8303EAB5F86A5C16CB33E302DABE2DF455EEB7372F6FD9B313DDBD297254
5A61C71870F0871DC1CE0F841C6403A8258ECBE73C8B547A6539CA7C9B72D7854FAAD26648121D65ADD6487
652923C2B5254CA7A73C1C24E9D6592C2967D45ABBD14D5D566C6F737569888FD35D652A66E0725B6F95290
012EF2391CE4F967472D720C260FF004BB144BD6A14A8BB6953663D2108370542A1565B7199504F2960A9BC
2CF9039E4F969936C184BE0E7696FEDBF6B4D7AA31760E93316A7FB588EC9AD3BDB32904949396C8EA38078
F5674735D6454D67A183F5E07FF00A6DA27DA03F95A4C31F72FA3FFD9','hex'), 'hbefh', 'gs', 3);
      INSERT INTO public.move_pattern
      (id, "object", description, "type", route)
      VALUES(16,
decode('FFD8FFE000104A4649460001010200250025000FFDB00430003020202020203020202030303030
406040404040408060605060908 0A0A090809090A0C0F0C0A0B0E0B09090D110D0E0F101011100A0C121312
10130F101010FFDB00430103030304030408040408100B090B10101010101010101010101010101010101
0101010101010101010101010101010101010101010101010101010101010101FFC000110800400028030111
00021101031101FFC4001D000001040301010000000000000000000000040205060700030801 09FFC40030100
0010303030205030207000000000000010203040506110012210731132223415108146115322433527181 91
B1FFC4001A010002030101000000000000000000000001020304050006FFC400261100020202020104010 50
0000000000000000102110321041231051332517122416162C1FFDA000C03010002110311003F00E332D1D6
C2679BA1253F3A6B019B746C5A3CC1CEB93A158B48F2E35CD91B4EEC20B4A1DD2750A9171C68D6A696790D9
C7FCD1EC234236F38D329581E8CDA3FD68F916AC52104F6C6348E54C651B448E93499357586988524A8AB68
CB4A4E4FF72359EF3A2F3C121EDAB51CA5DB950A95621BCC3AEC86E2C7DE929E7CC491F20E06BA39BB3D099
71748DB2152985254B5A50AD89EE40E35771BD146EDD055068AE5C0E4D8909655323C376634C2464BC96C65
607E42727FC6BA52A26846C6D64F8C84B8D9FDE320FE343BAA0BD3A3E99CB6AEF80ECA80C74E29F3DCA5310
9FAA8A5528BEA43729254C292946544A8255C60EDF73AC778CDFD356529F5A512B4E22CEA4C6B325539119B
724D45098DB56D24EDD9E26DEC71938EFA920E312AF261DE1A472E4F9F4F9DBA1C4736E728F513B7275771E
55E2CC458E69B6D0AB320DD943B929D72D1EDF913D14E9682EA5230879BCE168C9EE0A7235D932C68B58B1C
9B2E6A57D3136D56A6CD9520268B264FDDD3923CBE145732A4A559F7C6463F1AA73CCE5A48D2C7C26D769B2
D8B6FA8DD0E8F60419D6EDF7D69B42F0954CF0A2BCDBCB2C54AA296F18F2EE4EC2EF201DA36EB9F222B4CD2
870B2B92BF0CD9D3CEB0D0FA8C2A1D3AAE393A4DD14A5A1A9732AB27645AF4850F51A4B84F91D0470338D43
DA2DE86971E71BA5A36B960DB1E3BAA9B6EDCD4C05C504B7369EA7020A78294AC01BC7C119D1EADECAED43C
340136B3D2FB6DF54295322C3901412D462E05C97B8C82A613B948DDEDB881F3A916094B62F6843E2865B2B
AB54CB96FC8965542850ADF8B5362408B53AB4F0EBECBC94EE6C2991E9B4DF1839C9D1A8C42DC9F91A6951A
E18F4B79A6699506D09860D327459A169C8CE30DF239EDCF3A8E58E2DD93AE5E5824A2C8CC6B76A7294D479
7D259AF444A8AE4C390E867EE5E57257900E1448E71A1ED7D16573E5D1C64BC977F4AA0D06D7A5D4D5F505D
138573C572499B40A3C60CD457058710D2402F3AE27FA55E9EDCA771F9D4CF48CD49DB6C12F28566DD17E58
F6FD8165CEB26554E9CEC4932BF4A8822256A91E8995C8FE5B640F10649C818E3462DB5A673A40B6A5AFD12
6EE3975AA5D1DAB9EA28AA22932DF424A8C49082A43A508529292956D27710780351CB6AC9232FB13D39A85
B312D6A6D269571D2654C699FE2588D210B4A54544F1DBE7E3463427EC4BDD556000597948DC9F2F1C2B3ED
F91A6B38638B06BAD531F5CF9ED7DCA944A0456B6803F3B8924E8BDA0918BA271B761D1589970A67DC6B909
A7D3614E7C28CA0F11BD3E12782A1C633C0D75D215EC973D53B96D96175BEAADD74DB7DD68F870A894C69A7
5F241CED7D401EFC65291EFDF54B93CCC3C3C6F26695221CB9B1E08F6C92A1C7A7549E97F54AC3A2CEA5DAF
6B226CF69E50B725BE97AA709B0AC1F0DCCA5D29CF6E38D3352BB2E2920E1D2B6283311228776D769CC34A0
154D9CDA64461CF2028E169E3B7274633682E29F8009569DC94B555AB754BFA9CBA3B295CB32253010E426B
3C0F0D3CA923B0564E4FCEA5796314E52749104DA82B6F45374ABE5A9571469D685AF0575E11DD7575894D9
7263C718DCC8570CA7904719CE35879BD5F2F2AF1FA747B7F77F15F8FB3267EA13CADC38AAFF0097E0AEAD8
66F29D78D3EB9598AD43A7B9396896ECE25C4BD8FDC1C51EEAF723430FA64632F7B92DE4C9F6FC2FC220C1C
56E7EE66FD52FB7FE23FFFD9','hex'), 'dd', 'd', 3);
```

3. Диаграмма баз данных

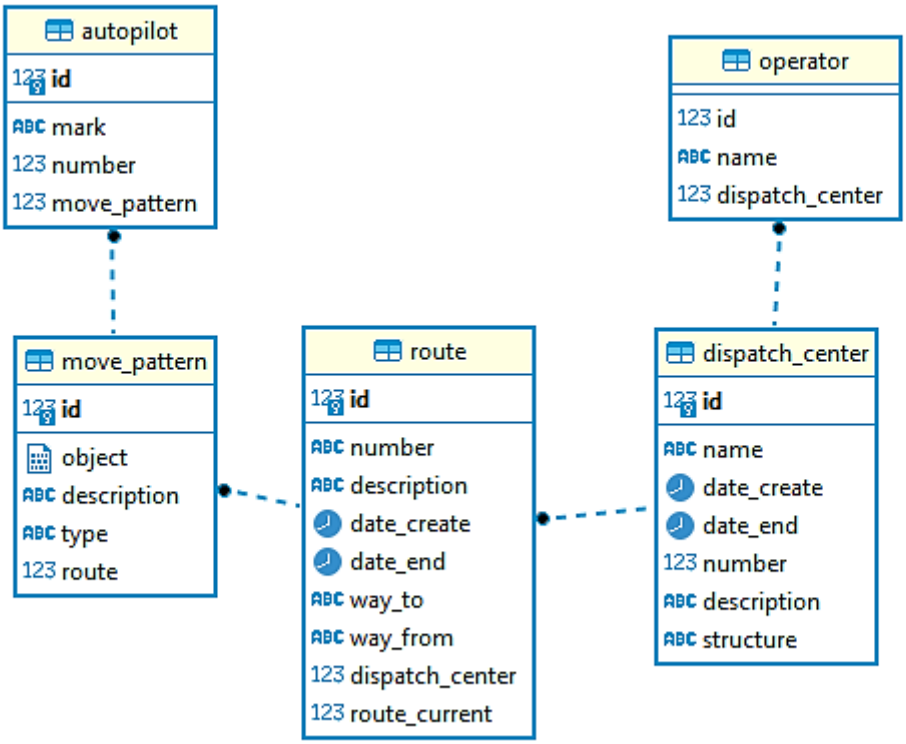


Рис. Диаграмма хранилища

ПРИЛОЖЕНИЕ В

Обобщенный программный код на python

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import r2_score, mean_absolute_error
import pandas as pd
from typing import Dict, List, Tuple, Optional
import warnings
from heapq import heappush, heappop
import time
from dataclasses import dataclass

warnings.filterwarnings('ignore')

np.random.seed(42)
torch.manual_seed(42)

@dataclass
class OptimizationMetrics:
    efficiency: float = 0.0
    quality_improvement: float = 0.0
    reliability: float = 0.0
    time_nn: float = 0.0
    time_hybrid: float = 0.0
    performance_nn: float = 0.0
    performance_hybrid: float = 0.0


class CriticalDBOptimizer:
    def __init__(self, num_features=6):
        self.num_features = num_features
        self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
        print(f"Используется устройство: {self.device}")

        self.feature_names = [
            'Время выполнения', 'Использование сети', 'Потребление CPU',
            'Использование памяти', 'Сложность запросов', 'Интенсивность операций'
        ]
        self.critical_limits = {
            'execution_time_max': 0.8,
            'network_usage_max': 0.7,
            'cpu_usage_max': 0.9,
            'memory_usage_min': 0.4,
            'query_complexity_max': 0.75,
            'operation_intensity_max': 0.8
        }

    def generate_realistic_performance_data(self, num_samples=1500):
        """Генерация реалистичных данных производительности БД"""
```

```
    X = np.zeros((num_samples, self.num_features))
    Y = np.zeros(num_samples)

    for i in range(num_samples):
        # Создаем коррелированные параметры
        base_load = np.random.uniform(0.2, 0.9)

        # Реалистичные зависимости
        execution_time = base_load * 0.7 + np.random.uniform(0.1, 0.3)
        network_usage = base_load * 0.6 + np.random.uniform(0.1, 0.2)
        cpu_usage = base_load * 0.8 + np.random.uniform(0.05, 0.15)
        memory_usage = 0.6 - base_load * 0.3 + np.random.uniform(-0.1, 0.1)
        query_complexity = base_load * 0.5 + np.random.uniform(0.2, 0.4)
        operation_intensity = base_load * 0.9 + np.random.uniform(0.05, 0.1)

        params = np.array([
            min(execution_time, 0.95),
            min(network_usage, 0.9),
            min(cpu_usage, 0.95),
            max(min(memory_usage, 0.9), 0.1),
            min(query_complexity, 0.85),
            min(operation_intensity, 0.9)
        ])
        X[i] = params
        Y[i] = self.calculate_performance(params)

    return X.astype(np.float32), Y.astype(np.float32)

def calculate_performance(self, params):
    """Расчет производительности системы с учетом ограничений"""
    base_performance = (
            0.25 * (1 - params[0]) +  # Время выполнения (меньше лучше)
            0.2 * (1 - params[1]) +  # Использование сети
            0.15 * (1 - params[2]) +  # Потребление CPU
            0.25 * params[3] +  # Использование памяти (больше лучше)
            0.1 * (1 - params[4]) +  # Сложность запросов
            0.05 * (1 - params[5])  # Интенсивность операций
    )

    # Штрафы за нарушение ограничений
    penalties = 0.0
    if params[0] > self.critical_limits['execution_time_max']:
   penalties += (params[0] - self.critical_limits['execution_time_max']) * 2.0

    if params[1] > self.critical_limits['network_usage_max']:
    penalties += (params[1] - self.critical_limits['network_usage_max']) * 1.5

    if params[2] > self.critical_limits['cpu_usage_max']:
        penalties += (params[2] - self.critical_limits['cpu_usage_max']) * 2.5

    if params[3] < self.critical_limits['memory_usage_min']:
  penalties += (self.critical_limits['memory_usage_min'] - params[3]) * 1.0

    if params[4] > self.critical_limits['query_complexity_max']:
  penalties += (params[4] - self.critical_limits['query_complexity_max']) * 1.2

    if params[5] > self.critical_limits['operation_intensity_max']:
```

```
    penalties += (params[5] - self.critical_limits['operation_intensity_max']) * 1.0

        final_performance = base_performance - penalties
        return max(0.1, min(0.95, final_performance))

    def prepare_data(self, num_samples=1500, test_size=0.2):
        X, Y = self.generate_realistic_performance_data(num_samples)

        split_idx = int(len(X) * (1 - test_size))
        indices = np.random.permutation(len(X))

        train_idx, test_idx = indices[:split_idx], indices[split_idx:]
        X_train, X_test = X[train_idx], X[test_idx]
        Y_train, Y_test = Y[train_idx], Y[test_idx]

        X_train_tensor = torch.from_numpy(X_train).to(self.device)
        Y_train_tensor = torch.from_numpy(Y_train).unsqueeze(1).to(self.device)
        X_test_tensor = torch.from_numpy(X_test).to(self.device)
        Y_test_tensor = torch.from_numpy(Y_test).unsqueeze(1).to(self.device)

        print("=== ДАННЫЕ ДЛЯ ОПТИМИЗАЦИИ БД ===")
        print(f"Обучающая выборка: {X_train.shape[0]} образцов")
        print(f"Тестовая выборка: {X_test.shape[0]} образцов")
        print(f"Диапазон производительности: {Y.min():.3f} - {Y.max():.3f}")

        return (X_train_tensor, Y_train_tensor, X_test_tensor, Y_test_tensor)


class DBPerformancePredictor(nn.Module):
    def __init__(self, input_dim, hidden_dims=[128, 64, 32], dropout_rate=0.2):
        super().__init__()

        layers = []
        prev_dim = input_dim

        for hidden_dim in hidden_dims:
            layers.extend([
                nn.Linear(prev_dim, hidden_dim),
                nn.ReLU(),
                nn.Dropout(dropout_rate)
            ])
            prev_dim = hidden_dim

        layers.extend([
            nn.Linear(prev_dim, 1),
            nn.Sigmoid()
        ])

        self.net = nn.Sequential(*layers)

    def forward(self, x):
        return self.net(x)


class NeuralNetworkOptimizer:
    """Оптимизатор на основе только нейросети"""
```

```
def __init__(self, critical_limits, device):
    self.critical_limits = critical_limits
    self.device = device

def train_model(self, X_train, Y_train, num_epochs=300):
    """Обучение нейросетевой модели"""
    model = DBPerformancePredictor(X_train.shape[1]).to(self.device)
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-4)
    criterion = nn.MSELoss()

    model.train()
    train_losses = []

    for epoch in range(num_epochs):
        optimizer.zero_grad()
        predictions = model(X_train)
        loss = criterion(predictions, Y_train)
        loss.backward()
        optimizer.step()
        train_losses.append(loss.item())

    return model, train_losses

def optimize_with_nn_only(self, X_train, Y_train, X_test, Y_test):
    """Оптимизация только с помощью нейросети"""
    start_time = time.time()

    print("Обучение нейросетевой модели...")
    model, train_losses = self.train_model(X_train, Y_train)

    # Поиск оптимальных параметров через расширенный случайный поиск
    model.eval()
    best_performance = -float('inf')
    best_params = None

    print(" Поиск оптимальных параметров...")
    with torch.no_grad():
        # Используем стратегию адаптивного случайного поиска
        for strategy in range(3):
            if strategy == 0:
                # Базовая стратегия: равномерный случайный поиск
                candidates = 1000
                search_space = [0.1, 0.9]
            elif strategy == 1:
# Стратегия 2: поиск в областях с высокой производительностью
                candidates = 500
                search_space = [0.3, 0.8]  # Более узкий диапазон
            else:
                # Стратегия 3: тонкая настройка вокруг лучшего решения
                candidates = 300
                if best_params is not None:
                    # Ищем вокруг лучшего решения
                    base = best_params.cpu().numpy()
                    search_space = [max(0.1, base[i] - 0.1) for i in range(6)], \
                        [min(0.9, base[i] + 0.1) for i in range(6)]
                else:
                    continue
```

```
                for i in range(candidates):
                    if strategy == 2:
                        # Для стратегии 3 генерируем вокруг лучшего решения
                        candidate = best_params.clone()
                        noise = torch.randn_like(candidate) * 0.05
                        candidate = torch.clamp(candidate + noise, 0.1, 0.9)
                    else:
                        # Для стратегий 1 и 2 - равномерный поиск
                        if strategy == 0:
                candidate = torch.rand(X_train.shape[1], device=self.device) * 0.8 + 0.1
                        else:
                            low, high = search_space
                            candidate = torch.rand(X_train.shape[1], device=self.device) * (high - low) + low

                    # Проверяем реальную производительность
                    real_performance = self.calculate_real_performance(candidate)

                    if real_performance > best_performance:
                        best_performance = real_performance
                        best_params = candidate.clone()

                print(f"  Стратегия {strategy + 1}: лучшая производительность = {best_performance:.4f}")

            nn_time = time.time() - start_time

            print(f"  Итог: производительность = {best_performance:.4f}")
            return best_params, best_performance, model, train_losses, nn_time

        def calculate_real_performance(self, params):
            """Расчет реальной производительности"""
            params_np = params.cpu().numpy()

            base_performance = (
                0.25 * (1 - params_np[0]) +
                0.2 * (1 - params_np[1]) +
                0.15 * (1 - params_np[2]) +
                0.25 * params_np[3] +
                0.1 * (1 - params_np[4]) +
                0.05 * (1 - params_np[5])
            )

            penalties = 0.0
            if params_np[0] > self.critical_limits['execution_time_max']:
                penalties += (params_np[0] - self.critical_limits['execution_time_max']) * 2.0

            if params_np[1] > self.critical_limits['network_usage_max']:
                penalties += (params_np[1] - self.critical_limits['network_usage_max']) * 1.5

            if params_np[2] > self.critical_limits['cpu_usage_max']:
                penalties += (params_np[2] - self.critical_limits['cpu_usage_max']) * 2.5

            if params_np[3] < self.critical_limits['memory_usage_min']:
                penalties += (self.critical_limits['memory_usage_min'] - params_np[3]) * 1.0

            if params_np[4] > self.critical_limits['query_complexity_max']:
```

```
            penalties += (params_np[4] - self.critical_limits['query_complexity_max']) * 1.2

        if params_np[5] > self.critical_limits['operation_intensity_max']:
            penalties += (params_np[5] - self.critical_limits['operation_intensity_max']) * 1.0

        final_performance = base_performance - penalties
        return max(0.1, min(0.95, final_performance))


class IntelligentHybridOptimizer:
    """Интеллектуальный гибридный оптимизатор: нейросеть + метод ветвей и границ"""

    def __init__(self, critical_limits, device):
        self.critical_limits = critical_limits
        self.device = device
        self.performance_cache = {}

    def evaluate_performance(self, params):
        """Быстрая оценка производительности с кэшированием"""
        if isinstance(params, torch.Tensor):
            params_np = params.detach().cpu().numpy()
        else:
            params_np = params

        params_key = tuple(np.round(params_np, 4))
        if params_key in self.performance_cache:
            return self.performance_cache[params_key]

        performance = self.calculate_real_performance(params_np)
        self.performance_cache[params_key] = performance
        return performance

    def calculate_real_performance(self, params):
        """Расчет реальной производительности"""
        base_performance = (
                0.25 * (1 - params[0]) +
                0.2 * (1 - params[1]) +
                0.15 * (1 - params[2]) +
                0.25 * params[3] +
                0.1 * (1 - params[4]) +
                0.05 * (1 - params[5])
        )

        penalties = 0.0
        if params[0] > self.critical_limits['execution_time_max']:
            penalties += (params[0] - self.critical_limits['execution_time_max']) * 2.0

        if params[1] > self.critical_limits['network_usage_max']:
            penalties += (params[1] - self.critical_limits['network_usage_max']) * 1.5

        if params[2] > self.critical_limits['cpu_usage_max']:
            penalties += (params[2] - self.critical_limits['cpu_usage_max']) * 2.5

        if params[3] < self.critical_limits['memory_usage_min']:
            penalties += (self.critical_limits['memory_usage_min'] - params[3]) * 1.0

        if params[4] > self.critical_limits['query_complexity_max']:
```

```
 penalties += (params[4] - self.critical_limits['query_complexity_max']) * 1.2

    if params[5] > self.critical_limits['operation_intensity_max']:
penalties += (params[5] - self.critical_limits['operation_intensity_max']) * 1.0

    final_performance = base_performance - penalties
    return max(0.1, min(0.95, final_performance))

def neural_network_guided_optimization(self, initial_params, model, max_iter=50):
    """Интеллектуальная оптимизация с руководством от нейросети"""
    start_time = time.time()

    best_solution = initial_params.clone()
    best_performance = self.evaluate_performance(initial_params)

    print(" Запуск интеллектуальной гибридной оптимизации...")
    print(f"   Начальная производительность: {best_performance:.4f}")

    history = []

    for iteration in range(max_iter):
        improvements_found = 0

        # Стратегия 1: Локальный поиск с руководством от нейросети
        with torch.no_grad():
            current_tensor = best_solution.unsqueeze(0)
            current_pred = model(current_tensor).item()

            # Генерируем кандидатов на основе предсказаний нейросети
            candidates = []

            # Адаптивный поиск в разных направлениях
            for dim in range(6):
                for direction in [-1, 1]:
                    candidate = best_solution.clone()
                    step_size = 0.05 * (1.0 - iteration / max_iter) + 0.01  # Уменьшаем шаг со временем
                    candidate[dim] += direction * step_size
                    candidate = torch.clamp(candidate, 0.1, 0.9)

                    # Быстрая оценка нейросетью
                    pred_perf = model(candidate.unsqueeze(0)).item()
                    candidates.append((candidate, pred_perf))

            # Добавляем случайные кандидаты для исследования
            for _ in range(10):
                noise = torch.randn_like(best_solution) * 0.1
                candidate = torch.clamp(best_solution + noise, 0.1, 0.9)
                pred_perf = model(candidate.unsqueeze(0)).item()
                candidates.append((candidate, pred_perf))

        # Оцениваем лучших кандидатов по реальной производительности
        candidates.sort(key=lambda x: x[1], reverse=True)
        top_candidates = candidates[:5]  # Берем топ-5 по предсказаниям

        for candidate, pred_perf in top_candidates:
            real_perf = self.evaluate_performance(candidate)
```

```
        # Шаг 2: Уточняем решение интеллектуальной гибридной оптимизацией
            print("Фаза 2: Уточнение решения гибридным методом...")
            final_params, final_performance, search_history, search_time = self.neural_network_guided_optimization(
                initial_params, model, max_iter=30
            )

            total_time = nn_time + search_time

            # Расчет улучшения
            improvement = final_performance - initial_performance
            improvement_percent = (improvement / initial_performance) * 100 if initial_performance > 0 else 0

            history = {
                'initial_performance': initial_performance,
                'final_performance': final_performance,
                'improvement': improvement,
                'improvement_percent': improvement_percent,
                'nn_losses': train_losses,
                'search_history': search_history,
                'timing': {
                    'nn_time': nn_time,
                    'search_time': search_time,
                    'total_time': total_time
                }
            }

            return final_params, final_performance, history, model

        def create_comprehensive_comparison(optimizer, nn_results, hybrid_results):
            """Создание комплексного сравнения методов"""
            fig = plt.figure(figsize=(18, 12))

            # 1. Основное сравнение производительности
            ax1 = plt.subplot2grid((2, 3), (0, 0), colspan=2)
            methods = ['Только\nнейросеть', 'Гибридный\nметод']
            performances = [nn_results['performance'], hybrid_results['performance']]

            bars = ax1.bar(methods, performances, color=['#ff6b6b', '#51cf66'], alpha=0.8, edgecolor='black')
            ax1.set_ylabel('Производительность системы', fontweight='bold', fontsize=12)
            ax1.set_title('СРАВНЕНИЕ КАЧЕСТВА РЕШЕНИЙ', fontweight='bold', fontsize=14)
            ax1.grid(True, alpha=0.3, axis='y')

            for bar, perf in zip(bars, performances):
                ax1.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
                         f'{perf:.4f}', ha='center', va='bottom', fontweight='bold', fontsize=11)

            # 2. Улучшение от гибридизации
            ax2 = plt.subplot2grid((2, 3), (0, 2))
            improvement = hybrid_results['improvement']
            improvement_percent = hybrid_results['improvement_percent']

            color = 'green' if improvement > 0 else 'red'
            ax2.bar(['Улучшение'], [improvement], color=color, alpha=0.7, edgecolor='black')
            ax2.set_ylabel('Абсолютное улучшение', fontweight='bold', fontsize=12)
            ax2.set_title('ЭФФЕКТИВНОСТЬ ГИБРИДИЗАЦИИ', fontweight='bold', fontsize=14)
```

```
        ax2.grid(True, alpha=0.3, axis='y')
        ax2.axhline(y=0, color='black', linestyle='-', alpha=0.5)

        ax2.text(0, improvement + (0.01 if improvement > 0 else -0.01),
                 f'{improvement:+.4f}\n({improvement_percent:+.1f}%)',
                 ha='center', va='bottom' if improvement > 0 else 'top',
                 fontweight='bold', fontsize=11, color=color)

        # 3. Временные характеристики
        ax3 = plt.subplot2grid((2, 3), (1, 0))
        times_nn = nn_results['time']
        times_hybrid = hybrid_results['timing']['total_time']

        time_breakdown = [times_nn, times_hybrid]
        time_labels = ['Только\nнейросеть', 'Гибридный\nметод']

        bars = ax3.bar(time_labels, time_breakdown, color=['#4ecdc4', '#45b7d1'], alpha=0.8,
edgecolor='black')
        ax3.set_ylabel('Время выполнения (секунды)', fontweight='bold', fontsize=12)
        ax3.set_title('СРАВНЕНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ', fontweight='bold', fontsize=14)
        ax3.grid(True, alpha=0.3, axis='y')

        for bar, time_val in zip(bars, time_breakdown):
            ax3.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.1,
                     f'{time_val:.1f}c', ha='center', va='bottom', fontweight='bold', fontsize=11)

        # 4. Процесс улучшения в гибридном методе
        ax4 = plt.subplot2grid((2, 3), (1, 1))
        if hybrid_results['search_history']:
            iterations = [x['iteration'] for x in hybrid_results['search_history']]
            performances = [x['best_performance'] for x in hybrid_results['search_history']]

            ax4.plot(iterations, performances, 'b-', linewidth=2, marker='o', markersize=4)
            ax4.axhline(y=hybrid_results['initial_performance'], color='red', linestyle='--',
                        linewidth=2, label=f'Начальное: {hybrid_results["initial_performance"]:.4f}')
            ax4.set_xlabel('Итерация гибридной оптимизации')
            ax4.set_ylabel('Лучшая производительность')
            ax4.set_title('ПРОЦЕСС УЛУЧШЕНИЯ В ГИБРИДНОМ МЕТОДЕ', fontweight='bold',
fontsize=14)
            ax4.legend()
            ax4.grid(True, alpha=0.3)

            # Показываем финальное улучшение
            final_improvement = performances[-1] - hybrid_results['initial_performance']
            ax4.text(0.05, 0.95, f'Финальное улучшение: {final_improvement:.4f}',
                     transform=ax4.transAxes, bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.8),
                     fontsize=10)

        # 5. Метрики эффективности
        ax5 = plt.subplot2grid((2, 3), (1, 2))
        efficiency = (1 - hybrid_results['timing']['total_time'] / nn_results['time']) * 100
        quality_improvement = hybrid_results['improvement_percent']

        metrics = ['Эффективность\n(E)', 'Улучшение\nкачества (Q)']
        values = [efficiency, quality_improvement]
        colors = ['lightblue', 'lightgreen']
```

```
        bars = ax5.bar(metrics, values, color=colors, alpha=0.7, edgecolor='black')
        ax5.set_ylabel('Значение (%)', fontweight='bold', fontsize=12)
        ax5.set_title('МЕТРИКИ ЭФФЕКТИВНОСТИ', fontweight='bold', fontsize=14)
        ax5.grid(True, alpha=0.3, axis='y')

        for bar, value in zip(bars, values):
            color = 'green' if value > 0 else 'red'
            ax5.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 1,
                    f'{value:+.1f}%', ha='center', va='bottom', fontweight='bold', color=color)

        plt.tight_layout()
        return fig


    def run_complete_analysis():
        """Запуск полного анализа"""
        print("ЗАПУСК ПОЛНОГО СРАВНИТЕЛЬНОГО АНАЛИЗА")
        print("=" * 80)

        # Инициализация
        optimizer = CriticalDBOptimizer(num_features=6)

        # Подготовка данных
        print(" Подготовка данных...")
        train_data = optimizer.prepare_data(num_samples=1200, test_size=0.15)
        X_train, Y_train, X_test, Y_test = train_data

        # Запуск оптимизации только нейросетью
        print("\nЗАПУСК ОПТИМИЗАЦИИ ТОЛЬКО НЕЙРОСЕТЬЮ...")
        nn_optimizer = NeuralNetworkOptimizer(optimizer.critical_limits, optimizer.device)
        nn_params, nn_performance, nn_model, nn_losses, nn_time =
nn_optimizer.optimize_with_nn_only(
            X_train, Y_train, X_test, Y_test
        )

        # Запуск интеллектуального гибридного метода
        print("\nЗАПУСК ИНТЕЛЛЕКТУАЛЬНОГО ГИБРИДНОГО МЕТОДА...")
        hybrid_optimizer = IntelligentHybridOptimizer(optimizer.critical_limits, optimizer.device)
        hybrid_params, hybrid_performance, hybrid_history, hybrid_model =
hybrid_optimizer.optimize_hybrid(
            X_train, Y_train, X_test, Y_test
        )

        # Подготовка результатов
        nn_results = {
            'performance': nn_performance,
            'time': nn_time,
            'params': nn_params,
            'losses': nn_losses
        }

        hybrid_results = {
            'performance': hybrid_performance,
            'initial_performance': hybrid_history['initial_performance'],
            'improvement': hybrid_history['improvement'],
            'improvement_percent': hybrid_history['improvement_percent'],
            'timing': hybrid_history['timing'],
```

```
            'search_history': hybrid_history['search_history'],
            'params': hybrid_params
        }
        # Создание визуализаций
        print("\nСоздание визуализаций...")
        fig = create_comprehensive_comparison(optimizer, nn_results, hybrid_results)

        # Анализ результатов
        print("\n" + "=" * 80)
        print("□ ДЕТАЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ")
        print("=" * 80)

        print(f"\n РЕЗУЛЬТАТЫ ОПТИМИЗАЦИИ:")
        print(f"   • Только нейросеть:         {nn_performance:.4f}")
        print(f"   • Гибридный метод:          {hybrid_performance:.4f}")
        print(f"   • Улучшение:                {hybrid_history['improvement']:+.4f}")
        print(f"   • Относительное улучшение:   {hybrid_history['improvement_percent']:+.2f}%")

        print(f"\n ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ:")
        print(f"   • Нейросеть:                {nn_time:.2f}c")
        print(f"Гибридный метод:          {hybrid_history['timing']['total_time']:.2f}c")
        print(f"    • Эффективность (E):            {(1 - hybrid_history['timing']['total_time'] / nn_time) *
100:+.1f}%")

        print(f"\n АНАЛИЗ ГИБРИДИЗАЦИИ:")
        if hybrid_history['improvement_percent'] > 2:
            print("ГИБРИДИЗАЦИЯ УСПЕШНА: интеллектуальная интеграция")
            print("      методов показывает значительное улучшение!")
        elif hybrid_history['improvement_percent'] > 0:
            print("  Гибридизация показывает небольшое улучшение")
        else:
            print(" Гибридизация не показала улучшения")

        return fig, nn_results, hybrid_results
    # --- ЗАПУСК ПРОГРАММЫ ---
    if __name__ == "__main__":
        try:
            fig, nn_results, hybrid_results = run_complete_analysis()

            # Сохранение результатов
            plt.savefig('hybrid_optimization_comparison.png',
                        dpi=300, bbox_inches='tight', facecolor='white')
            print(f"\n□ Результаты сохранены в 'hybrid_optimization_comparison.png'")

            plt.show()
            print(f"\nАНАЛИЗ УСПЕШНО ЗАВЕРШЕН!")

        except Exception as e:
            print(f" Ошибка при выполнении анализа: {e}")
            import traceback

            traceback.print_exc()
```

ПРИЛОЖЕНИЕ С

УТВЕРЖДАЮ
Директор Общества с ограниченной
ответственностью «Крым Диджитал Груп»
_____ О.В. Ермаков

«19» декабря 2025 г.

АКТ

о внедрении результатов диссертационной работы Мустафаевой Мерьем Ибраимовны на тему «Модели поддержки принятия решений по повышению производительности критических баз данных», по специальности 1.2.2 – «Математическое моделирование, численные методы и комплексы программ».

Техническая комиссия в составе: директора Калинина Ильи Алексеевича старшего инженера-программиста, Воловик Романа Владимировича старшего инженера-программиста, Доманецкого Даниила Константиновича старшего инженера-программиста, составила настоящий акт в том, что результаты диссертационной работы Мустафаевой М.И. - модели поддержки принятия решений по повышению производительности критических баз данных, позволили повысить эффективность по следующим направлениям деятельности - 61.02 «Разработка компьютерного программного обеспечения»:

- повысить скорость принятия решений и их эффективность при проектировании информационных взаимодействий в телекоммуникационных сетях предприятия;
- повысить производительность и надежность взаимодействия беспилотных летательных аппаратов с диспетчерским центром за счет оптимизации запросов к базам данных и прогнозирования их выполнения;
- реализованная система поддержки принятия решений обеспечивает эффективное регулирование нагрузок на КБД в условиях реального времени, что критически важно для устойчивой работы всей комплексной интегрированной системы «БПЛА-ДЦ».

Применение предложенных в диссертационной работе Мустафаевой М.И. моделей и алгоритмов оптимизации информационных взаимодействий с критическими базами данных при реализации базовых функциональных задач, позволяет повысить качество ситуационного прогнозирования при выработке решений в процессах проектирования и эксплуатации сетей предприятия, повысить надежность и разрабатываемых систем.

старший инженер-программист _____ Калинин И.А.
старший инженер-программист _____ Воловик Р.В.
старший инженер-программист _____ Доманецкий Д.К.