МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Анализ и разработка алгоритма резервного копирования данных в информационных системах»

Обучающийся К.В. Химич _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, О.В. Аникина
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема выпускной работы: «Анализ и разработка алгоритма резервного копирования данных в информационных системах»

Работа выполнена в объеме 43 страницы, включая 32 иллюстрации и 2 таблицы. Структура включает введение, три раздела, заключение, список использованной литературы.

Основным вопросом работы является разработка эффективного алгоритма для резервного копирования баз данных в информационных системах, обеспечивающего надежность и восстановление данных в случае сбоев.

Цель работы заключается в проектировании и анализе алгоритма резервного копирования для обеспечения устойчивости информационных систем, на примере конкретной базы данных.

Бакалаврская работа может быть структурирована следующим образом: анализ существующих методов резервного копирования; разработка требований к системе резервного копирования; выбор алгоритма и программного обеспечения; тестирование и оценка эффективности разработанного решения.

В завершении работы представлен обзор успешного использования алгоритмов резервного копирования в отечественной информационной системе, что позволило повысить степень защиты данных и обеспечить оперативное восстановление после сбоев.

В заключение следует подчеркнуть, что данная работа актуальна для повышения надежности и безопасности информационных систем, что является важным аспектом в современной информационной технологии.

Abstract

The title of the graduation work is «Analysis and Development of a Data Backup Algorithm in Information Systems». The graduation work consists of an introduction, three chapters, a conclusion, a list of references, and appendices, encompassing a total of 43 pages with 32 illustrations and 2 tables.

The key issue addressed in the graduation work is the analysis and development of a reliable data backup algorithm tailored for information systems to ensure data integrity and availability in case of data loss or corruption. The work focuses on creating an efficient backup algorithm that leverages modern technologies and meets the current needs for data protection and disaster recovery.

The aim of the work is to provide detailed information on the development of a robust data backup algorithm that enables information systems to efficiently backup and restore data, minimizing downtime and data loss. Additionally, the work offers strategies for scheduling backups, verifying backup integrity, and managing backup storage.

The graduation work can be divided into several logically connected parts: analysis of existing data backup algorithms, design of the new backup algorithm architecture, implementation of functional and non-functional requirements, and testing of the algorithm for reliability and performance.

As a result of the work, a comprehensive data backup algorithm has been developed that allows information systems to safeguard their data through efficient backup and recovery processes. The algorithm includes features such as automated backup scheduling, data integrity checks, and user-friendly management interfaces, thereby assisting systems in maintaining data continuity and security.

In conclusion, the results of the work are summarized, demonstrating that the developed data backup algorithm is an effective tool for data protection, contributing to enhanced data management and disaster recovery capabilities for information systems.

Содержание

Введение

В мире, где информация становится неотъемлемой частью нашей повседневной жизни, сохранность данных становится вопросом первостепенной важности. С каждым днем объемы цифровой информации, создаваемой, передаваемой и хранимой в информационных системах, постоянно растут. От личных фотографий до стратегически важных корпоративных данных, все это требует надежной защиты и безопасного хранения [10]. В такой ситуации понимание и эффективное применение методов резервного копирования данных становится ключевым элементом обеспечения устойчивости и непрерывности работы информационных систем.

Каждый день мы сталкиваемся с потенциальными угрозами, которые могут привести к потере данных: сбои оборудования, программные ошибки, вирусы, а также физические чрезвычайные ситуации, такие как пожары или наводнения. В случае отсутствия адекватной стратегии резервного копирования, эти угрозы могут иметь разрушительные последствия для организаций и частных лиц [6].

Вопросы, связанные с анализом и разработкой алгоритмов резервного копирования данных, остаются актуальными и востребованными в современном информационном обществе. Постоянное развитие технологий и изменение угроз требуют постоянного совершенствования методов и подходов к обеспечению безопасности и сохранности информации [14].

Актуальность темы исследования подтверждается недавним случаем с известной компанией. Технический сбой в IT-инфраструктуре компании был вызван хакерской атакой. В результате этой атаки была нарушена работоспособность клиентских сервисов, что привело к значительным убыткам, оцениваемым экспертами в сумму от 300 млн до 1 млрд рублей. Этот инцидент демонстрирует необходимость разработки эффективных методов защиты и восстановления данных, чтобы минимизировать ущерб и обеспечить непрерывность бизнес-процессов.

Цель работы – анализ и разработка эффективного алгоритма резервного копирования данных, который обеспечит надежную защиту информации в информационных системах.

Объект исследования – информационные системы, используемые для хранения и обработки данных.

Предмет исследования – алгоритмы резервного копирования данных в информационных системах.

Работа состоит из трех разделов.

В первом разделе проводится анализ современных технологий резервного копирования данных, включая изучение основных методов и подходов, а также рассмотрение программного обеспечения и инструментов.

Во втором разделе осуществляется разработка алгоритма резервного копирования данных, начиная с определения требований к нему на основе проведенного анализа. Затем происходит проектирование алгоритма и разработка программного обеспечения на его основе, включая создание необходимых модулей и интерфейса.

В третьем разделе проводится тестирование и анализ разработанного алгоритма. Для этого подготавливается тестовый набор данных, проводится тестирование программного обеспечения на различных сценариях и условиях, а затем анализируются результаты тестирования для оценки эффективности и надежности разработанного алгоритма резервного копирования данных.

1 Анализ современных технологий резервного копирования данных

1.1 Основные понятия и методология резервного копирования

Резервное копирование - это процесс создания дубликатов данных (бэкапов), обеспечивающий их сохранность и доступность в случае потери или повреждения оригинальных файлов или системы [4].

Резервные копии данных важны по ряду причин, рассмотренных ниже.

Аварийное восстановление: Когда оборудование отказывает, данные повреждаются или доступ к серверу пропадает по какой-либо причине, необходимо провести аварийное восстановление. Это может быть вызвано случайной ошибкой пользователя, кибератакой, сбоем сервера или другими факторами. Хотя вероятность конкретных аварий невелика, они могут произойти внезапно и требуют быстрой реакции для минимизации потерь данных [15].

Изменение решения: Часто пользователи удаляют данные, а затем осознают необходимость их восстановления. Это может быть вызвано ошибкой или изменением обстоятельств. Резервное копирование данных обеспечивает возможность вернуть удаленные данные в первоначальное состояние, способствуя сохранению целостности информации и продолжению бизнес-процессов [15].

Аудит: Иногда требуется восстановить данные или схему на определенный момент времени для анализа или проверки. Например, в случае юридических споров или обнаружения ошибок в программном обеспечении необходимо иметь доступ к данным на определенный момент времени. Резервные копии позволяют восстановить состояние данных на прошлые даты для анализа и документирования [15].

Тестирование: Проведение тестирования на реальных данных играет важную роль в обеспечении качества программного обеспечения. Периодическое обновление тестового сервера с использованием последних

рабочих данных облегчается благодаря возможности быстрого восстановления из резервных копий. Это помогает обнаружить и исправить ошибки до их появления в рабочей среде и обеспечивает более надежное функционирование программных продуктов [15].

Существует два метода резервного копирования: горячее и холодное.

Горячее резервное копирование – это метод создания резервных копий данных, который выполняется во время активной работы системы. Этот подход позволяет минимизировать прерывание работы системы, что особенно важно для бизнес-процессов, требующих непрерывной доступности данных. Горячее копирование обеспечивает создание копий с наименьшим временем простоя. Также этот метод обладает гибкостью в планировании процесса копирования, что позволяет адаптировать его к особенностям работы конкретной системы и бизнес-потребностям. Недостатком горячего копирования является более высокая нагрузка на ресурсы системы во время выполнения процесса копирования [13].

Холодное резервное копирование – это метод создания резервных копий данных, который использует периоды минимальной активности или приостановки работы системы. В отличие от горячего копирования, которое выполняется во время активной работы системы, холодное копирование происходит в моменты, когда система находится в состоянии покоя. Такой подход позволяет создавать более полные и точные копии данных, так как отсутствует риск изменений данных в процессе копирования. Однако для проведения холодного копирования требуется планирование периодов, когда система неактивна, что потребует дополнительных ресурсов и времени [17].

В резервном копировании играют ключевую роль два важных понятия: «целевая точка восстановления» (Recovery Point Objective, RPO) и «целевое время восстановления» (Recovery Time Objective, RTO) [15]. Они определяются в рамках стратегии восстановления после сбоя и помогают оценить, насколько быстро и эффективно можно восстановить систему и данные после инцидента.

RPO определяет максимально допустимую потерю данных в случае сбоя. Это время, измеряемое в периодах, за которое данные могут быть утрачены перед восстановлением. Например, если RPO равно одному часу, это означает, что в случае сбоя информация может быть восстановлена только до момента, наступившего за один час до сбоя. RPO определяется в зависимости от важности данных и бизнес-процессов [9].

RTO, с другой стороны, определяет максимально допустимое время простоя системы после сбоя. Это время, измеряемое в часах или минутах, за которое система должна быть восстановлена и полностью функционировать после сбоя. Например, если RTO равно четырем часам, это означает, что система должна быть полностью восстановлена и работоспособна в течение четырех часов после сбоя [18].

1.2 Способы резервного копирования

Существует два основных способа резервного копирования данных: логическое и копирование исходных файлов.

Логическое копирование данных, также известное как создание дампа, представляет собой метод резервного копирования, при котором данные сохраняются в виде файловой структуры или специального формата, который может быть восстановлен при необходимости. Этот метод обычно используется для резервного копирования баз данных или приложений, где сохраняется состояние системы в целом [7].

Логическое копирование данных обладает рядом преимуществ:

– удобство обработки: Файлы логической копии могут быть обработаны в стандартных текстовых редакторах и инструментах командной строки, что упрощает их просмотр и восстановление;

– гибкость восстановления: Для восстановления данных из логической копии достаточно передать файл по конвейеру на вход программы MySQL или воспользоваться программой mysqlimport;
– возможность работы по сети: Резервное копирование и восстановление данных можно выполнять по сети, что удобно при работе на удаленных машинах или в облачных сервисах;
– настройка процедуры: Инструмент mysqldump, который часто используется для создания логических копий, обладает множеством параметров, что позволяет гибко настраивать процесс резервного копирования;
– независимость от подсистем хранения: Логическая копия данных не зависит от конкретной подсистемы хранения, что позволяет легко перемещать данные между различными типами таблиц.

Однако есть и недостатки:

– загрузка процессора: Для создания логической копии требуется, чтобы сервер работал, что может повлечь за собой дополнительную нагрузку на процессор;
– объемность данных: Логические копии могут оказаться объемнее физических файлов из-за ASCII-представления данных, что требует больше места для хранения и дополнительных ресурсов для сжатия;
– потеря точности: При копировании и восстановлении данных из логической копии может происходить потеря точности из-за ошибок при интерпретации команд и перестройке индексов;
– затраты на восстановление: Восстановление данных из логической копии может быть трудоемким процессом, требующим много времени и дополнительных ресурсов сервера.

Физическое копирование данных представляет собой процесс создания точной копии физических файлов, содержащих данные, без интерпретации их

содержимого. Этот метод копирования включает в себя копирование самих файлов базы данных, журналов транзакций и других системных файлов, не зависящих от структуры данных [3].

Преимущества физического копирования данных включают:

- высокая производительность: Физическое копирование обычно происходит быстрее, чем логическое, так как не требует интерпретации данных и перекодировки;
- точность восстановления: Поскольку данные копируются без изменений, восстановление из физической копии обеспечивает точное восстановление исходного состояния данных;
- меньшие затраты процессорного времени: Физическое копирование не требует дополнительных вычислительных ресурсов для интерпретации данных и обработки команд;
- простота использования: Для создания физической копии часто используются стандартные инструменты файловой системы или программы управления базами данных, что делает процесс создания копии относительно простым и надежным.
- Однако у физического копирования также есть свои недостатки:
- зависимость от аппаратного обеспечения: Физическое копирование может быть связано с риском утери данных в случае отказа аппаратного обеспечения или повреждения файловой системы;
- больший объем хранения: Физические копии могут занимать больше места на диске, так как сохраняют полную копию всех файлов базы данных, включая неиспользуемое пространство;
- сложности восстановления частичных данных: Восстановление отдельных объектов или таблиц из физической копии может быть затруднено, так как данные хранятся в виде больших файлов, и

требуется полное восстановление всей копии для доступа к отдельным элементам.

Работать с физическими копиями обычно проще и эффективнее. Но не следует целиком полагаться на них при необходимости долговременного хранения или удовлетворения требований законодательства, время от времени нужно делать и логические копии [15].

1.3 Виды резервного копирования

Принято выделять три основных типа резервных копий.

Полные. Полная копия — это резервная копия системы с сохранением всех данных [16]. Данный метод создает полную копию набора исходных данных, поэтому является лучшим вариантом защиты с точки зрения управления и скорости восстановления данных. Обычно, полные резервные копии делают периодически и объединяют их с другими типами резервного копирования [1].

Дифференциальные. В дифференциальной резервной копии сохраняются различия между текущим состоянием системы и ее последней полной копией. Для восстановления из дифференциальной копии нужно, чтобы последняя по времени полная копия также была рабочей [16].

Инкрементные. В инкрементной резервной копии сохраняются различия между текущим состоянием системы и любой последней по времени резервной копией. Последняя копия может быть полной, дифференциальной или тоже инкрементной – для инкрементного копирования это не важно [16].

Схематичные представления полного, дифференциального и инкрементного резервных копирований представлены на рисунках 1, 2 и 3, соответственно.

Рисунок 1 – Полное резервное копирование

Рисунок 2 – Дифференциальное резервное копирование

Рисунок 3 – Инкрементное резервное копирование

При выборе одного из подходов к резервному копированию данных нужно выявить их индивидуальные преимущества и недостатки. Для оценки эффективности каждого метода была использована трехбалльная система. Она позволила выделить основные критерии и оценить соответствие методов требованиям. Была использована трехбалльная система оценивания, где «1» - это слабая оценка, а «3» - сильная. Полученные результаты представлены в таблице 1.

Таблица 1 – Достоинства и недостатки типов резервного копирования

Критерий оценивания	Полное резервное копирование	Дифференциальное резервное копирование	Инкрементное резервное копирование
Частота резервного копирования и кол-во точек восстановления (RPO)	1	2	3
Время на восстановление последней резервной копии (RTO)	3	2	1
Необходимый объем хранилища резервных копий	1	2	3
Нагрузка на сеть передачи данных	1	2	3
Требования к вычислительной мощности и оперативной памяти	3	2	2

Из данных таблицы можно понять, что каждый из подходов имеет свои положительные стороны, и должен применяться в зависимости от требований задачи.

Выводы по разделу 1

В рамках бакалаврской работы в первом разделе был проведен анализ современных технологий резервного копирования данных. Были изучены основные понятия, способы резервного копирования и их виды. Всё это является неотъемлемой частью разработки алгоритма резервного копирования баз данных.

2 Разработка алгоритма резервного копирования данных

2.1 Определение требований к разрабатываемому алгоритму

Для успешной разработки алгоритма необходимо четко определить требования, которым должен соответствовать алгоритм. В соответствии с предоставленными условиями заказчика, были выделены функциональные и нефункциональные требования.

К функциональным требованиям относятся:

- автоматизированное создание резервных копий данных по заданному расписанию;
- возможность настройки расписания выполнения бэкапов;
- сохранение резервных копий на облачный сервер;
- уведомление об успешном завершении процесса создания резервных копий;
- уведомление о невозможности выполнения резервного копирования с указанием причины (например, кончилось место на диске);
- наличие скрипта для восстановления данных из резервной копии на Яндекс Диске в указанную базу данных за выбранное число;
- опциональная возможность восстановления только определенной таблицы вместо всей базы данных;
- возможность настройки списка таблиц, которые необходимо исключить из резервного копирования.

К нефункциональным требованиям относятся:

- производительность: выполнение резервного копирования и восстановления данных должно быть эффективным и не вызывать значительных задержек в работе системы;

- надежность: обеспечение целостности и доступности резервных копий данных, а также надежность уведомлений о состоянии процесса;
- гибкость: возможность настройки и адаптации процесса резервного копирования под изменяющиеся потребности и условия использования;
- безопасность: обеспечение конфиденциальности и защиты данных во время их передачи и хранения;
- масштабируемость: способность алгоритма работать с различными объемами данных и адаптироваться к росту их количества.

Также в требованиях заказчик указал использовать в качестве облачного сервера Яндекс Диск.

Яндекс Диск – это облачное хранилище файлов, предоставляемое компанией Яндекс. Пользователи могут загружать свои файлы на серверы Яндекса, сохранять их там и иметь к ним доступ из любого места, подключенного к интернету. Яндекс Диск предоставляет возможность хранения файлов различных форматов, организации их в папки, обмена файлами с другими пользователями, а также синхронизации данных между различными устройствами, такими как компьютеры, смартфоны и планшеты. Пользователи могут управлять своими файлами через веб-интерфейс, специальные приложения для различных операционных систем и доступные API.

Дополнительно к этому было указано отправлять уведомления в Telegram.

Telegram – это мессенджер и социальная сеть, созданные компанией Telegram Messenger LLP. Он позволяет пользователям обмениваться сообщениями, проводить звонки, создавать групповые чаты, каналы и ботов.

Боты в Telegram – это программы, которые могут выполнять различные задачи автоматизации и обеспечивать доступ к разнообразной информации и сервисам. Они могут отправлять уведомления, отвечать на команды, предоставлять информацию о погоде, новостях, курсах валют, проводить опросы, игры и многое другое.

2.2 Обзор программного обеспечения для резервного копирования данных

В настоящее время существует множество инструментов резервного копирования. Предоставим краткую характеристику некоторых из них.

MySQL Enterprise Backup – этот инструмент, ранее известный как InnoDB Hot Backup, или ibbackup, является частью подписки на MySQL Enterprise от Oracle. Его использование не требует остановки MySQL, установки блокировок или прерывания нормальной работы базы данных, хотя и вызовет некоторое увеличение нагрузки на ваш сервер. Он поддерживает такие возможности, как изготовление сжатых резервных копий, инкрементное резервное копирование и потоковое резервное копирование на другой сервер. Это официальный инструмент резервного копирования для MySQL [15].

Percona XtraBackup - это бесплатный и открытый инструмент для резервного копирования баз данных MySQL и MariaDB. Он разработан компанией Percona и предоставляет возможность создавать резервные копии баз данных без блокировки таблиц или прерывания их работы, что обеспечивает непрерывность операций. Percona XtraBackup поддерживает такие функции, как инкрементное копирование, сжатие данных и шифрование, обеспечивая гибкость и безопасность процесса резервного копирования. Этот инструмент является популярным выбором среди администраторов баз данных благодаря своей надежности, производительности и простоте использования.

Mylvmbackup - это утилита резервного копирования для MySQL, которая создает копии баз данных, используя логическое томное устройство LVM (Logical Volume Manager). Она позволяет создавать копии данных без блокировки таблиц или прерывания работы базы данных, что обеспечивает непрерывность операций. mylvmbackup позволяет создавать полные или инкрементальные резервные копии, а также поддерживает сжатие данных для экономии места на диске. Этот инструмент обеспечивает простоту управления и конфигурации, делая процесс резервного копирования удобным и эффективным.

Zmanda Recovery Manager (ZRM) - это программное обеспечение для резервного копирования и восстановления данных, специально разработанное для баз данных MySQL. ZRM предоставляет обширные возможности по созданию и управлению резервными копиями, включая полные, инкрементальные и дифференциальные резервные копии, а также возможность управления жизненным циклом копий данных.

Рассмотрим, насколько каждое программное обеспечение соответствует конкретным требованиям работодателя и какие из них лучше всего удовлетворяют поставленным критериям. Результаты оценивания представлены в таблице 2.

Таблица 2 – Сравнение ПО для резервного копирования

Критерий	MySQL Enterprise Backup	XtraBackup	Mylvmbackup	ZRM
Автоматизированное создание резервных копий данных по заданному расписанию	+	+	+	+
Сохранение резервных копий на облачный сервер	-	-	-	+
Уведомление об успешном завершении процесса создания резервных копий	+	+	-	+
Восстановление данных из резервной копии на облачном сервере	-	-	+	-

Продолжение таблицы 2

Критерий	MySQL Enterprise Backup	XtraBackup	Mylvmbackup	ZRM
Опциональная возможность восстановления только определенной таблицы вместо всей базы данных	+	+	+	+
Возможность настройки списка таблиц, которые необходимо исключить из резервного копирования	+	+	+	+

Исходя из данных таблицы, был сделан вывод, что ни одно из рассмотренных программных решений не полностью соответствует всем поставленным требованиям. Поэтому было принято решение разработки собственного программного обеспечения, которое будет учитывать все указанные требования и обладать необходимым функционалом.

2.3 Проектирование алгоритма резервного копирования данных

Для эффективного проектирования алгоритма важно иметь представление о его структуре и взаимосвязях между его элементами. В этом помогают различные методологии, среди которых выделяется IDEF0.

IDEF0 (Integration Definition for Function Modeling) – это методология функционального моделирования, которая используется для анализа и проектирования бизнес-процессов. С ее помощью можно создавать графические модели, отображающие взаимосвязи между различными функциями в системе или процессе.

Эти модели помогают визуализировать структуру процесса, определить его основные функции, входы, выходы и управляющие механизмы, а также выявить возможные улучшения [8].

На рисунке 4 представлена контекстная диаграмма алгоритма резервного копирования.

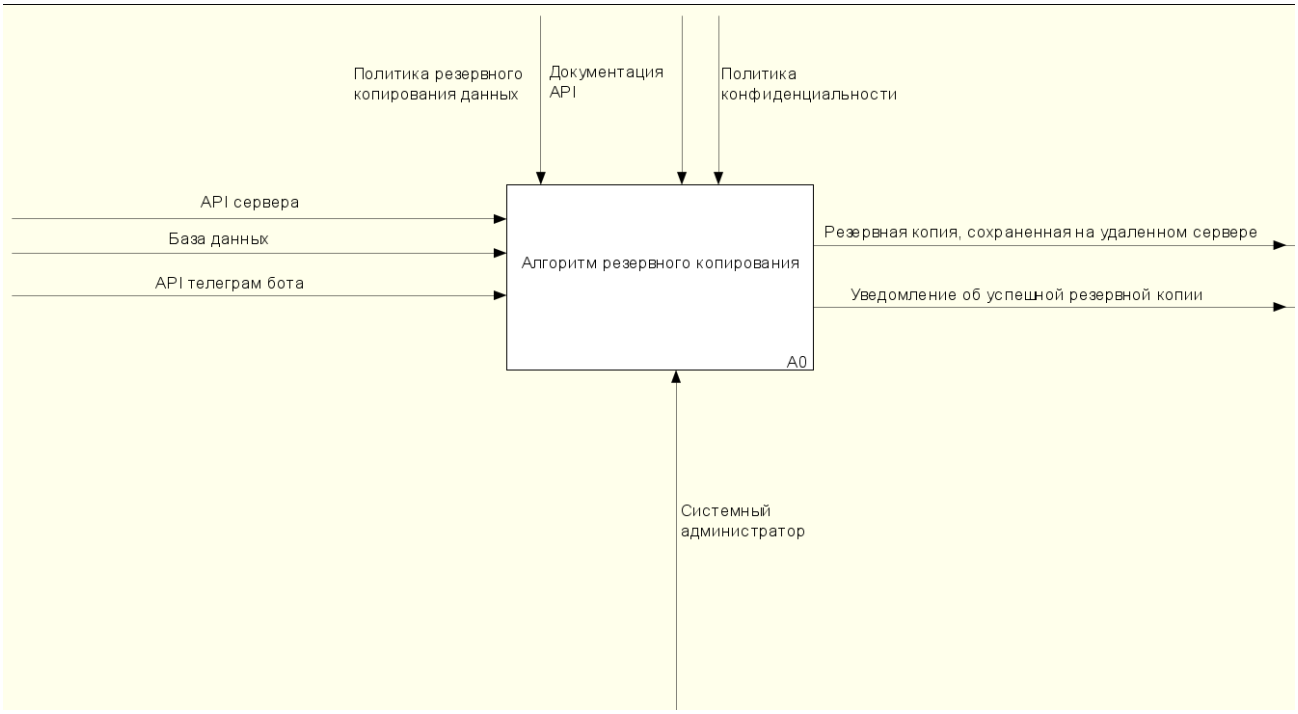

Рисунок 4 – Контекстная диаграмма

На данном рисунке представлена декомпозиция алгоритма резервного копирования.

Входные данные для алгоритма:

– база данных,

– API сервера,

– API телеграм-бота.

Выходные данные:

– резервная копия, сохраненная на сервере,

– уведомление об успешном выполнении резервного копирования.

Управление:

– политика резервного копирования данных,

– политика конфиденциальности,

– документация API.

В механизмы входит системный администратор.

На рисунке 5 представлена декомпозиция контекстной диаграммы.

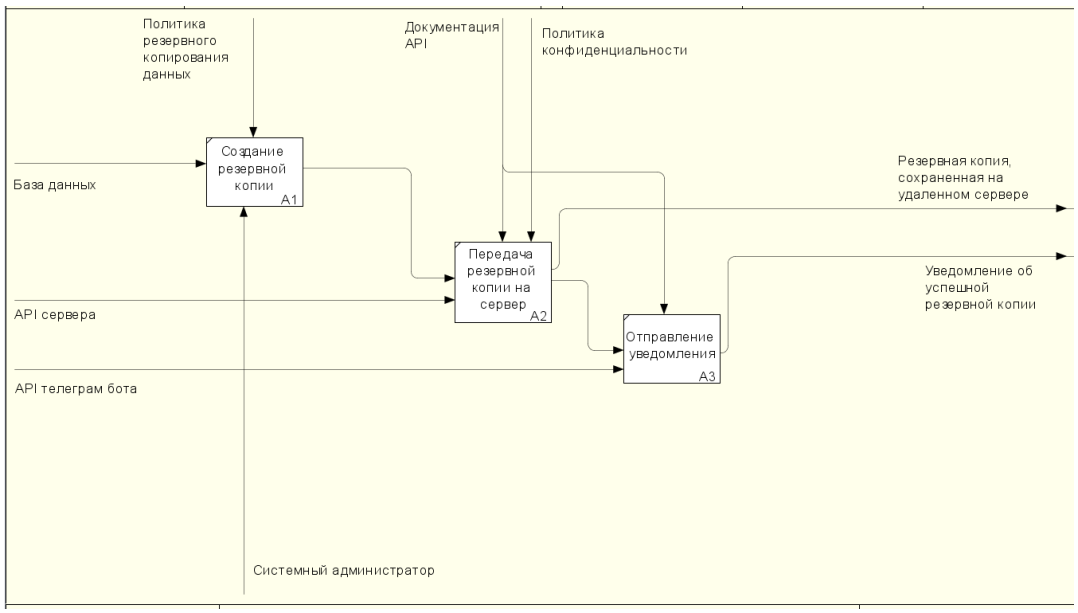

Рисунок 5 – Декомпозиция первого уровня

Эта диаграмма представляет собой подробное разделение процесса резервного копирования на три основных блока:

- создание резервной копии: В этом блоке осуществляется процесс создания резервной копии данных. Он включает в себя выполнение команд для резервного копирования баз данных;
- передача резервной копии на сервер: Этот блок отвечает за передачу созданной резервной копии на сервер;
- отправление уведомления: В данном блоке осуществляется отправка уведомления о выполненном процессе резервного копирования.

Для лучшей оптимизации алгоритм резервного копирования баз данных был разбит на четыре программы: полное копирование, инкрементное копирование, полное восстановление и инкрементное восстановление. Такой подход позволяет более детально и структурированно подойти к реализации процесса резервного копирования данных, упрощая его разработку, тестирование и последующую поддержку. Каждая программа отвечает за

выполнение конкретного этапа процесса, обеспечивая модульность и гибкость системы.

Алгоритм полного резервного копирования начинается с инициализации процесса, где считываются параметры из конфигурационного файла. Затем, для обеспечения целостности данных, текущий бинарный журнал закрывается, предотвращая потерю информации во время копирования.

Следующий этап включает создание полной копии всех баз данных и таблиц. После этого создается архив или другой тип резервной копии, который затем загружается на облачное хранилище, например, Яндекс Диск. По завершении загрузки пользователю отправляется уведомление в Telegram о завершении процесса резервного копирования.

Данный алгоритм представлен на рисунке 6. Программный код алгоритма полного резервного копирования представлен в приложении А.

Рисунок 6 – Алгоритм полного резервного копирования

Инкрементное резервное копирование начинается с чтения конфигурационного файла. Затем выполняется создание инкрементной резервной копии, где копируются только те данные, которые изменились с момента последнего полного или инкрементного копирования. Эта инкрементная копия загружается на Яндекс Диск. Пользователю отправляется уведомление в Telegram о завершении процесса копирования. В завершение, текущий бинарный журнал закрывается для обеспечения целостности данных. Алгоритм представлен на рисунке 7. Программный код алгоритма инкрементного резервного копирования представлен в приложении Б.

Рисунок 7 – Алгоритм инкрементного резервного копирования

Процесс полного восстановления данных начинается с чтения конфигурационного файла, который содержит параметры восстановления. После этого резервная копия базы данных загружается с Яндекс Диска. Восстановленные данные из резервной копии возвращаются в исходное состояние базы данных. Пользователь получает уведомление о завершении процесса восстановления через Телеграм. Этот алгоритм показан на рисунке 8. Программный код алгоритма полного восстановления базы данных представлен в приложении В.

Рисунок 8 – Алгоритм полного восстановления базы данных

Инкрементное восстановление начинается с инициализации процесса и чтения конфигурационного файла. Последняя инкрементная резервная копия базы данных загружается с Яндекс Диска. Затем данные из инкрементной

резервной копии восстанавливаются в базу данных, обновляя ее актуальной информацией. Пользователю отправляется уведомление о завершении процесса восстановления через Telegram. Алгоритм представлен на рисунке 9.

Программный код алгоритма восстановления инкрементной копии представлен в приложении Г.

Рисунок 9 – Алгоритм восстановления инкрементной копии

Для наглядной демонстрации всех возможностей, предоставляемых пользователю, была разработана диаграмма вариантов использования. Эта диаграмма иллюстрирует, какие действия и операции пользователь может выполнять. Она визуально отображает весь функционал системы, подчеркивая основные этапы взаимодействия пользователя с процессом управления

данными и обеспечивая четкое понимание всех доступных опций [12]. Диаграмма представлена на рисунке 10.

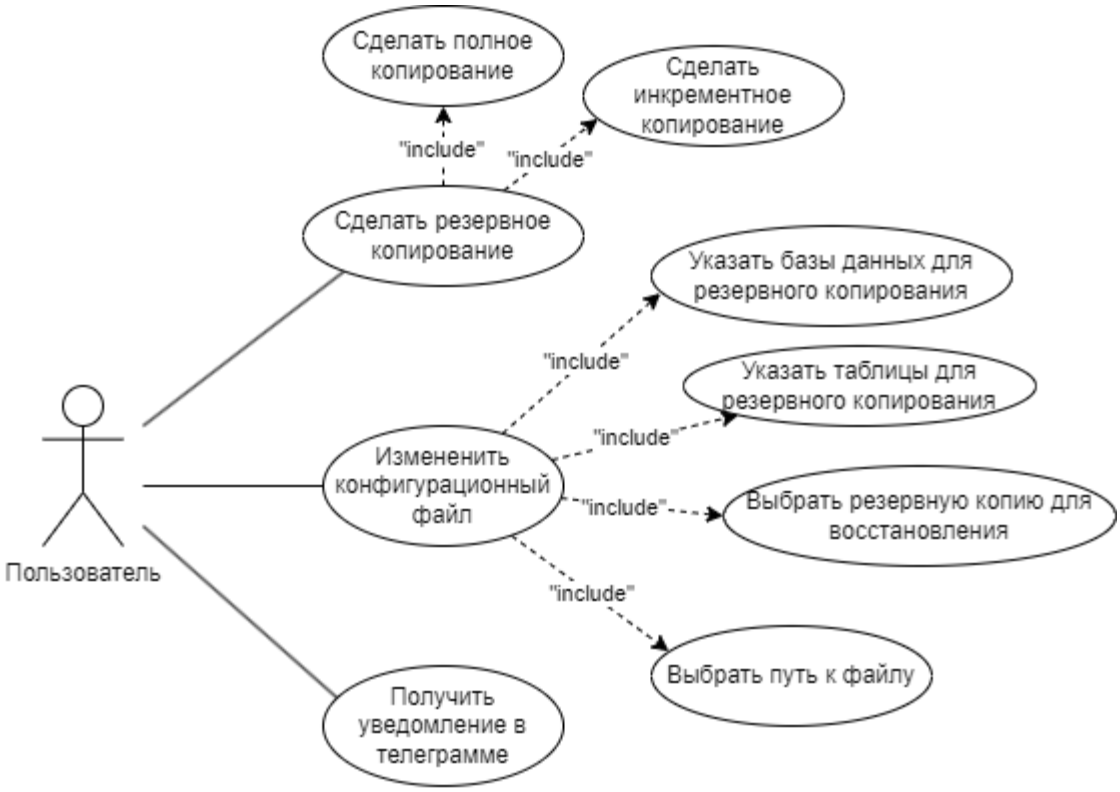

Рисунок 10 – Диаграмма вариантов использования

Процесс проектирования алгоритма резервного копирования завершён, и теперь можно переходить к этапу его реализации.

Выводы по разделу 2

В рамках бакалаврской работы во втором разделе была проведена разработка алгоритма резервного копирования данных. Были определены требования к разрабатываемому алгоритму. Был сделан обзор программного обеспечения для резервного копирования и спроектирован алгоритм резервного копирования.

3 Реализация и тестирование разработанного алгоритма

3.1 Разработка программного кода

После тщательного анализа и проектирования алгоритма резервного копирования данных наступила фаза его реализации и тестирования. В этом разделе представлен процесс создания программного кода, который будет осуществлять резервное копирование баз данных. Код будет разработан на языке Python, выбранном благодаря его гибкости, мощным инструментам обработки данных и широкой поддержке сообщества разработчиков [5].

Вся работа по реализации системы резервного копирования производилась на удалённом сервере. Для подключения к серверу использовался протокол SSH (Secure Shell), который обеспечивает защищённый доступ к серверу через сеть. Подключение осуществлялось с использованием SSH-ключей, что позволяет повысить уровень безопасности по сравнению с традиционным использованием паролей [20].

Для реализации алгоритма были необходимы следующие инструменты и библиотеки:

- Python3: Интерпретатор языка программирования Python версии три, который является основой для разработки и выполнения всех скриптов и программ, реализованных в рамках данного проекта;
- telegram-send: Python-библиотека, позволяющая отправлять уведомления и сообщения в Telegram через командную строку или скрипты;
- ydcmd: Библиотека для работы с API Yandex Disk, которая позволяет выполнять различные операции с файлами на облачном хранилище Яндекс Диска.

Далее были разработаны программы на основе спроектированных блок-схем, представленных во втором разделе.

Для визуализации работы программ была разработана диаграмма компонентов. Диаграмма компонентов используется для отображения высокоуровневой структуры системы и взаимодействий между ее различными частями или модулями. Она показывает, как компоненты системы связаны друг с другом и каким образом они взаимодействуют, чтобы обеспечить выполнение функциональности системы.

На рисунке 11 представлена диаграмма компонентов.

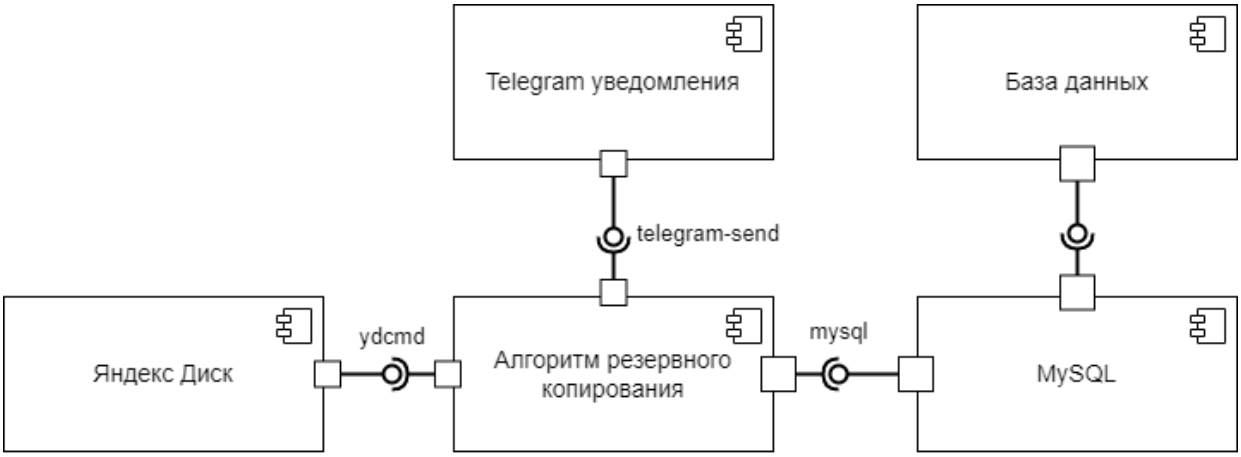

Рисунок 11 – Диаграмма компонентов

Ниже представлены основные функции каждой из программ.

На рисунке 12 представлена функция отправки уведомлений через Телеграм.

```
def send_telegram_message(message):
    try:
        command = f"telegram-send '{message}'"
        subprocess.run(command, shell=True, check=True)
        print("Уведомление успешно отправлено в Telegram.")
    except Exception as e:
        print(f"Ошибка при отправке уведомления в Telegram: {e}")
```

Рисунок 12 – Функция отправки уведомлений

Функция send_telegram_message предназначена для автоматической отправки сообщений в мессенджер Telegram из Python-скрипта. Внутри функции формируется команда для отправки заданного текстового сообщения, используя переданный аргумент message.

Если отправка сообщения прошла успешно, функция выводит уведомление о том, что сообщение успешно доставлено в Telegram. В случае возникновения ошибки (например, связанных с настройками сети или необходимыми установками), функция отлавливает исключение и выводит сообщение об ошибке для дальнейшего анализа или обработки.

На рисунке 13 представлена функция закрытия двоичного журнала.

```
def flush_logs(mysql_password):
    try:
        subprocess.run(f"mysqladmin -u root -p'{mysql_password}' flush-logs", shell=True, check=True)
        print("Бинарные логи MySQL успешно закрыты.")
    except Exception as e:
        print(f"Ошибка при закрытии бинарных логов MySQL: {e}")
```

Рисунок 13 – Функция закрытия двоичного журнала

Функция flush_logs предназначена для выполнения операции с MySQL, а именно закрытия бинарных логов. В начале функции формируется команда для выполнения операции flush-logs с помощью утилиты mysqladmin [11]. Если операция успешно выполнена (бинарные логи MySQL закрыты), функция выводит сообщение «Бинарные логи MySQL успешно закрыты.» в консоль. В случае возникновения ошибки (например, неправильный пароль, проблемы с доступом или другие причины), программа отлавливает исключение и выводит сообщение об ошибке, содержащее текст ошибки (e), что позволяет оператору или разработчику анализировать и исправлять проблему.

На рисунке 14 представлена функция чтения конфигурационного файла.

```
def read_config(file_path):
    with open(file_path, 'r') as file:
        config_data = json.load(file)
    return config_data
```

Рисунок 14 – Функция чтения конфигурационного файла

Функция read_config предназначена для чтения данных из файла конфигурации в формате JSON. В начале функции открывается указанный файл в режиме чтения. Затем данные из файла загружаются с помощью функции json.load, которая автоматически разбирает содержимое файла JSON и возвращает его в виде Python-структуры данных.

На рисунке 15 представлена функция загрузки резервной копии на Яндекс Диск.

```
def upload_to_yandex_disk(backup_file, yandex_disk_backup_directory):
    try:
command = f"ydcmd put {backup_file} {yandex_disk_backup_directory}"
subprocess.run(command, shell=True, check=True)
print(f"Файл резервной копии {backup_file} успешно загружен на Яндекс.Диск в каталог {yandex_disk_backup_directory}")
    except Exception as e:
error_message = f"Ошибка при загрузке файла резервной копии {backup_file} на Яндекс.Диск: {e}"
print(error_message)
send_telegram_message(error_message)
```

Рисунок 15 – Функция загрузки резервной копии на Яндекс Диск

Функция upload_to_yandex_disk предназначена для загрузки файлов резервных копий на Яндекс.Диск. В блоке try-except осуществляется попытка выполнения команды ydcmd put, которая используется для загрузки указанного файла backup_file в указанный каталог на Яндекс.Диске yandex_disk_backup_directory. Если операция завершается успешно, выводится сообщение о успешной загрузке файла. В случае возникновения ошибки (например, недоступность сервера Яндекс.Диска или некорректные

параметры), исключение перехватывается, формируется сообщение об ошибке и отправляется уведомление в Telegram через функцию send_telegram_message.

На рисунке 16 представлена функция для выгрузки базы данных с Яндекс Диска.

```
def download_from_yandex_disk(backup_file_name, yandex_disk_backup_directory):
    try:
        command = f"ydcmd get '{yandex_disk_backup_directory}{backup_file_name}'"
        subprocess.run(command, shell=True, check=True)
        print(f"Файл {backup_file_name} успешно скачан с Яндекс.Диска.")
        return backup_file_name
    except Exception as e:
        error_message = f"Ошибка при скачивании файла {backup_file_name} с Яндекс.Диска: {e}"
        print(error_message)
        send_telegram_message(error_message)
        return None
```

Рисунок 16 – Функция скачивания базы данных из Яндекс Диска

Функция download_from_yandex_disk предназначена для загрузки файлов с Яндекс.Диска. Она формирует команду для скачивания указанного файла из заданного каталога на Яндекс.Диске с использованием утилиты ydcmd. Если операция завершается успешно, функция выводит сообщение о успешном скачивании и возвращает имя скачанного файла. В случае ошибки функция ловит исключение, выводит сообщение об ошибке и отправляет уведомление в Telegram, а затем возвращает None.

На рисунке 17 представлена функция полного резервного копирования.

```python
def backup_mysql(database_name, tables, backup_directory, mysql_password):
    try:
        timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
        backup_file = os.path.join(backup_directory, f"{database_name}_{timestamp}.sql")
        tables_str = ' '.join(tables)
        command = f"mysqldump -u root -p'{mysql_password}' {database_name} {tables_str} > {backup_file}"
        subprocess.run(command, shell=True, check=True)
        print(f"Резервное копирование базы данных {database_name} успешно завершено.")
        return backup_file
    except Exception as e:
        error_message = f"Ошибка при резервном копировании базы данных {database_name}: {e}"
        print(error_message)
        send_telegram_message(error_message)
        return None
```

Рисунок 17 – Функция полного резервного копирования

Функция backup_mysql использует утилиту mysqldump для создания резервной копии базы данных MySQL. Она формирует команду с указанием имени пользователя, пароля, имени базы данных и списка таблиц для копирования [2]. Результат сохраняется в файл SQL с уникальным именем, включающим текущую временную метку. Если операция завершается успешно, выводится сообщение об успешном завершении операции. В случае ошибки выводится сообщение об ошибке и отправляется уведомление в Telegram.

На рисунке 18 представлена функция инкрементного резервного копирования.

```python
def backup_mysql_binlog(backup_directory, mysql_password):
    try:
        timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

        # Получение информации о текущем бинарном журнале
        result = subprocess.run(
            f"mysql -u root -p'{mysql_password}' -e 'SHOW MASTER STATUS;'",
            shell=True, check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, universal_newlines=True
        )

        # Обработка вывода SHOW MASTER STATUS
        lines = result.stdout.strip().split('\n')
        if len(lines) < 2:
            raise Exception("Не удалось получить информацию о бинарном журнале")

        # Распарсим данные
        binlog_info = lines[1].split()
        binlog_file = binlog_info[0]

        # Копирование текущего бинарного журнала в файл
        backup_file = os.path.join(backup_directory, f"{binlog_file}_{timestamp}.binlog")
        command = f"mysqlbinlog -s --user=root --password='{mysql_password}' /var/log/mysql/{binlog_file} > {backup_file}"
        subprocess.run(command, shell=True, check=True)

        print(f"Копирование бинарного журнала {binlog_file} успешно выполнено.")
        return backup_file
    except Exception as e:
        error_message = f"Ошибка при копировании бинарного журнала: {e}"
        print(error_message)
        send_telegram_message(error_message)
        return None
```

Рисунок 18 – Функция инкрементного резервного копирования

Функция backup_mysql_binlog предназначена для создания резервной копии текущего бинарного журнала MySQL. Она начинается с выполнения команды SHOW MASTER STATUS, чтобы получить информацию о текущем бинарном журнале. Полученные данные используются для определения имени текущего бинарного файла журнала [19]. Затем функция создает резервную копию бинарного журнала, используя утилиту mysqlbinlog. В случае успешного выполнения операции выводится сообщение о завершении, а при ошибке выводится сообщение об ошибке и отправляется уведомление в Telegram.

На рисунке 19 представлена функция полного восстановления базы данных.

```python
def restore_full_database_from_file(database_name, backup_file, mysql_password):
    try:
        command = f"mysql -u root -p'{mysql_password}' {database_name} < {backup_file}"
        subprocess.run(command, shell=True, check=True)
        print(f"Восстановление базы данных {database_name} из файла успешно завершено.")
    except Exception as e:
        print(f"Ошибка при восстановлении базы данных {database_name} из файла: {e}")
```

Рисунок 19 – Функция полного восстановления базы данных

Функция restore_full_database_from_file восстанавливает полную базу данных MySQL из предоставленного SQL-файла. Она использует команду mysql для выполнения SQL-скрипта из указанного файла backup_file. В случае успешного восстановления выводится сообщение об успешном завершении операции, а в случае ошибки выводится сообщение с описанием возникшей проблемы.

На рисунке 20 представлена функция инкрементного восстановления базы данных.

```python
def restore_table_from_binlog(mysql_password, binlog_file, database_name, table_name):
    try:
        # Команда для извлечения SQL из бинарного журнала только для указанной таблицы
        command = f"mysqlbinlog -D --database {database_name} --table {table_name} {binlog_file}|
        mysql -u root -p'{mysql_password}' {database_name}"
        subprocess.run(command, shell=True, check=True)
        print(f"Таблица {table_name} восстановлена из бинарного журнала {binlog_file}.")
    except Exception as e:
        error_message = f"Ошибка при восстановлении таблицы {table_name} из бинарного журнала: {e}"
        print(error_message)
        send_telegram_message(error_message)
```

Рисунок 20 – Функция инкрементного восстановления базы данных

Функция restore_table_from_binlog предназначена для восстановления конкретной таблицы из бинарного журнала MySQL. Она использует утилиту mysqlbinlog для извлечения SQL-запросов из указанного бинарного журнала только для заданной базы данных и таблицы. Полученные SQL-запросы передаются в команду mysql.

Для визуализации работы алгоритма резервного копирования была разработана диаграмма последовательности. Диаграмма последовательности используется для отображения взаимодействия между объектами системы в определенной последовательности, что позволяет понять порядок выполнения операций и обмена сообщениями между участниками процесса.

На рисунке 21 представлена диаграмма последовательности.

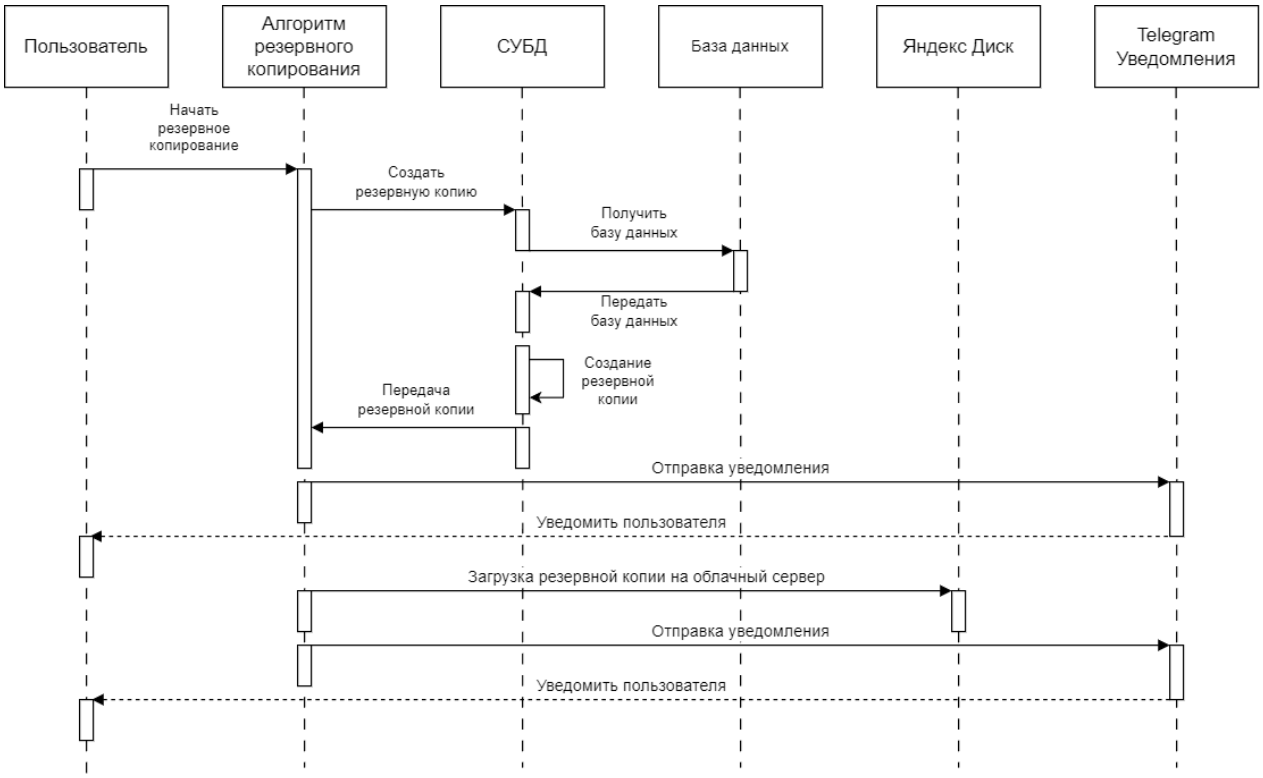

Рисунок 21 – Диаграмма последовательности алгоритма резервного копирования

После разработки функций для резервного копирования и восстановления данных в MySQL, следующим этапом является тестирование их работы. Для обеспечения надежности и эффективности алгоритмов в различных сценариях использования будет проведено тщательное тестирование.

3.2 Тестирование алгоритма

Для обеспечения надёжности и эффективности работы алгоритмов, разработанных для резервного копирования и восстановления данных в MySQL, было проведено тщательное тестирование. В этом подразделе представлены результаты тестирования каждой функции, а также анализ их работы в различных условиях. Целью тестирования было убедиться в правильности и стабильности функций в обработке больших объёмов данных и в различных сценариях использования.

Для проверки функциональности алгоритмов были созданы две тестовые базы данных: test_db1 и test_db2. Далее они были заполнены случайными данными. Результат представлен на рисунке 22.

```
+----+----------+-------------------++----+----------+-------+
| id | username | email             || id | name     | price |
+----+----------+-------------------++----+----------+-------+
|  1 | user1    | user1@example.com ||  1 | Product A | 19.99 |
|  2 | user2    | user2@example.com ||  2 | Product B | 29.99 |
|  3 | user3    | user3@example.com ||  3 | Product C |  9.99 |
+----+----------+-------------------++----+----------+-------+
```

Рисунок 22 – Таблицы баз данных для тестирования

После создания двух тестовых баз данных было выполнено полное резервное копирование, чтобы обеспечить сохранность их содержимого. Результат представлен на рисунках 23 и 24.

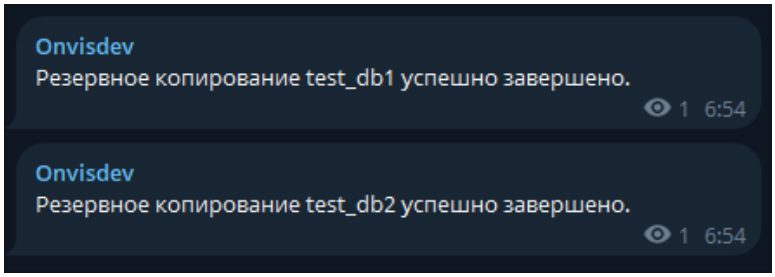

Рисунок 23 – Уведомления о полном резервном копировании

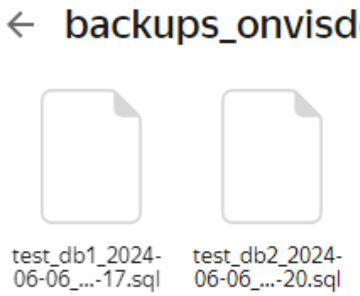

Рисунок 24 – Базы данных в Яндекс Диске

После создания полных резервных копий тестовых баз данных были внесены изменения в таблицы для подготовки к инкрементному резервному копированию. Результат представлен на рисунке 25.

id	username	email
1	user1	user1@example.com
2	user2	user2@example.com
3	user3	user3@example.com
4	newuser	newuser@example.com

id	name	price
1	Product A	19.99
2	Product B	29.99

Рисунок 25 – Измененные таблицы в тестовых базах данных

После внесения изменений в таблицы тестовых баз данных, было сделано инкрементное резервное копирование, результаты которого представлены на рисунках 26 и 27.

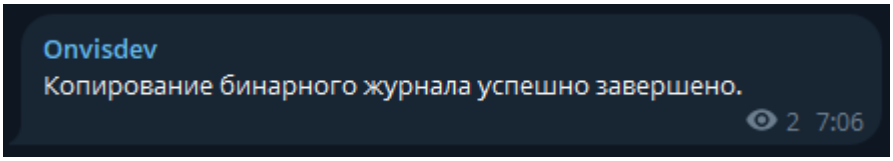

Рисунок 26 – Уведомление об инкрементном резервном копировании

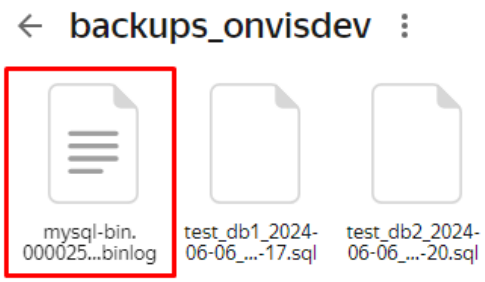

Рисунок 27 – Двоичный журнал на Яндекс Диске

После завершения инкрементного резервного копирования и перед дальнейшим восстановлением данных, было произведено удаление тестовых баз данных. Результат представлен на рисунке 28.

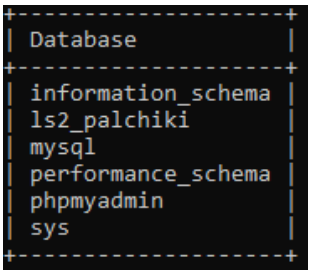

Рисунок 28 – Список баз данных

После удаления тестовых баз данных произведём их полное восстановление из ранее созданных полных резервных копий. Результаты представлены на рисунках 29 и 30.

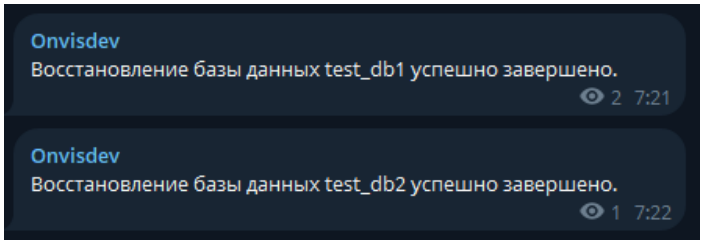

Рисунок 29 – Уведомления о восстановлении из полной резервной копии

```
mysql> select * from test_db1.users;
+----+----------+-------------------+
| id | username | email             |
+----+----------+-------------------+
|  1 | user1    | user1@example.com |
|  2 | user2    | user2@example.com |
|  3 | user3    | user3@example.com |
+----+----------+-------------------+
3 rows in set (0.00 sec)

mysql> select * from test_db2.products;
+----+-----------+-------+
| id | name      | price |
+----+-----------+-------+
|  1 | Product A | 19.99 |
|  2 | Product B | 29.99 |
|  3 | Product C |  9.99 |
+----+-----------+-------+
3 rows in set (0.01 sec)
```

Рисунок 30 – Восстановленные таблицы баз данных

После удаления исходных баз данных было выполнено восстановление данных из инкрементной резервной копии. Результат представлен на рисунках 31 и 32.

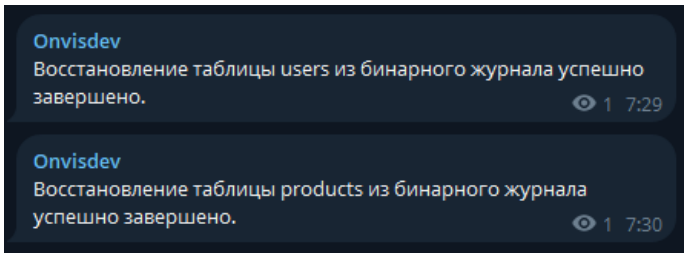

Рисунок 31 – Уведомления о восстановлении из инкрементной резервной копии

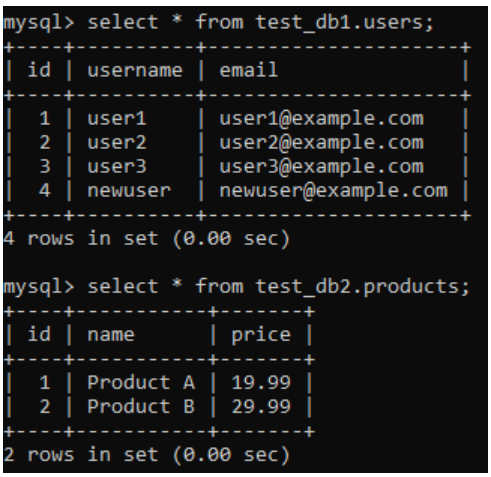

Рисунок 32 – Восстановленные таблицы из инкрементной резервной копии

Тестирование прошло успешно, и восстановление данных было завершено без проблем. Была продемонстрирована надёжность разработанного алгоритма резервного копирования, а также восстановления, обеспечивая защиту и целостность данных.

Выводы по разделу 3

В ходе работы были разработаны методы создания полных и инкрементных резервных копий. Тестирование алгоритма подтвердило его надёжность и эффективность, обеспечивая систему готовностью к обеспечению сохранности данных и оперативному восстановлению в случае необходимости.

Заключение

В данной работе была выполнена всесторонняя разработка алгоритма резервного копирования данных для информационных систем. Основной целью исследования было создание надежного и эффективного механизма, который обеспечивает целостность и доступность данных в случае их утраты или повреждения. На основе проведенного анализа современных технологий резервного копирования были выделены ключевые подходы и методы, используемые в данной области. Было рассмотрено множество существующих решений, что позволило выявить их сильные и слабые стороны. Это дало возможность сформулировать требования к разрабатываемому алгоритму, учитывающие актуальные потребности и вызовы в области защиты данных.

Разработанный алгоритм резервного копирования на языке Python представляет собой комплексное решение, включающее в себя следующие ключевые элементы:

- автоматизация процесса резервного копирования: Алгоритм обеспечивает возможность автоматического создания резервных копий в заданные временные интервалы, что минимизирует риск утраты данных из-за человеческого фактора;
- проверка целостности данных: Важной составляющей является механизм верификации резервных копий, что позволяет убедиться в их корректности и пригодности для восстановления;
- эффективное использование ресурсов: Предусмотрена оптимизация использования дискового пространства путем внедрения методов инкрементного и дифференциального копирования, что снижает объемы хранимых данных и ускоряет процесс резервного копирования;
- гибкость и масштабируемость: Алгоритм адаптирован для работы с различными типами данных и конфигурациями систем, что

делает его универсальным инструментом для разнообразных информационных систем.

В процессе тестирования алгоритм продемонстрировал высокую надежность и стабильность работы в различных сценариях использования. Это подтверждает его готовность к внедрению в реальных условиях и способность обеспечивать защиту данных в случае сбоев и аварийных ситуаций.

В ходе работы были также выявлены направления для дальнейшего развития проекта. Среди них можно отметить:

- расширение функциональности: Включение дополнительных возможностей, таких как шифрование данных и управление доступом, что повысит уровень безопасности и конфиденциальности резервных копий.
- оптимизация производительности: Дальнейшая оптимизация алгоритмов копирования и восстановления данных для сокращения времени выполнения операций и улучшения эффективности использования ресурсов системы.
- интеграция с другими системами управления данными: Разработка интерфейсов для взаимодействия с существующими системами управления базами данных и системами мониторинга, что упростит процесс интеграции и повысит общую эффективность управления данными.

В заключение, можно отметить, что разработанный алгоритм резервного копирования данных является важным шагом в направлении обеспечения безопасности и надежности хранения данных в информационных системах. Его внедрение позволит значительно повысить уровень защиты данных, снизить риски их утраты и обеспечить оперативное восстановление в случае аварийных ситуаций. Таким образом, данный алгоритм представляет собой значительный вклад в область защиты и управления данными, обеспечивая их доступность и целостность в современных информационных системах.

Список используемой литературы

1. Блюмин М.И. Системы управления базами данных. М.: Наука, 2010. - 352 с.
2. Головин М.Б., Крюков А.Ю. PostgreSQL: основы языка SQL. СПб.: БХВ-Петербург, 2015. - 640 с.
3. Губернаторов А.Ю., Серов Н.И. Архитектура баз данных. М.: МФТИ, 2018. - 296 с.
4. Журавлев А.А. Технологии резервного копирования. СПб.: Питер, 2016. - 432 с.
5. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
6. Иванов И.И. Современные методы защиты данных. М.: Юрайт, 2019. - 280 с.
7. Ковалев В.А. СУБД и системы управления данными. М.: Физматлит, 2014. - 400 с.
8. Кузнецов С.М. Проектирование баз данных. СПб.: БХВ-Петербург, 2017. - 520 с.
9. Михайлов А.В. Резервное копирование и восстановление данных. М.: НИУ ВШЭ, 2012. - 370 с.
10. Петров П.П. Информационная безопасность: учебник. М.: Юнити-Дана, 2015. - 448 с.
11. Сергеев В.Н. MySQL. Подробное руководство. М.: ДМК Пресс, 2016. - 750 с.
12. Смирнов В.А. Управление данными в информационных системах. М.: Юрайт, 2013. - 360 с.
13. Тарасов К.С. Резервное копирование в корпоративных сетях. СПб.: Питер, 2020. - 480 с.
14. Федоров А.Б. Защита информации и безопасность данных. СПб.: БХВ-Петербург, 2011. - 392 с.

15. Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму. 3-е изд. СПб.: Питер, 2018. - 864 с.

16. Acronis. Backup for Dummies®. Special Edition. 2020.

17. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th Edition. Pearson, 2014. - 1440 p.

18. Date C.J. An Introduction to Database Systems. 8th Edition. Pearson, 2003. - 1024 p.

19. Elmasri R., Navathe S.B. Fundamentals of Database Systems. 7th Edition. Pearson, 2016. - 1272 p.

20. Mullins C.S. Database Administration: The Complete Guide to Practices and Procedures. Addison-Wesley Professional, 2012. - 688 p.

Приложение А

Алгоритм полного резервного копирования

```
import os
import subprocess
import json
from datetime import datetime

def read_config(file_path):
    with open(file_path, 'r') as file:
        config_data = json.load(file)
    return config_data

def flush_logs(mysql_password):
    try:
        subprocess.run(f"mysqladmin -u root -p'{mysql_password}' flush-logs", shell=True, check=True)
        print("Бинарные логи MySQL успешно закрыты.")
    except Exception as e:
        print(f"Ошибка при закрытии бинарных логов MySQL: {e}")

def backup_mysql(database_name, tables, backup_directory, mysql_password):
    try:
        timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
        backup_file = os.path.join(backup_directory, f"{database_name}_{timestamp}.sql")
        tables_str = ' '.join(tables)
        command = f"mysqldump -u root -p'{mysql_password}' {database_name} {tables_str} > {backup_file}"
        subprocess.run(command, shell=True, check=True)
        print(f"Резервное копирование базы данных {database_name} успешно завершено.")
        return backup_file
    except Exception as e:
```

```
            error_message = f"Ошибка при резервном копировании базы данных {database_name}: {e}"
            print(error_message)
            send_telegram_message(error_message)
            return None

        def upload_to_yandex_disk(backup_file, yandex_disk_backup_directory):
            try:
                command = f"ydcmd put {backup_file} {yandex_disk_backup_directory}"
                subprocess.run(command, shell=True, check=True)
                print(f"Файл резервной копии {backup_file} успешно загружен на Яндекс.Диск в каталог {yandex_disk_backup_directory}")
            except Exception as e:
                error_message = f"Ошибка при загрузке файла резервной копии {backup_file} на Яндекс.Диск: {e}"
                print(error_message)
                send_telegram_message(error_message)

        def send_telegram_message(message):
            try:
                command = f"telegram-send '{message}'"
                subprocess.run(command, shell=True, check=True)
             print("Уведомление успешно отправлено в Telegram.")
            except Exception as e:
                print(f"Ошибка при отправке уведомления в Telegram: {e}")
        if __name__ == "__main__":
            config_data = read_config('config_file.json')
            mysql_password = config_data['mysql_password']
            backup_directory = config_data['backup_directory']
            yandex_disk_backup_directory = config_data['yandex_disk_backup_directory']
            databases = config_data['databases']
```

```
        # Закрытие текущего бинарного журнала MySQL
        flush_logs(mysql_password)

        for database_name, tables in databases.items():
            backup_file = backup_mysql(database_name, tables, backup_directory, mysql_password)
            if backup_file:
                upload_to_yandex_disk(backup_file, yandex_disk_backup_directory)
                send_telegram_message(f"Резервное копирование {database_name} успешно завершено.")
```

Приложение Б

Алгоритм инкрементного резервного копирования

```
import os
import subprocess
import json
from datetime import datetime

def read_config(file_path):
with open(file_path, 'r') as file:
    config_data = json.load(file)
  return config_data

def flush_logs(mysql_password):
  try:
    subprocess.run(f"mysqladmin -u root -p'{mysql_password}' flush-logs", shell=True, check=True)
    print("Бинарные логи MySQL успешно закрыты.")
  except Exception as e:
    print(f"Ошибка при закрытии бинарных логов MySQL: {e}")

def backup_mysql_binlog(backup_directory, mysql_password):
  try:
    timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

    # Получение информации о текущем бинарном журнале
    result = subprocess.run(
      f"mysql -u root -p'{mysql_password}' -e 'SHOW MASTER STATUS;'",
      shell=True, check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, universal_newlines=True
    )

    # Обработка вывода SHOW MASTER STATUS
    lines = result.stdout.strip().split('\n')
```

```
            if len(lines) < 2:

        raise Exception("Не удалось получить информацию о бинарном журнале")

            # Распарсим данные
            binlog_info = lines[1].split()
            binlog_file = binlog_info[0]

            # Копирование текущего бинарного журнала в файл
            backup_file = os.path.join(backup_directory, f"{binlog_file}_{timestamp}.binlog")
            command = f"mysqlbinlog -s --user=root --password='{mysql_password}' /var/log/mysql/{binlog_file} > {backup_file}"
            subprocess.run(command, shell=True, check=True)

            print(f"Копирование бинарного журнала {binlog_file} успешно выполнено.")
            return backup_file
        except Exception as e:
            error_message = f"Ошибка при копировании бинарного журнала: {e}"
            print(error_message)
            send_telegram_message(error_message)
            return None

    def upload_to_yandex_disk(backup_file, yandex_disk_backup_directory):
        try:
            command = f"ydcmd put '{backup_file}' '{yandex_disk_backup_directory}'"
            subprocess.run(command, shell=True, check=True)
            print(f"Файл бинарного журнала {backup_file} успешно загружен на Яндекс.Диск в каталог {yandex_disk_backup_directory}")
        except Exception as e:
            error_message = f"Ошибка при загрузке файла бинарного журнала {backup_file} на Яндекс.Диск: {e}"
            print(error_message)
```

```
            send_telegram_message(error_message)

        def send_telegram_message(message):
            try:
                command = f"telegram-send '{message}'"
                subprocess.run(command, shell=True, check=True)
                print("Уведомление успешно отправлено в Telegram.")
            except Exception as e:
                print(f"Ошибка при отправке уведомления в Telegram: {e}")

        if __name__ == "__main__":
            config_data = read_config('config_file.json')
            mysql_password = config_data['mysql_password']
            backup_directory = config_data['backup_directory']
            yandex_disk_backup_directory = config_data['yandex_disk_backup_directory']

            backup_file = backup_mysql_binlog(backup_directory, mysql_password)
            if backup_file:
                upload_to_yandex_disk(backup_file, yandex_disk_backup_directory)
                send_telegram_message(f"Копирование бинарного журнала успешно
завершено.")
                # Закрытие текущего бинарного журнала MySQL
                flush_logs(mysql_password)
```

Приложение В

Алгоритм полного восстановления базы данных

```
import os
import subprocess
import json
from datetime import datetime

def read_config(file_path):
    with open(file_path, 'r') as file:
        config_data = json.load(file)
    return config_data

def download_from_yandex_disk(backup_file_name, yandex_disk_backup_directory):
  try:
    command = f"ydcmd get '{yandex_disk_backup_directory}{backup_file_name}'"
    subprocess.run(command, shell=True, check=True)
    print(f"Файл {backup_file_name} успешно скачан с Яндекс.Диска.")
    return backup_file_name
  except Exception as e:
    error_message = f"Ошибка при скачивании файла {backup_file_name} с Яндекс.Диска: {e}"
    print(error_message)
    send_telegram_message(error_message)
    return None

def restore_full_database_from_file(database_name, backup_file, mysql_password):
    try:
        command = f"mysql -u root -p'{mysql_password}' {database_name} < {backup_file}"
        subprocess.run(command, shell=True, check=True)
        print(f"Восстановление базы данных {database_name} из файла успешно завершено.")
    except Exception as e:
```

```
            print(f"Ошибка при восстановлении базы данных {database_name} из файла: {e}")

        def send_telegram_message(message):
            try:
                command = f"telegram-send '{message}'"
                subprocess.run(command, shell=True, check=True)
                print("Уведомление успешно отправлено в Telegram.")
            except Exception as e:
                print(f"Ошибка при отправке уведомления в Telegram: {e}")

        if __name__ == "__main__":
            config_data = read_config('config_file.json')
            mysql_password = config_data['mysql_password']
            database_name = config_data['database_name']
            yandex_disk_backup_directory = config_data['yandex_disk_backup_directory']
            backup_file_name = config_data['backup_file_name']  # Название файла резервной копии на Яндекс.Диске
            # Скачивание файла с Яндекс.Диска
            backup_file = download_from_yandex_disk(backup_file_name, yandex_disk_backup_directory)

            if backup_file:
                # Восстановление базы данных из файла
                restore_full_database_from_file(database_name, backup_file, mysql_password)
                send_telegram_message(f"Восстановление базы данных {database_name} успешно завершено.")
                # После восстановления можно удалить скачанный файл, если это необходимо
                os.remove(backup_file)
```

Приложение Г

Алгоритм восстановления из инкрементной копии

```
import os
import subprocess
import json
from datetime import datetime

def read_config(file_path):
    with open(file_path, 'r') as file:
        config_data = json.load(file)
    return config_data

def download_binlog_from_yandex_disk(binlog_file_name, yandex_disk_backup_directory):
    try:
        command = f"ydcmd get '{yandex_disk_backup_directory}{binlog_file_name}'"
        subprocess.run(command, shell=True, check=True)
        print(f"Файл бинарного журнала {binlog_file_name} успешно скачан с Яндекс.Диска.")
        return binlog_file_name
    except Exception as e:
        error_message = f"Ошибка при скачивании файла бинарного журнала {binlog_file_name} с Яндекс.Диска: {e}"
        print(error_message)
        send_telegram_message(error_message)
        return None

def restore_table_from_binlog(mysql_password, binlog_file, database_name, table_name):
    try:
        # Команда для извлечения SQL из бинарного журнала только для указанной таблицы
```

```
            command = f"mysqlbinlog -D --database {database_name} --table {table_name} {binlog_file}|
            mysql -u root -p'{mysql_password}' {database_name}"
            subprocess.run(command, shell=True, check=True)
            print(f"Таблица {table_name} восстановлена из бинарного журнала {binlog_file}.")
        except Exception as e:
            error_message = f"Ошибка при восстановлении таблицы {table_name} из бинарного журнала: {e}"
            print(error_message)
            send_telegram_message(error_message)

    def send_telegram_message(message):
        try:
            command = f"telegram-send '{message}'"
            subprocess.run(command, shell=True, check=True)
            print("Уведомление успешно отправлено в Telegram.")
        except Exception as e:
            print(f"Ошибка при отправке уведомления в Telegram: {e}")

    if __name__ == "__main__":
        config_data = read_config('config_file.json')
        mysql_password = config_data['mysql_password']
        database_name = config_data['database_name']
        yandex_disk_backup_directory = config_data['yandex_disk_backup_directory']
        binlog_file_name = config_data['binlog_file_name']  # Название файла бинарного журнала на Яндекс.Диске
        table_name = config_data['table_name']  # Имя таблицы для восстановления

        # Скачивание файла бинарного журнала с Яндекс.Диска
        binlog_file = download_binlog_from_yandex_disk(binlog_file_name, yandex_disk_backup_directory)
```

```
        if binlog_file:
            # Восстановление таблицы из бинарного журнала
            restore_table_from_binlog(mysql_password, binlog_file, database_name, table_name)
            send_telegram_message(f"Восстановление таблицы {table_name} из бинарного журнала успешно завершено.")
            # После восстановления можно удалить скачанный файл бинарного журнала, если это необходимо
            os.remove(binlog_file)
```