

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра компьютерных систем

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) Программная инженерия и компьютерные науки

ОТЧЕТ

о прохождении производственной практики (преддипломной практики)

(указывается наименование практики)

Обучающегося Винтер Алёны Викторовны группы №21201 курса 4

(Ф.И.О. полностью)

Тема задания: Проектирование и разработка инструмента-расширения, реализующего метод оптимизации журнала предзаписи в СУБД PostgreSQL

Место прохождения практики: Общество с ограниченной ответственностью «Постгрес Профессиональный» (ООО «ППГ»), 630090, Новосибирская область, г. Новосибирск, пр. Лаврентьева 2/2, офис 419

(полное наименование организации и структурного подразделения, индекс, адрес)

Сроки прохождения практики: с 03.02.2025 г. по 03.05.2025 г.

Руководитель практики
от профильной организации

Рутман Михаил Валерьевич,
ведущий разработчик программного
обеспечения Департамента разработки
продуктов (Обособленное подразделение
г. Новосибирск)

(Ф.И.О. полностью, должность)

(подпись)

Руководитель практики от НГУ Пищик Борис Николаевич,
КафКС ФИТ НГУ, доцент

(Ф.И.О. полностью, должность)

(подпись)

Руководитель ВКР

Пищик Борис Николаевич

(Ф.И.О. полностью)

КафКС ФИТ НГУ, доцент

(должность)

Оценка по итогам защиты отчета: _____

(неудовлетворительно, удовлетворительно, хорошо, отлично)

Отчет заслушан на заседании кафедры _____

(наименование кафедры)

протокол _____ от «_____» _____ 20____ г.

Новосибирск 2025

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	2
ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	3
ВВЕДЕНИЕ.....	4
1. ОБНОВЛЕНИЕ ИНФОРМАЦИИ О КОНТРОЛЬНОЙ ТОЧКЕ.....	5
2. УЧЕТ НОТ-ОПТИМИЗАЦИИ ПРИ СВОРАЧИВАНИИ ЦЕПОЧЕК.....	6
3. НАЛОЖЕНИЕ СТРОК.....	7
4. ССЫЛАЕМОСТЬ НА ПРЕДЫДУЩИЕ ВЕРСИИ СТРОКИ.....	8
5. ССЫЛАЕМОСТЬ ИНДЕКСОВ.....	10
6. ПРОЦЕССЫ ОЧИСТКИ.....	11
7. ВЛИЯНИЕ НА ТРАНЗАКЦИИ.....	12
8. РЕДАКТИРОВАНИЕ ЖУРНАЛА ПРЕДЗАПИСИ.....	13
9. ВЛИЯНИЕ НА СОГЛАСОВАННОСТЬ ДАННЫХ ПРИ ФИЗИЧЕСКОЙ РЕПЛИКАЦИИ.....	14
ЗАКЛЮЧЕНИЕ.....	16
СПИСОК ИСПОЛЪЗУЕМЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ.....	17

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

СУБД — система управления базами данных.

Кластер баз данных — набор баз, управляемых одним экземпляром работающего сервера PostgreSQL.

WAL (Write Ahead Log) — журнал предзаписи в СУБД PostgreSQL.

PGDATA (PostgreSQL Data Directory) — это корневой каталог кластера баз данных PostgreSQL, содержащий все необходимые для функционирования СУБД данные и метаданные.

Блок —

1) это минимальная единица хранения и операций ввода-вывода в PostgreSQL, представляющая собой непрерывный участок данных фиксированного размера (по умолчанию 8 КБ) (в данном контексте также используется термин “**страница**”);

2) это часть WAL-записи, соответствующая изменениям в одном блоке данных на диске.

LSN (log sequence number) — то уникальный идентификатор позиции в журнале предзаписи, равный смещению от начала журнала предзаписи.

Репликация — это механизм автоматического копирования и синхронизации данных между серверами СУБД, обеспечивающий отказоустойчивость и масштабируемость системы.

Отставание реплики (в контексте репликации кластера баз данных) — это временная задержка между применением изменений на первичном (мастер) узле и их отражением на вторичных (репликах) узлах кластера.

Модуль — это автономная, независимая единица функциональности в системе, предназначенная для выполнения конкретной задачи. В программировании модуль обычно представляет собой файл или набор связанных кода, функций или классов, которые можно использовать повторно в разных частях приложения.

Расширение — это дополнительный компонент или программное обеспечение, которое улучшает или расширяет функциональность существующей системы, приложения или программы. Обычно оно интегрируется без изменений в основную систему, добавляя новые функции или возможности.

TPS (Transactions Per Second) — метрика, используемая для измерения производительности системы, которая показывает количество транзакций, обрабатываемых системой за одну секунду.

Бэкап (Backup) — копия данных, созданная для защиты от потери данных, их повреждения или сбоя оборудования.

Реплика (Replica) — копия базы данных, сервера или системы, которая синхронизируется с оригиналом.

API (Application Programming Interface, интерфейс прикладного программирования) — набор правил, протоколов и инструментов, которые позволяют различным программным приложениям взаимодействовать друг с другом.

Хук (Hook) — функция, которая позволяет разработчикам добавлять пользовательский код в определенные точки потока выполнения существующей программы, тем самым изменяя или расширяя ее функциональность без изменения исходной кодовой базы.

Колбэк (Callback, функция обратного вызова) — функция, которая передаётся в другую функцию в качестве аргумента.

ВВЕДЕНИЕ

В системе управления базами данных обеспечение целостности и долговечности данных имеет решающее значение [1]. Одним из ключевых механизмов СУБД PostgreSQL для достижения этих целей является журналирование [2,3]. Журналирование — это процесс, при котором все изменения, внесённые в базу данных, записываются до того, как они будут сохранены в основных файлах. Это гарантирует, что в случае сбоя система может быть восстановлена до согласованного состояния путём воспроизведения записей журнала. Однако при активной работе сервера баз данных генерируется большое количество журнальных записей, которые занимают значительный объём памяти [4].

Актуальность исследования обусловлена негативным влиянием значительного объема журнала предзаписи на производительность СУБД PostgreSQL. Чрезмерный рост WAL создает ряд серьезных проблем: возрастает нагрузка на подсистему ввода-вывода, значительно удлиняется время восстановления после сбоев [5], а также возникают задержки при репликации из-за увеличенного объема передаваемых данных [6]. Кроме того, большой объем WAL требует дополнительных вычислительных ресурсов для обработки и существенно увеличивает требования к дисковому пространству [6]. Все эти факторы в совокупности снижают общую производительность системы, особенно заметно это проявляется в высоконагруженных средах. Оптимизация объема WAL позволяет минимизировать эти негативные эффекты и повысить эффективность работы СУБД.

Целью данной работы является разработка инструмента-расширения для PostgreSQL, оптимизирующего работу механизма журнала предзаписи с целью уменьшения объема журнальных записей и ускорения процессов восстановления и репликации. Для достижения этой цели были поставлены следующие задачи:

1. Проектирование архитектуры инструмента.
2. Разработка и реализация инструмента для системы журналирования с целью уменьшения объема хранимых данных и ускорения восстановления и репликации баз данных.

Научная новизна работы состоит в том, что предложен новый метод хранения записей журнала предзаписи с удалёнными промежуточными значениями строк, который в перспективе уменьшает объем хранимых данных журнала предзаписи и ускоряет процессы восстановления и репликации кластера баз данных.

В результате практической работы будет разработан прототип инструмента-расширения для системы журналирования в СУБД PostgreSQL, которое предоставит пользователю возможность сокращать объем хранимых записей журнала предзаписи, что решит текущую проблему необходимости хранения больших объемов резервных данных и позволит быстрее реплицировать и восстанавливать базу данных до консистентного состояния, повышая эффективность управления ресурсами и обеспечивая стабильную работу системы.

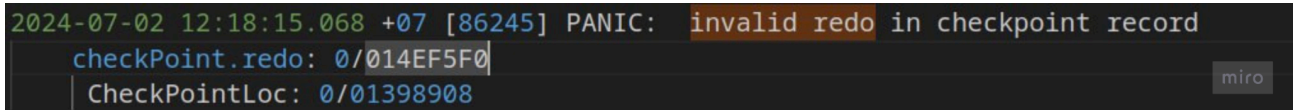
В данной работе рассматривается СУБД PostgreSQL версии 16.3.

Практика проводилась в обществе с ограниченной ответственностью «Постгрес Профессиональный» (ООО «ППГ»), 630090, Новосибирская область, г. Новосибирск, пр. Лаврентьева 2/2, офис 419

Postgres Professional — российский разработчик системы управления базами данных Postgres Pro на основе PostgreSQL. СУБД Postgres Pro входит в Единый реестр российского программного обеспечения. Компания дополняет Postgres новой функциональностью (включая разработку ядра), обеспечивает безопасность. Новосибирское подразделение компании занимается вопросами отказоустойчивости, разработкой и улучшением функциональности кластеров, а также внедрением передовых решений для масштабирования и управления базами данных.

1. ОБНОВЛЕНИЕ ИНФОРМАЦИИ О КОНТРОЛЬНОЙ ТОЧКЕ

После полного чтения сегмента журнала и свёртки «цепочек» записей происходит физическое смещение данных в обработанном сегменте. Это приводит к расхождению между фактическим положением контрольной точки в сегменте и данными в файле *pg_control*, хранящемся на диске. Таким образом, LSN последней контрольной точки перестает соответствовать её реальному местоположению после обработки сегмента (рис. 1).



```
2024-07-02 12:18:15.068 +07 [86245] PANIC: invalid redo in checkpoint record
checkPoint.redo: 0/014EF5F0
CheckPointLoc: 0/01398908
```

Рис. 1

Решение проблемы заключается в следующем: после завершения обработки сегмента необходимо обновить информацию о LSN последней контрольной точки в файле *pg_control*, чтобы обеспечить её соответствие актуальному местоположению в обработанном сегменте.

2. УЧЕТ НОТ-ОПТИМИЗАЦИИ ПРИ СВОРАЧИВАНИИ ЦЕПОЧЕК

При сворачивании "цепочек" необходимо учитывать механизм Heap-Only Tuple. Если первая запись в "цепочке" содержит префикс и/или суффикс, требуется дополнительное чтение страницы таблицы для восстановления полной версии строки, чтобы обеспечить корректное наложение изменений.

Данный подход увеличивает время обработки сегмента за счет обращения к странице таблицы.

3. НАЛОЖЕНИЕ СТРОК

При обработке WAL-записей возникает фундаментальная проблема: структуры данных строк не содержат информации о длине и типах атрибутов. Для корректного наложения изменений необходимо обращаться к описанию таблицы в системном каталоге `pg_class`, что невозможно в стандартном процессе архивации.

Для решения этой проблемы был модифицирован исходный код процесса архивации. Теперь он работает как фоновый процесс (*background worker*) с двухэтапной инициализацией. На первом этапе инициализируются базовые структуры, а на втором - при обработке конкретных записей - происходит полное подключение к нужной базе данных, идентификатор которой становится известен из WAL-записи. Процесс работает с правами суперпользователя, что устраняет необходимость аутентификации.

Особую сложность представляет работа в кластерной среде, где WAL является общим для всех баз данных. Каждое переключение между базами при обработке разных записей создает значительные накладные расходы. Кэширование метаданных лишь частично решает эту проблему, так как объем сохраняемых структур остается существенным.

После подключения к конкретной базе данных процесс получает доступ к системным таблицам `pg_class` и `pg_attribute`, которые всегда индексируются, что ускоряет поиск нужной информации. Полученное описание таблицы копируется в структуру `TupleDesc` для последующего использования. Однако здесь возникает новая проблема: поскольку обработка сегмента WAL происходит постфактум, структура таблицы могла измениться или таблица могла быть удалена. Это создает риск ошибок при попытке наложения строк по устаревшей схеме.

В качестве компромиссного решения предлагается ограничить применение алгоритма свертки заранее определенным списком таблиц, для которых запрещаются структурные изменения. Такой подход обеспечивает корректность обработки, но снижает гибкость системы и добавляет дополнительные накладные расходы на контроль структуры таблиц. Особенно это заметно в системах с частыми изменениями схемы данных.

4. ССЫЛАЕМОСТЬ НА ПРЕДЫДУЩИЕ ВЕРСИИ СТРОКИ

Анализ структуры UPDATE-записей выявил ключевую проблему обработки цепочек обновлений. Стандартный механизм PostgreSQL использует ссылки на предыдущие версии строк в последующих изменениях. При свертке цепочки промежуточные версии удаляются, что нарушает исходные ссылки и делает невозможным корректное восстановление данных (Рис. 2).

Страница:

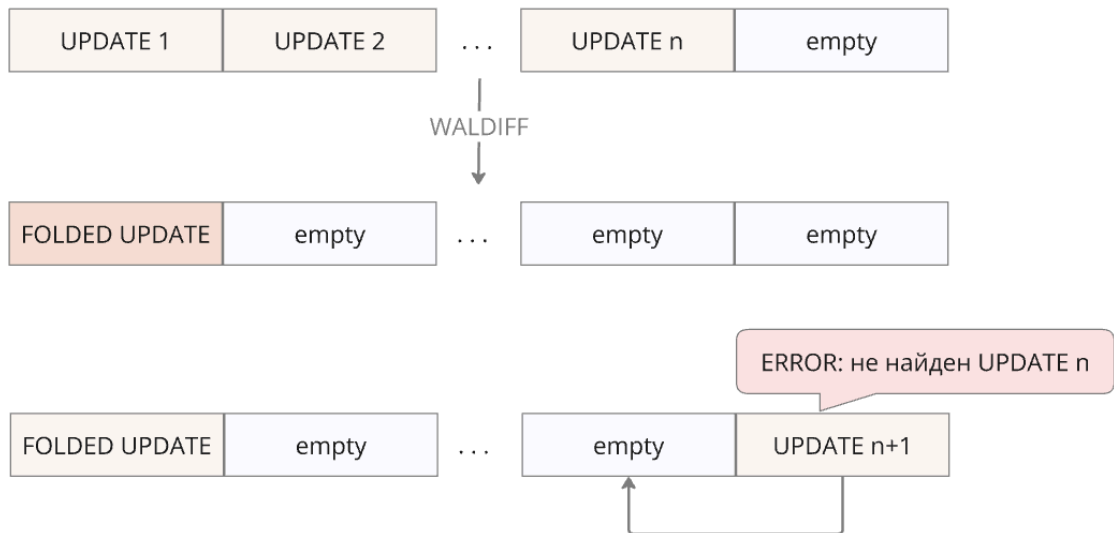


Рис. 2

Таким образом, возникает необходимость изменять адрес результирующей версии строки всей "цепочки" обновлений: нужно сохранять не адрес начала "цепочки", а адрес её конца. Это позволит корректно находить актуальную версию строки при последующих обновлениях (рис. 3).

Страница:

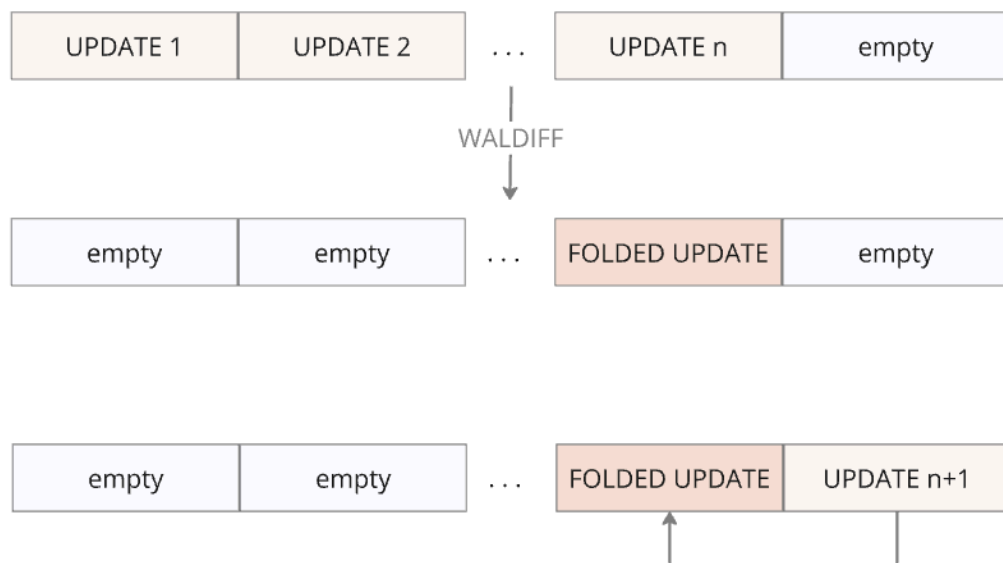


Рис. 3

Однако это решение имеет важное ограничение - оно несовместимо с флагом *XLOG_HEAP_INIT_PAGE*, отмечающим записи инициализации новых страниц. При сдвиге записей во время свёртки может нарушиться порядок обработки: запись, требующая существующей страницы, может быть обработана раньше записи её создающей.

Для сохранения работоспособности системы необходимо оставлять первые записи "цепочки" с флагом *XLOG_HEAP_INIT_PAGE* неизменными, чтобы обеспечить корректное воспроизведение всех записей.

5. ССЫЛАЕМОСТЬ ИНДЕКСОВ

С учетом принятого решения в главе 4 для корректной очистки и управления указателями в индексах, ссылающихся на НОТ-цепочки, необходимо сохранять информацию о начале цепочки, поскольку индекс всегда указывает на первую запись. При вставке результирующей версии строки требуется:

- 1) добавить на страницу запись о начале цепочки;
- 2) пометить её флагом *lp_redirect*;
- 3) направить ссылку индекса на результирующую версию строки.

Для реализации этого механизма рассматривались два подхода: создание нового менеджера ресурсов или расширение функциональности существующего или сохранение в WAL двух записей — сначала вставка первой строки цепочки, затем результирующей версии.

Был выбран второй вариант как более простой в реализации.

6. ПРОЦЕССЫ ОЧИСТКИ

Стандартная логика очистки предполагает, что после её завершения на странице не должно оставаться указателей на строки (*linp*) с флагами, отличными от *lp_normal*. С учетом принятого решения в главе 5 необходимо найти обход этого ограничения. Были найдены следующие решения:

- 1) Модификация процессов очистки — разрешить существование указателей на строки с флагом *lp_redirect* на момент начала очистки, а также возможность их вставки без ссылки на строку.
- 2) Введение нового флага *lp_plug* — аналога *lp_redirect*, но без данных строки (только указатель *linp*, выполняющий роль заглушки).

Однако первый подход усложняет логику очистки и требует значительных доработок исходных кодов PostgreSQL, что может негативно сказаться на производительности и стабильности. А также наличие указателей *linp* с флагом *lp_redirect* без ссылки на строку повышает риск ошибок при обработке данных.

При реализации второго решения обнаружилось ограничение на уровне хранения флагов: в заголовке строки под флаги выделено всего 2 бита, что позволяет использовать только четыре состояния:

- *lp_normal* (обычная запись),
- *lp_unused* (удалённая строка),
- *lp_redirect* (ссылка на актуальную версию),
- *lp_dead* (логически удалённая строка, ожидающая очистки).

Добавление нового флага *lp_plug* потребовало бы расширения заголовка как минимум на 1 бит, что привело бы к увеличению размера каждой записи из-за выравнивания памяти и ввиду этого росту потребления памяти (каждая запись стала бы занимать на 32 бита больше). Такой подход был признан неприемлемым из-за значительных накладных расходов. В результате для решения проблемы был выбран первый подход (модификация логики очистки), несмотря на его сложность в реализации.

7. ВЛИЯНИЕ НА ТРАНЗАКЦИИ

Если извлеченные записи не затрагивают критичные данные о состоянии транзакций, то их отсутствие не нарушит процесс восстановления. Однако возможна ситуация, когда изменения, внесённые транзакцией, были удалены или модифицированы при фильтрации журнала и сама транзакция остается в журнале предзаписи и воспроизводится стандартным образом.

В результате такая транзакция не окажет реального влияния на базу данных, так как связанные с ней изменения уже отсутствуют. Это приводит к появлению «пустых» транзакций — операций, которые выполняются, но не модифицируют данные. Обработка таких транзакций бесполезна, но требует вычислительных ресурсов, а также «пустые» транзакции попадают в данные статистики, делая ее некорректной.

В высоконагруженных системах даже незначительные издержки могут сказаться на производительности — для минимизации влияния «пустых» транзакций требуется анализ записей WAL на предмет их актуальности перед воспроизведением и исключение незначимых транзакций из процесса восстановления. Однако реализация таких механизмов повышает сложность системы (требуются дополнительные алгоритмы валидации) и увеличивает накладные расходы на поддержку логики фильтрации.

Несмотря на затратность, оптимизация обработки «пустых» транзакций может быть оправдана в системах с высокой нагрузкой, где критична даже минимальная экономия ресурсов. Для этого необходимы эффективные алгоритмы, позволяющие балансировать между точностью восстановления и производительностью.

8. РЕДАКТИРОВАНИЕ ЖУРНАЛА ПРЕДЗАПИСИ

Изменения, сохраняемые в журнале предзаписи, представляют собой не отдельные и изолированные события, а тесно взаимосвязанные элементы единой системы. Эти взаимосвязи играют ключевую роль в обеспечении целостности данных и корректности их обработки. Однако именно эта особенность делает невозможным простое извлечение отдельных записей из журнала без риска возникновения ошибок. Например, удаление промежуточных записей, связанных с изменениями в одной строке, требует обязательного учета сопутствующих записей, относящихся к очистке и обработке версий этих изменений. Если такие сопутствующие записи не будут правильно обработаны, то, например, попытка удалить запись, которая уже не существует, приведет к ошибке в процессе очистки.

Количество таких взаимосвязанных записей в журнале предзаписи может быть значительным, особенно в масштабных системах с высокой нагрузкой. Это накладывает серьезные ограничения и влечет за собой значительные накладные расходы как на вычислительные ресурсы, так и на сложность реализации алгоритмов обработки данных, так как необходимо отслеживать и соответствующим образом обрабатывать записи, обращающиеся или модифицирующие удаленные в процессе сворачивания записи. В высоконагруженных системах, где объем WAL может достигать огромных размеров, даже небольшая ошибка в обработке записей способна иметь далеко идущие последствия. Она может привести не только к падению производительности, но и к утрате целостности данных.

9. ВЛИЯНИЕ НА СОГЛАСОВАННОСТЬ ДАННЫХ ПРИ ФИЗИЧЕСКОЙ РЕПЛИКАЦИИ

Один из предполагаемых сценариев использования метода удаления промежуточных значений строк заключается в применении его в физической репликации. Удаление промежуточных версий строк уменьшает общий объем передаваемых данных и количество операций, необходимых для применения изменений, но создаёт задержку между репликами и мастером в один сегмент, что может быть критично для некоторых систем.

Помимо проблемы отставания реализованный подход создает серьезные проблемы согласованности. Основная сложность возникает при смене ролей серверов - когда бывшая реплика становится новым основным сервером. В таком случае обработанные WAL-записи (например, с номерами смещений 7, 8 и 9 на Рис. 4), которые были удалены на исходном основном сервере, создают расхождение в истории изменений между серверами. Новый основной сервер (N2) не имеет информации об этих записях, в то время как бывший основной сервер (N1) продолжает считать их действительными.

страница основного сервера

off = 5	off = 6	off = 7
off = 8	off = 9	off = 10

WALDIFF



страница реплики

off = 5	off = 6	off = 7
off = 8	off = 9	off = 10

Рис. 4

Такая ситуация создаёт серьезные проблемы, поскольку сервер N2 не будет пытаться очистить строки, связанные с этими смещениями. Это может привести к тому, что свободное для сервера N2 место будет занято другими строками, в то время как на сервере N1, для которого эти записи всё ещё существуют – попытки применения изменений на N1 могут завершаться ошибками из-за расхождений в структуре страниц. Решение этих проблем требует глубоких изменений в механизме управления страницами. Необходимо разработать методы обнаружения и обработки расхождений в истории WAL. Было решено внести изменения в код, управляющий обработкой данных страниц.

При возникновении подобных «коллизий» (когда обнаружено, что `lptr` с заданным смещением уже существует) требуется очистить конкретную версию строки на странице. Так как строки могут иметь разные длины, то просто перезаписать уже существующую строку не выйдет – образует «разрыв», на который никто не ссылается, что может привести к тому, что он не будет учтен при очистке страницы на этапе дефрагментации. Это создаёт риск сбоя работы системы. Для предотвращения подобных ситуаций требуется немедленно выполнить дефрагментацию страницы. Реализация таких изменений существенно усложнит код ядра PostgreSQL. К тому же операция дефрагментации очень затратна по времени и ресурсам, что

делает её неприемлемой для высоконагруженных систем, и в случае отстающей реплики увеличит её отставание.

ЗАКЛЮЧЕНИЕ

В рамках практики были выполнены все поставленные задачи. Была спроектирована архитектура инструмента-расширения для системы журналирования и разработан его прототип, чья цель — уменьшение объёма хранимых данных и ускорение восстановления и репликации баз данных. В ходе разработки инструмента-расширения «WALDIFF» были выявлены ключевые технические сложности, препятствующие дальнейшей реализации и описаны способы решения этих проблем или же доказана невозможность их устранения в рамках текущей архитектуры PostgreSQL.

Таким образом, внедрение предложенного метода сопряжено с серьезными техническими сложностями, включая необходимость кардинального изменения логики работы процессов СУБД, разработки новых механизмов обеспечения согласованности данных и создания дополнительных процедур контроля целостности;

Практическая работа позволила мне сформировать компетенции в области анализа систем хранения данных и проектирования решений для их оптимизации. Я приобрела навыки работы с СУБД PostgreSQL, включая глубокое понимание внутренней структуры и механизмов журналирования, восстановления, архивации и репликации. Работа над проектом способствовала развитию аналитического мышления, навыков проектирования архитектуры программного обеспечения и оценки эффективности разрабатываемых решений.

Я оцениваю свою работу на практике положительно, поскольку поставленные задачи были успешно выполнены, а приобретённые знания и навыки имеют большую ценность для дальнейшего профессионального роста.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. PostgreSQL Documentation: Reliability [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/wal-reliability.html> (дата обращения: 01.05.2025).
2. PostgreSQL Documentation: Write-Ahead Logging (WAL) [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/wal-intro.html> (дата обращения: 01.05.2025).
3. Mohan C., Haderle D. Algorithms for Flexible Space Management in Transaction Systems Supporting Fine-Granularity Locking // Proceedings of the International Conference on Extending Database Technology (EDBT). 1994. P. 289–291. DOI: <https://doi.org/10.1145/289.291>.
4. Pavel T. PostgreSQL: Why and How WAL Bloats [Электронный ресурс] // Dzone. URL: <https://dzone.com/articles/postgresql-why-and-how-wal-bloats> (дата обращения: 01.05.2025).
5. PostgreSQL Documentation: WAL Configuration [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/wal-configuration.html> (дата обращения: 01.05.2025).
6. What to Look for if Your PostgreSQL Replication is Lagging [Электронный ресурс] // PGEdge. URL: <https://www.pgedge.com/blog/understanding-and-reducing-postgresql-replication-lag> (дата обращения: 01.05.2025).
7. Джошуа Дрейк. Репликация Postgres на практике [Электронный ресурс] // PGConf.Russia. URL: <https://pgconf.ru/talk/1588507> (дата обращения: 01.05.2025).
8. PostgreSQL Documentation: Warm Standby [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/warm-standby.html> (дата обращения: 01.05.2025).
9. Алексей Лесовский. Поточная репликация на практике [Электронный ресурс] // PGConf.Russia. URL: <https://pgconf.ru/talk/1587639> (дата обращения: 01.05.2025).
10. PostgreSQL для профессионалов: Управление производительностью и отказоустойчивостью. Настройка и оптимизация WAL [Электронный ресурс] // Postgres Professional. URL: https://edu.postgrespro.ru/dba2/dba2_11_wal_tuning.pdf (дата обращения: 01.05.2025).
11. PostgreSQL WAL Archiving [Электронный ресурс] // OpsDash Blog. URL: <https://www.opsdash.com/blog/postgresql-wal-archiving-backup.html> (дата обращения: 01.05.2025).
12. Репозиторий с исходным кодом СУБД PostgreSQL [Электронный ресурс] // GitHub. URL: https://github.com/postgres/postgres/tree/REL_16_STABLE (дата обращения: 01.05.2025).