ООО «Малленом Системс»

Руководство администратора подсистемы хранения данных системы мониторинга состояния конвейерных лент

Череповец

2025 год

Содержание

[Введение 3](#_Toc200823356)

[1. Общие сведения о системе и подсистеме хранения данных 4](#_Toc200823357)

[1.1 Описание системы мониторинга состояния конвейерных лент 4](#_Toc200823358)

[1.2 Описание подсистемы хранения данных 4](#_Toc200823359)

[2. Архитектура подсистемы хранения данных 6](#_Toc200823360)

[2.1 Используемые технологии 6](#_Toc200823361)

[2.2 Схема базы данных PostgreSQL 6](#_Toc200823362)

[2.3 Репликация данных 8](#_Toc200823363)

[2.4 Инструменты управления 8](#_Toc200823364)

[3 Управление базой данных 9](#_Toc200823365)

[3.1 Доступ к базе данных 9](#_Toc200823366)

[3.2 Мониторинг состояния базы данных 9](#_Toc200823367)

[3.3 Обслуживание базы данных 10](#_Toc200823368)

[4 Управление сервисом обработки данных о повреждениях 12](#_Toc200823369)

[4.1 API сервиса 12](#_Toc200823370)

[4.2 Мониторинг сервиса 13](#_Toc200823371)

[5 Резервное копирование и восстановление данных 14](#_Toc200823372)

[5.1 Стратегия резервного копирования 14](#_Toc200823373)

[5.2 Тестирование процедуры восстановления 15](#_Toc200823374)

[6 Безопасность данных 16](#_Toc200823375)

[6.1 Контроль доступа 16](#_Toc200823376)

[6.2 Шифрование данных 16](#_Toc200823377)

[6.3 Аудит 16](#_Toc200823378)

[6.4 Безопасность сервиса обработки данных 17](#_Toc200823379)

[7 Управление инцидентами 18](#_Toc200823380)

[7.1 Процедура обработки инцидентов 18](#_Toc200823381)

[7.2 Эскалация инцидентов 18](#_Toc200823382)

[8 Приложения 19](#_Toc200823383)

[8.1 Список полезных команд и скриптов 19](#_Toc200823384)

[8.2 Глоссарий терминов 19](#_Toc200823385)

# Введение

Настоящее руководство предназначено для администраторов, ответственных за поддержание и управление подсистемой хранения данных системы мониторинга состояния конвейерных лент (далее – Система). Руководство содержит детальную информацию о архитектуре, управлении, обслуживании, резервном копировании, безопасности и управлении инцидентами, касающихся подсистемы хранения данных. Руководство предназначено для использования квалифицированным персоналом с опытом администрирования баз данных и серверного оборудования.

# Общие сведения о системе и подсистеме хранения данных

## Описание системы мониторинга состояния конвейерных лент

Система мониторинга состояния конвейерных лент предназначена для непрерывного отслеживания состояния конвейерных лент с целью раннего выявления повреждений, предотвращения аварийных остановок и оптимизации технического обслуживания. Система включает в себя:

* Датчики: Датчики, установленные на конвейерной ленте, собирают данные о ее состоянии (например, оптические датчики, датчики вибрации, датчики температуры).
* Сервис обработки данных о повреждениях: Сервис, который анализирует данные, поступающие от датчиков, выявляет признаки повреждений и записывает информацию о повреждениях в подсистему хранения данных. Этот сервис также предоставляет API для добавления, удаления и изменения записей о повреждениях.
* Подсистема хранения данных: Централизованное хранилище данных о повреждениях, параметрах работы ленты и конфигурации системы.
* Интерфейс пользователя (UI): Графический интерфейс для визуализации данных, мониторинга состояния ленты, просмотра отчетов и управления системой.

## Описание подсистемы хранения данных

Подсистема хранения данных является критически важным компонентом Системы. Она отвечает за надежное хранение и оперативный доступ к следующим типам данных:

* События повреждений: Идентификатор события, время обнаружения, местоположение повреждения на ленте (в метрах от начала), тип повреждения (разрыв, прокол, износ), степень серьезности (критическое, умеренное, незначительное), описание, фотографии (ссылки на файлы), статус (открыто, в работе, закрыто), ответственный за устранение.
* Параметры работы ленты: Время записи, скорость ленты (м/с), нагрузка (тонны/час), натяжение (Н), температура ленты (°C), данные вибрации (значение амплитуды).
* Конфигурация системы: Параметры датчиков (ID, тип, местоположение, калибровочные данные), параметры алгоритмов обработки данных (пороговые значения, правила классификации), параметры оповещений.
* Пользователи и их права: ID пользователя, Имя пользователя, Пароль (хешированный), Роль (администратор, оператор, наблюдатель), Дата создания, Дата последнего входа.

# Архитектура подсистемы хранения данных

## Используемые технологии

1. База данных: PostgreSQL
2. Операционная система: Ubuntu Server 22.04 LTS
3. Аппаратное обеспечение: Два сервера в кластере с репликацией данных. Каждый сервер:
   * Процессор: Intel Xeon Gold 6248R
   * Оперативная память: 64 GB DDR4
   * Дисковая подсистема: 2 x 1TB NVMe SSD (RAID 1)
4. Язык программирования: С#

## Схема базы данных PostgreSQL

База данных состоит из следующих таблиц:

1. events (события повреждений):
   * event\_id SERIAL PRIMARY KEY - Уникальный идентификатор события.
   * timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL - Время обнаружения повреждения.
   * location REAL NOT NULL - Местоположение повреждения на ленте (в метрах от начала).
   * damage\_type VARCHAR(255) NOT NULL - Тип повреждения (разрыв, прокол, износ).
   * severity VARCHAR(50) NOT NULL - Степень серьезности (критическое, умеренное, незначительное).
   * description TEXT - Подробное описание повреждения.
   * image\_url VARCHAR(255) - Ссылка на изображение повреждения (опционально).
   * status VARCHAR(50) NOT NULL DEFAULT 'открыто' - Статус события (открыто, в работе, закрыто).
   * assigned\_to INTEGER REFERENCES users(user\_id) - ID пользователя, ответственного за устранение.
2. sensor\_data (параметры работы ленты):
   * data\_id SERIAL PRIMARY KEY - Уникальный идентификатор записи данных.
   * timestamp TIMESTAMP WITHOUT TIME ZONE NOT NULL - Время записи данных.
   * belt\_speed REAL - Скорость ленты (м/с).
   * load REAL - Нагрузка (тонны/час).
   * tension REAL - Натяжение (Н).
   * temperature REAL - Температура ленты (°C).
   * vibration REAL - Значение вибрации.
3. sensors (конфигурация датчиков):
   * sensor\_id SERIAL PRIMARY KEY - Уникальный идентификатор датчика.
   * sensor\_type VARCHAR(255) NOT NULL - Тип датчика.
   * location VARCHAR(255) NOT NULL - Местоположение датчика на ленте.
   * calibration\_data TEXT - Калибровочные данные датчика.
4. users (пользователи):
   * user\_id SERIAL PRIMARY KEY - Уникальный идентификатор пользователя.
   * username VARCHAR(255) UNIQUE NOT NULL - Имя пользователя.
   * password\_hash VARCHAR(255) NOT NULL - Хеш пароля.
   * role VARCHAR(50) NOT NULL - Роль пользователя (администратор, оператор, наблюдатель).
   * created\_at TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT NOW().
   * last\_login TIMESTAMP WITHOUT TIME ZONE - Дата последнего входа.

## Репликация данных

Для обеспечения высокой доступности и отказоустойчивости используется встроенная в PostgreSQL потоковая репликация. Один сервер является основным (Primary), а второй – резервным (Standby). Данные с основного сервера непрерывно реплицируются на резервный сервер. В случае отказа основного сервера резервный сервер автоматически становится основным.

## Инструменты управления

* psql: Командная строка PostgreSQL для выполнения SQL-запросов и администрирования базы данных.
* pgAdmin: Графический интерфейс для управления базами данных PostgreSQL.

# Управление базой данных

## Доступ к базе данных

1. Учетные записи пользователей: Доступ к базе данных осуществляется через учетные записи пользователей, определенные в таблице users.
2. Роли и привилегии:
   * Администратор: Полный доступ ко всем таблицам и функциям базы данных.
   * Оператор: Доступ на чтение и запись к таблицам events и sensor\_data. Доступ на чтение к таблицам sensors и users.
   * Наблюдатель: Доступ только на чтение ко всем таблицам.
3. Создание новых пользователей: sql CREATE USER <username> WITH PASSWORD '<password>'; GRANT <role> TO <username>; Замените <username>, <password> и <role> на соответствующие значения.
4. Изменение пароля пользователя: sql ALTER USER <username> WITH PASSWORD '<new\_password>';
5. Удаление пользователя: sql DROP USER <username>;
6. Рекомендации по паролям: Пароли должны быть сложными (не менее 12 символов, содержать буквы верхнего и нижнего регистра, цифры и специальные символы) и регулярно меняться (не реже одного раза в три месяца).

## Мониторинг состояния базы данных

1. Метрики производительности:
   * Загрузка процессора: Используйте команды top или htop для мониторинга загрузки процессора.
   * Использование памяти: Используйте команды free -m или htop для мониторинга использования памяти.
   * Дисковая активность: Используйте команды iostat или iotop для мониторинга дисковой активности.
   * Время отклика на запросы: Используйте команду EXPLAIN ANALYZE перед SQL-запросами для анализа времени их выполнения.
   * Количество активных соединений: Выполните SQL-запрос SELECT count(\*) FROM pg\_stat\_activity;
   * Размер базы данных: Выполните SQL-запрос SELECT pg\_size\_pretty(pg\_database\_size('<database\_name>')); Замените <database\_name> на имя вашей базы данных.
2. Инструменты мониторинга:
   * pgAdmin: Предоставляет графический интерфейс для мониторинга производительности PostgreSQL.
   * Grafana: Используется для визуализации метрик производительности, собранных с помощью Prometheus.
3. Настройка оповещений:
   * Prometheus Alertmanager: Используется для отправки оповещений по электронной почте или в Slack при достижении определенных пороговых значений (например, высокая загрузка процессора, недостаточно места на диске).

## Обслуживание базы данных

1. Оптимизация запросов:
   * Использование индексов: Создавайте индексы на столбцах, которые часто используются в фильтрах (WHERE clause) и при соединениях (JOIN clause). sql CREATE INDEX <index\_name> ON <table\_name> (<column\_name>);
   * Анализ запросов: Используйте команду EXPLAIN ANALYZE для анализа плана выполнения запроса и выявления узких мест.
   * Переписывание запросов: Иногда переписывание запроса другим способом может значительно улучшить его производительность.
2. Оптимизация хранения данных:
   * Удаление устаревших данных: Регулярно удаляйте устаревшие данные из таблиц events и sensor\_data.
   * Архивирование данных: Перемещайте устаревшие данные в архивные таблицы или файлы для экономии места на диске.
3. Обновление статистики: Регулярно выполняйте команду ANALYZE для обновления статистики таблиц. Это необходимо для правильной работы оптимизатора запросов. sql ANALYZE <table\_name>;
4. Vacuuming: Регулярно выполняйте команду VACUUM для очистки неиспользуемого места в базе данных. Это помогает повысить производительность и избежать раздувания базы данных. sql VACUUM FULL <table\_name>; Внимание: VACUUM FULL блокирует таблицу на время выполнения. Рассмотрите использование VACUUM без FULL.

# Управление сервисом обработки данных о повреждениях

## API сервиса

Сервис обработки данных о повреждениях предоставляет REST API для взаимодействия с подсистемой хранения данных.

1. Добавление нового события повреждения:
   * POST /events
   * JSON payload: json { "timestamp": "2023-10-27T10:00:00Z", "location": 15.5, "damage\_type": "разрыв", "severity": "критическое", "description": "Обнаружен продольный разрыв ленты", "image\_url": "https://example.com/image.jpg" }
   * Response: 201 Created (успешно), 400 Bad Request (неверные данные)
2. Удаление события повреждения:
   * DELETE /events/{event\_id}
   * Response: 204 No Content (успешно), 404 Not Found (событие не найдено)
   * Требуется аутентификация и авторизация для предотвращения несанкционированного удаления.
3. Редактирование события повреждения:
   * PUT /events/{event\_id}
   * JSON payload: Содержит поля для обновления (аналогично POST).
   * Response: 200 OK (успешно), 400 Bad Request (неверные данные), 404 Not Found (событие не найдено)
   * Требуется аутентификация и авторизация.
4. Получение данных о событии повреждения:
   * GET /events/{event\_id}
   * Response: 200 OK (успешно) - JSON с данными события, 404 Not Found (событие не найдено)
5. Получение списка событий повреждений (с фильтрацией и пагинацией):
   * GET /events?status=открыто&severity=критическое&page=1&page\_size=10
   * Response: 200 OK - JSON с массивом событий.

## Мониторинг сервиса

1. Статус сервиса: Используйте команду systemctl status damage-service для проверки статуса сервиса.
2. Логи сервиса: Анализируйте логи сервиса в файле /var/log/damage-service.log для выявления ошибок и проблем. Используйте tail -f /var/log/damage-service.log для мониторинга в реальном времени.
3. Метрики производительности сервиса:
   * Время обработки запросов: Мониторинг времени обработки каждого типа API-запроса.
   * Количество обработанных событий: Мониторинг количества событий, добавленных, удаленных и обновленных за определенный период времени.
   * Использование ресурсов (CPU, memory): Аналогично мониторингу базы данных.

# Резервное копирование и восстановление данных

## Стратегия резервного копирования

1. Тип резервного копирования: Полное резервное копирование выполняется раз в неделю (воскресенье в 02:00), инкрементное резервное копирование выполняется ежедневно в 02:00.
2. Частота резервного копирования:
   * Полное: Раз в неделю.
   * Инкрементное: Ежедневно.
3. Место хранения резервных копий: Резервные копии хранятся на выделенном NAS (Network Attached Storage) сервере в отдельной сети.
4. Хранение резервных копий:
   * Полные резервные копии хранятся в течение одного месяца.
   * Инкрементные резервные копии хранятся в течение одной недели.
5. Автоматизация резервного копирования: Используется cron для автоматического запуска скриптов резервного копирования.
6. Восстановление из инкрементных резервных копий (если необходимо): Применяйте инкрементные резервные копии в порядке их создания после восстановления полной резервной копии. Процесс восстановления из WAL архива подробно описан в документации PostgreSQL.
7. Проверка целостности данных: Выполните выборочные запросы для проверки целостности данных.
8. Запуск сервиса обработки данных о повреждениях: systemctl start damage-service
9. Мониторинг: Внимательно следите за работой системы после восстановления.

## Тестирование процедуры восстановления

Регулярно (не реже одного раза в месяц) проводите тестирование процедуры восстановления на тестовой среде для проверки ее работоспособности и актуальности.

# Безопасность данных

## Контроль доступа

* Разграничение прав доступа: Каждому пользователю назначается роль с соответствующими правами доступа к базе данных.
* Аутентификация: Доступ к базе данных осуществляется по паролю.
* Усиление аутентификации: Рассмотрите возможность внедрения двухфакторной аутентификации для повышения безопасности.
* Ограничение доступа по IP-адресу: Настройте firewall (например, iptables или UFW) для ограничения доступа к PostgreSQL только с доверенных IP-адресов.
* Использование SSL/TLS: Включите SSL/TLS для шифрования трафика между клиентами и сервером PostgreSQL.

## Шифрование данных

* Шифрование паролей: Пароли пользователей хранятся в базе данных в виде хешей (bcrypt).
* Шифрование диска: Рассмотрите возможность шифрования дисков с использованием LUKS для защиты данных от физического доступа.
* Шифрование данных "в покое" (Data-at-rest encryption): PostgreSQL поддерживает шифрование на уровне столбцов или таблиц, но это может повлиять на производительность.

## Аудит

* Ведение журналов аудита: Включите аудит в PostgreSQL для записи информации о действиях пользователей с базой данных. sql ALTER SYSTEM SET log\_statement = 'all'; # Записывать все SQL-запросы ALTER SYSTEM SET log\_connections = on; # Записывать подключения ALTER SYSTEM SET log\_disconnections = on; # Записывать отключения После изменения конфигурации необходимо перезапустить PostgreSQL.
* Анализ журналов аудита: Регулярно анализируйте журналы аудита для выявления подозрительной активности. Используйте инструменты для автоматического анализа логов.

## Безопасность сервиса обработки данных

* Включите аутентификацию и авторизацию для всех API сервиса.
* Используйте HTTPS для шифрования трафика между клиентами и сервисом.
* Проводите регулярные проверки безопасности кода сервиса.

# Управление инцидентами

## Процедура обработки инцидентов

1. Обнаружение инцидента: Обнаружение инцидента (например, сбой сервера, повреждение данных, подозрительная активность).
2. Идентификация инцидента: Определение типа и масштаба инцидента.
3. Регистрация инцидента: Запись информации об инциденте в систему отслеживания инцидентов (например, Jira, ServiceNow).
4. Принятие мер по сдерживанию инцидента: Принятие мер для предотвращения дальнейшего распространения инцидента (например, отключение поврежденного сервера, блокировка учетной записи пользователя).
5. Устранение инцидента: Устранение причины инцидента и восстановление работоспособности системы.
6. Восстановление данных (если необходимо): Восстановление данных из резервной копии.
7. Анализ инцидента: Проведение анализа причины инцидента и разработка мер для предотвращения подобных инцидентов в будущем.
8. Закрытие инцидента: Закрытие инцидента в системе отслеживания инцидентов.

## Эскалация инцидентов

* Критические инциденты (например, полная потеря данных, остановка системы) эскалируются немедленно руководителю ИТ-отдела.
* Важные инциденты (например, частичная потеря данных, снижение производительности системы) эскалируются в течение 2 часов.
* Незначительные инциденты эскалируются в течение 24 часов.

# Приложения

## Список полезных команд и скриптов

* psql: Командная строка PostgreSQL.
* pg\_dump: Утилита для резервного копирования базы данных PostgreSQL.
* pg\_restore: Утилита для восстановления базы данных PostgreSQL.
* systemctl: Утилита для управления системными сервисами (systemd).
* top, htop: Утилиты для мониторинга использования ресурсов системы.
* iostat, iotop: Утилиты для мониторинга дисковой активности.
* /opt/backup/full\_backup.sh: Скрипт для полного резервного копирования.
* /opt/backup/incremental\_backup.sh: Скрипт для инкрементного резервного копирования.

## Глоссарий терминов

* WAL (Write-Ahead Logging): Механизм журналирования в PostgreSQL, который обеспечивает надежность и отказоустойчивость.
* NAS (Network Attached Storage): Сетевое хранилище данных.
* SSL/TLS: Протоколы шифрования для защиты данных при передаче по сети.
* LUKS (Linux Unified Key Setup): Стандарт шифрования дисков в Linux.