1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Анализ журналов транзакций**

по дисциплине «Основы построения защищенных баз данных»

1. Выполнила
2. студентка гр. 4851003/90801 Кулеева А.Г.

1. Руководитель
2. ассистент Зубков Е.А.
3. Санкт-Петербург
4. 2023

# **Цель**

Получить навыки работы с механизмами журнализации и восстановления в СУБД. обеспечения доступности баз данных и расследования инцидентов с использованием журналов WAL.

# Ход работы

## Доступные характеристики журнала транзакций

Чтобы получить перечень доступных характеристик журнала транзакций необходимо ввести команду select \* from pg\_settings WHERE category LIKE 'Журнал WAL%'. Получим следующий результат (см. рисунок 1):

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Рисунок 1 ― Проверка текущего состояния журнала

Чтобы включить архивирование, необходимо изменить некоторые настройки в конфигурационном файле и перезапустить службу.

Требуется создать бд и таблицу, на которых будем ставить эксперименты.

CREATE DATABASE lab4;

\c lab4

CREATE TABLE MOCKDATA(

first\_name VARCHAR(500),

last\_name VARCHAR(500),

gender VARCHAR(5),

country VARCHAR(500));

Сгенерируем случайные данные с помощью сервиса генерации данных. Было создано 3 файла размером 10, 100 и 1000 записей. Импортируем данные в таблицу и посмотрим, как изменилась таблица логов (рисунок 3).

\i C:/Users/heath/Downloads/ExportSQL(10).sql

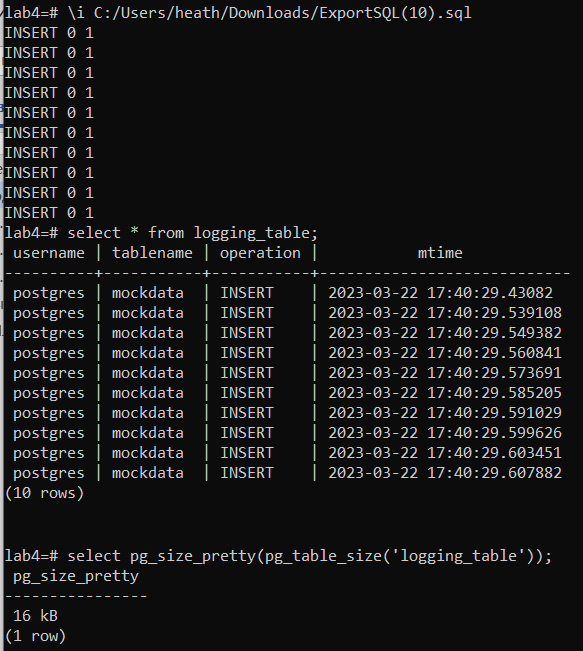


Рисунок 2 ― Вставка в таблицу

Для получения характеристик WAL в postgresql используется следующая команда, результат выполнения представлен на Рисунке 3:

SELECT \* FROM pg\_stat\_bgwriter;

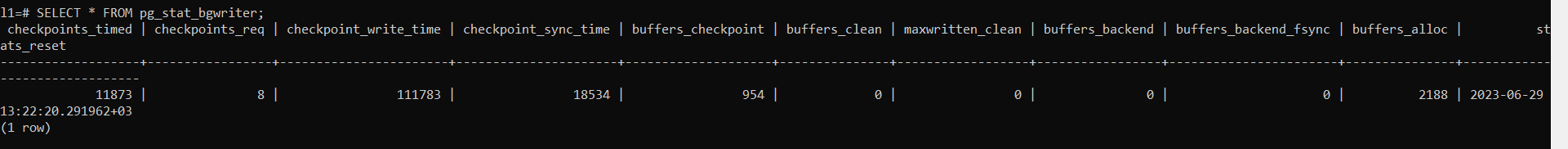


Рисунок 3 ― Характеристики журнала

Использованная команда выводит следующие характеристики WAL:

* Checkpoints Requested: Количество запросов на выполнение контрольной точки.
* Checkpoints Timed: Количество контрольных точек, которые были выполнены по расписанию.
* Buffers Allocated: Количество выделенных буферов для хранения данных.
* Buffers Backend: Количество буферов, используемых фоновыми процессами.
* Buffers Backend Fsync: Количество буферов, которые требуют синхронизации с диском фоновыми процессами.
* Buffers Allocated By Backend: Количество буферов, выделенных фоновыми процессами.
* Buffers Freed By Backend: Количество буферов, освобожденных фоновыми процессами.
* Max Written Clean: Максимальное количество записанных чистых (неизмененных) буферов.
* Max Written Dirty: Максимальное количество записанных грязных (измененных) буферов.
* Buffers Alloc: Количество выделенных буферов в данный момент.
* Buffers Clean: Количество чистых (неизмененных) буферов в данный момент.
* Buffers Backend Fsync Requested: Количество запросов на синхронизацию с диском фоновыми процессами.

Для определения местоположения файла данных была использована команда SHOW data\_directory, файл данных расположен на диске С. Файл WAL также расположен на диске C, его имя 0000000100000001000000FA.

## Оценка скорости запросов

Далее было проведено исследование средней скорости запросов к базе данных. В качестве первого запроса использовалась простая выборка, второй запрос объединяет операции выборки и обновления, третий ― операции вставки и удаления. Для замера времени запроса будем использовать инструмент \timing on. Результаты выполнения представлены в таблице 1.

Таблица 1 ― Оценка скорости запросов (WAL и Data на одном диске)

|  |  |
| --- | --- |
| Запрос | Время (мс) |
| select \* from mockdata0 where gender = 'M' and country = ‘Russia’; | 1.286 ms |
| UPDATE mockdata  SET country = (  SELECT country  FROM mockdata  WHERE last\_name LIKE 'Forester' AND first\_name LIKE 'Eve'  )  WHERE first\_name = 'John'; | 22.205 ms |
| CREATE TABLE deleted\_mock(  first\_name VARCHAR(500),  last\_name VARCHAR(500),  gender VARCHAR(5),  country VARCHAR(500));  WITH deleted\_rows AS (  DELETE FROM mockdata  WHERE country LIKE 'Jordan'  RETURNING \*  )  INSERT INTO deleted\_mock (first\_name, last\_name, gender, country)  SELECT first\_name, last\_name, gender, country  FROM deleted\_rows; | 45.395 ms |

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 ― Результат выполнения запроса

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 ― Результат выполнения запроса

Далее требуется разместить файлы журнала и данных на разных физических дисках.

* Останавливаем службу PostgreSQL;
* Переносим каталог pg\_wal в новое расположение;
* Даем права на папку в новом месте расположения пользователю, под которым запущена служба PostgreSQL. В Windows это по умолчанию NETWORK SERVICE;
* Делаем символическую ссылку на новый каталог из старого. В Windows для этого используется программа mklink. Нужно запустить командную строку от имени Администратора и ввести следующую команду:

mklink /D “PGDATA\pg\_wal” “новое\_расположение\_pg\_wal”

* Запускаем службу PostgreSQL.

В Таблице 2 представлены результаты запросов, когда данные и журнал находятся на разных дисках.

Таблица 2 ― Оценка скорости запросов (WAL и Data на разных дисках)

| Запрос | Время (мс) |
| --- | --- |
| select \* from mockdata0 where gender = 'M' and country = ‘Russia’; | 11.543 ms |
| UPDATE mockdata  SET country = (  SELECT country  FROM mockdata  WHERE last\_name LIKE 'Forester' AND first\_name LIKE 'Eve'  )  WHERE first\_name = 'Jack'; | 35.488 ms |
| CREATE TABLE deleted\_mock(  first\_name VARCHAR(500),  last\_name VARCHAR(500),  gender VARCHAR(5),  country VARCHAR(500));  WITH deleted\_rows AS (  DELETE FROM mockdata  WHERE country LIKE 'India'  RETURNING \*  )  INSERT INTO deleted\_mock (first\_name, last\_name, gender, country)  SELECT first\_name, last\_name, gender, country  FROM deleted\_rows; | 60.220 ms |

Как видим, время доступа несколько увеличилось, в среднем на 10 мс.

## Манипулирование журналом

Для получения характеристик WAL в postgresql используется следующая команда, результат выполнения представлен на Рисунке 6. Сравним с данными на рисунке 3, очевидно, они изменились после многих манипуляций.

SELECT \* FROM pg\_stat\_bgwriter;

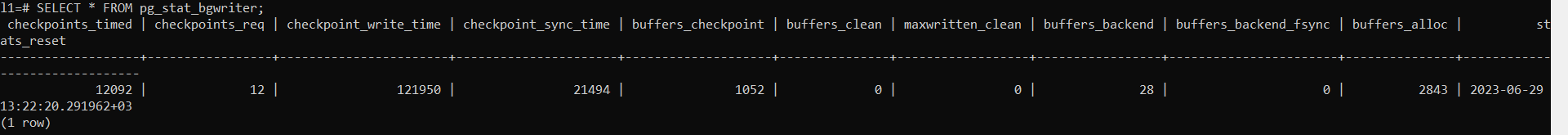


Рисунок 6 ― Характеристики журнала

Для получения данных необходимо применить утилиту pg\_waldump к файлу WAL. Так как WAL содержит большой объем данных, для получения информации по конкретной транзакции можно использовать флаг xid с идентификатором текущей транзакции. Для его получения можно использовать функцию txid\_current(). Результаты, полученные из WAL приведены ниже на Рисунках 7-8.

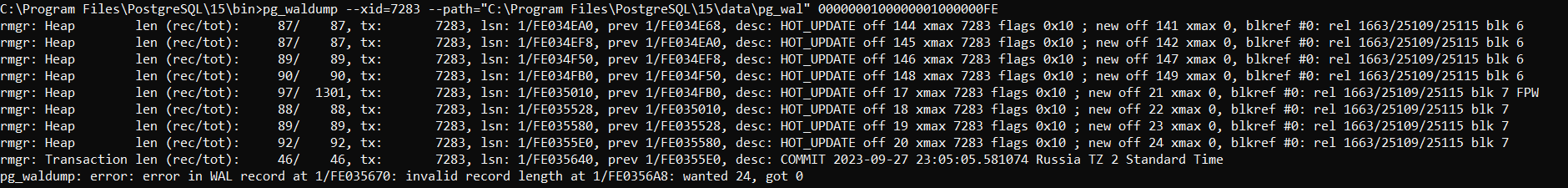


Рисунок 7 ― Результат вывода pg\_waldump (выборка + обновление)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 8 ― Результат вывода pg\_waldump (выборка + удаление)

Как можно видеть на представленных выше выдержках из файла, в записи журнала WAL в PostgreSQL содержатся следующие данные:

rmgr (Resource Manager): Идентификатор ресурсного менеджера, который указывает на тип операции, выполняемой в транзакции. Например, rmgr: HEAP указывает на операцию с кучей (heap).

lsn (Log Sequence Number): Уникальный идентификатор для каждой записи WAL, который позволяет определить порядок записей в журнале и установить точку восстановления.

prev (Previous LSN): Указывает на предыдущую запись WAL, которая была применена перед текущей записью.

desc (Description): Описание операции, выполненной в транзакции. Например, HOT\_UPDATE — Обновление данных, которое использует HOT (Heap-Only Tuples) механизм для эффективного управления данными в куче.

Далее было проведено исследование журнала в ситуациях, когда выполнение транзакции произошло с откатом. Для вызова отката была разработана представленная ниже функция, использующая команду ROLLBACK. В результате работы функции данные в incidents не были изменены. Как видно на Рисунке 9, последняя запись в журнале содержит команду ABORT, что говорит о том, что был проведен откат транзакции.

create or replace PROCEDURE rollback\_func()

language plpgsql as

$$

DECLARE

i INT;

BEGIN

i := 1;

RAISE NOTICE '%', txid\_current();

WHILE i < 2 LOOP

UPDATE mockdata

SET country = (

SELECT country

FROM mockdata

WHERE last\_name LIKE 'Ross' AND first\_name LIKE ' Alexander '

)

WHERE first\_name = 'Abdul';

i := i + 1;

END LOOP;

ROLLBACK;

END

$$;

CALL rollback\_func();

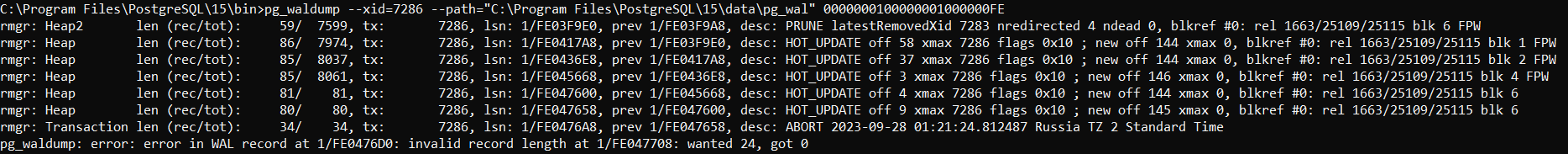


Рисунок 9 — Запись WAL о транзакции с откатом

## Восстановление журнала

Для оценки модели восстановления было необходимо определить текущий режим резервирования с помощью команды SHOW archive\_mode. Для этого был проанализирован файл postgresql.conf, в котором заданы параметры восстановления: archive\_mode = on, что позволяет использовать функционал postgresql по резервному восстановлению. Настройка параметров контрольных точек может быть проведена в разделе Checkpoints.

Далее в консоли была создана резервная копия базы данных с помощью команды (Рисунок 10):

pg\_dump --file=testrestore --username=postgres --dbname=BSUBD\_l1

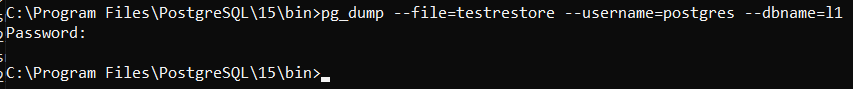


Рисунок 10 — Создание резервной копии бд

На Рисунке 11 видим, что резервная копия базы данных — это файл с кодом, который будет выполнен при восстановлении, что соответствует модели восстановления из кода.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11 — Файл кода бд

Данная команда сохранила информацию в файл testrestore. После этого отношение было очищено от данных. Для их восстановления была использована команда (Рисунок 12):

psql --username=postgres l1 < testrestore

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 12 — Восстановление бд

Усечение WAL в PostgreSQL используется для обеспечения надежности и восстановления данных в случае сбоев или сбоев системы. Он записывает все изменения данных, происходящие в базе данных, в журнал транзакций перед их фактическим применением к самой базе данных.

Усечение выполняется для удаления устаревших файлов журнала транзакций, которые больше не нужны для восстановления базы данных. Это позволяет освободить дисковое пространство и улучшить производительность.

Для усечения журнала необходимо остановить службу и выполнить следующую команду (Рисунок 13). Из множества файлов остался только один.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 13 — WAL после усечения.

# Вывод

В ходе выполнения данной лабораторной работы были получены знания о системе журналирования в PostgreSQL. Были изучены характеристики журнала, получены основы работы с транзакциями и подсистемой восстановления.