

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**МИРЭА** − **Российский технологический университет**»

**РТУ МИРЭА**

**Институт комплексной безопасности и цифровых технологий (ИКБ) Кафедра КБ-14 «Цифровые технологии обработки данных»**

**Направление подготовки**

**09.03.02 Информационные системы и технологии**

**Практическая работа №6-8**

**ОТЧЕТ**

Выполнил студент группы:

БСБО-07-20

Любовский С.В.

**Москва 2023г.**

**Выполнение задания**

Компания, в рамках развития своей IT-инфраструктуры, нуждается в улучшении и настройках политики безопасности баз данных и системе автоматических бэкапов

На основе эталонной модели внести следующие изменения или дополнения:

1. В системе должен присутствовать отдельный кластер для как минимум одной таблицы, которая будет часто использоваться (добавление, изменение и удаление данных).

CREATE INDEX printers\_idx ON printers(printer\_name);

CLUSTER  printers USING printers\_idx;

1. Настроить транзакционные параметры и фактор заполнения для оптимизации производительности.

-- Установить транзакционный уровень READ COMMITTED

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Установить фактор заполнения 80% для таблицы printers

ALTER TABLE printers SET (fillfactor = 80);

1. Предоставить отдельные допуски для доступа к кластеру.

CREATE ROLE worker;

CREATE ROLE manager;

GRANT SELECT, INSERT ON printers TO worker;

GRANT SELECT, INSERT, UPDATE ON printers TO manager;

1. На основном кластере должна быть настроена автоматическая очистка и бэкапирование для как минимум 2 таблиц, которые являются важными для работы системы и могут меняться со временем.

-- Настройка автоматической очистки

-- Установить параметр autovacuum для таблицы important\_table1

ALTER TABLE employees SET (autovacuum\_enabled = on);

ALTER TABLE contact\_persons SET  (autovacuum\_enabled  = on);

-- Установить параметры max\_worker\_processes, max\_parallel\_workers\_per\_gather и max\_parallel\_workers в конфигурационном файле

-- для управления параллельным выполнением очистки таблиц

ALTER SYSTEM SET max\_worker\_processes = 6;

ALTER SYSTEM SET max\_parallel\_workers\_per\_gather = 2;

ALTER SYSTEM SET max\_parallel\_workers = 4;

-- Установить параметры autovacuum\_vacuum\_scale\_factor и autovacuum\_analyze\_scale\_factor

-- для выполнения очистки и анализа таблицы important\_table1

ALTER TABLE employees SET (autovacuum\_vacuum\_scale\_factor = 0.1, autovacuum\_analyze\_scale\_factor = 0.05);

ALTER TABLE contact\_persons SET (autovacuum\_vacuum\_scale\_factor = 0.1, autovacuum\_analyze\_scale\_factor = 0.05);

Подготовим следующий shell скрипт для бэкапирования таблиц:

#!/bin/bash

pg\_dump.exe -t employees -t contact\_persons -U postgres -f "/backups/db.bak" clients\_database

echo Команда выполнена.

Далее в crontab -e выставляем следующие параметры:

0 3 \* \* \* /sripts/db\_back.sh

1. Настроить гибкую настраиваемую автоочистку и бэкапирование всей системы.

Пример настройки параметров очистки в конфигурационном файле:

autovacuum = on

autovacuum\_naptime = 3min

autovacuum\_vacuum\_threshold = 100

autovacuum\_analyze\_threshold = 50

autovacuum\_vacuum\_scale\_factor = 0.2

autovacuum\_analyze\_scale\_factor = 0.1

Далее делаем то же самое:

Скрипт:

#!/bin/bash

pg\_dumpall.exe -U postgres -f "/backups/db\_all.bak"

echo Команда выполнена.

Crontab:

0 1 \* \* \* /sripts/db\_back\_all.sh

1. Необходимо вести журналы посещения пользователей, отслеживая их активность в системе, и журналы использования самой нагруженной таблицы для анализа производительности

Выставляем следующие параметры в конф. файле:

log\_connections = on

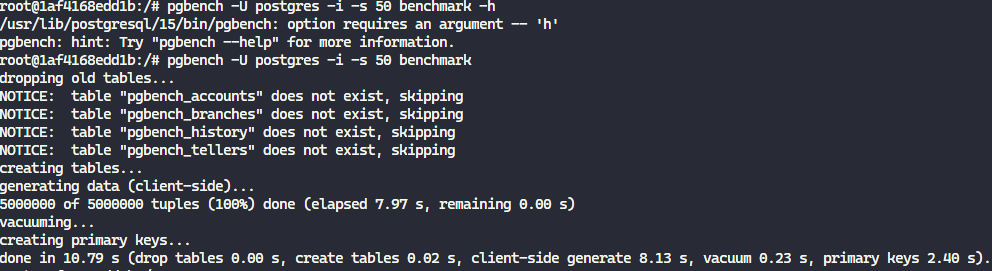
log\_disconnections = on

log\_statement = 'all'

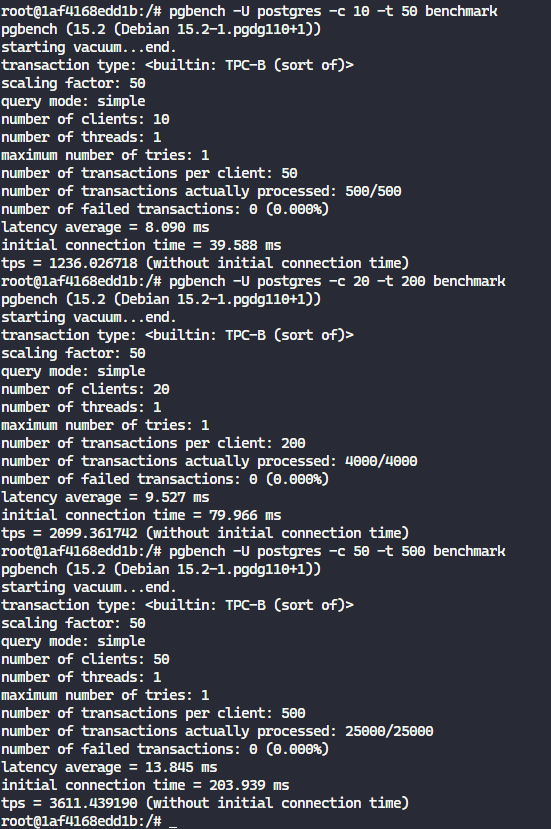
Данные параметры отвечают за логирование действий, происходящих в базе данных. log\_connections включает логирование подключений к базе данных, log\_disconnections включает логирование отключений от базы данных, а log\_statement включает логирование SQL-запросов, которые отправляются к базе данных.

1. Необходимо определить производительность системы при различных уровнях нагрузки. Для проведения нагрузочного тестирования базы данных использовать pgbench.

Подготовка конфигурации к тестам:



Тестирование с разным уровнем нагрузки:



1. Разработать стратегию мониторинга и оповещения для контроля состояния системы и оперативного реагирования на возможные проблемы. Это включает мониторинг производительности, использования ресурсов и состояния журналов, а также оповещения при превышении пороговых значений или возникновении ошибок.

import psutil

import time

MEMORY\_THRESHOLD = 80.0

CPU\_THRESHOLD = 80.0

DISK\_THRESHOLD = 80.0

while True:

    # мониторинг использования ресурсов процесса Postgres

    for proc in psutil.process\_iter(["pid", "name", "cpu\_percent", "memory\_percent"]):

        if proc.info["name"] == "postgres":

            pid = proc.info["pid"]

            cpu\_percent = proc.info["cpu\_percent"]

            mem\_percent = proc.info["memory\_percent"]

            mem\_info = proc.memory\_info()

            rss = mem\_info.rss / (1024 \* 1024)  # в мегабайтах

            vms = mem\_info.vms / (1024 \* 1024)  # в мегабайтах

            print(

                f"Postgres process (PID {pid}): CPU usage: {cpu\_percent}%, "

                f"memory usage: {mem\_percent}% ({rss:.2f} MB RSS, {vms:.2f} MB VMS)"

            )

            if mem\_percent > MEMORY\_THRESHOLD:

                print(

                    f"Postgres process (PID {pid}) "

                    f"is using too much memory: {mem\_percent}%"

                )

            if cpu\_percent > CPU\_THRESHOLD:

                print(

                    f"Postgres process (PID {pid}) "

                    f"is using too much CPU: {cpu\_percent}%",

*color*="yellow",

                )

    cpu\_percent = psutil.cpu\_percent()

    mem = psutil.virtual\_memory()

    mem\_percent = mem.percent

    mem\_used = mem.used / (1024 \* 1024 \* 1024)  # в гигабайтах

    disk = psutil.disk\_usage("/")

    disk\_percent = disk.percent

    disk\_used = disk.used / (1024 \* 1024 \* 1024)  # в гигабайтах

    print(

        f"System usage: CPU usage: {cpu\_percent}%, "

        f"memory usage: {mem\_percent}% ({mem\_used:.2f} GB used), "

        f"disk usage: {disk\_percent}% ({disk\_used:.2f} GB used)"

    )

    if mem\_percent > MEMORY\_THRESHOLD:

        print(f"System is using too much memory: {mem\_percent}%")

    if cpu\_percent > CPU\_THRESHOLD:

        print(f"System is using too much CPU: {cpu\_percent}%")

    if disk\_percent > DISK\_THRESHOLD:

        print(f"System is using too much disk: {disk\_percent}%")

    time.sleep(60)

Данный код представляет собой бесконечный цикл, который мониторит использование ресурсов на уровне системы и процесса Postgres. Для мониторинга используется библиотека psutil. Если использование памяти, CPU или диска превышает заданные пороговые значения (MEMORY\_THRESHOLD, CPU\_THRESHOLD и DISK\_THRESHOLD соответственно), то выводится сообщение с предупреждением о превышении порога. Для вывода цветного текста используется библиотека termcolor. Все проверки осуществляются в бесконечном цикле с интервалом в 60 секунд.