**Министерство науки и высшего образования Российской Федерации**
Федеральное государственное автономное
образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

ОТЧЕТ по лабораторной работе
«Этап 3. Администрирование и оптимизация»
по дисциплине «**Проектирование баз данных**»

| | |
|---|---|
| Автор: | Константинова Елизавета Анатольевна |
| Факультет: | ФИТиП |
| Группа: | M3203 |
| Преподаватели: | Шевчик Софья Владимировна |

ИТМО

Санкт-Петербург 2024

## Запросы к бд:

Вывести всех пользователей и их роли на определенном сервере которые сейчас онлайн

```
SELECT u.username, ur.role_id
FROM servers AS s
LEFT JOIN users u ON u.user_id = s.owner_id
LEFT JOIN user_roles AS ur ON s.server_id = ur.server_id and u.user_id =
ur.user_id
WHERE u.online_status = TRUE and s.server_id = 97664
```

Вывести пары людей которые сидят как минимум на двух одинаковых серверах

```
SELECT DISTINCT
    u1.username AS username1,
    u2.username AS username2
FROM
    user_roles ur1
        JOIN user_roles ur2 ON ur1.server_id = ur2.server_id AND ur1.user_id
<> ur2.user_id
        JOIN users u1 ON ur1.user_id = u1.user_id
        JOIN users u2 ON ur2.user_id = u2.user_id
GROUP BY
    u1.username, u2.username
HAVING
    COUNT(DISTINCT ur1.server_id) >= 2;
```

Вывести все сервера и их обладателей у которых есть хотя бы один текстовый канал

```
SELECT
    s.server_name,
    u.username AS owner_username
FROM
    servers s
        JOIN channels c ON s.server_id = c.server_id
        JOIN users u ON s.owner_id = u.user_id
WHERE
    c.channel_type = 'text_channel'
GROUP BY
    s.server_name, u.username;
```

## Сервис, который вызывает Explain Analyze для каждого запроса и измеряет Cost:

Dockerfile:

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build-env
WORKDIR /discord

COPY . ./


RUN dotnet restore

RUN dotnet publish -c Release -o out
```

```
FROM mcr.microsoft.com/dotnet/runtime:8.0
WORKDIR /discord
COPY --from=build-env /discord/out .
EXPOSE 8080
ENTRYPOINT ["sh", "-c", "dotnet discord.dll"]
```

docker-compose:

```
analyze_app:
  restart: no
  container_name: analyze_app
  environment:
    ATTEMPT_COUNT: ${ATTEMPT_COUNT}
  build:
    context: ./discord
    dockerfile: Dockerfile
  volumes:
    - ./itogi:/docker-entrypoint-initdb.d/itogi
  depends_on:
    db:
      condition: service_healthy
  env_file:
    - .env
  ports:
    - "8080:8080"
```

Analyze.cs:

```
using System.Text.RegularExpressions;
using Npgsql;

namespace discord;

class Analyze
{
    static void Main(string[] args)
    {
        var timestamp = DateTime.Now.ToString("yyyyMMddHHmmss");
        var filePath = $"/docker-entrypoint-
initdb.d/itogi/query_itogi_{timestamp}.txt";
        var connectionString =
"Host=db;Username=postgres;Password=mypassword;Database=discord-
db;Port=5432;";
        Console.WriteLine("ANALYZE SERVICE STARTED");

        var attemptCount =
int.Parse(Environment.GetEnvironmentVariable("ATTEMPT_COUNT") ?? "4");

        string[] queries =
        {
            """
            SELECT u.username, ur.role_id
            FROM servers AS s
            LEFT JOIN users u ON u.user_id = s.owner_id
```

```
            LEFT JOIN user_roles AS ur ON s.server_id = ur.server_id and
u.user_id = ur.user_id
            WHERE u.online_status = TRUE and s.server_id = 97664
            """,
            """
            SELECT DISTINCT
                u1.username AS username1,
                u2.username AS username2
            FROM
                user_roles ur1
                    JOIN user_roles ur2 ON ur1.server_id = ur2.server_id AND
ur1.user_id <> ur2.user_id
                    JOIN users u1 ON ur1.user_id = u1.user_id
                    JOIN users u2 ON ur2.user_id = u2.user_id
            GROUP BY
                u1.username, u2.username
            HAVING
                COUNT(DISTINCT ur1.server_id) >= 2;
            """,
            """
            SELECT
                s.server_name,
                u.username AS owner_username
            FROM
                servers s
                    JOIN channels c ON s.server_id = c.server_id
                    JOIN users u ON s.owner_id = u.user_id
            WHERE
                c.channel_type = 'text_channel'
            GROUP BY
                s.server_name, u.username;
            """
        };

        var costs = new double[queries.Length][];
        for (var i = 0; i < costs.Length; i++)
        {
            costs[i] = new double[attemptCount];
        }

        try
        {
            using (var writer = new StreamWriter(filePath))
            {
                using (var connection = new
NpgsqlConnection(connectionString))
                {
                    connection.Open();

                    for (var q = 0; q < queries.Length; q++)
                    {
                        var query = queries[q];
                        Console.WriteLine($"Executing query {q +
1}/{queries.Length}: {query}");
                        for (var a = 0; a < attemptCount; a++)
                        {
                            try
```

```
                            {
                                using (var command = new
NpgsqlCommand($"EXPLAIN ANALYZE {query}", connection))
                                {
                                    using (var reader =
command.ExecuteReader())
                                    {
                                        var res = new List<double>();

                                        while (reader.Read())
                                        {
                                            var result = reader.GetString(0);
                                            var match = Regex.Match(result,
@"cost=(\d+\.\d+)..(\d+\.\d+)");

                                            if (match.Success)
                                            {
                                                var startCost =
double.Parse(match.Groups[1].Value);

                                                var endCost =
double.Parse(match.Groups[2].Value);

                                                res.Add(endCost);
                                            }
                                        }

                                        if (a == attemptCount - 1)
                                        {
                                            var minC = res.Min();
                                            var maxC = res.Max();
                                            var avgC = res.Average();
                                            writer.WriteLine($"query:
{query}\n");

                                            writer.WriteLine($"best case
time: {minC}");

                                            writer.WriteLine($"worst case
time: {maxC}");

                                            writer.WriteLine($"average case
time: {avgC}");

                                            writer.WriteLine('\n');

                                        }
                                    }
                                }
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"ERROR: {ex.Message}");
                            }
                        }
                    }
                }
            }

            Console.WriteLine($"RESULTS SAVED: {filePath}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"error writing to file: {ex.Message}");
```

```
        }
    }
}
```

Добавленные индексы:

```
CREATE INDEX idx_servers_owner_id ON servers (owner_id);
CREATE INDEX idx_user_roles_server_user_id ON user_roles (server_id,
user_id);
CREATE INDEX idx_user_roles_user_id_server_id ON user_roles (user_id,
server_id);
CREATE INDEX idx_channels_channel_type ON channels (channel_type);
```

Сравнительная таблица:

| изначально | 1 запрос | 2 запрос | 3 запрос |
|---|---|---|---|
| Best | 8.3 | 1454.06 | 0.38 |
| Worst | 1814.93 | 25352528.95 | 3869.82 |
| Avg | 728.044 | 5058657.189999999 | 2428.444285714286 |
| После индексов | | | |
| Best | 0.88 | 1404.62 | 0.37 |
| Worst | 17.51 | 16744308.71 | 3682.29 |
| Avg | 10.324 | 4240299.414285715 | 1909.295 |
| Лучше на | 7180% | 19,2% | 27,2% |

Создание партиции таблицы ролей пользователей:

```
ALTER TABLE user_roles RENAME TO user_roles_old;

CREATE TABLE user_roles
(
    user_role_id SERIAL PRIMARY KEY,
    user_id      INTEGER NOT NULL REFERENCES users (user_id),
    role_id      INTEGER NOT NULL REFERENCES roles (role_id),
    server_id    INTEGER NOT NULL REFERENCES servers (server_id)
) PARTITION BY RANGE (user_role_id);

ALTER TABLE user_roles_old
    ADD CONSTRAINT user_roles_old
        CHECK (user_role_id >= 1 AND user_role_id <= 100000);

CREATE TABLE user_roles_1 PARTITION OF user_roles
    FOR VALUES FROM
(
    1
) TO
(
    50000
```

```
);

CREATE TABLE user_roles_2 PARTITION OF user_roles
    FOR VALUES FROM
(
    50000
) TO
(
    100001
);

WITH moved_rows AS (
DELETE
FROM user_roles_old a
WHERE user_role_id >= 1
  AND user_role_id < 50000 RETURNING a.*
)
INSERT
INTO user_roles_1
SELECT *
FROM moved_rows;

WITH moved_rows AS (
DELETE
FROM user_roles_old a
WHERE user_role_id >= 50000
  AND user_role_id < 100001 RETURNING a.*
)
INSERT
INTO user_roles_2
SELECT *
FROM moved_rows;

ALTER TABLE user_roles_old DROP CONSTRAINT user_roles_old;
```

| изначально | 1 запрос | 2 запрос | 3 запрос |
|---|---|---|---|
| Best | 0.88 | 1404.62 | 0.37 |
| Worst | 17.51 | 16744308.71 | 3682.29 |
| Avg | 10.324 | 4240299.414285715 | 1909.295 |
| После партиции | | | |
| Best | 8.3 | 565.91 | 0.37 |
| Worst | 1820.78 | 16829892.86 | 3682.29 |
| Avg | 778.18 | 3313748.391 | 1909.295 |
| Лучше на | -98% | 30% | 0% |

Скрипт, который создает бэкапы базы данных каждые n-часов, последние m-бекапов хранятся

backup.sh:

```
#!/bin/bash

DB_USER=${DB_SUPERUSER}
DB_NAME=${POSTGRES_DB}
DB_PASSWORD=${POSTGRES_PASSWORD}

BACKUP_DIR="/backup"

BACKUP_COUNT=${M}

TIMESTAMP=$(date +'%Y%m%d%H%M%S')
BACKUP_FILE="$BACKUP_DIR/backup_$TIMESTAMP.sql"
sleep_time=${N}

export PGPASSWORD="$DB_PASSWORD"

mkdir -p "$BACKUP_DIR"

sleep "$sleep_time"

pg_dump -h db -p 5432 -U "$DB_USER" -d "$DB_NAME" > "$BACKUP_FILE"

BACKUP_FILES_COUNT=$(ls -1 "$BACKUP_DIR"/*.sql | wc -l)
if [ "$BACKUP_FILES_COUNT" -gt "$BACKUP_COUNT" ]; then
    ls -1t "$BACKUP_DIR"/*.sql | tail -n +$(($BACKUP_COUNT + 1)) | xargs rm -
f
fi

unset PGPASSWORD

echo "backup"
```

Dockerfile:

```
FROM alpine:latest
RUN apk add --no-cache postgresql-client
COPY backup.sh ./backup.sh
RUN chmod +x /backup.sh
EXPOSE 8081

CMD ["sh", "/backup.sh"]
```

docker-compose:

```
backup:
  restart: always
  container_name: backup
```

```
  environment:
    POSTGRES_DB: ${POSTGRES_DB}
    POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    DB_USERNAME: ${DB_USERNAME}
    DB_SUPERUSER: ${DB_SUPERUSER}
    N: ${N}
    M: ${M}
  build:
    context: ./backup
    dockerfile: Dockerfile
  volumes:
    - ./backup:/backup
    - ./backup/backup.sh:/docker-entrypoint-initdb.d/backup.sh
  depends_on:
    db:
      condition: service_healthy
  env_file:
    - .env
  ports:
    - "8081:8081"
```

.env:

```
POSTGRES_DB="discord-db"
POSTGRES_PASSWORD="mypassword"
DB_USERNAME="postgres"
DB_SUPERUSER="postgres"
FILLING_AMOUNT=100000
ATTEMPT_COUNT=10
N=10m
M=48
```

2 реплики Postgres с использованием Patroni:

```
docker-compose:
patroni:
    image: cybertec/pgwatch2
    container_name: patroni
    restart: unless-stopped
    depends_on:
      - db
    ports:
      - "2392:2392"
    environment:
      PATRONI_RESTAPI_CONNECT_ADDRESS: "0.0.0.0:2392"

  replica1:
    image: postgres
    container_name: replica1
    restart: unless-stopped
    environment:
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_USER: ${DB_SUPERUSER}
      POSTGRES_DB: ${POSTGRES_DB}
```

```
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - replica1_data:/var/lib/postgresql/data
    depends_on:
      - db

  replica2:
    image: postgres
    container_name: replica2
    restart: unless-stopped
    environment:
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_USER: ${DB_SUPERUSER}
      POSTGRES_DB: ${POSTGRES_DB}
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - replica2_data:/var/lib/postgresql/data
    depends_on:
      - db

volumes:
  replica2_data:
  replica1_data:
```

Итоговая структура проекта: