МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

**Лабораторная работа 1**

**по курсу «Базы Данных»**

Выполнил: Куценко М.Д.

Группа: М8О-312Б-22

Преподаватель: А.В. Малахов

Москва, 2025

Оглавление

[1.0 Данные 2](#__RefHeading___Toc269_596683903)

[1.1 Индексы 4](#__RefHeading___Toc271_596683903)

[1.1.1 BTREE 4](#__RefHeading___Toc273_596683903)

[1.1.2 BRIN 4](#__RefHeading___Toc285_596683903)

[1.1.3 GIN 5](#__RefHeading___Toc291_596683903)

[1.2 Транзакции 6](#__RefHeading___Toc319_596683903)

[1.2.1 READ COMMITED 6](#__RefHeading___Toc293_596683903)

[1.2.2 REPEATABLE READ 7](#__RefHeading___Toc295_596683903)

[1.2.3 SERIALIZABLE 7](#__RefHeading___Toc297_596683903)

[1.3 Транзакции 8](#__RefHeading___Toc323_596683903)

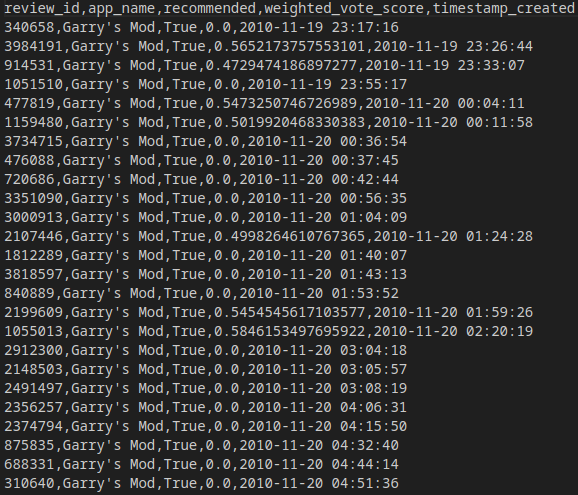
[1.3.1 pg\_bigm, pg\_trgm 8](#__RefHeading___Toc299_596683903)

[1.3.2 pgcrypto 9](#__RefHeading___Toc309_596683903)

[Вывод 10](#__RefHeading___Toc321_596683903)

# **1.0 Данные**

В качестве датасета взял следующий с обзорами Steam: https://www.kaggle.com/datasets/najzeko/steam-reviews-2021/data

После отбора нескольких колонок, 10 млн записей и сортировки по возрастанию дат получил следующий набор данных в формате csv:

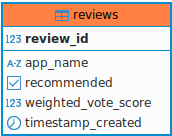
Написал следующий Dockerfile:

version: "1.0"  
services:  
postgres:  
  container\_name: "index\_testing"  
  image: postgres:16.4  
  environment:  
    POSTGRES\_DB: "db"  
    POSTGRES\_USER: "Maxim"  
    POSTGRES\_PASSWORD: "12345"  
  volumes:  
    - ./mapped:/data  
    - ./migrations:/docker-entrypoint-initdb.d  
  ports:  
    - "5432:5432"

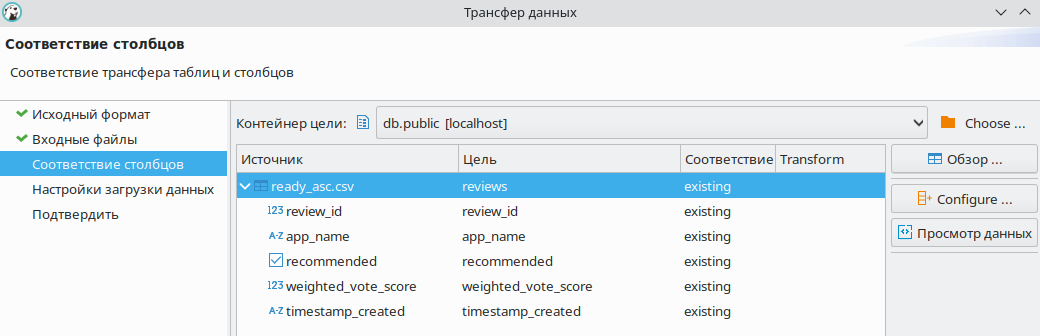
Он запускает следующий SQL-файл для создания таблицы:

|  |
| --- |
| CREATE TABLE "reviews"(   "review\_id" SERIAL NOT NULL,   "app\_name" TEXT NOT NULL,   "recommended" BOOLEAN NOT NULL,   "weighted\_vote\_score" FLOAT NOT NULL,   "timestamp\_created" TIMESTAMP NOT NULL ); ALTER TABLE   "reviews" ADD PRIMARY KEY("review\_id"); |

В Dbeaver увидел, что таблица создалась:



Загрузил данные через инструментарий dbeaver:



# **1.1 Индексы**

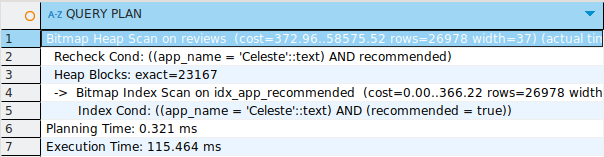
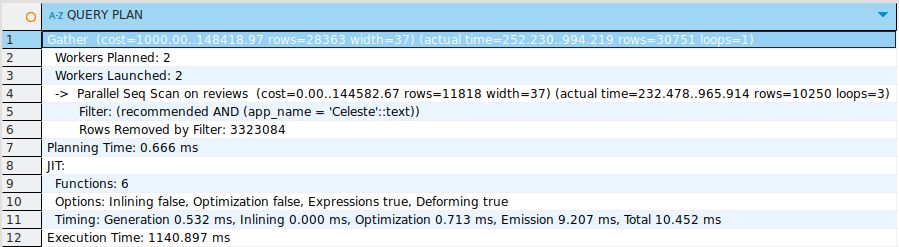
## 1.1.1 BTREE

Индекс BTREE эффективен для широкого набора задач и может быть использован практически для любых столбцов где реализовано сравнение.

Запрос без индекса и с индексом:

EXPLAIN analyze select \* from reviews WHERE app\_name = 'Celeste' AND recommended = true  
  
CREATE INDEX idx\_app\_recommended ON reviews(app\_name, recommended);  
EXPLAIN analyze select \* from reviews WHERE app\_name = 'Celeste' AND recommended = true;  
DROP INDEX idx\_app\_recommended;

Сравним время выполнения:

Стало быстрее почти в 10 раз.

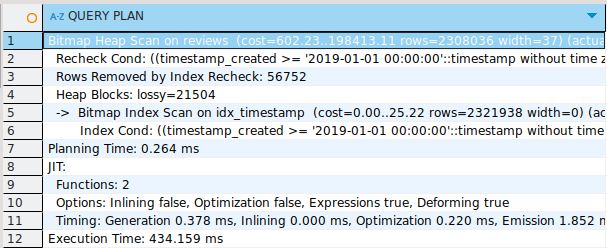
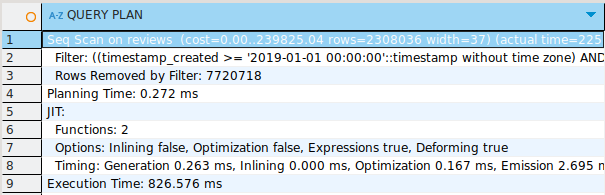
## 1.1.2 BRIN

Индекс эффективен для отсортированных величин, например в случае когда данные заносятся в БД с указыванием их времени добавления.

Запрос:  
explain analyze SELECT \* FROM reviews WHERE timestamp\_created BETWEEN '2019-01-01' AND '2020-02-01';

CREATE INDEX idx\_timestamp ON reviews USING brin(timestamp\_created);  
explain analyze SELECT \* FROM reviews WHERE timestamp\_created BETWEEN '2019-01-01' AND '2020-02-01';  
DROP INDEX idx\_timestamp;

Сравним время выполнения:

Стало быстрее в 2 раза

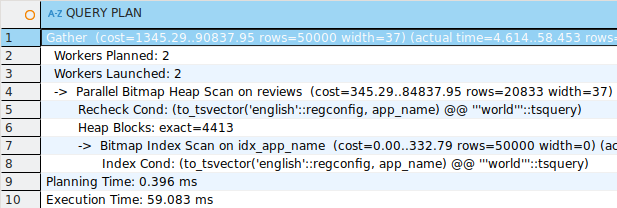
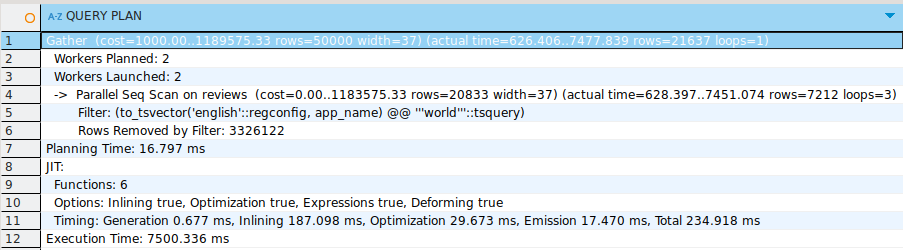
## 1.1.3 GIN

Данный индекс как правило используется для поиска в массивах, больших текстах, а также полнотекстового поиска.

Запрос:

EXPLAIN analyze SELECT \* FROM reviews WHERE to\_tsvector('english', app\_name) @@ to\_tsquery('english', 'world');  
  
CREATE INDEX idx\_app\_name ON reviews USING gin(to\_tsvector('english', app\_name));  
EXPLAIN analyze SELECT \* FROM reviews WHERE to\_tsvector('english', app\_name) @@ to\_tsquery('english', 'world');  
DROP INDEX idx\_app\_name;

Сравним время выполнения:



Полнотекстовый поиск стал быстрее более чем в 100 раз.

# **1.2 Транзакции**

## 1.2.1 READ COMMITED

Данный запрос демонстрирует Non-repeatable read и Phantom read на уровне READ COMMITED:

|  |
| --- |
| -- Transaction 1 begin transaction; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT AVG(weighted\_vote\_score) FROM reviews WHERE app\_name LIKE '%Witcher%'; -- Transaction 2 runs here SELECT AVG(weighted\_vote\_score) FROM reviews WHERE app\_name LIKE '%Witcher%'; -- We get non-repeatable read SELECT COUNT(\*) FROM reviews where app\_name LIKE '%Witcher%' and EXTRACT(EPOCH FROM (NOW() - timestamp\_created)) < 3600; -- We get phantom read COMMIT;   -- Transaction 2 begin transaction; UPDATE reviews SET weighted\_vote\_score = weighted\_vote\_score / 1.1 WHERE app\_name LIKE '%Witcher%'; INSERT INTO reviews (app\_name, recommended, weighted\_vote\_score, timestamp\_created) VALUES ('Witcher', true, 0.5, NOW()); COMMIT; |

## 1.2.2 REPEATABLE READ

Запрос, демонстрирующий защищённость от проблем из предыдущего пункта на уровне REPEATABLE READ:

-- Transaction 1  
BEGIN;  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
SELECT COUNT(\*) FROM reviews WHERE app\_name = 'Cyberpunk 2077';  
-- Transaction 2 runs here  
SELECT COUNT(\*) FROM reviews WHERE app\_name = 'Cyberpunk 2077';  
-- Protected from non-repeatable read  
SELECT COUNT(\*) FROM reviews  
where app\_name = 'Cyberpunk 2077' and EXTRACT(EPOCH FROM (NOW() - timestamp\_created)) < 3600;  
-- Also protected from phantom read  
COMMIT;

-- Transaction 2  
BEGIN;  
INSERT INTO reviews (app\_name, recommended, weighted\_vote\_score, timestamp\_created)  
VALUES ('Cyberpunk 2077', true, 0.85, NOW());  
COMMIT;

## 1.2.3 SERIALIZABLE

Запрос, демонстрирующий ошибку сериализации на уровне SERIALIZABLE:

-- Transaction 1  
BEGIN;  
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SELECT AVG(weighted\_vote\_score) FROM reviews WHERE app\_name LIKE '%Witcher%';  
-- Transaction 2 runs  
UPDATE reviews  
SET weighted\_vote\_score = weighted\_vote\_score \* 1.1  
WHERE app\_name LIKE '%Witcher%';  
COMMIT; -- Probable serialization error

-- Transaction 2  
BEGIN;  
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SELECT AVG(weighted\_vote\_score) FROM reviews WHERE app\_name LIKE '%Witcher%';  
UPDATE reviews  
SET weighted\_vote\_score = weighted\_vote\_score \* 0.9  
WHERE app\_name LIKE '%Witcher%';  
COMMIT;

# **1.3 Транзакции**

## 1.3.1 pg\_bigm, pg\_trgm

Данные расширения предоставляют функционал для работы с текстами, в том числе возможность создавать индексы на текстовых колонках, имея почти идентичную функциональность и схожее устройство.

Пример запросов:

CREATE EXTENSION IF NOT EXISTS pg\_trgm;

SELECT distinct app\_name, similarity(app\_name, 'witch') AS similarity\_score  
FROM reviews  
ORDER BY similarity\_score DESC  
LIMIT 10;

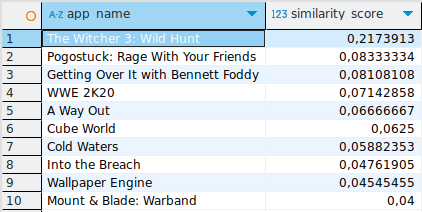
CREATE INDEX idx\_app\_name ON reviews USING gin (app\_name gin\_trgm\_ops);  
DROP INDEX idx\_app\_name;

CREATE EXTENSION IF NOT EXISTS pg\_bigm;

explain analyze SELECT distinct app\_name, bigm\_similarity(app\_name, 'witch') AS similarity\_score  
FROM reviews  
ORDER BY similarity\_score DESC  
LIMIT 10;

CREATE INDEX idx\_app\_name ON reviews USING gin (app\_name gin\_bigm\_ops);  
DROP INDEX idx\_app\_name;

Результат выполнения запроса на сходство:



## 1.3.2 pgcrypto

Позволяет зашифровывать и расшифровывать сообщения, а также предоставляет другой функционал, связанный с хэшированием и т.п. Полезен в целях безопасности.

Запрос:

CREATE EXTENSION IF NOT EXISTS pgcrypto;

INSERT INTO reviews (app\_name, recommended, weighted\_vote\_score, timestamp\_created)  
VALUES (  
  pgp\_sym\_encrypt('SECRET', 'code'),  
  true,  
  0.85,  
  NOW()  
);

SELECT pgp\_sym\_decrypt(app\_name::bytea, 'code') as decrypted\_app\_name  
FROM (select app\_name  
from reviews  
order by timestamp\_created desc  
limit 1);

Результат выполнения:



# **Вывод**

В ходе выполнения данной лабораторной работы вспомнил информацию об индексах и транзакциях. Закрепил знания об индексах на практике, использовав индексы btree, gin, brin; замерил, какой прирост по скорости запросов дают данные индексы. Закрепил знания о транзкациях на практике, использовав разные уровни изоляции и повторив разные аномалии.

Поработал с дополнениями pg\_bigm, pg\_trigm, pgcrypto и рассмотрел в каких сценариях они полезны, применил их в работе.