Задание №2: Производительность индексов

# Тестовый стенд

Тесты производились на официальном Docker образе PostgreSQL 15.4.

Была создана база данных для заготовки социальной сети с одной таблицей:

CREATE TABLE "User" (   
        "Id"            SERIAL NOT NULL,   
        "PublicId"      uuid NOT NULL,   
        "Firstname"     varchar(100) NOT NULL,   
        "Secondname"    varchar(200) NOT NULL,   
        "BirthDate"     date,   
        "Biography"     varchar(5000),   
        "City"          varchar(100),   
        "PasswordHash"  varchar(300)   
    );

Таблица не содержала ни одного индекса и была заполнена случайно сгенерированными данными в количестве 1 000 000 строк.

Был получен, запрос, который используется монолитным Api при поиске пользователей по имени и фамилии:

SELECT    
    "PublicId",    
    "Firstname",    
    "Secondname",    
    "BirthDate",    
    "Biography",    
    "City"             
 FROM    
     "User"    
WHERE    
    LOWER("Firstname") LIKE 'wil%'    
    AND    
    LOWER("Secondname") LIKE 'tur%'

Этот запрос использовался для нагрузочного тестирования БД с помощью встроенной утилиты **pgbench** для 1/10/100/1000 параллельных выполнений.

# До индекса

Explain запрос до создания индекса:

otus\_social\_network=> EXPLAIN SELECT "PublicId", "Firstname", "Secondname", "BirthDate", "Biography", "City" FROM "User" WHERE LOWER("Firstname") LIKE 'wil%' AND LOWER("Secondname") LIKE 'tur%';   
                                                    QUERY PLAN                                                       
------------------------------------------------------------------------------------------------------------------   
 Gather  (cost=1000.00..29247.83 rows=25 width=57)   
   Workers Planned: 2   
   ->  Parallel Seq Scan on "User"  (cost=0.00..28245.33 rows=10 width=57)   
         Filter: ((lower(("Firstname")::text) ~~ 'wil%'::text) AND (lower(("Secondname")::text) ~~ 'tur%'::text))   
(4 rows)

Был выбран тип индекса по-умолчанию: **B-tree**, с добавлением в индекс точного выражения, включающего в себя функцию LOWER.

Скрипт создания индекса:

CREATE INDEX fn\_ln ON "User" (Lower("Firstname") varchar\_pattern\_ops, Lower("Secondname") varchar\_pattern\_ops);

Explain запроса после создания индекса:

otus\_social\_network=> EXPLAIN SELECT "PublicId", "Firstname", "Secondname", "BirthDate", "Biography", "City" FROM "User" WHERE LOWER("Firstname") LIKE 'wil%' AND LOWER("Secondname") LIKE 'tur%';   
                                                                                                        QUERY PLAN                                                             
                                                  
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------   
-----------------------------------------------   
 Bitmap Heap Scan on "User"  (cost=127.43..225.27 rows=25 width=57)   
   Filter: ((lower(("Firstname")::text) ~~ 'wil%'::text) AND (lower(("Secondname")::text) ~~ 'tur%'::text))   
   ->  Bitmap Index Scan on fn\_ln  (cost=0.00..127.42 rows=25 width=0)   
         Index Cond: ((lower(("Firstname")::text) ~>=~ 'wil'::text) AND (lower(("Firstname")::text) ~<~ 'wim'::text) AND (lower(("Secondname")::text) ~>=~ 'tur'::text) AND    
(lower(("Secondname")::text) ~<~ 'tus'::text))   
(4 rows)

Результаты прогона тестов:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метрика** | **Количество потоков** | **До индекса** | **После индекса** |
| Latency, ms | 1 | 101,06 | 0,571 |
| Throughput, tps | 1 | 9,89 | 1750,087 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метрика** | **Количество потоков** | **До индекса** | **После индекса** |
| Latency, ms | 10 | 423,64 | 0,898 |
| Throughput, tps | 10 | 23,60 | 11130,899 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метрика** | **Количество потоков** | **До индекса** | **После индекса** |
| Latency, ms | 100 | 3908,23 | 9,402 |
| Throughput, tps | 100 | 25,58 | 10635,695 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метрика** | **Количество потоков** | **До индекса** | **После индекса** |
| Latency, ms | 1000 | 38932,52 | 96,815 |
| Throughput, tps | 1000 | 25,68 | 10328,977 |