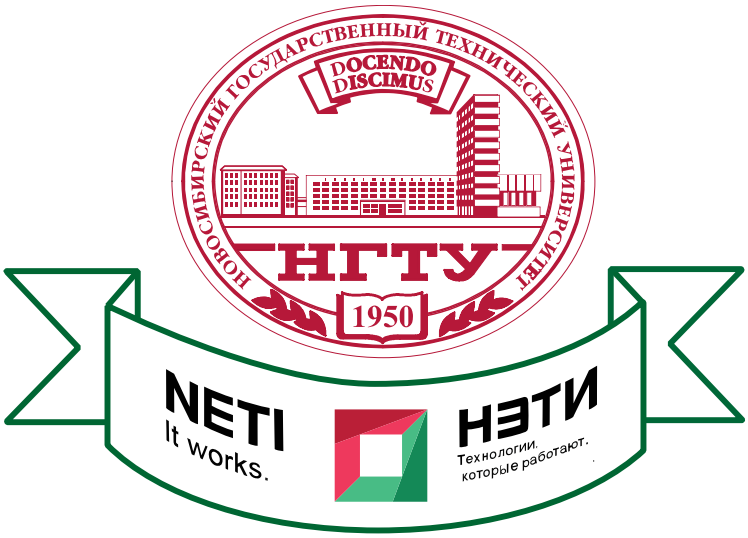
Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

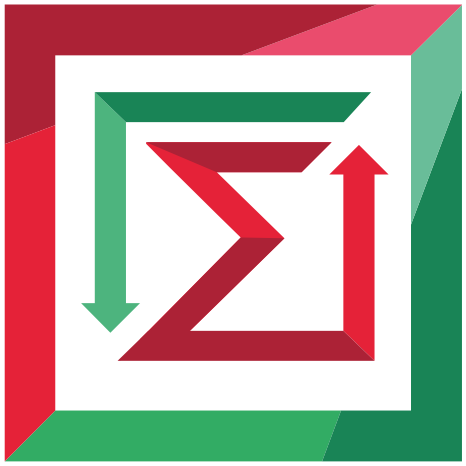


Кафедра теоретической и прикладной информатики

Лабораторная работа № 7

по дисциплине «Администрирование информационных систем»

ВЫПОЛНЕНИЕ ЗАПРОСОВ



Факультет:	ПМИ
Группа:	ПМИ-02
Бригада:	8
Студенты:	Сидоров Даниил,
	Дюков Богдан
Преподаватель:	Аврунев О.Е.

Новосибирск

2026

Ход работы

1. Создали таблицу со следующей структурой (вместе с autovacuum_enabled=false):

Вариант	Столбцы, индексы
8	Идентификатор и два числовых столбца индексы по второму.

```
demo=# CREATE SCHEMA lab7schema;
CREATE SCHEMA
demo=# CREATE TABLE lab7schema.my_table (
demo(# id SERIAL PRIMARY KEY,
demo(# num1 INTEGER,
demo(# num2 INTEGER
demo(# ) WITH (autovacuum_enabled=false);
CREATE TABLE
demo=# CREATE INDEX idx_num2 ON lab7schema.my_table (num2);
CREATE INDEX
demo=#
```

2. Заполнили таблицу данными:

Вариант	Количество строк
8	200000

Использовали для генерации сценарий pgsql, сразу после убедились в корректности вставки:

```
demo=# DO $$
demo$# BEGIN
demo$#     FOR i IN 1..200000 LOOP
demo$#         INSERT INTO lab7schema.my_table (num1, num2)
demo$#         VALUES (random() * 100, random() * 100);
demo$#     END LOOP;
demo$# END;
demo$# $$ LANGUAGE plpgsql;
DO
demo=# SELECT * FROM lab7schema.my_table LIMIT 10;
 id | num1 | num2
----+------+------
  1 |   16 |   55
  2 |   55 |   83
  3 |    8 |   30
  4 |   88 |   51
  5 |   64 |    2
  6 |   80 |   58
  7 |    6 |   31
  8 |    3 |   34
  9 |   95 |   74
 10 |   66 |    2
(10 rows)

demo=# SELECT COUNT(*) FROM lab7schema.my_table;
 count
--------
 200000
(1 row)

demo=#
```

3. Привели значения статистики данных для таблицы в целом (количество строк и страниц):

```
demo=# ANALYZE lab7schema.my_table;
ANALYZE
demo=# SELECT
demo-#     reltuples AS rows,
demo-#     relpages AS pages
demo-# FROM
demo-#     pg_class
demo-# WHERE
demo-#     oid = 'lab7schema.my_table'::regclass;
  rows  | pages
--------+-------
 200000 |  1082
(1 row)
```

Привели значения статистики данных для каждого из столбцов. Для удобства, сначала выводим: название столбца, кол-во различных значений, корреляция, ширина столбца:

```
demo=# SELECT
    attname AS column_name,
    n_distinct AS distinct_values,
    correlation,
    avg_width AS average_width
FROM
    pg_stats
WHERE
    tablename = 'my_table' AND schemaname = 'lab7schema';
 column_name | distinct_values |  correlation   | average_width
-------------+-----------------+----------------+---------------
 id          |              -1 |              1 |             4
 num1        |             101 |    0.016371738 |             4
 num2        |             101 | -0.00067745027 |             4
(3 rows)
```

Теперь выводим: название столбца, наиболее часто встречающиеся значения в столбце:

```
demo=# SELECT
    attname AS column_name,
    most_common_vals AS common_values
FROM
    pg_stats
WHERE
    tablename = 'my_table' AND schemaname = 'lab7schema';
 column_name |
                                                                 common_values


-------------+-------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------
 id          |
 num1        | {20,80,14,31,93,35,96,27,43,91,25,54,4,79,58,64,3,9,18,32,51,33,70,87
,92,61,42,48,73,99,7,44,49,1,94,6,13,68,84,97,46,17,10,23,37,39,41,86,21,60,62,74,36
,95,82,11,28,66,89,5,75,81,50,59,65,52,55,57,2,26,22,67,76,24,30,47,77,53,85,19,56,6
3,78,83,16,71,72,98,12,40,29,45,69,34,38,90,15,88,8}
 num2        | {12,25,9,99,83,70,20,29,67,80,5,35,42,88,11,58,75,34,44,89,46,68,22,6
,19,7,17,37,79,1,24,93,8,14,41,43,47,61,73,90,94,52,96,64,82,27,32,50,71,39,56,97,10
,40,84,98,18,26,55,45,48,53,54,57,78,85,92,66,31,38,72,91,69,74,4,15,2,36,60,62,63,8
1,33,77,86,51,3,87,13,21,59,95,23,28,49,76,30,65,16}
(3 rows)
```

4. Выполнили запрос:

Вариант	Столбцы, индексы
8	Получение строк из таблицы, условие вида between на проиндексированный столбец.

```
demo=# SELECT *
demo-# FROM lab7schema.my_table
demo-# WHERE num2 BETWEEN 10 AND 20;
   id   | num1 | num2
--------+------+------
     23 |   79 |   16
     34 |   96 |   11
     37 |   75 |   15
     40 |   72 |   15
     43 |   88 |   11
     52 |   57 |   14
     61 |   59 |   14
     73 |   88 |   11
     77 |   39 |   17
     89 |   97 |   12
     99 |   70 |   19
    122 |   63 |   18
    150 |   28 |   14
```

5. Привели план выполнения и время выполнения запроса. Используем explain analyze:

```
demo=# EXPLAIN ANALYZE
demo-# SELECT *
demo-# FROM lab7schema.my_table
demo-# WHERE num2 BETWEEN 10 AND 20;
EXPLAIN ANALYZE
SELECT *
FROM lab7schema.my_table
WHERE num2 BETWEEN 10 AND 20;
                                                         QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on my_table  (cost=335.10..1766.57 rows=23298 width=12) (actual time=2.456..20.147 rows=22016 loops=1)
   Recheck Cond: ((num2 >= 10) AND (num2 <= 20))
   Heap Blocks: exact=1082
   ->  Bitmap Index Scan on idx_num2  (cost=0.00..329.28 rows=23298 width=0) (actual time=2.067..2.068 rows=22016 loops=1)
         Index Cond: ((num2 >= 10) AND (num2 <= 20))
 Planning Time: 0.998 ms
 Execution Time: 22.463 ms
(7 rows)
```

На основе полученных результатов можно сказать следующее:

- сначала движок базы данных выполняет сканирование индекса, чтобы быстро найти строки, которые удовлетворяют условию (создается битовая карта “Bitmap”, указывающая, какие строки в таблице соответствуют условию);
- затем движок базы данных выполняет сканирование кучи (фактические данные таблицы), используя битовую карту, чтобы эффективно извлечь только те строки, которые были идентифицированы во время сканирования индекса.

Время планирования составило 0.998мс, а общее время выполнения - 22.463мс.

6. Увеличили количество строк в таблице в два раза. Для этого вставили в неё текущие строки ещё раз (сразу после вставки убедились, что строк стало действительно в 2 раза больше):

```
demo=# INSERT INTO lab7schema.my_table (num1, num2)
demo-# SELECT num1, num2
demo-# FROM lab7schema.my_table;
INSERT 0 200000
demo=# SELECT COUNT(*) FROM lab7schema.my_table;
 count
--------
 400000
(1 row)

demo=#
```

7. Повторно привели план и время выполнения запроса из п.4.

```
demo=# EXPLAIN ANALYZE
SELECT *
FROM lab7schema.my_table
WHERE num2 BETWEEN 10 AND 20;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on my_table  (cost=665.82..3527.44 rows=46575 width=12) (actual time=5.847..39.323 rows=44032 loops=1)
   Recheck Cond: ((num2 >= 10) AND (num2 <= 20))
   Heap Blocks: exact=2163
   ->  Bitmap Index Scan on idx_num2  (cost=0.00..654.17 rows=46575 width=0) (actual time=5.028..5.029 rows=44032 loops=1)
         Index Cond: ((num2 >= 10) AND (num2 <= 20))
 Planning Time: 1.183 ms
 Execution Time: 44.117 ms
(7 rows)
```

8. Собрали статистику данных для таблицы:

```
demo=# ANALYZE lab7schema.my_table;
ANALYZE
demo=#
```

9. Привели значения статистики данных из п.3.
   Статистика для таблицы в целом:

```
demo=# SELECT
    reltuples AS rows,
    relpages AS pages
FROM
    pg_class
WHERE
    oid = 'lab7schema.my_table'::regclass;
  rows   | pages
--------+-------
 400000 |  2163
(1 row)

demo=#
```

   Статистика для каждого из столбцов (также разделили на два запроса):

```
demo=# SELECT
    attname AS column_name,
    n_distinct AS distinct_values,
    correlation,
    avg_width AS average_width
FROM
    pg_stats
WHERE
    tablename = 'my_table' AND schemaname = 'lab7schema';
 column_name | distinct_values | correlation  | average_width
-------------+-----------------+--------------+---------------
 num1        |             101 | 0.0057165693 |             4
 num2        |             101 |  0.005393079 |             4
 id          |              -1 |            1 |             4
(3 rows)

demo=#
```

```
demo=# SELECT
    attname AS column_name,
    most_common_vals AS common_values
FROM
    pg_stats
WHERE
    tablename = 'my_table' AND schemaname = 'lab7schema';
 column_name |
                                                                      common_values


-------------+--------------------------------------------------------------------------
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
-------------------------------------------------------
 num1        | {76,57,81,93,95,90,22,40,70,50,58,15,44,21,19,27,31,16,30,49,56,64,68
,97,65,92,99,12,25,53,83,33,39,86,10,42,60,52,89,82,43,36,77,8,23,29,54,9,13,34,78,8
7,24,71,6,7,51,61,63,75,32,48,84,17,28,35,47,3,59,62,69,88,5,96,20,37,41,85,79,26,46
,45,74,80,67,2,38,66,4,14,94,72,18,98,55,11,1,91,73}
 num2        | {2,98,38,12,20,81,89,39,48,33,41,66,73,18,42,8,27,40,58,69,99,52,54,7
,76,79,80,45,59,77,78,75,4,44,16,17,32,88,28,61,10,15,50,25,51,65,72,87,91,26,37,96,
53,34,43,46,11,24,36,57,70,30,35,93,97,47,13,23,82,22,5,9,71,68,49,74,19,95,1,6,67,8
4,21,31,85,3,83,94,14,86,29,60,64,55,56,63,62,92,90}
 id          |
(3 rows)
```

10. Привели план и время выполнения запроса:

```
demo=# EXPLAIN ANALYZE
SELECT *
FROM lab7schema.my_table
WHERE num2 BETWEEN 10 AND 20;
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on my_table  (cost=669.40..3536.28 rows=46925 width=12) (actual time=5.636..43.431 rows=44032 loops=1)
   Recheck Cond: ((num2 >= 10) AND (num2 <= 20))
   Heap Blocks: exact=2163
   ->  Bitmap Index Scan on idx_num2  (cost=0.00..657.67 rows=46925 width=0) (actual time=4.795..4.796 rows=44032 loops=1)
         Index Cond: ((num2 >= 10) AND (num2 <= 20))
 Planning Time: 1.973 ms
 Execution Time: 48.346 ms
(7 rows)
```

Сравнили планируемое и реальное время выполнения для пунктов:

Пункт 5:
- Планируемое время выполнения: 1766.57 мс
- Реальное время выполнения: 20.147 мс

Пункт 7:
- Планируемое время выполнения: 3527.44 мс
- Реальное время выполнения: 39.323 мс

Пункт 10:
- Планируемое время выполнения: 3536.28 мс
- Реальное время выполнения: 43.431 мс

Как видно, реальное время выполнения каждого запроса значительно меньше планируемого времени. Планировщик запросов PostgreSQL делает консервативные оценки времени выполнения.

Также эти значения показывают, что реальное время выполнения увеличивается после вставки дополнительных строк, что ожидаемо, поскольку большее количество строк в таблице обычно приводит к большему времени выполнения запроса.

Хотя ANALYZE помогает планировщику запросов делать более обоснованные решения, он не гарантирует, что время выполнения запроса всегда будет уменьшаться. В нашем случае реальное время даже немного увеличилось (с 39 до 43 мс).

Мы можем попробовать ещё раз собрать статистику и привести план и время выполнения запроса:

```
demo=# ANALYZE lab7schema.my_table;
ANALYZE
demo=# EXPLAIN ANALYZE
SELECT *
FROM lab7schema.my_table
WHERE num2 BETWEEN 10 AND 20;
                                                        QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on my_table  (cost=642.81..3482.48 rows=45111 width=12) (actual time=4.524..25.623 rows=44032 loops=1)
   Recheck Cond: ((num2 >= 10) AND (num2 <= 20))
   Heap Blocks: exact=2163
   ->  Bitmap Index Scan on idx_num2  (cost=0.00..631.53 rows=45111 width=0) (actual time=3.706..3.707 rows=44032 loops=1)
         Index Cond: ((num2 >= 10) AND (num2 <= 20))
 Planning Time: 1.156 ms
 Execution Time: 30.603 ms
(7 rows)
```

В данном случае реальное время уменьшилось в разы (стало 25мс).