Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа программной инженерии

Отчет по лабораторной работе

3.1 «Проектирование аналитической схемы базы данных»

по дисциплине

«Системы управления базами данных»

Выполнила

Студентка гр. 5130904/10101 Никифорова Е. А.

Руководитель Прокофьев О. В.

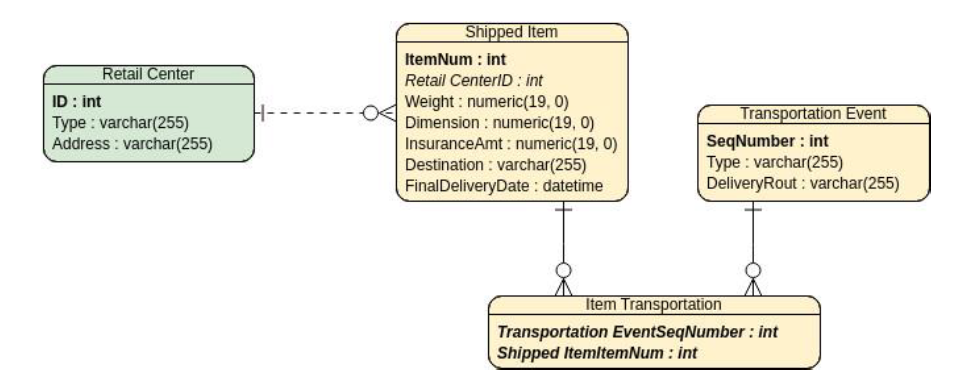
Санкт-Петербург

2024

Постановка задачи

Седьмое практическое задание связано с проектированием схемы базы данных для аналитики. Будем исходить из того, что приложение, для которого была сделана база данных в задании стала очень популярной и по ней каждый день можно собирать большой объем статистической информации. Результатом данного практического задания являются: скрипты создания базы данных, хранимая процедура (генератор) для ее заполнения, анализ плана выполнения запроса.

Пример используемой ER диаграммы (система доставки UPS)



Требования к БД

• Одна таблица (Item transportation) должна содержать не меньше 10 млн. записей, которые со временем теряют актуальность.

• Другая таблица (Shipped Item), связанная с первой, должна содержать не меньше 1 млн. записей.

• Остальные таблицы должны содержать по 5-10 записей.

• В одной из таблиц с количеством записей 1 млн. должна быть колонка с текстом, по которой будет необходимо настроить полнотекстовый поиск.

Практическая часть включает:

1) наполнение таблицы, для этого нужно написать хранимую процедуру - генератор на языке ***plpython3u***, которая использует словари (для строковых типов), случайные значения (для строковых, числовых типов).

2) оценку скорости выполнения запросов.

Для этого могут быть использованы механизмы секционирования, наследования и индексов. Необходимо подготовить два запроса:

• Запрос к одной таблице, содержащий фильтрацию по нескольким полям.

• Запрос к нескольким связанным таблицам, содержащий фильтрацию по нескольким полям.

3) оценка скорости полнотекстового поиска (использование функций to\_tsvector to\_tsquery)

Для каждого из этих запросов необходимо провести следующие шаги:

• Получить план выполнения запроса без использования индексов (удаление индекса или отключение его использования в плане запроса).

• Получить статистику (IO и Time) выполнения запроса без использования индексов.

• Создать нужные индексы, позволяющие ускорить запрос.

• Получить план выполнения запроса с использованием индексов и сравнить с первоначальным планом.

• Получить статистику выполнения запроса с использованием индексов и сравнить с первоначальной статистикой.

• Оценить эффективность выполнения оптимизированного запроса.

Решение:

За основу была принята база данных из задания.

Скрипты создания таблиц:

CREATE TABLE RetailCenter (

ID SERIAL PRIMARY KEY,

Type VARCHAR,

Address VARCHAR

);

CREATE TABLE TransportationEvent (

SeqNumber SERIAL PRIMARY KEY,

Type VARCHAR,

DeliveryRoute VARCHAR

);

CREATE TABLE ShippedItem (

ItemNum SERIAL PRIMARY KEY,

RetailCenterID INT,

Dimension NUMERIC,

InsuranceAmt NUMERIC,

Destination VARCHAR,

FinalDeliveryDate TIMESTAMP,

FOREIGN KEY (RetailCenterID) REFERENCES RetailCenter(ID)

);

CREATE TABLE ItemTransportation (

TransportationEventSeqNumber  INT REFERENCES TransportationEvent(SeqNumber),

ShippedItemItemNum INT REFERENCES ShippedItem(ItemNum)

);

Скрипты заполнения таблиц:

CREATE EXTENSION plpython3u;

CREATE OR REPLACE FUNCTION fill\_shipped\_item()

RETURNS VOID AS $$

import random

from datetime import datetime, timedelta

destinations = ["Street1", "Street2", "Street3", "Street4", "Street5", "Street6", "Street7", "Street8", "Street9", "Street10"]

shipped\_item\_insert = plpy.prepare("INSERT INTO ShippedItem (RetailCenterID, dimension, insuranceAmt, destination, finalDeliverydate) VALUES ($1, $2, $3, $4, $5)", ["int", "int", "int", "text", "timestamp"])

for \_ in range(1\_000\_000):

retail\_center\_id = random.randint(1, 6)

dimension = random.randint(1, 100)

insurance\_amt = random.randint(1, 1000)

destination = random.choice(destinations)

final\_delivery\_date = datetime.now() + timedelta(days=random.randint(1, 30)) + timedelta(hours=random.randint(1, 30))

plpy.execute(shipped\_item\_insert, [retail\_center\_id, dimension, insurance\_amt, destination, final\_delivery\_date])

$$ LANGUAGE plpython3u;

CREATE OR REPLACE FUNCTION fill\_item\_transport()

RETURNS VOID AS $$

import random

plpy.info('Started')

item\_transport\_insert = plpy.prepare("INSERT INTO ItemTransportation (TransportationEventSeqNumber, ShippedItemItemNum) VALUES ($1, $2)", ["int", "int"])

rows\_num = 10000000

for i in range(rows\_num):

transportation\_event\_seq\_number = random.randint(1, 15)

shipped\_item\_item\_num = random.randint(2, 1000000)

plpy.execute(item\_transport\_insert, [transportation\_event\_seq\_number, shipped\_item\_item\_num])

plpy.info('Finished')

$$ LANGUAGE plpython3u;

Однотабличный запрос:

SET enable\_indexscan = OFF;

SET enable\_bitmapscan = OFF;

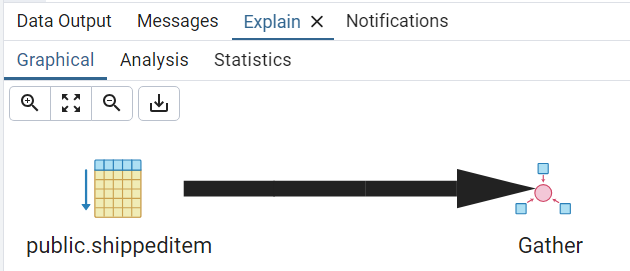
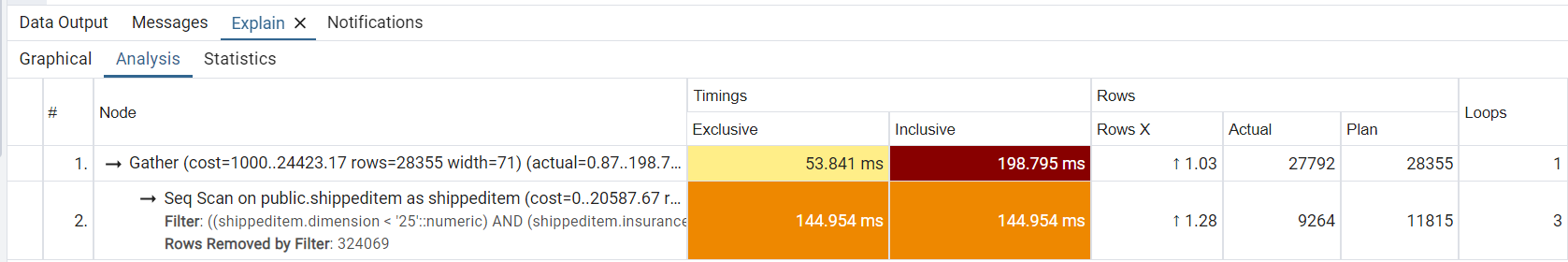
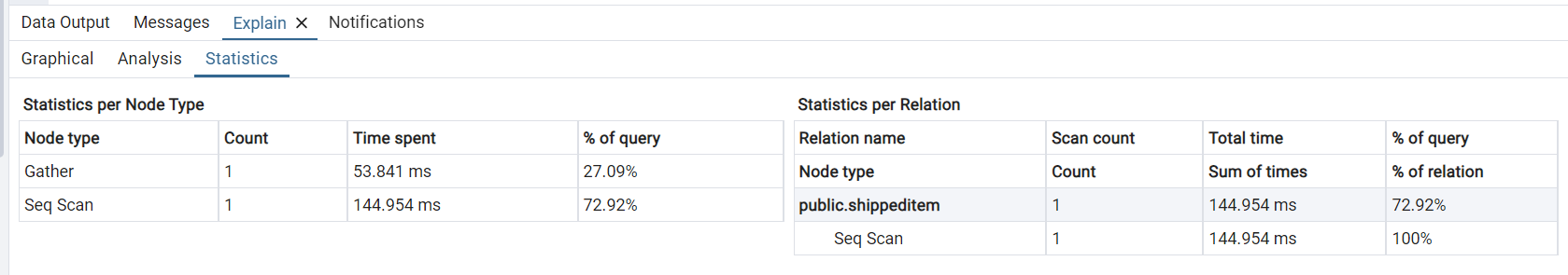
EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

SELECT \* FROM ShippedItem

WHERE RetailCenterID = 2

AND Dimension < 25

AND InsuranceAmt > 300;

x 

[

{

"Plan": {

"Node Type": "Gather",

"Parallel Aware": false,

"Async Capable": false,

"Startup Cost": 1000.00,

"Total Cost": 24423.17,

"Plan Rows": 28355,

"Plan Width": 71,

"Actual Startup Time": 0.870,

"Actual Total Time": 198.795,

"Actual Rows": 27792,

"Actual Loops": 1,

"Output": ["itemnum", "retailcenterid", "dimension", "insuranceamt", "destination", "finaldeliverydate", "vectorized"],

"Workers Planned": 2,

"Workers Launched": 2,

"Single Copy": false,

"Plans": [

{

"Node Type": "Seq Scan",

"Parent Relationship": "Outer",

"Parallel Aware": true,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "shippeditem",

"Startup Cost": 0.00,

"Total Cost": 20587.67,

"Plan Rows": 11815,

"Plan Width": 71,

"Actual Startup Time": 0.034,

"Actual Total Time": 144.954,

"Actual Rows": 9264,

"Actual Loops": 3,

"Output": ["itemnum", "retailcenterid", "dimension", "insuranceamt", "destination", "finaldeliverydate", "vectorized"],

"Filter": "((shippeditem.dimension < '25'::numeric) AND (shippeditem.insuranceamt > '300'::numeric) AND (shippeditem.retailcenterid = 2))",

"Rows Removed by Filter": 324069,

"Workers": [

{

"Worker Number": 0,

"Actual Startup Time": 0.023,

"Actual Total Time": 125.271,

"Actual Rows": 8115,

"Actual Loops": 1

},

{

"Worker Number": 1,

"Actual Startup Time": 0.026,

"Actual Total Time": 125.748,

"Actual Rows": 8497,

"Actual Loops": 1

}

]

}

]

},

"Planning Time": 0.352,

"Triggers": [

],

"Execution Time": 200.280

}

]

Однотабличный запрос с индексом:

CREATE INDEX idx\_shipped\_filter ON ShippedItem (RetailCenterID, Dimension, InsuranceAmt);

SET enable\_indexscan = ON;

SET enable\_bitmapscan = ON;

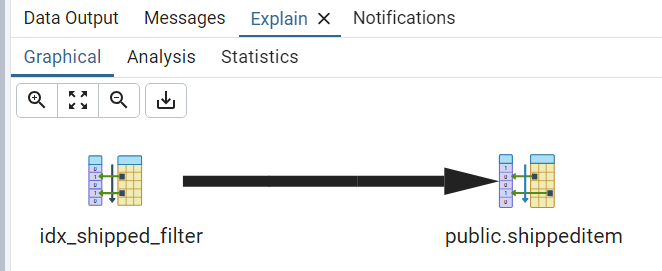
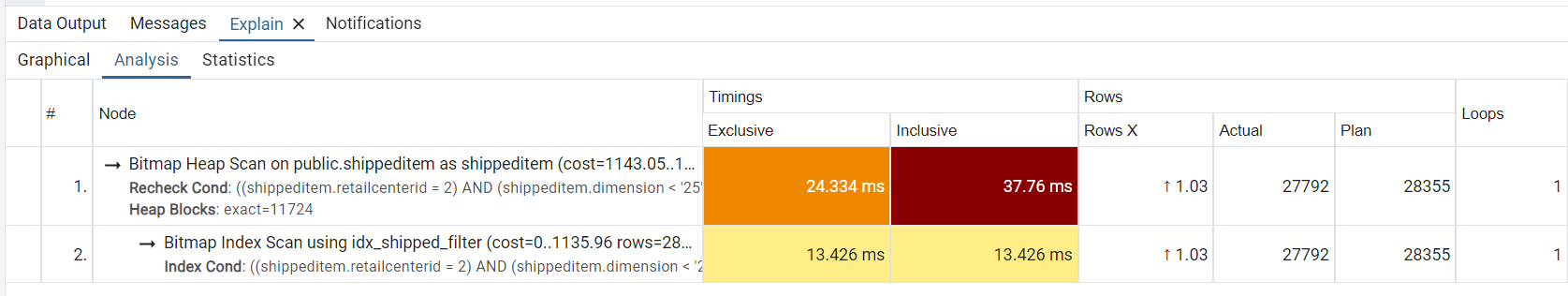
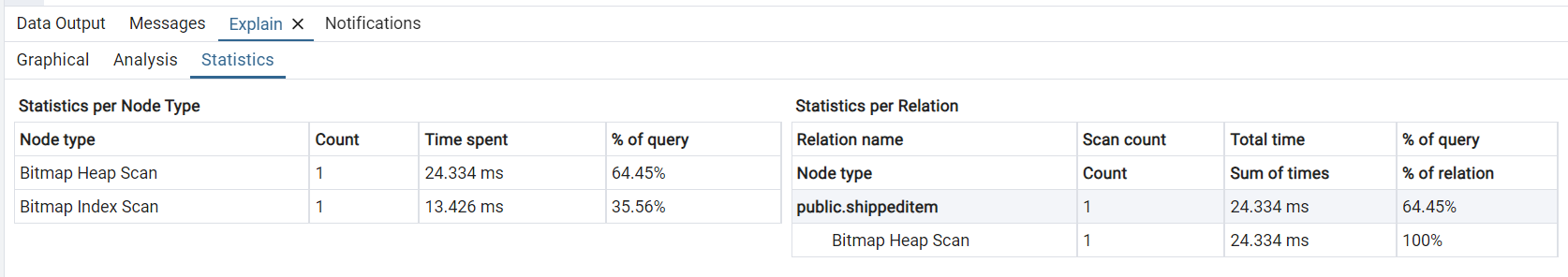
EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

SELECT \* FROM ShippedItem

WHERE RetailCenterID = 2

AND Dimension < 25

AND InsuranceAmt > 300;

[

{

"Plan": {

"Node Type": "Bitmap Heap Scan",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "shippeditem",

"Startup Cost": 1143.05,

"Total Cost": 14935.26,

"Plan Rows": 28355,

"Plan Width": 71,

"Actual Startup Time": 15.423,

"Actual Total Time": 37.760,

"Actual Rows": 27792,

"Actual Loops": 1,

"Output": ["itemnum", "retailcenterid", "dimension", "insuranceamt", "destination", "finaldeliverydate", "vectorized"],

"Recheck Cond": "((shippeditem.retailcenterid = 2) AND (shippeditem.dimension < '25'::numeric) AND (shippeditem.insuranceamt > '300'::numeric))",

"Rows Removed by Index Recheck": 0,

"Exact Heap Blocks": 11724,

"Lossy Heap Blocks": 0,

"Plans": [

{

"Node Type": "Bitmap Index Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Index Name": "idx\_shipped\_filter",

"Startup Cost": 0.00,

"Total Cost": 1135.96,

"Plan Rows": 28355,

"Plan Width": 0,

"Actual Startup Time": 13.426,

"Actual Total Time": 13.426,

"Actual Rows": 27792,

"Actual Loops": 1,

"Index Cond": "((shippeditem.retailcenterid = 2) AND (shippeditem.dimension < '25'::numeric) AND (shippeditem.insuranceamt > '300'::numeric))"

}

]

},

"Planning Time": 2.123,

"Triggers": [

],

"Execution Time": 39.296

}

]

Многотабличный запрос:

SET enable\_indexscan = off;

SET enable\_bitmapscan = off;

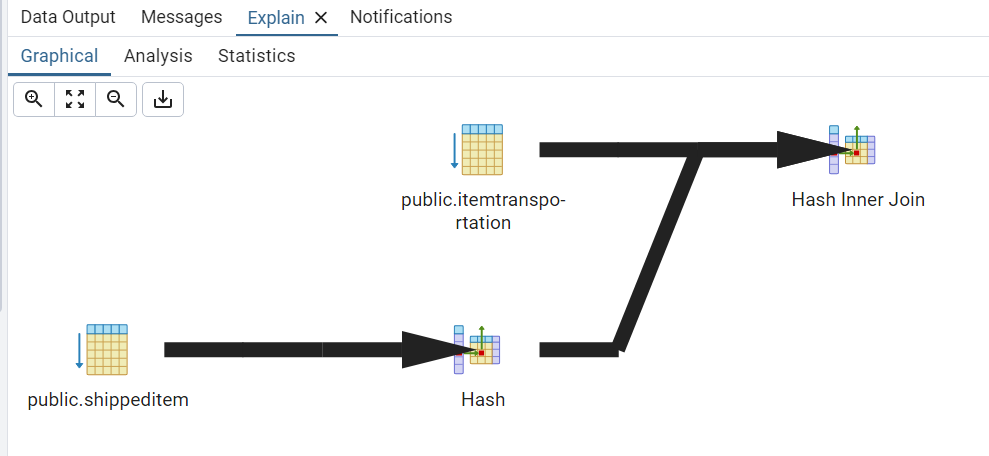
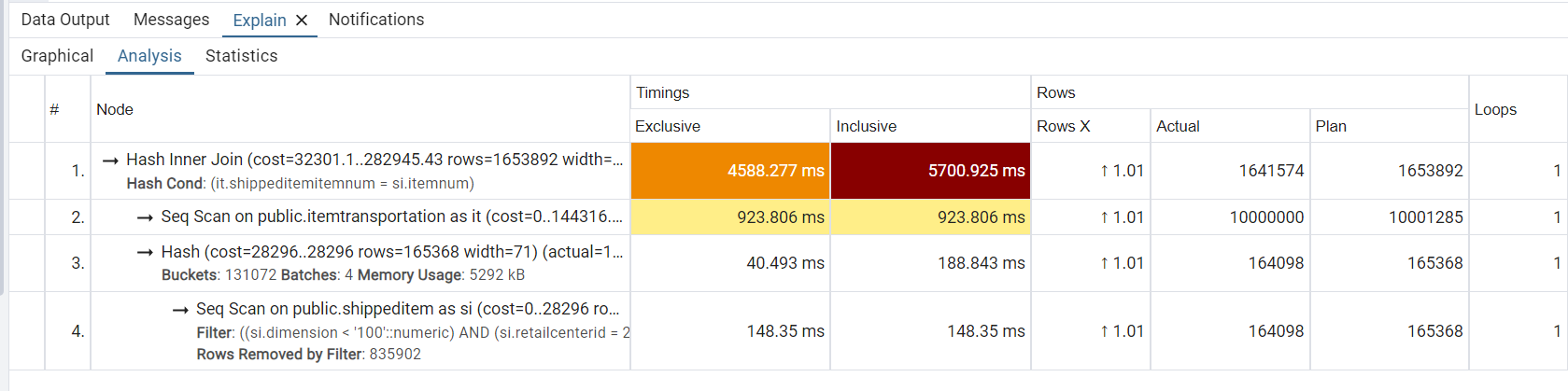
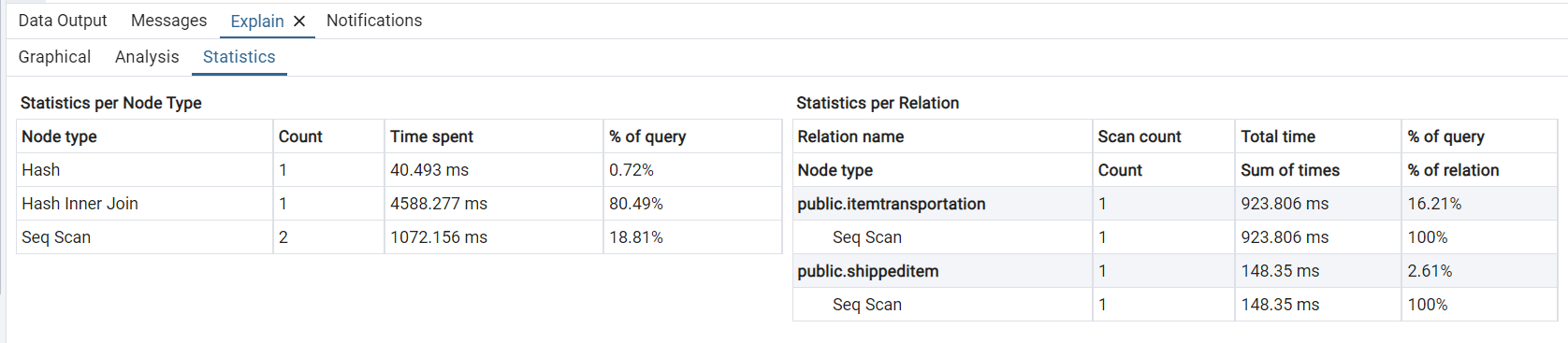
EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

SELECT \* FROM ShippedItem SI

JOIN ItemTransportation IT ON SI.ItemNum = IT.ShippedItemItemNum

WHERE SI.RetailCenterID = 2

AND SI.Dimension < 100;

[

{

"Plan": {

"Node Type": "Hash Join",

"Parallel Aware": false,

"Async Capable": false,

"Join Type": "Inner",

"Startup Cost": 32301.10,

"Total Cost": 282945.43,

"Plan Rows": 1653892,

"Plan Width": 79,

"Actual Startup Time": 189.966,

"Actual Total Time": 5700.925,

"Actual Rows": 1641574,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized", "it.transportationeventseqnumber", "it.shippeditemitemnum"],

"Inner Unique": true,

"Hash Cond": "(it.shippeditemitemnum = si.itemnum)",

"Plans": [

{

"Node Type": "Seq Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "itemtransportation",

"Schema": "public",

"Alias": "it",

"Startup Cost": 0.00,

"Total Cost": 144316.85,

"Plan Rows": 10001285,

"Plan Width": 8,

"Actual Startup Time": 0.127,

"Actual Total Time": 923.806,

"Actual Rows": 10000000,

"Actual Loops": 1,

"Output": ["it.transportationeventseqnumber", "it.shippeditemitemnum"]

},

{

"Node Type": "Hash",

"Parent Relationship": "Inner",

"Parallel Aware": false,

"Async Capable": false,

"Startup Cost": 28296.00,

"Total Cost": 28296.00,

"Plan Rows": 165368,

"Plan Width": 71,

"Actual Startup Time": 188.841,

"Actual Total Time": 188.843,

"Actual Rows": 164098,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized"],

"Hash Buckets": 131072,

"Original Hash Buckets": 131072,

"Hash Batches": 4,

"Original Hash Batches": 4,

"Peak Memory Usage": 5292,

"Plans": [

{

"Node Type": "Seq Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "si",

"Startup Cost": 0.00,

"Total Cost": 28296.00,

"Plan Rows": 165368,

"Plan Width": 71,

"Actual Startup Time": 0.020,

"Actual Total Time": 148.350,

"Actual Rows": 164098,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized"],

"Filter": "((si.dimension < '100'::numeric) AND (si.retailcenterid = 2))",

"Rows Removed by Filter": 835902

}

]

}

]

},

"Planning Time": 0.192,

"Triggers": [

],

"Execution Time": 5750.033

}

]

Многотабличный запрос с индексом:

CREATE INDEX idx\_shippeditem\_rcid ON ShippedItem (RetailCenterID);

SET enable\_indexscan = ON;

SET enable\_bitmapscan = ON;

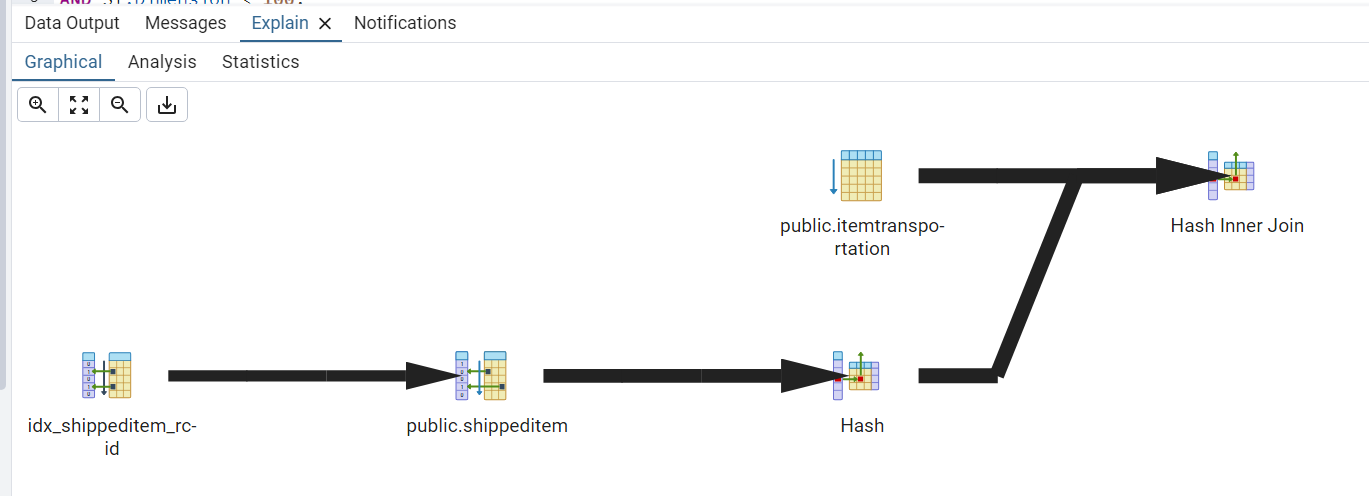
EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

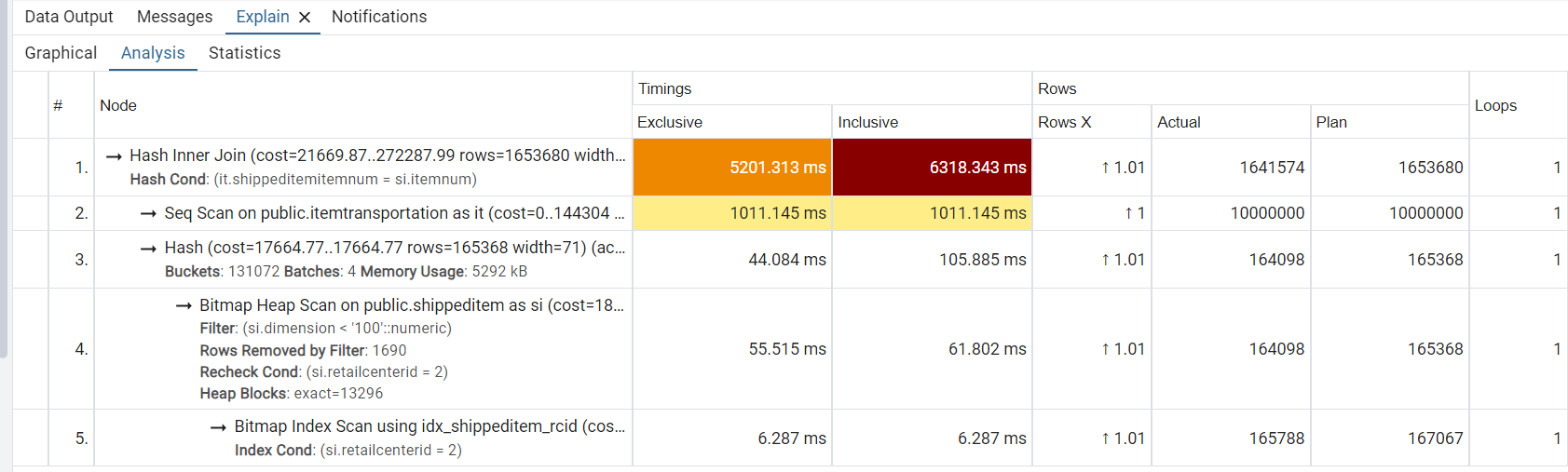
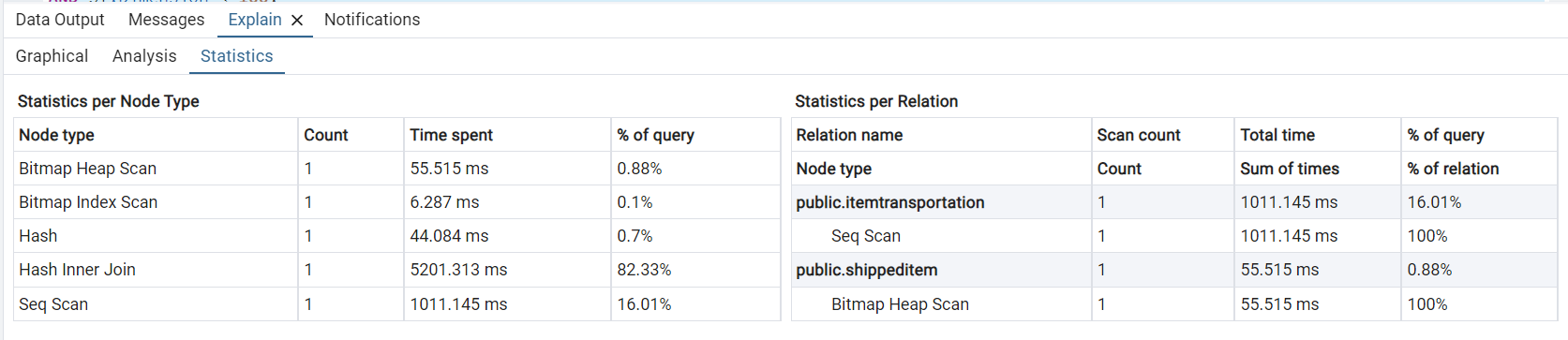
SELECT \* FROM ShippedItem SI

JOIN ItemTransportation IT ON SI.ItemNum = IT.ShippedItemItemNum

WHERE SI.RetailCenterID = 2

AND SI.Dimension < 100;



[

{

"Plan": {

"Node Type": "Hash Join",

"Parallel Aware": false,

"Async Capable": false,

"Join Type": "Inner",

"Startup Cost": 21669.87,

"Total Cost": 272287.99,

"Plan Rows": 1653680,

"Plan Width": 79,

"Actual Startup Time": 107.654,

"Actual Total Time": 6318.343,

"Actual Rows": 1641574,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized", "it.transportationeventseqnumber", "it.shippeditemitemnum"],

"Inner Unique": true,

"Hash Cond": "(it.shippeditemitemnum = si.itemnum)",

"Plans": [

{

"Node Type": "Seq Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "itemtransportation",

"Schema": "public",

"Alias": "it",

"Startup Cost": 0.00,

"Total Cost": 144304.00,

"Plan Rows": 10000000,

"Plan Width": 8,

"Actual Startup Time": 0.057,

"Actual Total Time": 1011.145,

"Actual Rows": 10000000,

"Actual Loops": 1,

"Output": ["it.transportationeventseqnumber", "it.shippeditemitemnum"]

},

{

"Node Type": "Hash",

"Parent Relationship": "Inner",

"Parallel Aware": false,

"Async Capable": false,

"Startup Cost": 17664.77,

"Total Cost": 17664.77,

"Plan Rows": 165368,

"Plan Width": 71,

"Actual Startup Time": 105.882,

"Actual Total Time": 105.885,

"Actual Rows": 164098,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized"],

"Hash Buckets": 131072,

"Original Hash Buckets": 131072,

"Hash Batches": 4,

"Original Hash Batches": 4,

"Peak Memory Usage": 5292,

"Plans": [

{

"Node Type": "Bitmap Heap Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "si",

"Startup Cost": 1862.77,

"Total Cost": 17664.77,

"Plan Rows": 165368,

"Plan Width": 71,

"Actual Startup Time": 8.055,

"Actual Total Time": 61.802,

"Actual Rows": 164098,

"Actual Loops": 1,

"Output": ["si.itemnum", "si.retailcenterid", "si.dimension", "si.insuranceamt", "si.destination", "si.finaldeliverydate", "si.vectorized"],

"Recheck Cond": "(si.retailcenterid = 2)",

"Rows Removed by Index Recheck": 0,

"Filter": "(si.dimension < '100'::numeric)",

"Rows Removed by Filter": 1690,

"Exact Heap Blocks": 13296,

"Lossy Heap Blocks": 0,

"Plans": [

{

"Node Type": "Bitmap Index Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Index Name": "idx\_shippeditem\_rcid",

"Startup Cost": 0.00,

"Total Cost": 1821.43,

"Plan Rows": 167067,

"Plan Width": 0,

"Actual Startup Time": 6.287,

"Actual Total Time": 6.287,

"Actual Rows": 165788,

"Actual Loops": 1,

"Index Cond": "(si.retailcenterid = 2)"

}

]

}

]

}

]

},

"Planning Time": 1.103,

"Triggers": [

],

"Execution Time": 6369.613

}

]

Полнотекстовый поиск:

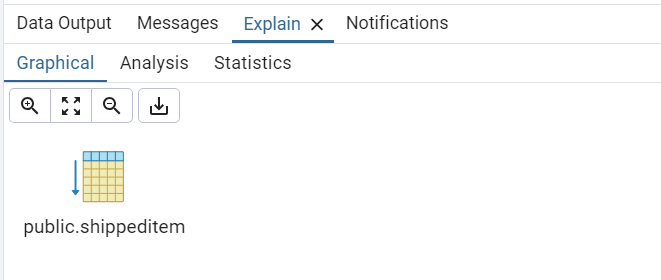
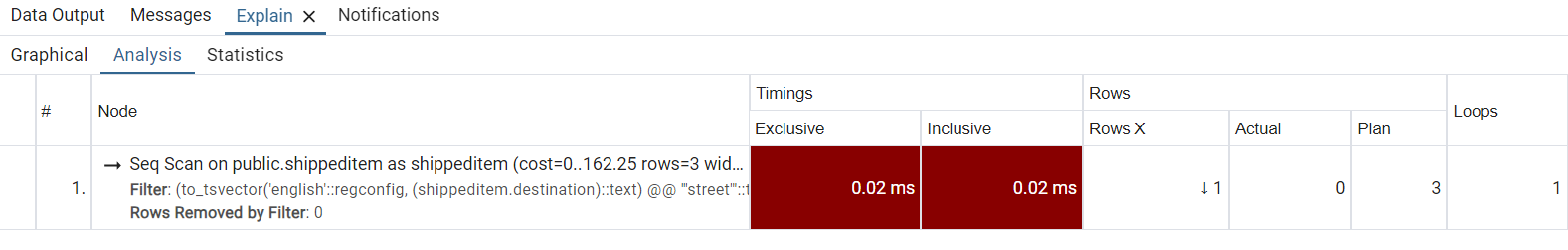
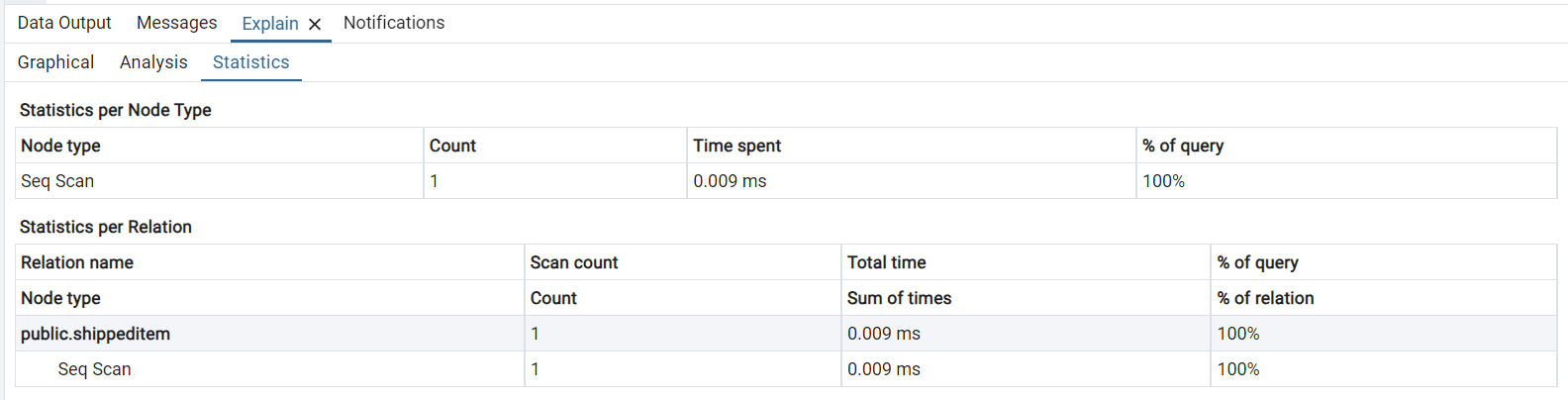
SET enable\_indexscan = OFF;

SET enable\_bitmapscan = OFF;

EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

SELECT \* FROM ShippedItem

WHERE to\_tsvector('english', Destination) @@ to\_tsquery('english', 'Street1');

[

{

"Plan": {

"Node Type": "Seq Scan",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "shippeditem",

"Startup Cost": 0.00,

"Total Cost": 162.25,

"Plan Rows": 3,

"Plan Width": 112,

"Actual Startup Time": 0.008,

"Actual Total Time": 0.009,

"Actual Rows": 0,

"Actual Loops": 1,

"Output": ["itemnum", "retailcenterid", "dimension", "insuranceamt", "destination", "finaldeliverydate"],

"Filter": "(to\_tsvector('english'::regconfig, (shippeditem.destination)::text) @@ '''street'''::tsquery)",

"Rows Removed by Filter": 0

},

"Planning Time": 0.150,

"Triggers": [

],

"Execution Time": 0.023

}

]

Полнотекстовый поиск с индексом:

CREATE INDEX idx\_destination\_text\_search ON ShippedItem USING GIN (to\_tsvector('english', Destination));

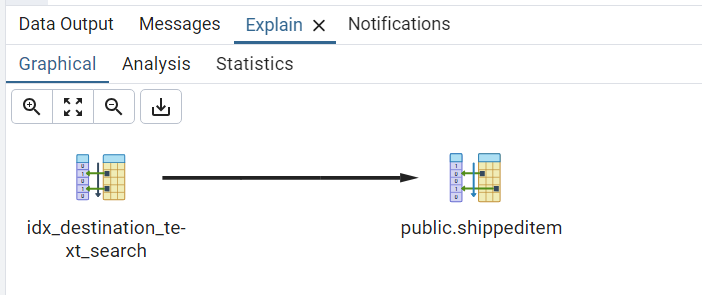
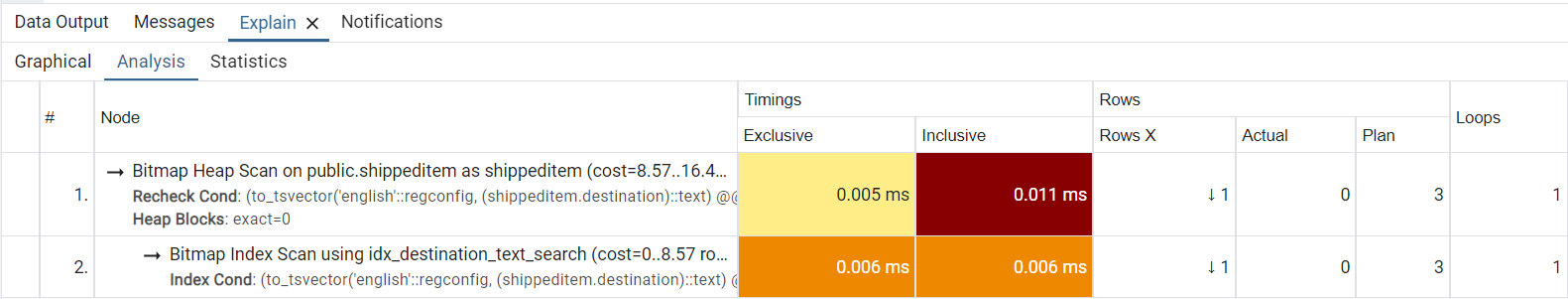
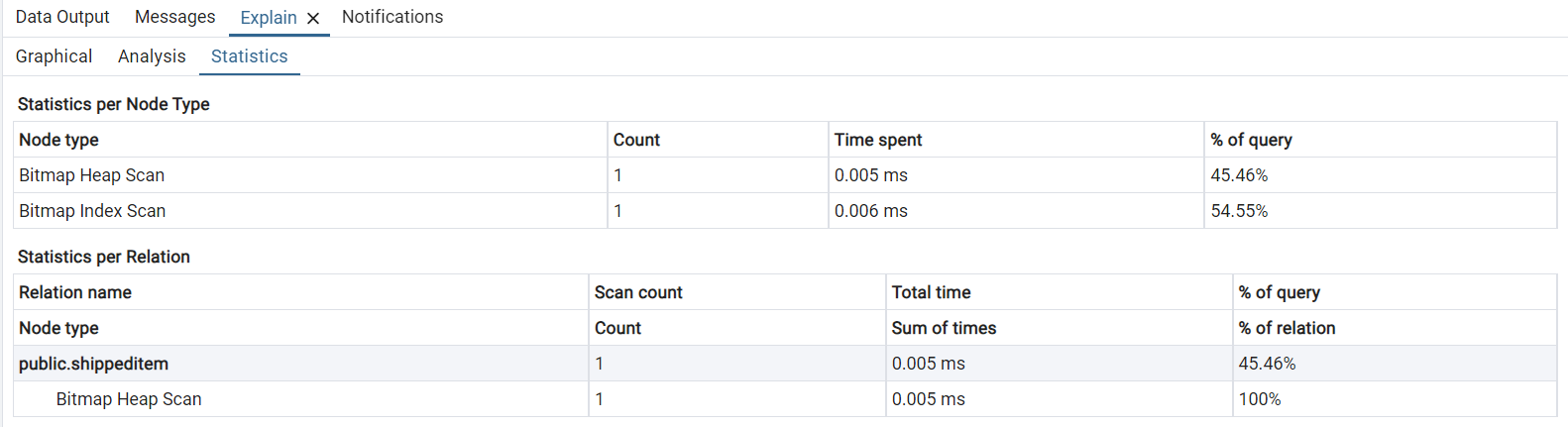
SET enable\_indexscan = ON;

SET enable\_bitmapscan = ON;

EXPLAIN (ANALYZE, VERBOSE, COSTS, TIMING, FORMAT JSON)

SELECT \* FROM ShippedItem

WHERE to\_tsvector('english', Destination) @@ to\_tsquery('english', 'S');

[

{

"Plan": {

"Node Type": "Bitmap Heap Scan",

"Parallel Aware": false,

"Async Capable": false,

"Relation Name": "shippeditem",

"Schema": "public",

"Alias": "shippeditem",

"Startup Cost": 8.57,

"Total Cost": 16.43,

"Plan Rows": 3,

"Plan Width": 112,

"Actual Startup Time": 0.010,

"Actual Total Time": 0.011,

"Actual Rows": 0,

"Actual Loops": 1,

"Output": ["itemnum", "retailcenterid", "dimension", "insuranceamt", "destination", "finaldeliverydate"],

"Recheck Cond": "(to\_tsvector('english'::regconfig, (shippeditem.destination)::text) @@ '''street'''::tsquery)",

"Rows Removed by Index Recheck": 0,

"Exact Heap Blocks": 0,

"Lossy Heap Blocks": 0,

"Plans": [

{

"Node Type": "Bitmap Index Scan",

"Parent Relationship": "Outer",

"Parallel Aware": false,

"Async Capable": false,

"Index Name": "idx\_destination\_text\_search",

"Startup Cost": 0.00,

"Total Cost": 8.57,

"Plan Rows": 3,

"Plan Width": 0,

"Actual Startup Time": 0.006,

"Actual Total Time": 0.006,

"Actual Rows": 0,

"Actual Loops": 1,

"Index Cond": "(to\_tsvector('english'::regconfig, (shippeditem.destination)::text) @@ '''street'''::tsquery)"

}

]

},

"Planning Time": 0.133,

"Triggers": [

],

"Execution Time": 0.032

}

]

Вывод

В ходе лабораторной работы был изучен язык ***plpython3u*** для генерации данных таблицы, были проведены оценки скорости различных запросов с использованием индексов и без, а также познакомились с полнотекстовыми запросами.