ГУАП

КАФЕДРА № 53

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, канд. техн. наук |  |  |  | А. В. Бржезовский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| индексация данных |
| по курсу: методы и средства проектирования информационных систем и технологий |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 3641 |  |  |  | М.Б.Фомин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

*Задание:*

Произвести генерацию и вставку тестовых данных в БД, выполнить запросы из ЛР 3..5 или аналогичные им, зафиксировать планы и время выполнения запросов, создать систему индексов для ускорения выполнения запросов, повторно выполнить запросы, зафиксировать планы и время выполнения.

*Выполнение работы:*

Для генерации и вставки случайных тестовых данных использовалась функция генерации данных DBeaver EE. В таблицу apartments, house, district, street, city, client было вставлено 1000000 строк.

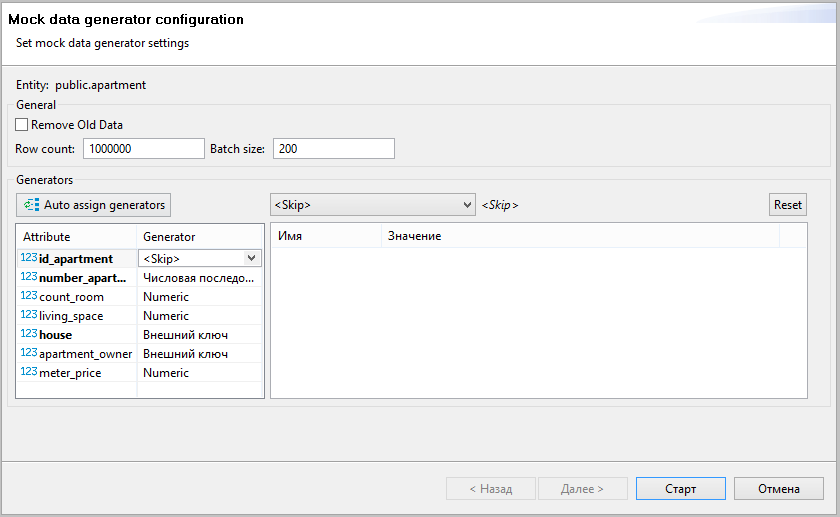


Рисунок 1 – Генератор для заполнения таблиц

Далее были рассмотрены запросы из предыдущих работ с индексами и без, а также их планы. После выполнения каждого запроса с индексом и без выполняется очистка кэша для чистоты показаний.

**Запрос A**) перечень однокомнатных квартир, продаваемых в Московском районе;

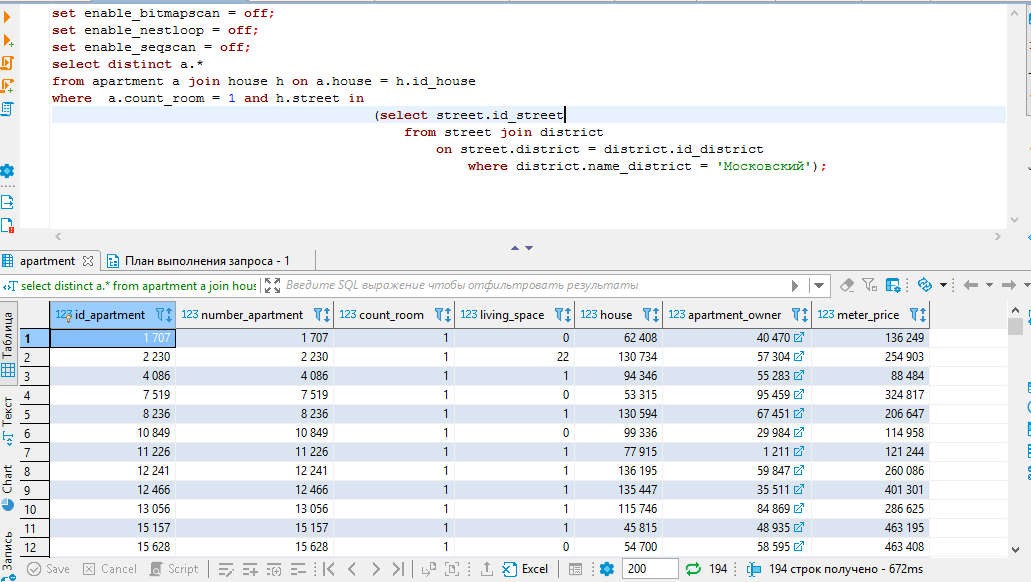


Рисунок 2 – Результат работы запроса А (без индекса)

Запрос представляет собой выполнение запроса без индексов. Время выполнения определяется путем вычисления разницы во времени до выполнения запроса и после. В примере видно, что на выполнение запроса было затрачено 672 милисекунд.

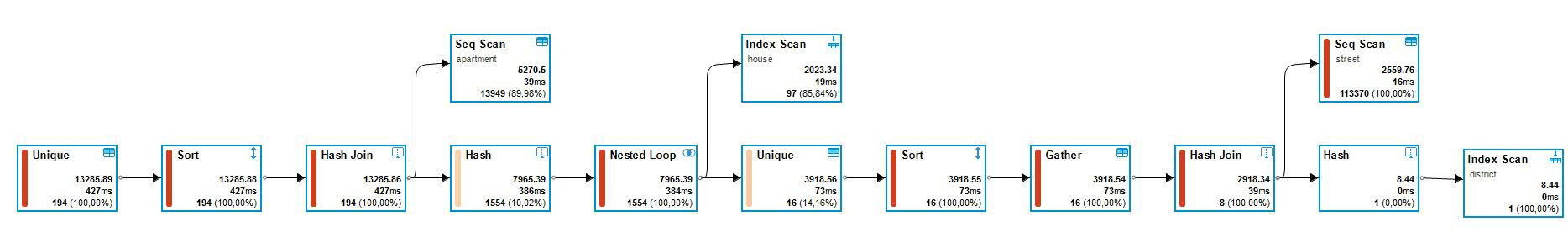


Рисунок 3 – План выполнения запроса а (без индекса)

Далее с помощью запроса создаются индексы в соответствующие таблицы. В таблицах были проиндексированы поля, которые содержат внешние ключи и которые наиболее часто применяется для отбора данных в запросах. Время выполнения 0,484 секунды.

**CREATE** **INDEX** apartment\_count\_room\_idx **ON** public.apartment **USING** btree (count\_room);

**CREATE** **UNIQUE** **INDEX** apartment\_id\_apartment\_idx **ON** public.apartment **USING** btree (id\_apartment);

**CREATE** **UNIQUE** **INDEX** apartment\_number\_apartment\_idx **ON** public.apartment **USING** btree (number\_apartment, house);

**CREATE** **UNIQUE** **INDEX** house\_id\_house\_idx **ON** public.house **USING** btree (id\_house);

**CREATE** **UNIQUE** **INDEX** house\_number\_house\_idx **ON** public.house **USING** btree (number\_house, street);

**CREATE** **UNIQUE** **INDEX** street\_id\_street\_idx **ON** public.street **USING** btree (id\_street);

**CREATE** **UNIQUE** **INDEX** street\_name\_street\_idx **ON** public.street **USING** btree (name\_street, district);

**CREATE** **INDEX** district\_id\_district\_idx **ON** public.district **USING** btree (id\_district);

**CREATE** **INDEX** district\_name\_district\_idx **ON** public.district **USING** btree (name\_district);

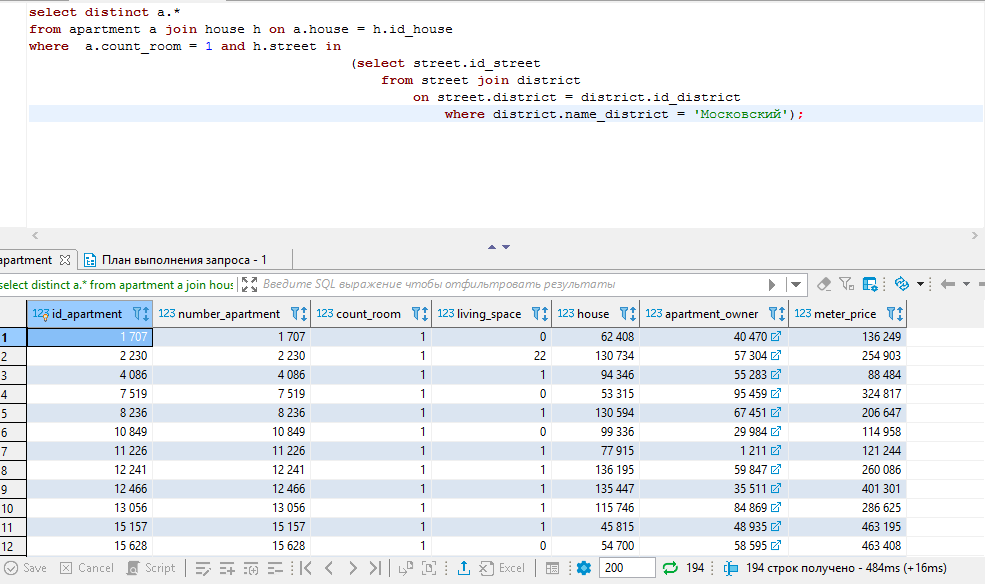


Рисунок 4 – Результат работы запроса а (с индексом)

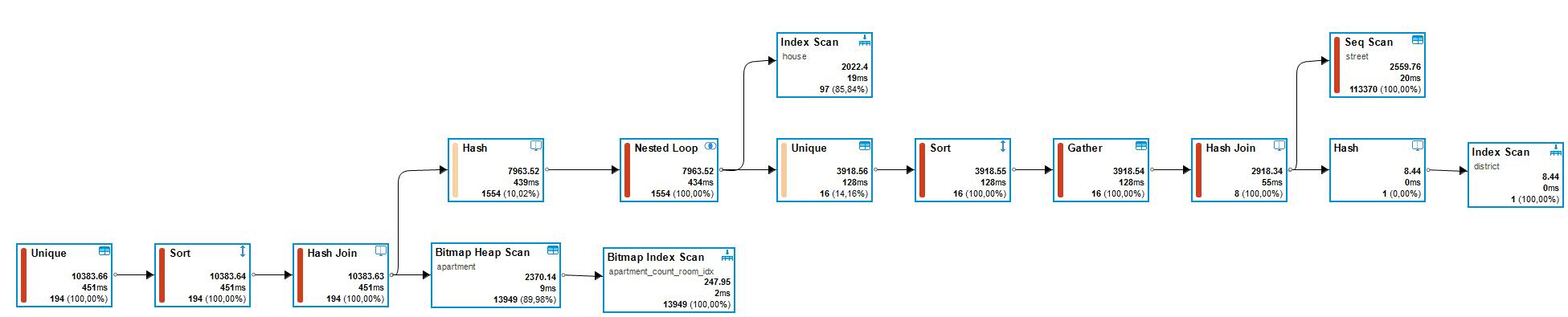


Рисунок 5 – План выполнения запроса а (с индексом)

**Запрос Г**) средняя цена однокомнатной квартиры в городе;

Запрос представляет собой выполнение запроса без индексов. В примере видно, что на выполнение запроса было затрачено 0,234 секунды.

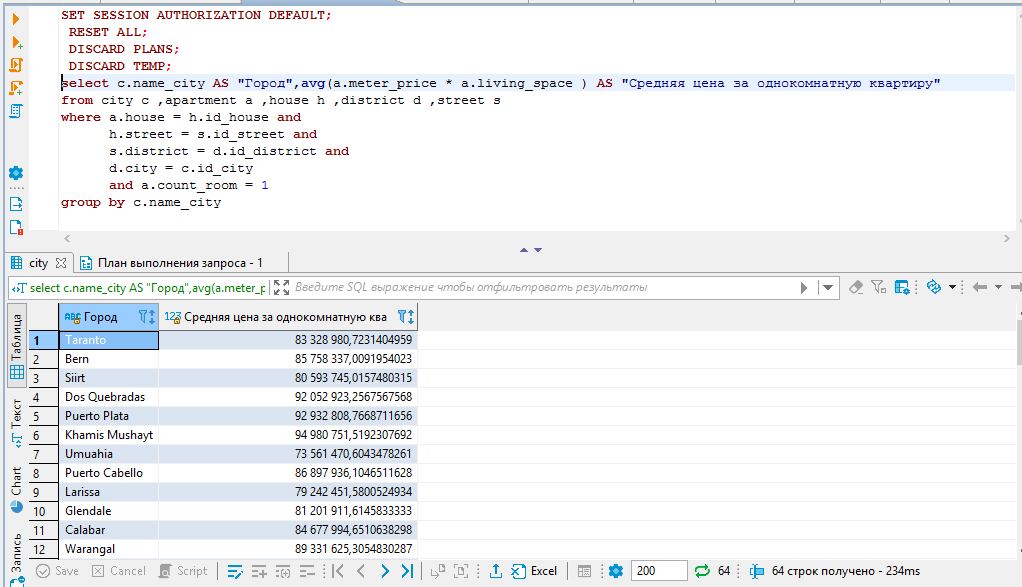


Рисунок 6 – Результат работы запроса г (без индекса)

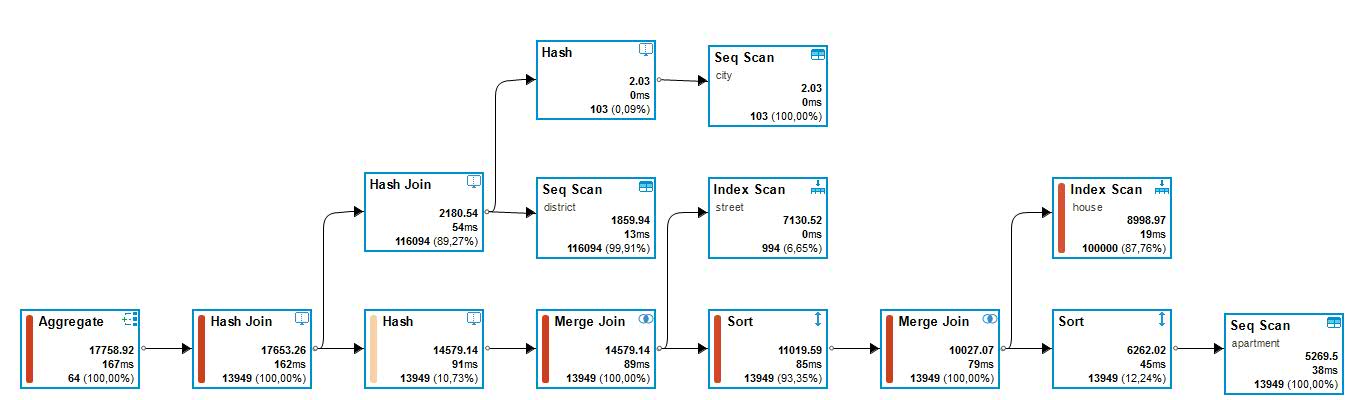


Рисунок 7 – План выполнения запроса б (без индекса)

Далее с помощью запроса создаются индексы в соответствующие таблицы. В таблицах были проиндексированы поля, которые содержат внешние ключи и которые наиболее часто применяется для отбора данных в запросах. Время выполнения 0,156 секунды.

**CREATE** **INDEX** apartment\_count\_room\_idx **ON** public.apartment **USING** btree (count\_room);

**CREATE** **UNIQUE** **INDEX** apartment\_id\_apartment\_idx **ON** public.apartment **USING** btree (id\_apartment);

**CREATE** **UNIQUE** **INDEX** apartment\_number\_apartment\_idx **ON** public.apartment **USING** btree (number\_apartment, house);

**CREATE** **UNIQUE** **INDEX** house\_id\_house\_idx **ON** public.house **USING** btree (id\_house);

**CREATE** **UNIQUE** **INDEX** house\_number\_house\_idx **ON** public.house **USING** btree (number\_house, street);

**CREATE** **UNIQUE** **INDEX** street\_id\_street\_idx **ON** public.street **USING** btree (id\_street);

**CREATE** **UNIQUE** **INDEX** street\_name\_street\_idx **ON** public.street **USING** btree (name\_street, district);

**CREATE** **INDEX** district\_id\_district\_idx **ON** public.district **USING** btree (id\_district);

**CREATE** **INDEX** district\_name\_district\_idx **ON** public.district **USING** btree (name\_district);

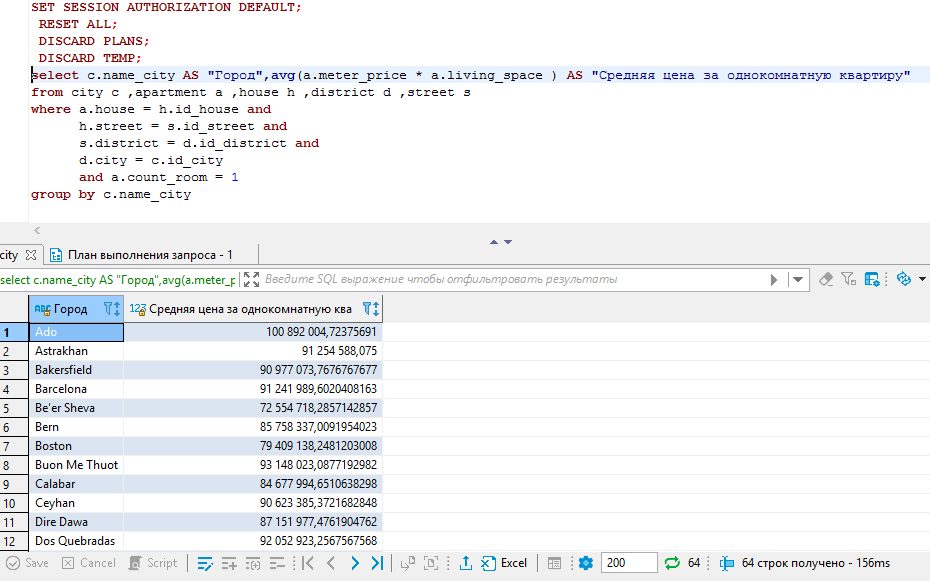


Рисунок 8 – Результат работы запроса г (с индексом)

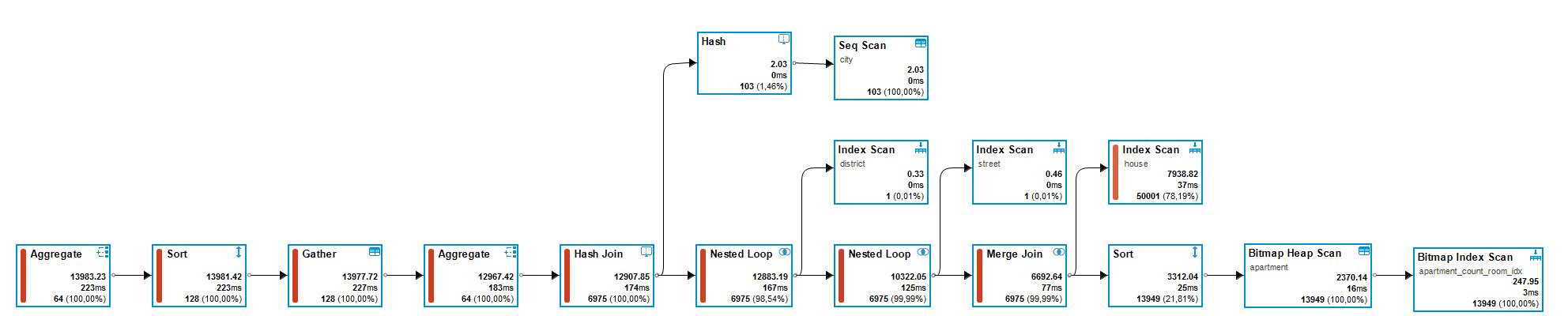


Рисунок 9 – План выполнения запроса г (с индексом)

**Запрос З**) районы, в которых не продаются однокомнатные квартиры

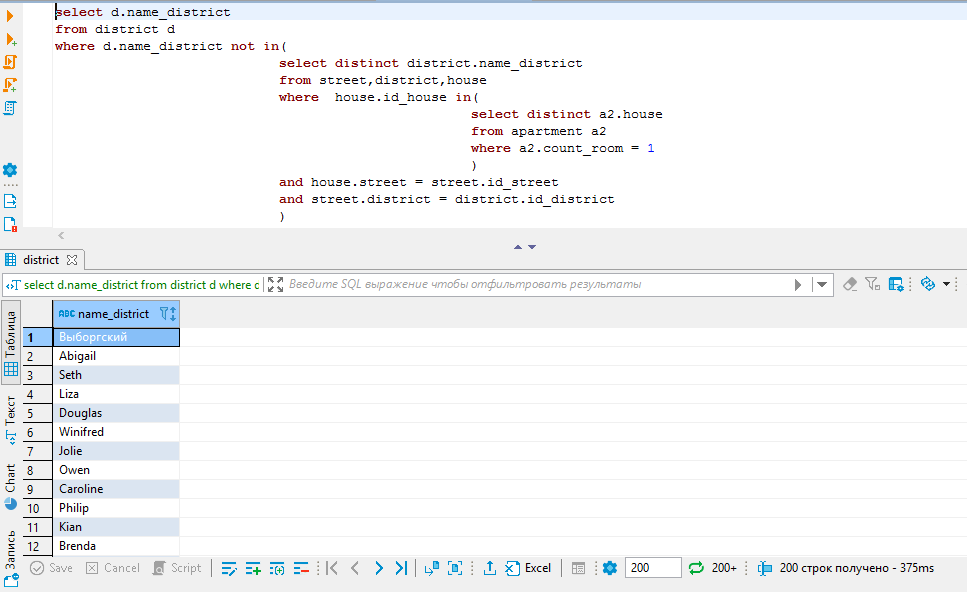


Рисунок 10 – Результат работы запроса в (без индекса)

Запрос представляет собой выполнение запроса без индексов. В примере видно, что на выполнение запроса было затрачено 0,375 секунды.

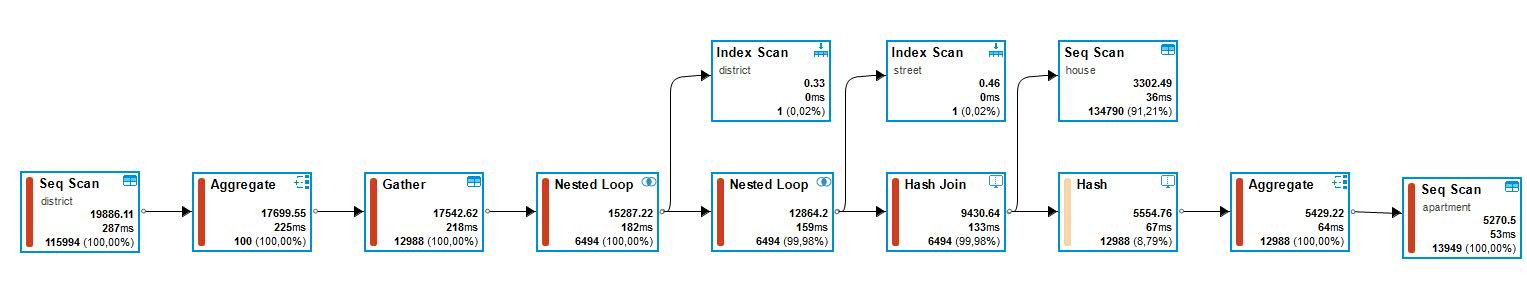


Рисунок 11 – План выполнения запроса в (без индекса)

Далее с помощью запроса создаются индексы в соответствующие таблицы. В таблицах были проиндексированы поля, которые содержат внешние ключи и которые наиболее часто применяется для отбора данных в запросах. Время выполнения 0,187 секунд.

**CREATE** **INDEX** apartment\_count\_room\_idx **ON** public.apartment **USING** btree (count\_room);

**CREATE** **UNIQUE** **INDEX** apartment\_id\_apartment\_idx **ON** public.apartment **USING** btree (id\_apartment);

**CREATE** **UNIQUE** **INDEX** apartment\_number\_apartment\_idx **ON** public.apartment **USING** btree (number\_apartment, house);

**CREATE** **UNIQUE** **INDEX** house\_id\_house\_idx **ON** public.house **USING** btree (id\_house);

**CREATE** **UNIQUE** **INDEX** house\_number\_house\_idx **ON** public.house **USING** btree (number\_house, street);

**CREATE** **UNIQUE** **INDEX** street\_id\_street\_idx **ON** public.street **USING** btree (id\_street);

**CREATE** **UNIQUE** **INDEX** street\_name\_street\_idx **ON** public.street **USING** btree (name\_street, district);

**CREATE** **INDEX** district\_id\_district\_idx **ON** public.district **USING** btree (id\_district);

**CREATE** **INDEX** district\_name\_district\_idx **ON** public.district **USING** btree (name\_district);

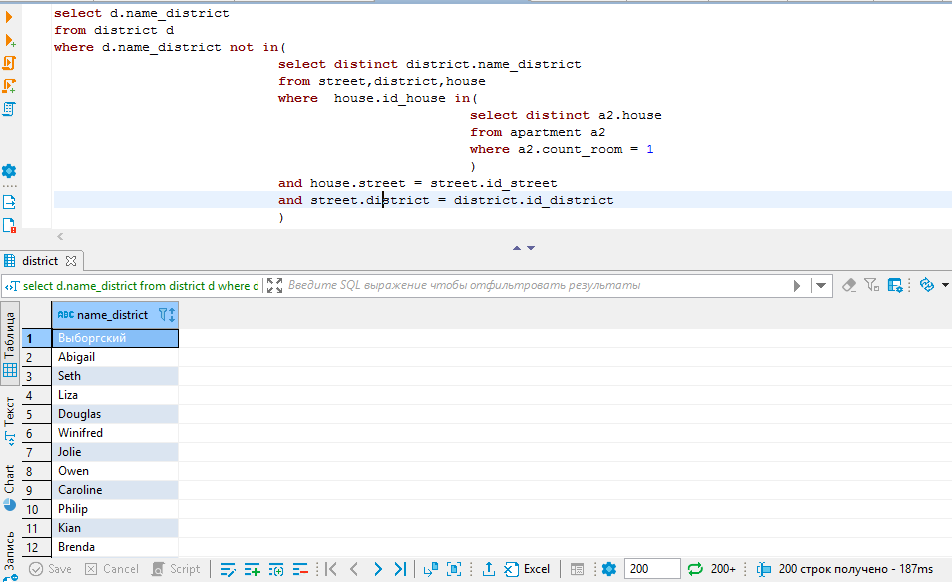


Рисунок 12 – Результат работы запроса в (с индексом)

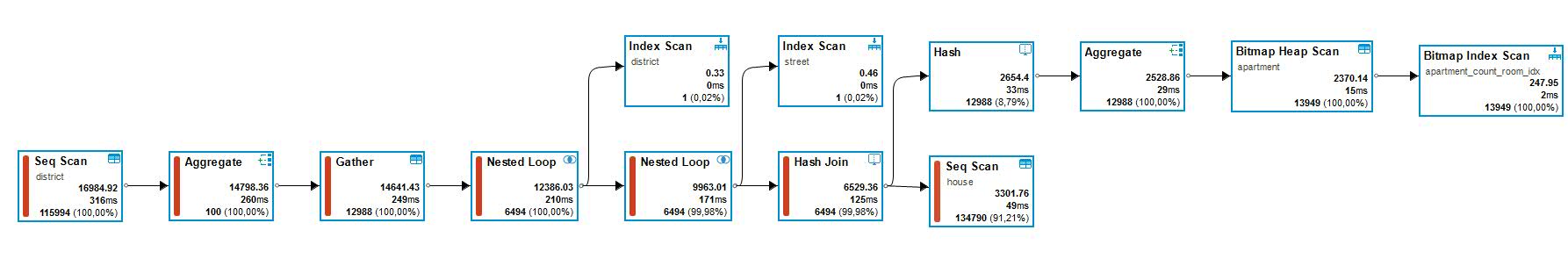


Рисунок 13 – План выполнения запроса в (с индексом)

**Альтернатива(HINTS)**

В postgresql можно прямо «запрещать» конкретные операции с помощью переменных сессии. Удобно то, что их не надо менять в конфиге и перезагружать БД, их значение меняется только в текущей открытой сессии и не влияет на остальные сессии, так что можно экспериментировать прямо на реальных данных. Вот их список со значениями по умолчанию. Почти все операции включены:

| **Используемые операции** | **Значение по умолчанию** |
| --- | --- |
| enable\_bitmapscan enable\_hashagg enable\_hashjoin enable\_indexscan enable\_indexonlyscan enable\_material enable\_mergejoin  enable\_parallel\_append enable\_seqscan enable\_sort enable\_tidscan enable\_parallel\_hash enable\_partition\_pruning enable\_nestloop | on |
| enable\_partitionwise\_join enable\_partitionwise\_aggregate | off |

Запрещая или разрешая отдельные операции, мы заставляем планировщик выбирать другие планы, которые мы можем увидеть все той же командой EXPLAIN. На самом деле, «запрет» операций не запрещает их использование, а просто сильно увеличивает их затратность. В PostgreSQL каждой «запрещенной» операции автоматически «накидывается» затратность равная 10 миллиардам условных единиц. При этом в EXPLAIN суммарные веса плана могут получиться запредельно высокими, но на фоне этих десятков миллиардов вес остальных операций хорошо просматривается, так как он обычно укладывается в меньшие порядки.

Рассмотрим используемый ранее запрос А:

**План 1.** Со всеми разрешенными операциями суммарная затратность наиболее оптимального плана 0,484 с

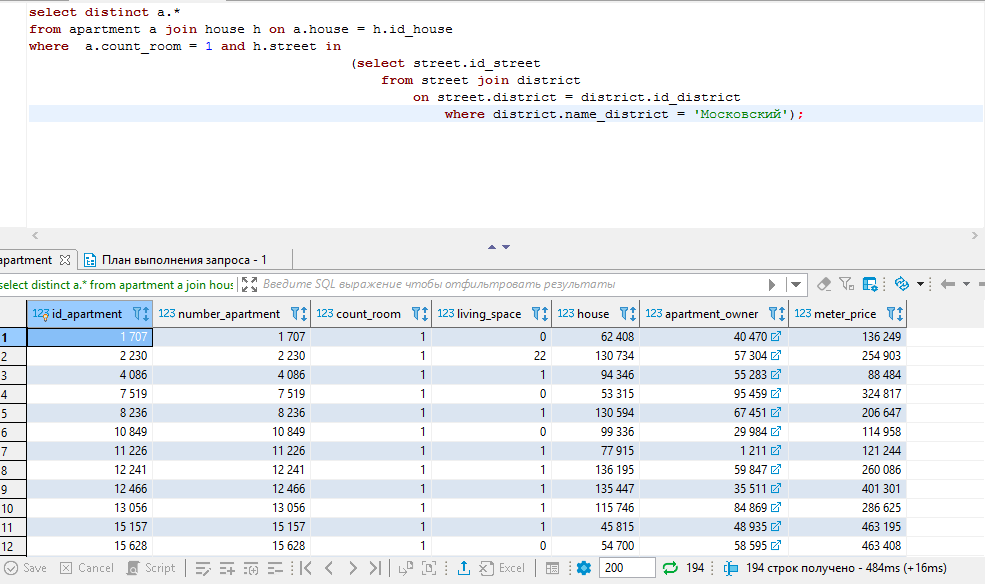


Рисунок 4 – Результат работы запроса а

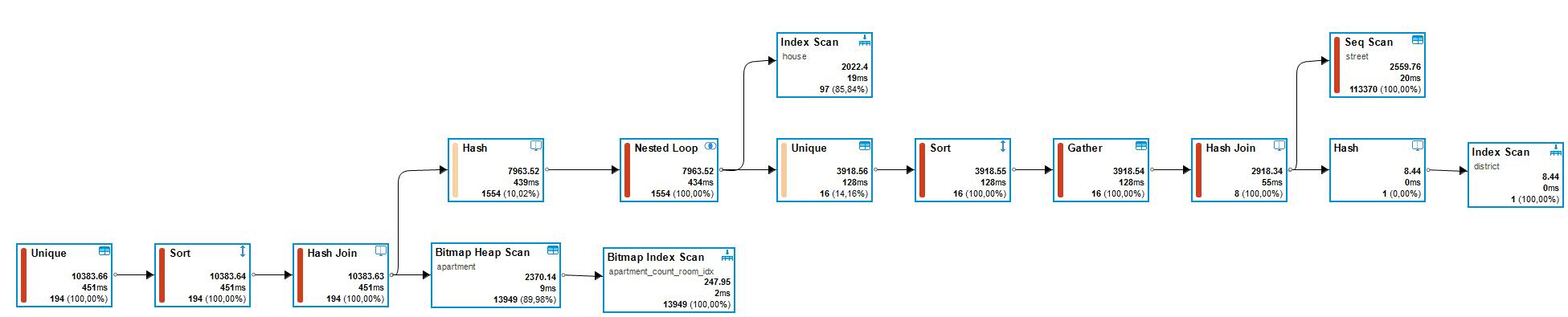


Рисунок 5 – План выполнения запроса а

**План 2**. С «запрещенным» nested loop затратность выросла до 0,641 c. Он, как видно, примерно в 1.5 раза превышает затратность оптимального плана, то есть достаточно близок к нему.

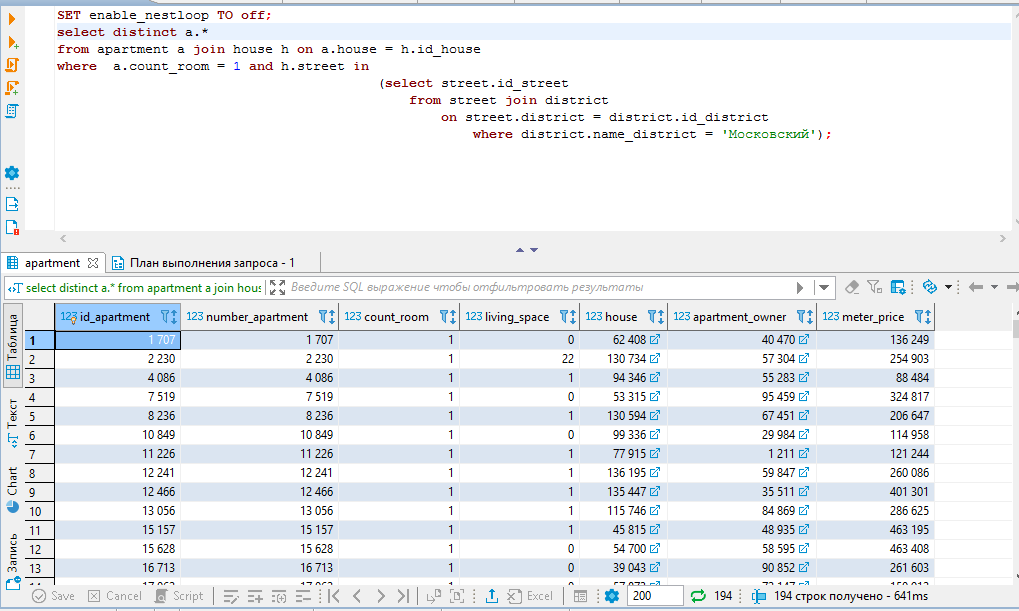


Рисунок 4 – Результат работы запроса а(С «запрещенным» nested loop)

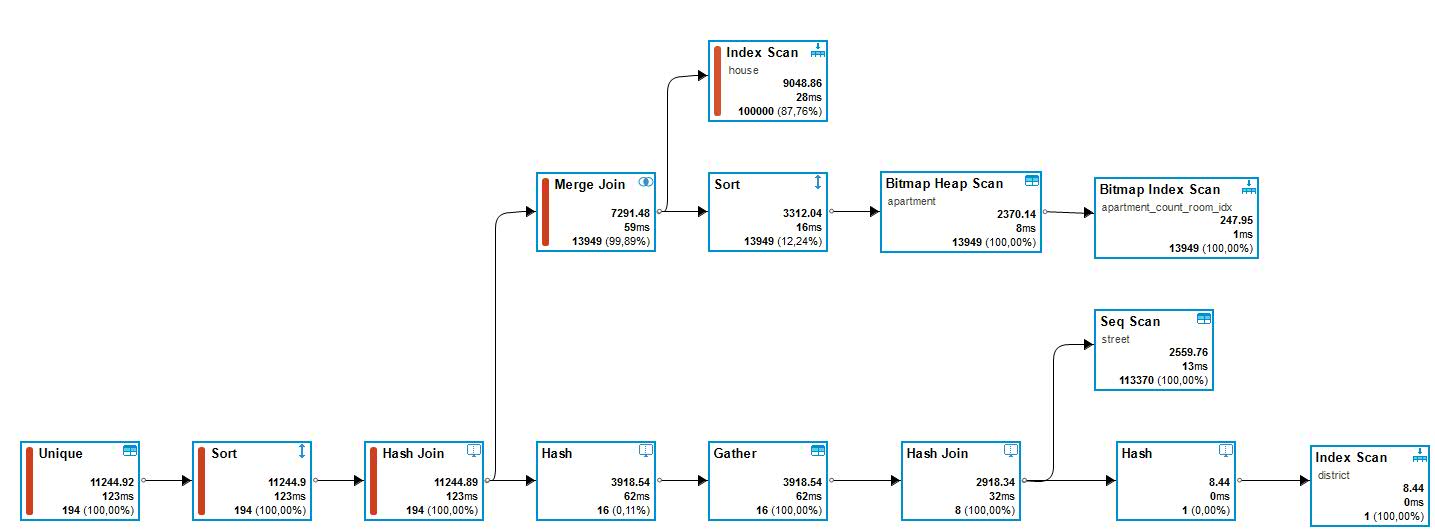


Рисунок 5 – План выполнения запроса а (С «запрещенным» nested loop)

**План 3**. С «запрещенным» Hash Join затратность выросла до 0,516 c. Он, как видно, примерно в 1.3 раза превышает затратность оптимального плана, то есть достаточно близок к нему.

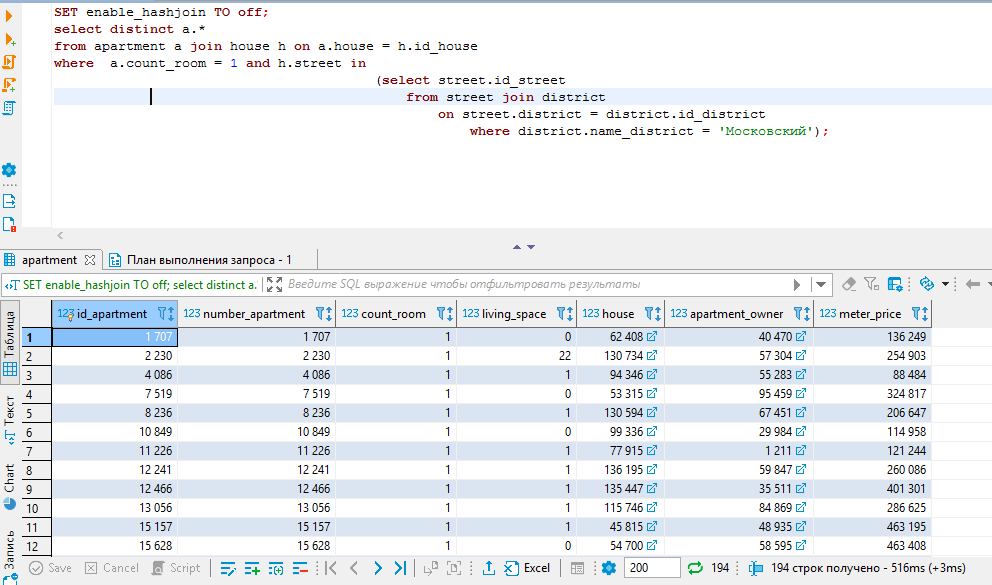


Рисунок 4 – Результат работы запроса а(С «запрещенным» Hash Join)

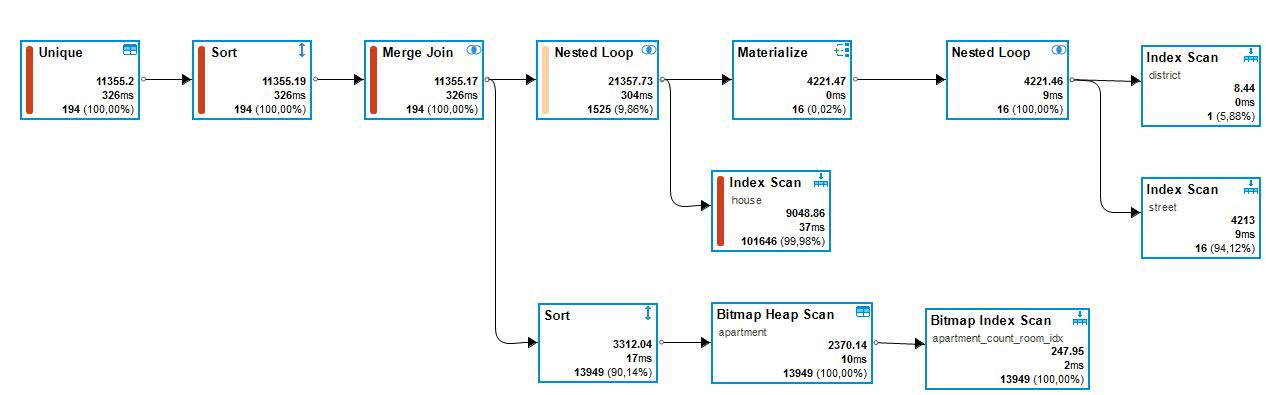


Рисунок 5 – План выполнения запроса а (С «запрещенным» Hash Join)

*Вывод:*

Поскольку оптимизатор запросов postgresql обычно выбирает лучший план выполнения запроса, изменять план рекомендуется c крайней осторожностью, т.к в случае неправильного выбора выполнения запроса результирующая скорость выполнения запросов может быть значительно снижена.