

## Exercise 1

### Task 1 & 2 - предполагаемый план выполнения

SQLQuery2.sql - DE...GIPLN6\boris (59))\* X 51 - Lab Exercise 1...RGIPLN6\boris (56)

```
USE AdventureWorks2017;  
GO  
  
-- Script 7.1  
  
SELECT * FROM dbo.DatabaseLog;  
GO
```

100 %

Сообщения План выполнения

Запрос 1: стоимость запроса (по отношению к пакету): 0%  
USE AdventureWorks2017;

T-SQL

USE DATABASE  
Стоимость: 0 %

Запрос 2: стоимость запроса (по отношению к пакету): 100%  
-- Script 7.1 SELECT \* FROM dbo.DatabaseLog

Просмотр строк таблицы  
[DatabaseLog]  
Стоимость: 100 %

SELECT  
Стоимость: 0 %

Просмотр строк таблицы

Просмотр строк таблицы	
Физическая операция	Просмотр строк таблицы
Логическая операция	Table Scan
Предполагаемый режим выполнения	Row
Хранилище	RowStore
Предполагаемая стоимость операций ввода-вывода	0,581644
Предполагаемая стоимость оператора	0,583556 (100%)
Предполагаемая стоимость ЦП	0,0019126
Предполагаемая стоимость поддержки	0,583556
Предполагаемое количество выполнений	1
Расчетное число строк на выполнение	1596
Приблизительное число считываемых строк	1596
Предполагаемый размер строки	8593 Б
Отсортировано	False
Идентификатор узла	0
Объект	[AdventureWorks2017].[dbo].[DatabaseLog]
Список выходных столбцов	[AdventureWorks2017].[dbo].[DatabaseLog].DatabaseLogID; [AdventureWorks2017].[dbo].[DatabaseLog].PostTime; [AdventureWorks2017].[dbo].[DatabaseLog].DatabaseUser; [AdventureWorks2017].[dbo].[DatabaseLog].Event; [AdventureWorks2017].[dbo].[DatabaseLog].Schema; [AdventureWorks2017].[dbo].[DatabaseLog].Object; [AdventureWorks2017].[dbo].[DatabaseLog].TSQL; [AdventureWorks2017].[dbo].[DatabaseLog].XmlEvent

Запрос успешно выполнен.

## Task 3 - XML план выполнения

```
План выполнения.xml  SQLQuery2.sql - DE...GIPLN6\boris (59))*  51 - Lab Exercise 1...RGIPLN6\boris (56))
<?xml version="1.0" encoding="utf-16"?>
<ShowPlanXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMlSc
  <BatchSequence>
    <Batch>
      <Statements>
        <StmtUseDb StatementCompId="1" StatementId="1" StatementText="USE AdventureWorks2017;*&#xD;&#xA;" S
      </Statements>
    </Batch>
    <Batch>
      <Statements>
        <StmtSimple StatementCompId="1" StatementId="1" StatementText="*&#xD;&#xA;;-- Script 7.2*&#xD;&#xA;SE
      </Statements>
    </Batch>
    <Batch>
      <Statements>
        <StmtSimple StatementCompId="1" StatementEstRows="1596" StatementId="1" StatementOptmLevel="TRIVIA
        <StatementSetOptions ANSI_NULLS="true" ANSI_PADDING="true" ANSI_WARNINGS="true" ARITHABORT="true
        <QueryPlan NonParallelPlanReason="NoParallelPlansInDesktopOrExpressEdition" CachedPlanSize="16"
          <MemoryGrantInfo SerialRequiredMemory="0" SerialDesiredMemory="0" GrantedMemory="0" MaxUsedMem
          <OptimizerHardwareDependentProperties EstimatedAvailableMemoryGrant="154058" EstimatedPagesCac
          <TraceFlags IsCompileTime="true">
            <TraceFlag Value="8017" Scope="Global" />
          </TraceFlags>
          <RelOp AvgRowSize="8593" EstimateCPU="0.0019126" EstimateIO="0.581644" EstimateRebinds="0" Est
          <OutputList>
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
            <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]" Colu
          </OutputList>
          <TableScan Ordered="false" ForcedIndex="false" ForceScan="false" NoExpandHint="false" Storag
          <DefinedValues>
            <DefinedValue>
              <ColumnReference Database="[AdventureWorks2017]" Schema="[dbo]" Table="[DatabaseLog]"
```

Task4 - реальный план выполнения, выводится после выполнения скрипта, на основании данных об использованных ресурсах. Сюда входит Компонент Database Engine — время, затраченное на выполнение, и время ЦП.

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a script named 'SQLQuery2.sql' with the following content:

```
USE AdventureWorks2017;  
GO  
  
-- Script 7.3  
  
SELECT * FROM dbo.DatabaseLog;  
GO
```

The bottom pane shows the execution plan for the query. The 'Results' tab is selected, displaying the following information:

Запрос 1: стоимость запроса (по отношению к пакету): 100%  
SELECT \* FROM dbo.DatabaseLog

The execution plan shows a single step: 'Просмотр строк таблицы [DatabaseLog]'. The cost is 100%, and the execution time is 0.107s. The plan also indicates that 1596 rows were scanned out of 1596 total rows, with a 100% scan percentage.

At the bottom, a status bar indicates: 'Запрос успешно выполнен.'

Task5 - безуспешная попытка создать предполагаемый план т.к. таблица еще не создана

SQLQuery2.sql - DE...GIPLN6\boris (59))\* 51 - Lab Exerc

```
USE AdventureWorks2017;
GO

CREATE TABLE dbo.SomeTable
(
  SomeTableID INT IDENTITY(1, 1) PRIMARY KEY,
  FullName varchar(35)
);
INSERT INTO dbo.SomeTable VALUES('Hello'),
SELECT * FROM dbo.SomeTable;
DROP TABLE dbo.SomeTable;
GO
```

100 %

Сообщения План выполнения

(затронута одна строка)  
Сообщение 208, уровень 16, состояние 1, строка 8  
Недопустимое имя объекта "dbo.SomeTable".

Время выполнения: 2020-06-16T12:15:16.3498134+03:00

Task 6 - реальный план выполнения

100 %

Результаты Сообщения План выполнения

Запрос 1: стоимость запроса (по отношению к пакету): 75%  
INSERT INTO [dbo].[SomeTable] values(@1), (@2)

Операция	Стоимость	Процент
INSERT	0 %	
Clustered Index Insert	100 %	0.056s
Вычисление скалярного значен...	0 %	0.002s
Constant Scan	0 %	0.000s

Запрос 2: стоимость запроса (по отношению к пакету): 25%  
SELECT \* FROM dbo.SomeTable

Операция	Стоимость	Процент
SELECT	0 %	
Clustered Index Scan (Cluste...	100 %	0.000s

## Task 7 - кэш планов

```
SELECT cp.objtype AS PlanType,
       OBJECT_NAME(st.objectid,st.dbid) AS ObjectName,
       cp.refcounts AS ReferenceCounts,
       cp.usecounts AS UseCounts,
       st.text AS SQLBatch,
       qp.query_plan AS QueryPlan
FROM sys.dm_exec_cached_plans AS cp
CROSS APPLY sys.dm_exec_query_plan(cp.plan_handle) AS qp
CROSS APPLY sys.dm_exec_sql_text(cp.plan_handle) AS st;
GO
```

0 %

PlanType	ObjectName	ReferenceCounts	UseCounts	SQLBatch	QueryPlan
Adhoc	NULL	2	1	-- Script 7.5  SELECT cp.objtype AS PlanTyp...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
Adhoc	NULL	2	1	SET STATISTICS XML ON	<ShowPlanXML xmlns="http://schemas.microsoft.com...

Функция `sys.dm_exec_cached_plans` возвращает строку для каждого закэшированного плана запросов. Запросы кэшируются для более быстрого выполнения

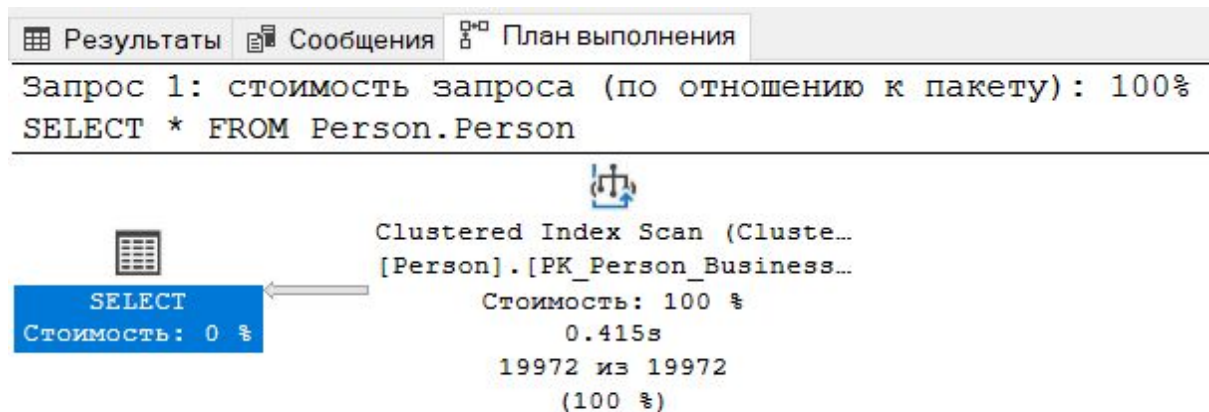
`sys.dm_exec_query_plan` Возвращает события инструкции Showplan в XML-формате для пакета, указанного в дескрипторе плана.

`sys.dm_exec_sql_text` Возвращает текст пакета SQL, идентифицируемого указанным аргументом

## Exercise 2

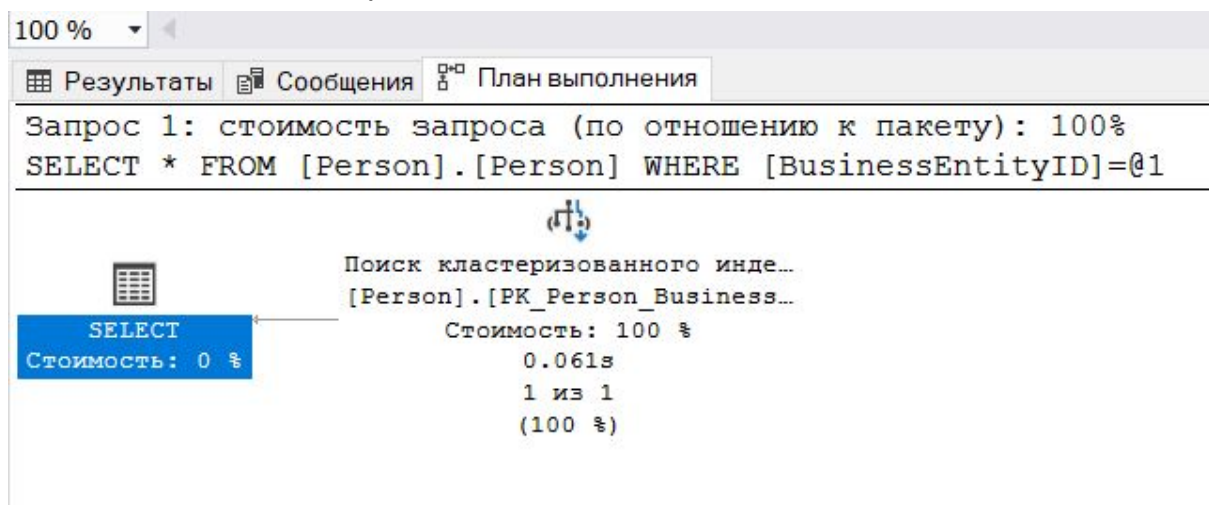
### Task 2

Выполняется просмотр всей таблицы по главному ключу



### Task 3

Выполняется поиск конкретного значения главного ключа

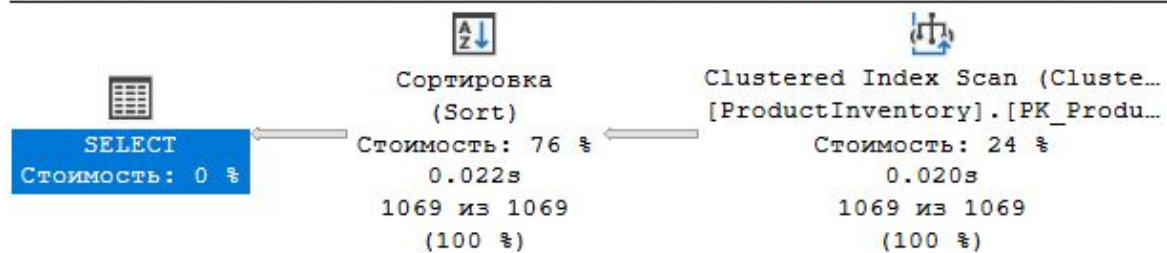




#### Task 4

Выполняется просмотр таблицы по главному ключу, затем полученный результат сортируется

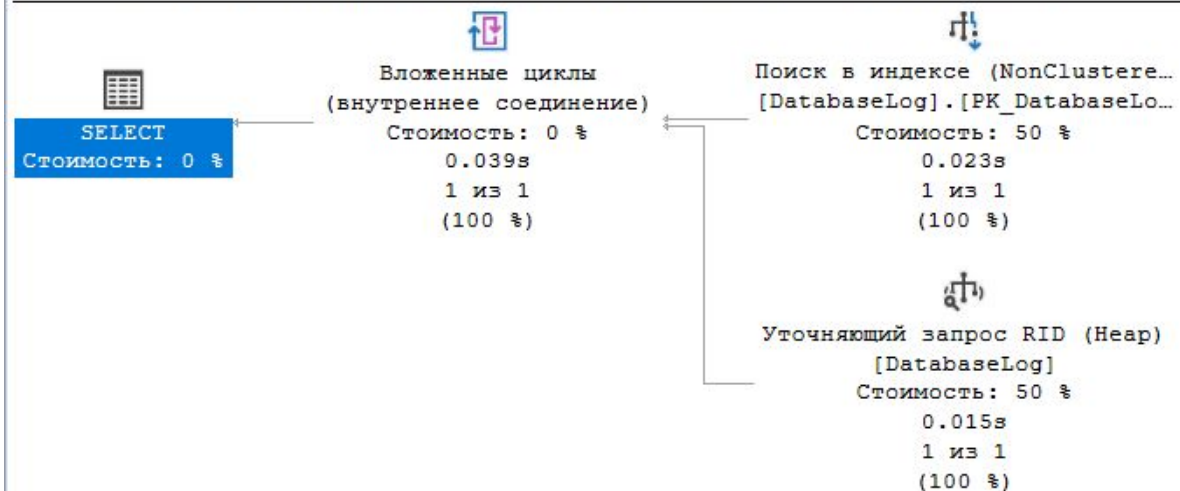
Запрос 1: стоимость запроса (по отношению к пакету): 100%  
 SELECT \* FROM Production.ProductInventory ORDER BY Shelf



#### Task 5

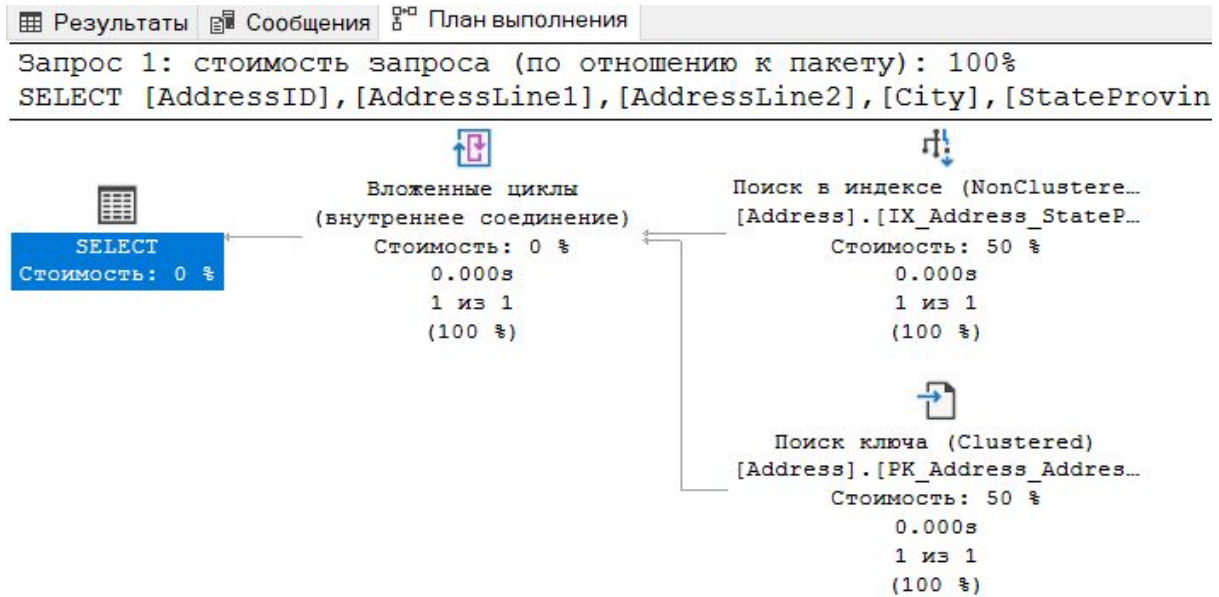
Т.к. индекс является некластеризованным, поиск выполняется с применением уточняющего запроса

Запрос 1: стоимость запроса (по отношению к пакету): 100%  
 SELECT \* FROM [dbo].[DatabaseLog] WHERE [DatabaseLogID]=@1



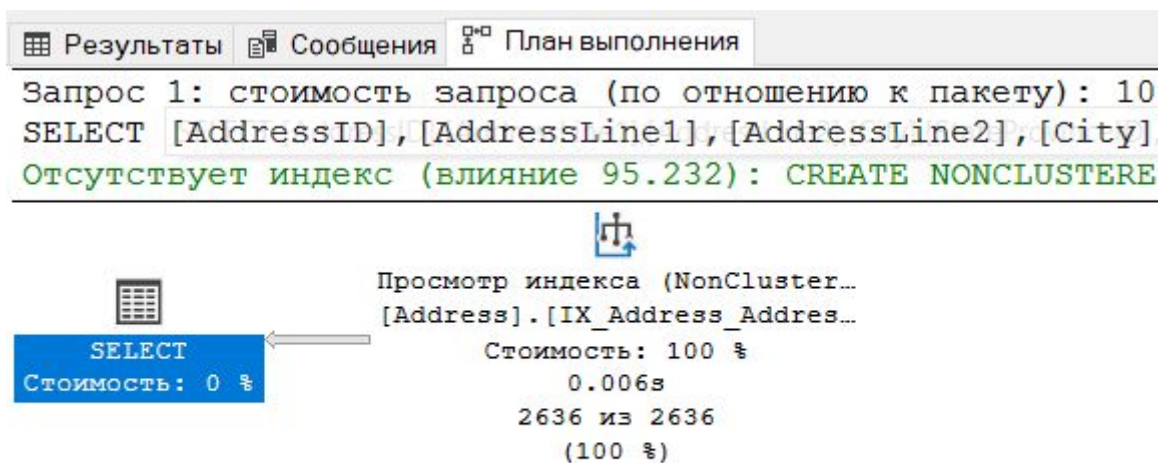
## Task 6

Выполняется поиск диапазона строк некластеризованного индекса, и в нем ищется конкретное значение ключа



## Task 7

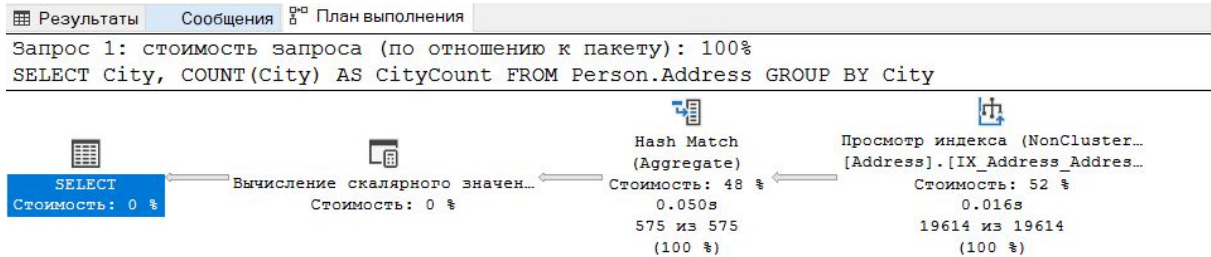
Похоже что здесь выполняется просмотр всей таблицы вместо поиска по индексу. Я обнаружил такой же результат для других часто встречающихся в таблице значений. Мое предположение в том, что т.к некластеризованный индекс хранит в себе указатели на строки, то SQL быстрее просмотреть всю таблицу и выбрать нужные значения без использования индекса





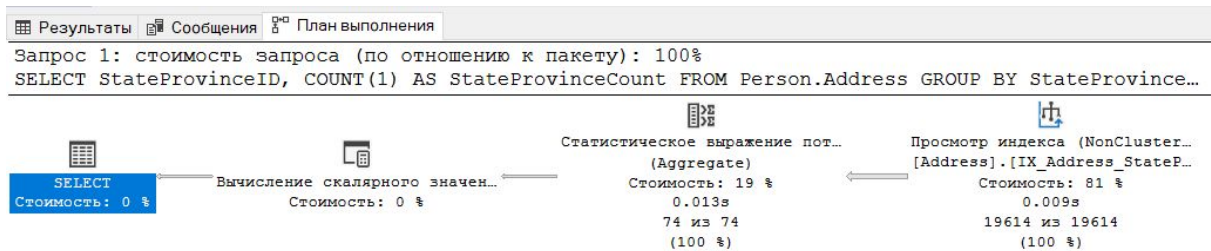
## Task 8

Выполняется просмотр всей таблицы, затем выполняется агрегирующая функция. Т.к. искомая величина (City) не представлена отдельным индексом то в качестве агрегирующей функции применяется Hash match. Функция создает в памяти таблицу хэшей и сравнивает поступающие значения с этой таблицей.



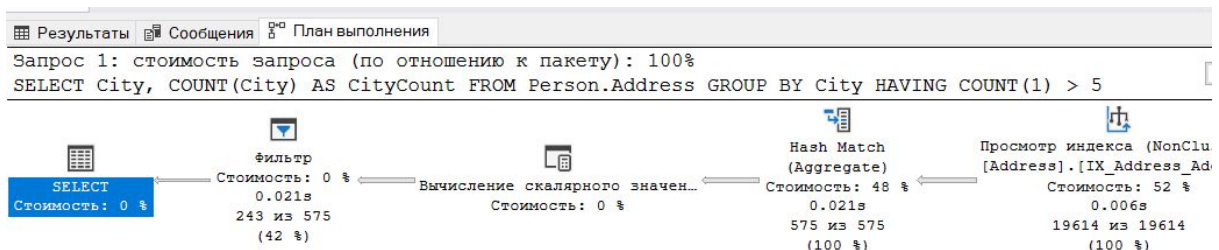
## Task 9

Отличие в том, что теперь таблица просматривается по индексу State\_Province\_ID, что увеличило производительность. Т.к. искомая величина представлена отдельным индексом, то используется другая агрегирующая функция. Статистическое выражение потока просто считает поступающие строки



## Task 10

Разница с заданием 8 состоит только в появлении дополнительного фильтра HAVING. Отличие HAVING от WHERE состоит в том, что HAVING выполняется после GROUP BY а не до, что позволяет ему работать со столбцами используемыми в GROUP BY. Стоимость просмотра таблицы и функции Hash Match в процентом соотношении не изменилась, время выполнения уменьшилось из-за кеширования.



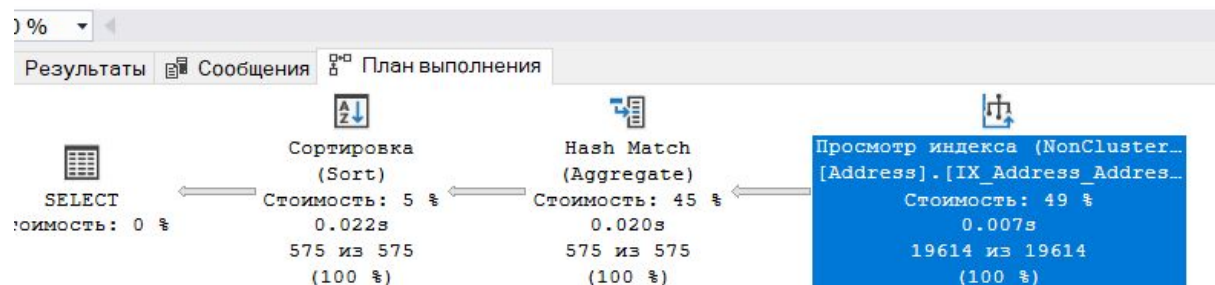
### Exercise 3

В данном примере отличий между двумя запросами нет. Distinct создает таблицу для хранения дубликатов, и выбирает только уникальные значения. Group by обычно используется для других функций (к примеру count()). В данном примере он делает то же самое что и Distinct. Поэтому мы получаем два одинаковых плана выполнения, но второй выполняется быстрее т.к. он закеширован (я менял запросы местами, результат тот же - второй быстрее)

```
USE AdventureWorks2017;  
GO
```

```
-- Script 7.15
```

```
SELECT DISTINCT a.City  
FROM Person.Address AS a  
ORDER BY a.City;  
  
SELECT a.City  
FROM Person.Address AS a  
GROUP BY a.City  
ORDER BY a.City;  
GO
```



Запрос 2: стоимость запроса (по отношению к пакету): 50%

SELECT a.City FROM Person.Address AS a GROUP BY a.City ORDER BY a.City

