

Indeksy - Karta pracy nr 2

Imię i Nazwisko: Oscar Teeninga

Swoje odpowiedzi wpisuj w **czerwone pola**. Preferowane są zrzuty ekranu, **wymagane** komentarze.

Co jest potrzebne?

Do wykonania ćwiczenia potrzebne są:

- **MS SQL Server** wersja co najmniej 2016,
- przykładowa baza danych **AdventureWorks2017**.

Przygotowanie

Stwórz swoją bazę danych o nazwie **XYZ**. Jeśli jednak dzielisz z kimś serwer, to użyj swoich inicjałów:

```
CREATE DATABASE XYZ  
GO
```

```
USE XYZ  
GO
```

Dokumentacja

Obowiązkowo:

- <https://docs.microsoft.com/en-us/sql/relational-databases/indexes/indexes>
- <https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide>
- <https://www.simple-talk.com/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/>

Materiały rozszerzające:

- <https://www.sqlshack.com/sql-server-query-execution-plans-examples-select-statement/>

—

Zadanie 1 – Indeksy klastrowane i nieklastrowane

Celem zadania jest poznanie indeksów **klastrowanych** i **nieklastrowanych**...

Skopiuj tablicę Customer do swojej bazy danych:

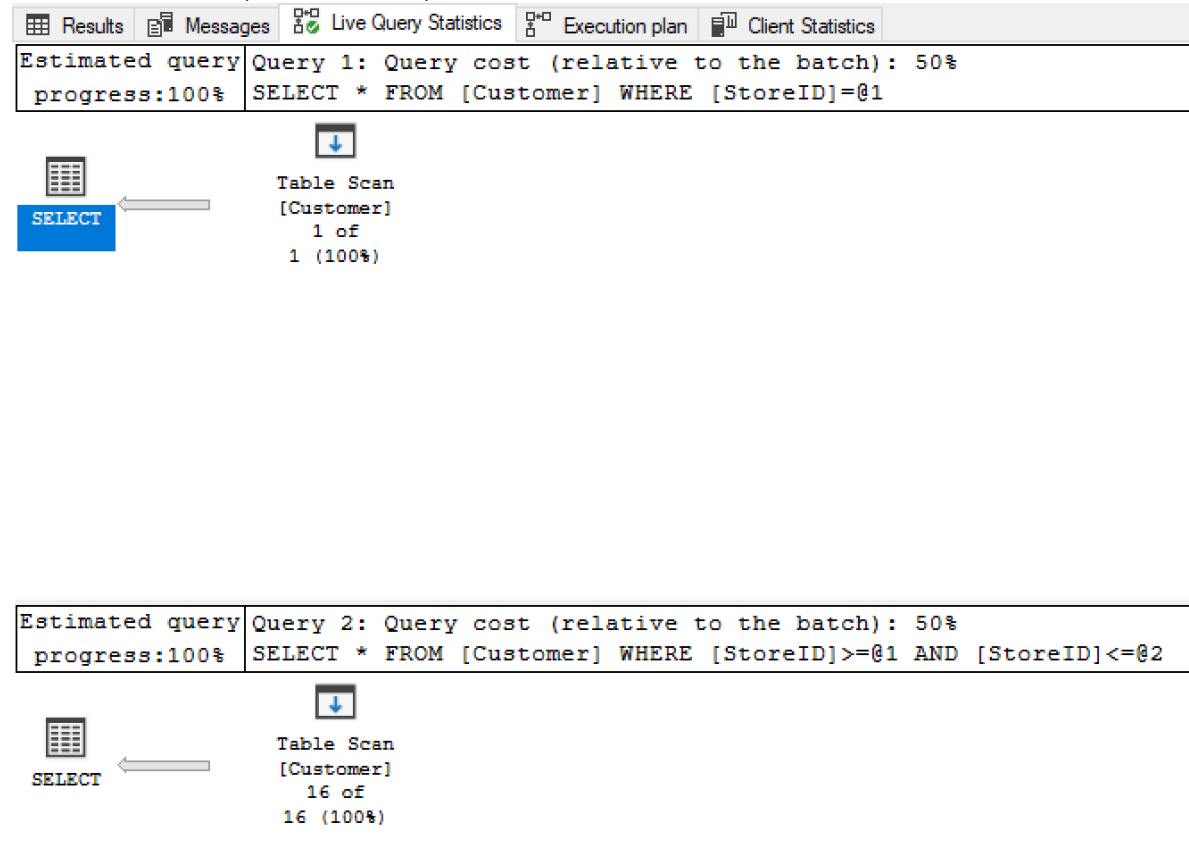
```
SELECT * INTO [Customer] FROM [AdventureWorks2017].[Sales].[Customer]
```

Wykonaj analizy zapytań:

```
SELECT * FROM Customer WHERE StoreID = 594
SELECT * FROM Customer WHERE StoreID BETWEEN 594 AND 610
```

Zanotuj czas zapytania, jego koszt:

Koszt: 50% / 50%, Czas: 80 ms, 101ms.

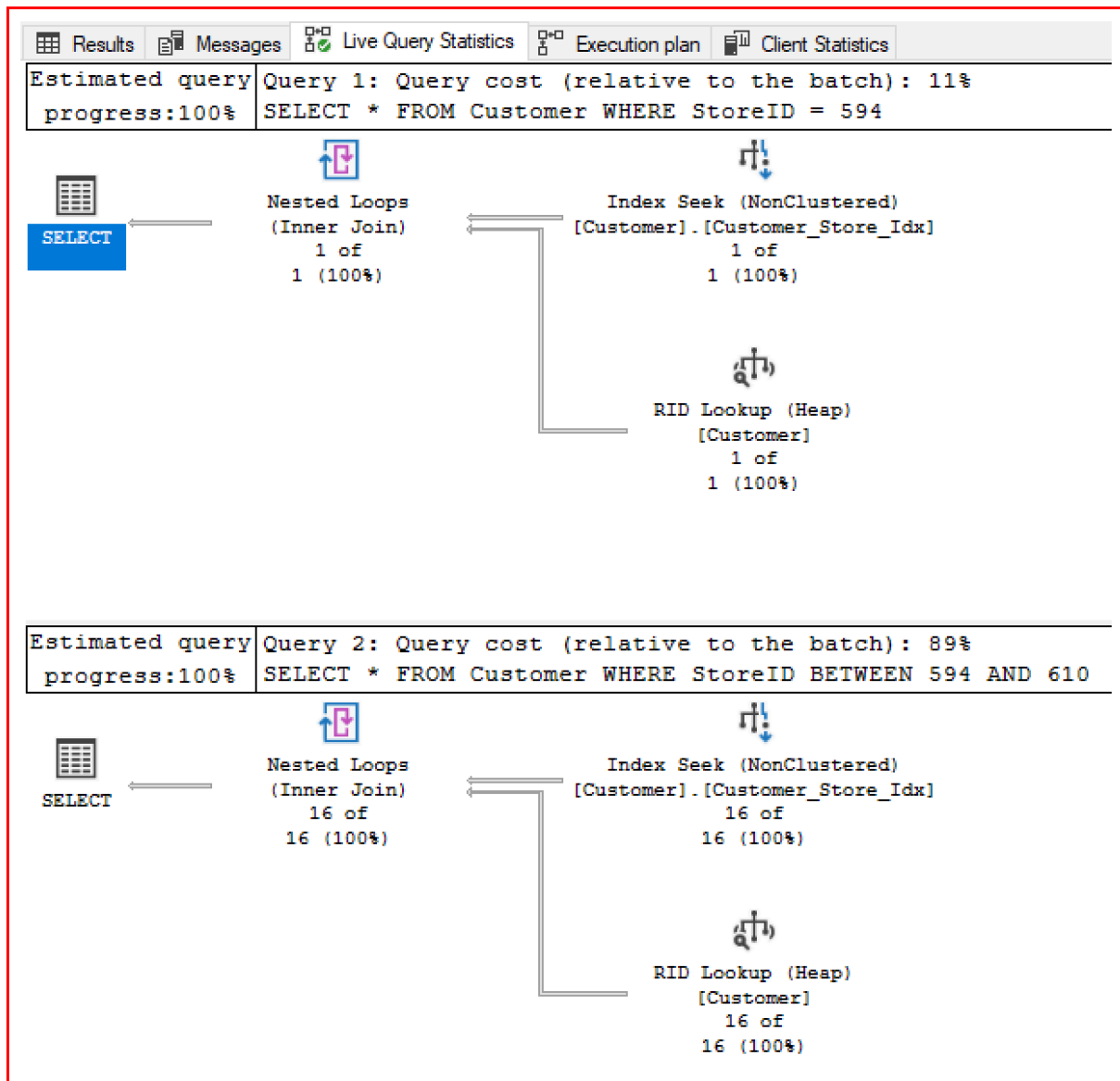


Dodaj indeks:

```
CREATE INDEX Customer_Store_Idx ON Customer(StoreID)
```

Jak zmienił się plan i czas? Czy jest możliwość optymalizacji?

Koszt: 11% / 89%, Czas: 93ms, 87 ms.

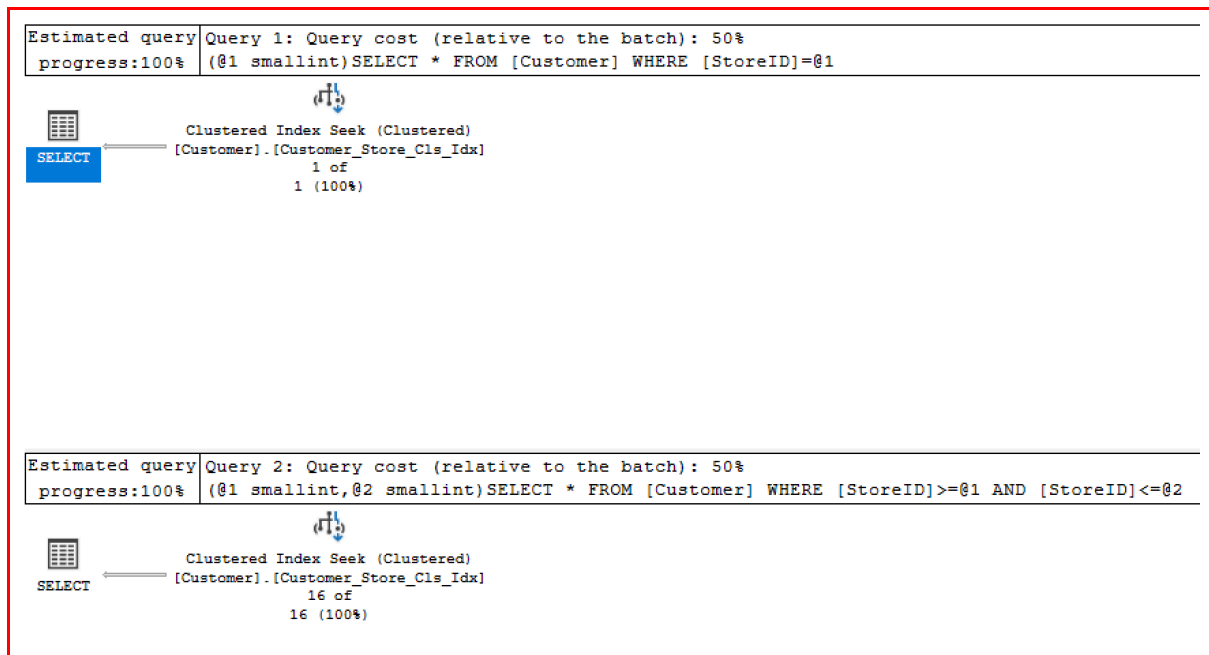


Dodaj indeks klastrowany:

```
CREATE CLUSTERED INDEX Customer_Store_Cls_Idx ON Customer(StoreID)
```

Czy zmienił się plan i czas? Skomentuj dwa podejścia w wyszukiwaniu krotek.

Koszt: 50% / 50%, Czas: 89 ms i 142 ms. Po planach widać, że w przypadku nieklastrowanych indeksów mamy na myśli przechowywany stop indeksów dołączany do tabeli i łączony przy pomocny inner join. Natomiast klastrowany jest na stałe przypisany do krotek w tabeli, dlatego można zdefiniować tylko jeden indeks klastrowany na tabelę.



Zadanie 2 – Indeksy zawierające dodatkowe dane z kolumn

Celem zadania jest poznanie indeksów z przechowywaniem kolumn...

Skopiuj tablicę Person do swojej bazy danych:

```
SELECT [BusinessEntityID]
      ,[PersonType]
      ,[NameStyle]
      ,[Title]
      ,[FirstName]
      ,[MiddleName]
      ,[LastName]
      ,[Suffix]
      ,[EmailPromotion]
      ,[rowguid]
      ,[ModifiedDate]
INTO [Person]
FROM [AdventureWorks2017].[Person].[Person]
```

Wykonaj analizę planu dla trzech zapytań:







```
SELECT * FROM [Person] WHERE LastName = 'Agbonile'

SELECT * FROM [Person] WHERE LastName = 'Agbonile' AND FirstName = 'Osarumwense'

SELECT * FROM [Person] WHERE FirstName = 'Osarumwense'
```

Co można o nich powiedzieć?

Bez indeksów mamy do czynienia z prostym przeszukiwaniem liniowym tabeli.

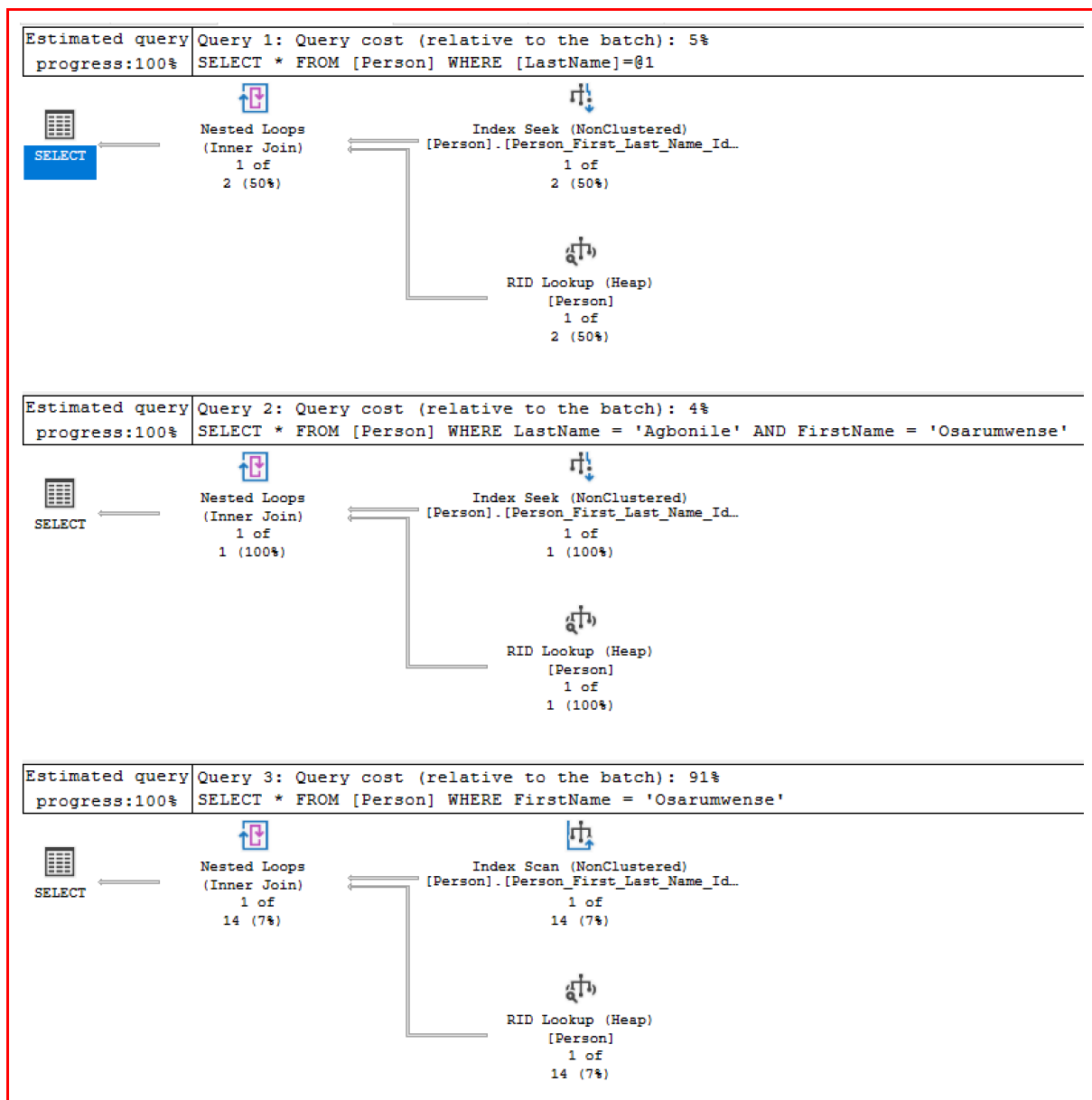
Estimated query progress:100%	Query 1: Query cost (relative to the batch): 33% SELECT * FROM [Person] WHERE [LastName]=@1
	 Table Scan [Person] 1 of 2 (50%)
Estimated query progress:100%	Query 2: Query cost (relative to the batch): 33% SELECT * FROM [Person] WHERE [LastName]=@1 AND [FirstName]=@2
	 Table Scan [Person] 1 of 1 (100%)
Estimated query progress:100%	Query 3: Query cost (relative to the batch): 33% SELECT * FROM [Person] WHERE [FirstName]=@1
	 Table Scan [Person] 1 of 14 (7%)

Przygotuj indeks obejmujący te zapytanie:

```
CREATE INDEX Person_First_Last_Name_Idx
ON Person (LastName, FirstName)
```

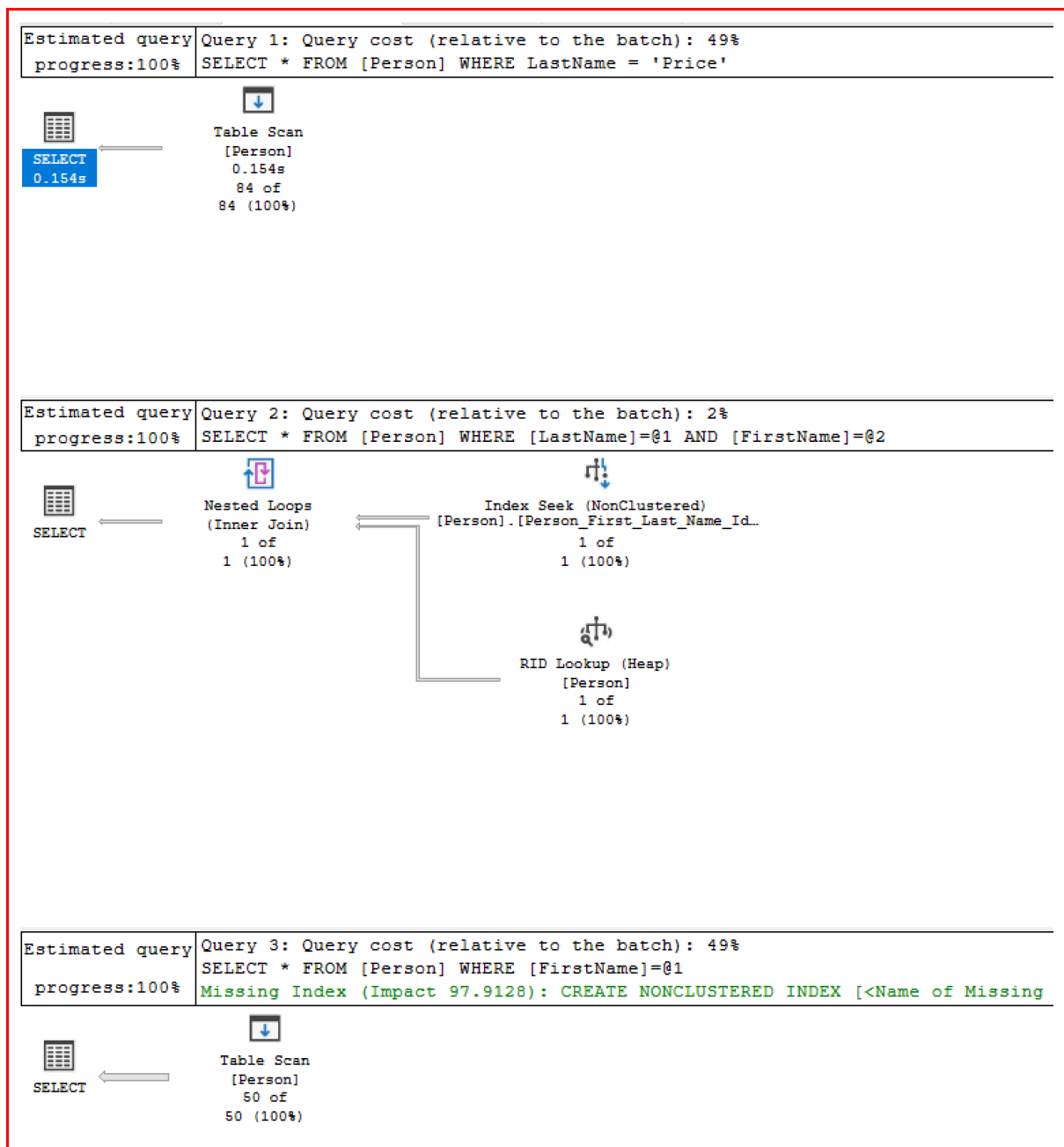
Sprawdź plan zapytania. Co się zmieniło?

Został dodany wcześniej wspomniany Inner Join i stworzony został nie klastrowany indeks na dwóch kolumnach. Przez to widać, że w przypadku wyszukiwania po nazwisku bądź imieniu i nazwisku kosztuje znacznie mniej niż po imieniu (ponieważ index zaczyna się nazwiskiem)



Przeprowadź ponownie analizę zapytań tym razem dla parametrów: FirstName = 'Angela' LastName = 'Price'. (Trzy zapytania, różna kombinacja parametrów). Czym różni się ten plan od zapytania o 'Osarumwense Agbonile'. Dlaczego tak jest?

Jak widać, tylko w przypadku drugiego zapytania wykorzystane zostały indeksy. Stało się tak zapewne ze względu na to, że w tym przypadku zarówno 'Angela', jak i 'Price' nie są unikalne w tabeli w przeciwieństwie do 'Osarumwense' i 'Agbonile'.



Zadanie 3

Skopiuj tablicę PurchaseOrderDetail do swojej bazy danych:

```
SELECT * INTO [PurchaseOrderDetail] FROM
[AdventureWorks2017].[Purchasing].[PurchaseOrderDetail]
```

Wykonaj analizę zapytania:

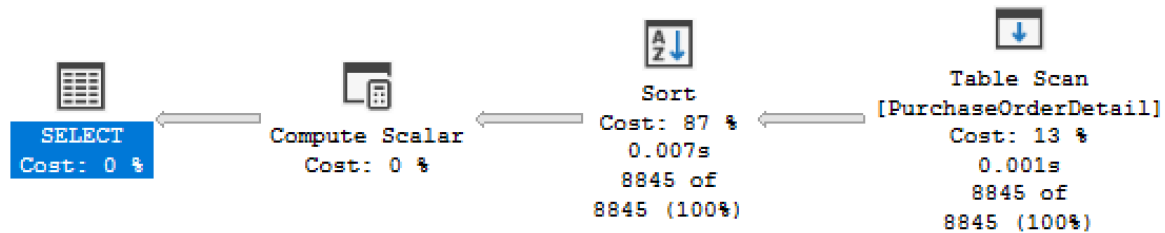
```
SELECT RejectedQty, ((RejectedQty/OrderQty)*100) AS RejectionRate,
ProductID, DueDate
FROM PurchaseOrderDetail
ORDER BY RejectedQty DESC, ProductID ASC
```

Która część zapytania ma największy koszt?

Największy koszt miał 'Sort' – 87%.

Query 1: Query cost (relative to the batch): 100%

```
SELECT RejectedQty, ((RejectedQty/OrderQty)*100) AS RejectionRate,
```



Jaki indeks można zastosować aby zoptymalizować koszt zapytania? Napisz go:

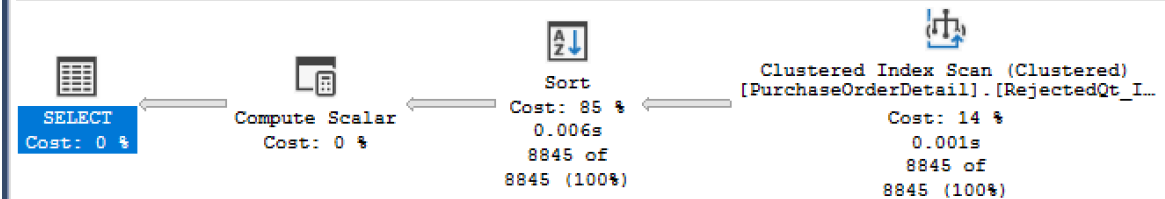
Zastosowanie klastrowanego indeksu na RejectedQty

```
CREATE CLUSTERED INDEX RejectedQt_Idx_1  
ON PurchaseOrderDetail(RejectedQty)
```

Wklej obrazek z planem:

Query 1: Query cost (relative to the batch): 100%

```
SELECT RejectedQty, ((RejectedQty/OrderQty)*100) AS RejectionRate, ProductID, Du
```



Zadanie 4

Celem zadania jest porównanie indeksów zawierających wszystkie kolumny z przechowywującym kolumny.

Skopiuj tabelicę Address do swojej bazy danych:

```
SELECT * INTO [Address] FROM [AdventureWorks2017].[Person].[Address]
```

W tej części będziemy analizować następujące zapytanie:

```
SELECT AddressLine1, AddressLine2, City, StateProvinceID, PostalCode  
FROM Address  
WHERE PostalCode BETWEEN N'98000' and N'99999'
```

Stwórz dwa indeksy:

```
CREATE INDEX Address_PostalCode_1  
ON Address (PostalCode)  
INCLUDE (AddressLine1, AddressLine2, City, StateProvinceID);  
GO
```

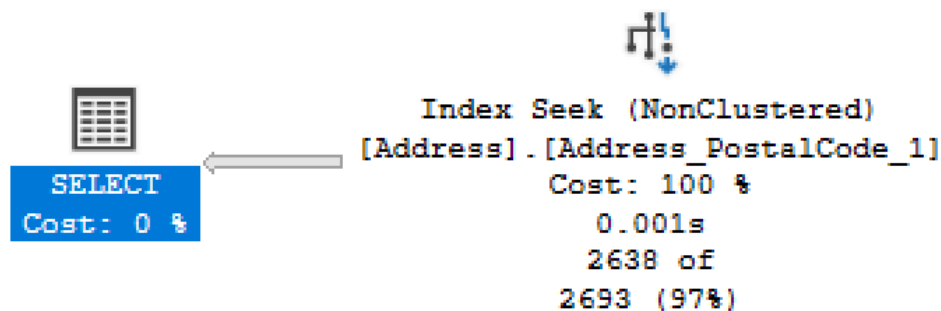


```
CREATE INDEX Address_PostalCode_2
ON Address (PostalCode, AddressLine1, AddressLine2, City,
StateProvinceID);
GO
```

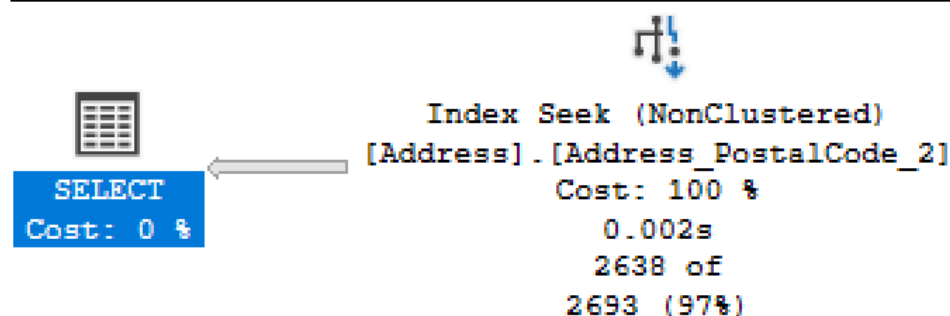
Czy jest widoczna różnica w zapytaniach? Jeśli tak to jaka? Aby wymusić użycie indeksu użyj WITH(INDEX(Address_PostalCode_1)) po FROM:

Czas dla indeksu Address_PostalCode_2 jest większy.

Query 1: Query cost (relative to the batch): 100%
 SELECT AddressLine1, AddressLine2, City, StateProv:



Query 1: Query cost (relative to the batch): 100%
 SELECT AddressLine1, AddressLine2, City, StateProv



Sprawdź rozmiar Indeksów:

```
SELECT i.[name] AS IndexName, SUM(s.[used_page_count]) * 8 AS IndexSizeKB
FROM sys.dm_db_partition_stats AS s
INNER JOIN sys.indexes AS i ON s.[object_id] = i.[object_id] AND
s.[index_id] = i.[index_id]
WHERE i.[name] = 'Address_PostalCode_1' OR i.[name] =
'Address_PostalCode_2'
GROUP BY i.[name]
GO
```

Który jest większy? Jak można skomentować te dwa podejścia? Które kolumny wpływają na to?

Im więcej kolumn tym większy rozmiar indeksu.

	IndexName	IndexSizeKB
1	Address_PostalCode_1	1784
2	Address_PostalCode_2	1808

Zadanie 5 – Indeksy z filtrami

Celem zadania jest poznanie indeksów z filtrami.

Skopiuj tablicę BillOfMaterials do swojej bazy danych:

```
SELECT * INTO BillOfMaterials
FROM [AdventureWorks2017].[Production].BillOfMaterials
```

W tej części analizujemy zapytanie:

```
SELECT ProductAssemblyID, ComponentID, StartDate
FROM BillOfMaterials
WHERE EndDate IS NOT NULL
      AND ComponentID = 327
      AND StartDate >= '2010-08-05'
```

Zastosuj indeks:

```
CREATE NONCLUSTERED INDEX BillOfMaterials_Cond_Idx
ON BillOfMaterials (ComponentID, StartDate)
WHERE EndDate IS NOT NULL
```

Sprawdź czy działa. Wykonaj plan poniższego zapytania:

```
SELECT ProductAssemblyID, ComponentID, StartDate
FROM BillOfMaterials
WHERE ComponentID = 327
      AND StartDate > '2010-08-05'
```

Czy indeks został użyty? Dlaczego?

Nie został, ponieważ indeks był stworzony dla rekordów, które w polu EndDate nie mają null. Oznacza to, że analizując nasze zapytanie istnieją rekordy, które potencjalnie mogą zostać zwrócone, a mają EndDate równy null, a więc na nich indeks nie został stworzony.

Spróbuj wymusić indeks. Co się stało, dlaczego takie zachowanie?

Zapytanie wywaliło się, została zwrócona informacja

Query processor could not produce a query plan because of the hints defined in this query. Resubmit the query without specifying any hints and without using SET FORCEPLAN.