

# Indeksy - Karta pracy nr 2

Imię i Nazwisko: Wojciech Koszyła

Swoje odpowiedzi wpisuj w **czzerwone 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]
```

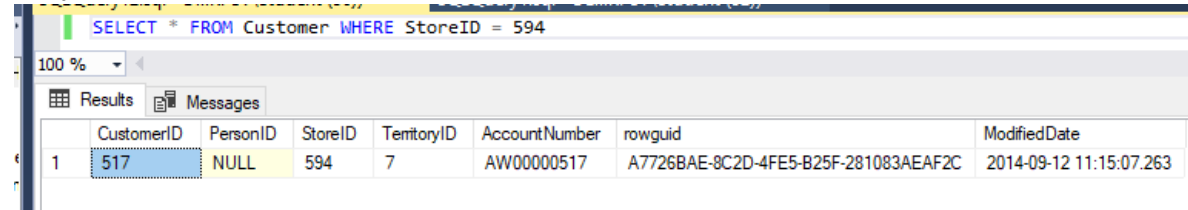
Wykonaj analizy zapytań:

```
SELECT * FROM Customer WHERE StoreID = 594
```

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

Zanotuj czas zapytania, jego koszt:

Zapytanie 1:



SQL Query (12.sql) - D:\KPSV\Student (30) - SQL Query (12.sql) - D:\KPSV\Student (30)

```
SELECT * FROM Customer WHERE StoreID = 594
```

100 %

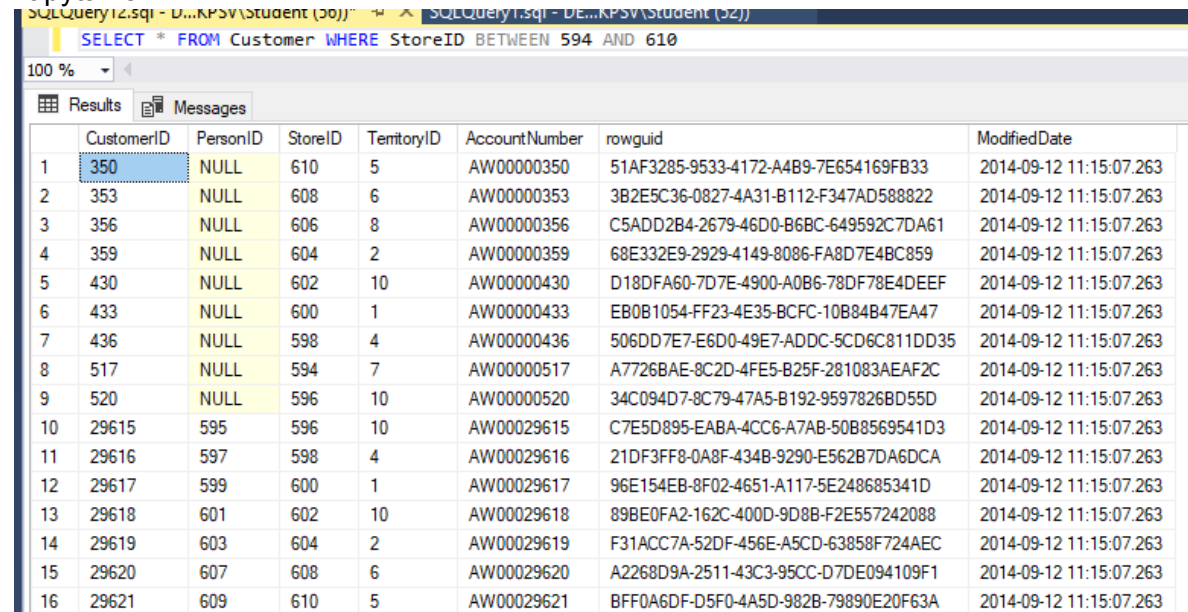
Results Messages

	CustomerID	PersonID	StoreID	TerritoryID	AccountNumber	rowguid	ModifiedDate
1	517	NULL	594	7	AW00000517	A7726BAE-8C2D-4FE5-B25F-281083AEAF2C	2014-09-12 11:15:07.263

Czas 1:

XYZ 00:00:01 1 rows

Zapytanie 2:



SQL Query (12.sql) - D:\KPSV\Student (30) - SQL Query (12.sql) - D:\KPSV\Student (30)

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

100 %

Results Messages

	CustomerID	PersonID	StoreID	TerritoryID	AccountNumber	rowguid	ModifiedDate
1	350	NULL	610	5	AW00000350	51AF3285-9533-4172-A4B9-7E654169FB33	2014-09-12 11:15:07.263
2	353	NULL	608	6	AW00000353	3B2E5C36-0827-4A31-B112-F347AD588822	2014-09-12 11:15:07.263
3	356	NULL	606	8	AW00000356	C5ADD2B4-2679-46D0-B6BC-649592C7DA61	2014-09-12 11:15:07.263
4	359	NULL	604	2	AW00000359	68E332E9-2929-4149-8086-FA8D7E4BC859	2014-09-12 11:15:07.263
5	430	NULL	602	10	AW00000430	D18DFA60-7D7E-4900-A0B6-78DF78E4DEEF	2014-09-12 11:15:07.263
6	433	NULL	600	1	AW00000433	EB0B1054-FF23-4E35-BCFC-10B84B47EA47	2014-09-12 11:15:07.263
7	436	NULL	598	4	AW00000436	506DD7E7-E6D0-49E7-ADDC-5CD6C811DD35	2014-09-12 11:15:07.263
8	517	NULL	594	7	AW00000517	A7726BAE-8C2D-4FE5-B25F-281083AEAF2C	2014-09-12 11:15:07.263
9	520	NULL	596	10	AW00000520	34C094D7-8C79-47A5-B192-9597826BD55D	2014-09-12 11:15:07.263
10	29615	595	596	10	AW00029615	C7E5D895-EABA-4CC6-A7AB-50B8569541D3	2014-09-12 11:15:07.263
11	29616	597	598	4	AW00029616	21DF3FF8-0A8F-434B-9290-E562B7DA6DCA	2014-09-12 11:15:07.263
12	29617	599	600	1	AW00029617	96E154EB-8F02-4651-A117-5E248685341D	2014-09-12 11:15:07.263
13	29618	601	602	10	AW00029618	89BE0FA2-162C-400D-9D8B-F2E557242088	2014-09-12 11:15:07.263
14	29619	603	604	2	AW00029619	F31ACC7A-52DF-456E-A5CD-63858F724AEC	2014-09-12 11:15:07.263
15	29620	607	608	6	AW00029620	A2268D9A-2511-43C3-95CC-D7DE094109F1	2014-09-12 11:15:07.263
16	29621	609	610	5	AW00029621	BFF0A6DF-D5F0-4A5D-982B-79890E20F63A	2014-09-12 11:15:07.263

Czas 2:

XYZ 00:00:00 16 rows

Dodaj indeks:

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

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

Czas 1:

XYZ 00:00:00 1 rows

Czas 2:

XYZ 00:00:02 16 rows

Baza danych znajduje się na wirtualnej maszynie umieszczonej u mnie na dość "high-endowym" komputerze (12 rdzeni procesora). W przypadku zapytania pierwszego czas spadł do poniżej jednej sekundy, lecz w przypadku drugiego czas wzrósł. Zmiany te są jednak na pograniczu błędu pomiarowego, więc nie dają nam żadnej rzetelnej miary wzrostu/spadku wydajności systemu.

Z czysto logicznego punktu widzenia wydajność powinna wzrosnąć.

Moim pomysłem na dodatkową optymalizację byłoby posortowanie tablicy Customer po StoreID, lecz dodaje to narzut przy operacjach dodawania. Pozwalałoby to jednak na filtracji poprzez znalezienie pierwszego i ostatniego spełniającego - wszystko w środku również spełniałoby założenia filtra.

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.

Czas 1: 

XYZ	00:00:00	1 rows
-----	----------	--------

Czas 2: 

XYZ	00:00:00	16 rows
-----	----------	---------

Czas poprawił się.

Tworzenie indeksu klastrowanego polega na posortowaniu danych według klucza, dzięki czemu dostęp do danych jest "klastrowy". Wskazuje on na blok danych, natomiast "nieklastrowy" przechowuje kopię wartości i wskaźnik na te dane.

Klastrowy jest naturalnie szybszy i jego działanie jest podobne do działania dysków twardych, lecz jedynie jeden taki indeks może istnieć na jedną 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]
```

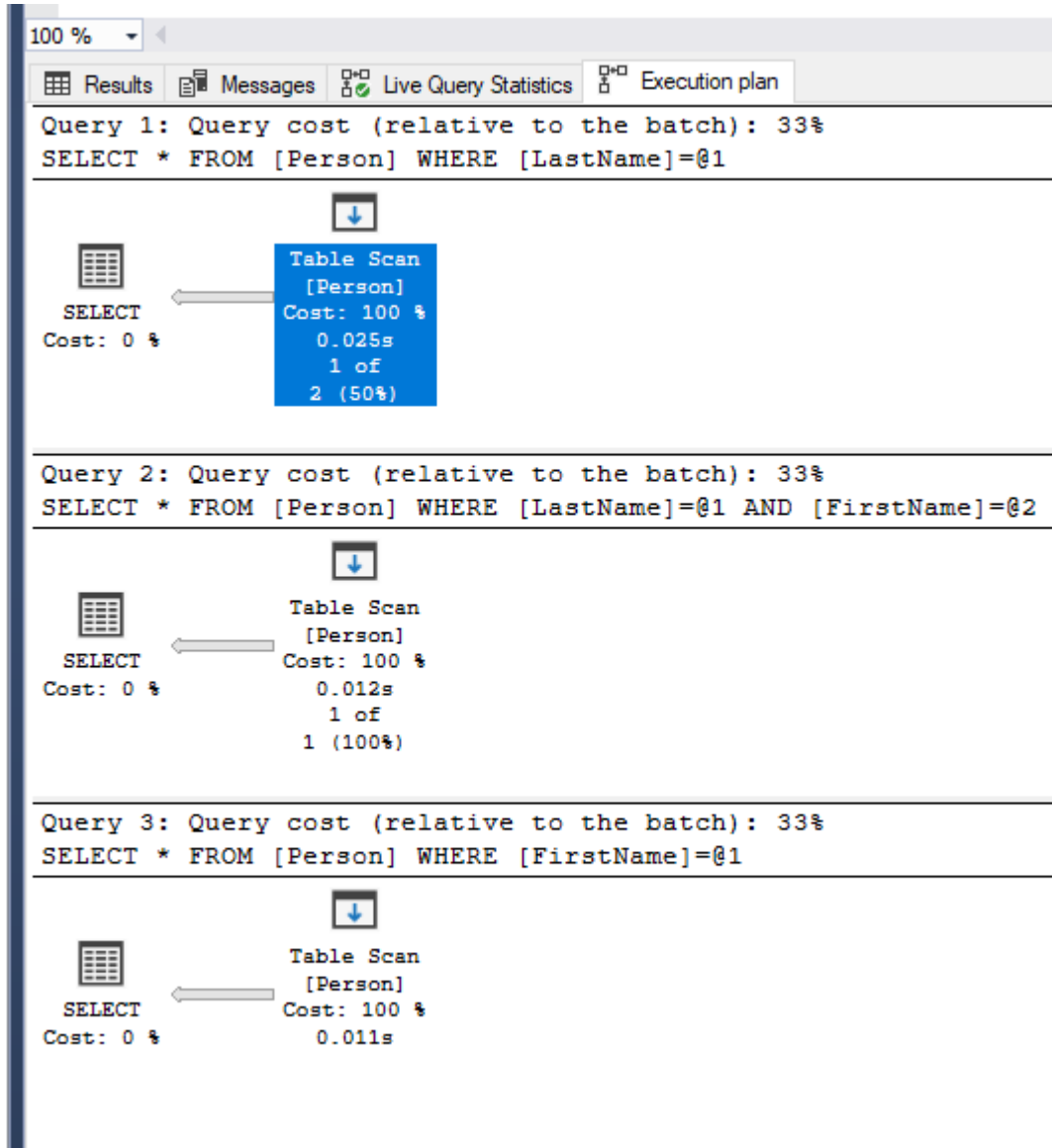
Wykonaj analizę planu dla trzech zapytań:

```
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?

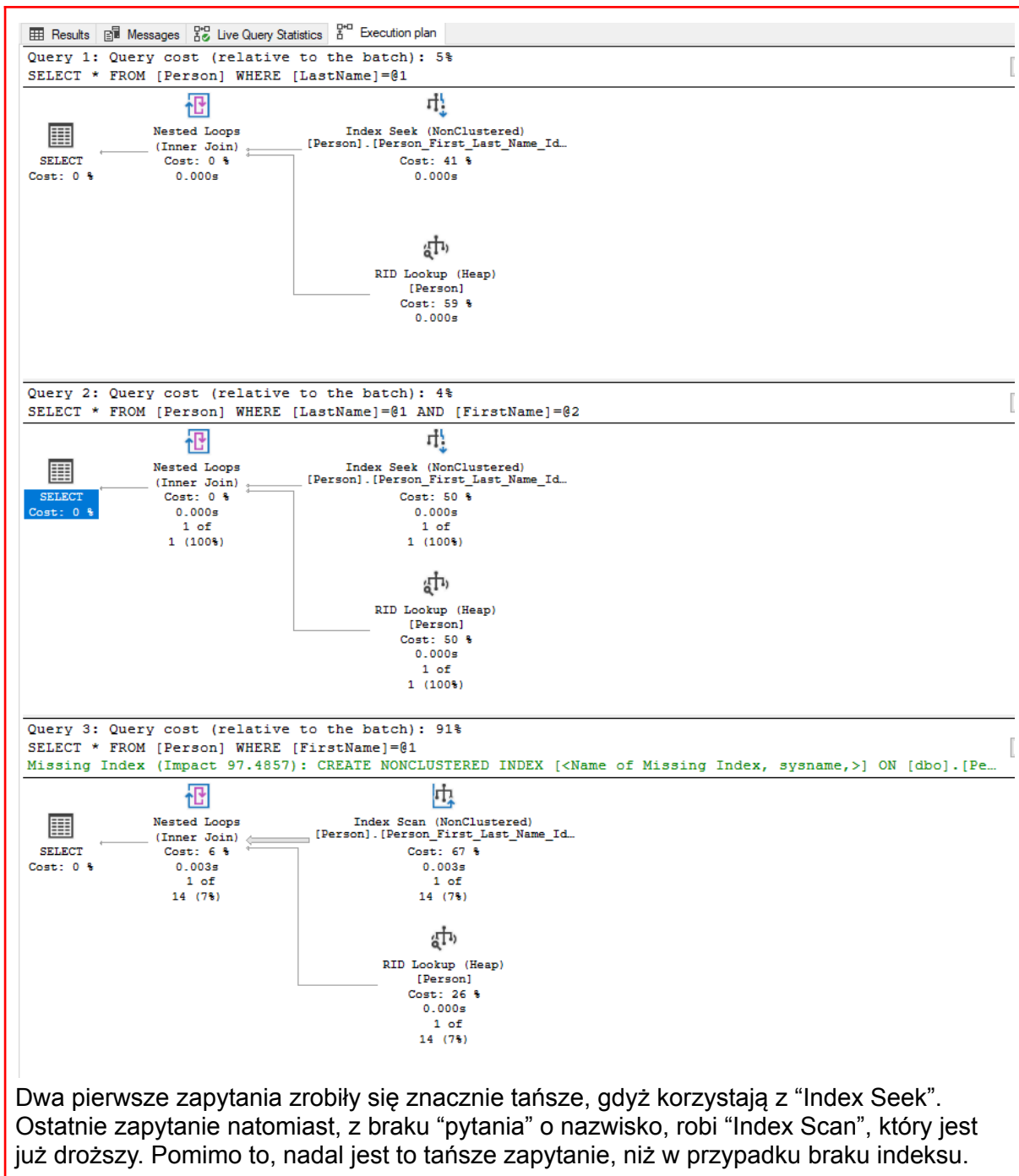


Wszystkie 3 zapytania mają podobny do siebie koszt, gdyż wszystkie czytają taką samą ilość wierszy "Numbers of rows read: 19972".

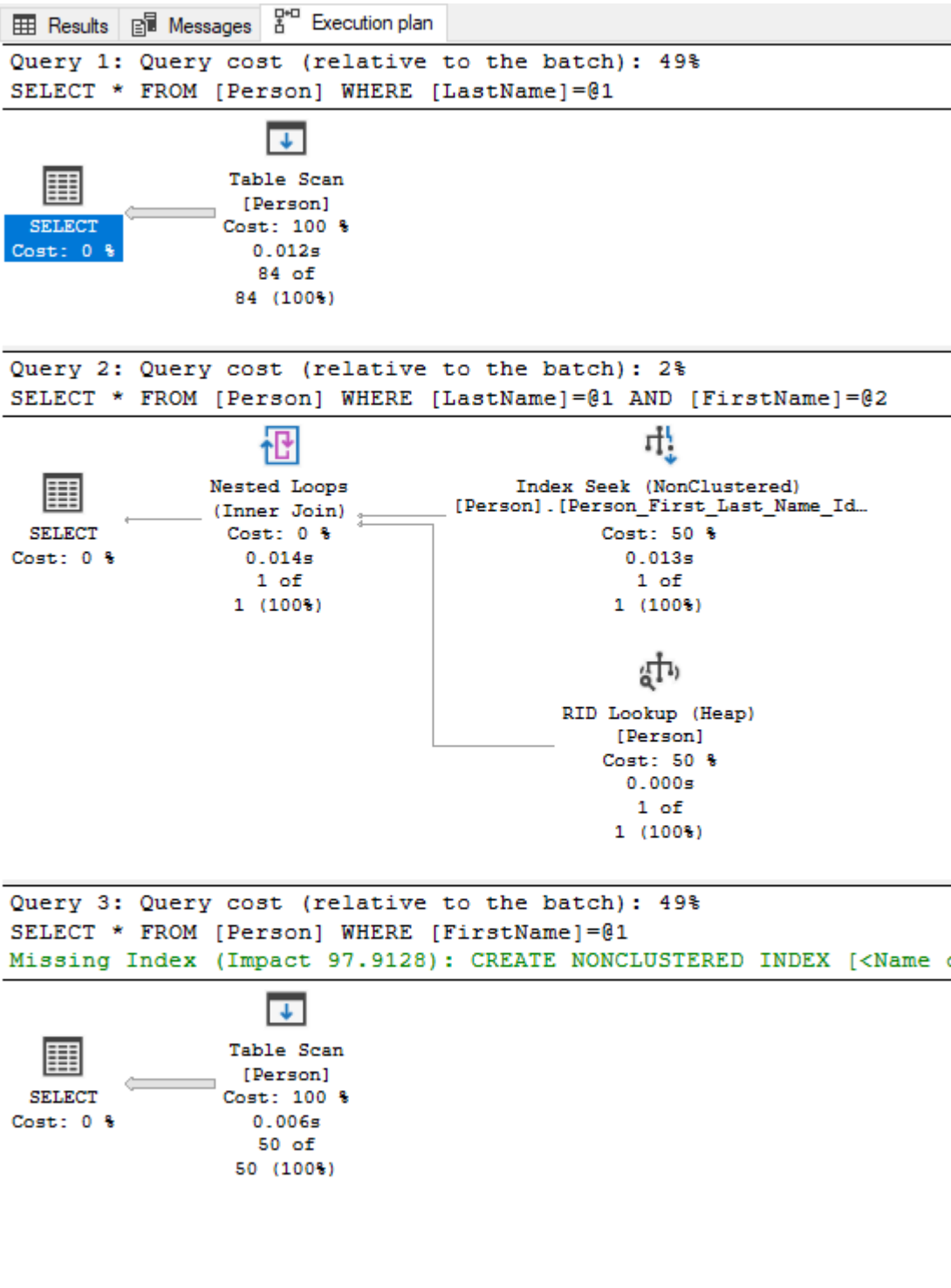
Przygotuj indeks obejmujący te zapytanie:

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

Sprawdź plan zapytania. Co się zmieniło?



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?



Tym razem tylko drugie zapytanie korzysta z indeksu.

Wydaje mi się, że jest to spowodowane ilością wyników spełniających te parametry - w przypadku "Angela"/"Price", jest ich bardzo dużo

100 %

Results Messages Execution plan

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName	Su
25	12980	IN	0	NULL	Emily	A	Price	N
26	13019	IN	0	NULL	Hannah	A	Price	N
27	13067	IN	0	NULL	Madison	NULL	Price	N
28	13094	IN	0	NULL	Fernando	NULL	Price	N
29	13136	IN	0	NULL	Spencer	NULL	Price	N
30	13181	IN	0	NULL	Antonio	E	Price	N
31	13810	IN	0	NULL	Marcus	R	Price	N
32	13862	IN	0	NULL	Ian	J	Price	N
33	13907	IN	0	NULL	Lucas	W	Price	N

<

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName	Suf
1	7642	IN	0	NULL	Angela	D	Price	NL

<

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName	Suf
1	115	EM	0	NULL	Angela	W	Barbariol	NL
2	401	SC	0	Ms.	Angela	NULL	Barbariol	NL
3	7642	IN	0	NULL	Angela	D	Price	NL
4	7643	IN	0	NULL	Angela	E	Bennett	NL
5	7644	IN	0	NULL	Angela	M	Wood	NL
6	7646	IN	0	NULL	Angela	M	Barnes	NL
7	7649	IN	0	NULL	Angela	D	Ross	NL
8	7653	IN	0	NULL	Angela	T	Henders...	NL

natomiast w przypadku mało popularnych “Osarumwense”/”Agbonile”, spełnia je tylko jedna osoba

100 %

Results Messages Execution plan

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName
1	4388	IN	0	NULL	Osarumwense	Uwafiokun	Agbonile

<

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName
1	4388	IN	0	NULL	Osarumwense	Uwafiokun	Agbonile

<

	BusinessEntityID	Person Type	NameStyle	Title	FirstName	MiddleName	LastName
1	4388	IN	0	NULL	Osarumwense	Uwafiokun	Agbonile

Indeks, który wcześniej stworzyliśmy, jest indeksem nie-klastrowym, dlatego w przypadku “Osarumwense”/”Agbonile” zadziałał natychmiastowo, a z “Angela”/”Price” musiał zostać przeprowadzony “Table Scan”.

## Zadanie 3

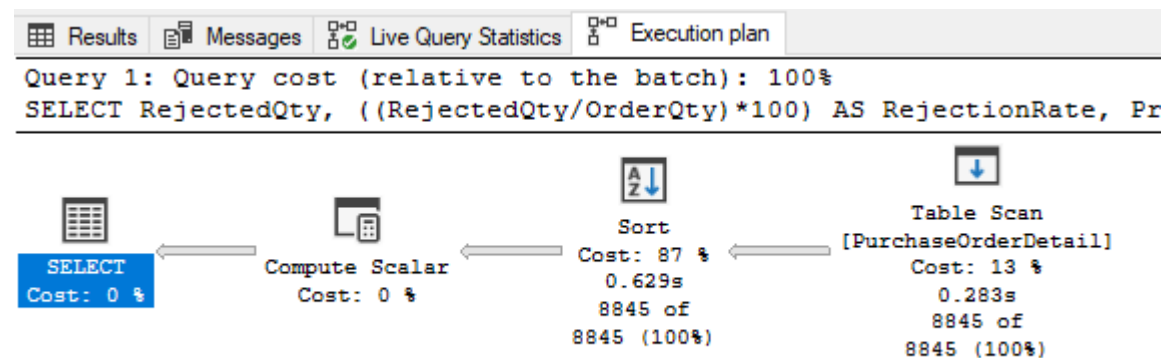
Skopiuj tablicę PurchaseOrderDetail do swojej bazy danych:

```
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?

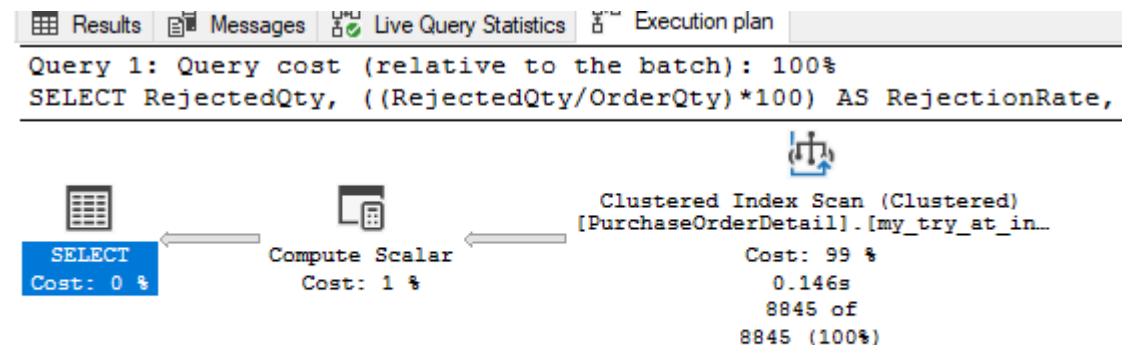


Największy koszt ma sortowanie wyników.

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

```
CREATE CLUSTERED INDEX my_try_at_indexing ON PurchaseOrderDetail(RejectedQty
DESC, ProductID ASC)
```

Wklej obrazek z planem:



## 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:

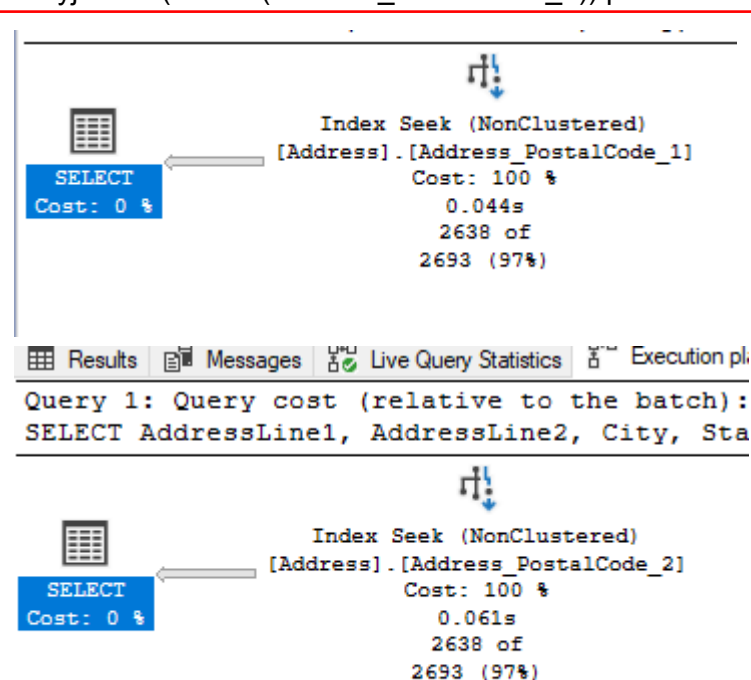
```
SELECT AddressLine1, AddressLine2, City, StateProvinceID, PostalCode
FROM Address
WHERE PostalCode BETWEEN N'98000' and N'99999'
```

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:



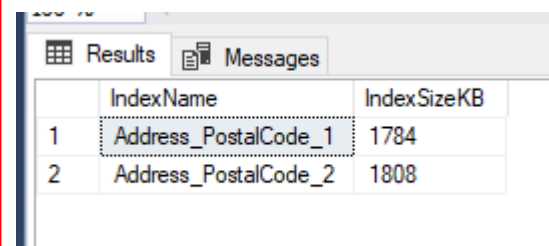
W zapytaniach jedyną widoczną różnicą może być inny czas wykonywania, lecz jest na tyle mały, że może wynikać z warunków środowiska komputerowego.

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?



	IndexName	IndexSizeKB
1	Address_PostalCode_1	1784
2	Address_PostalCode_2	1808

W pierwszym przypadku, tylko PostalCode jest w kluczu, a reszta kolumn jest dołączona do liści w B-drzewie indeksu.

W drugim przypadku, wszystkie kolumny są w kluczu, więc struktura B-drzewa powinna być trochę większa..

## 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'
```

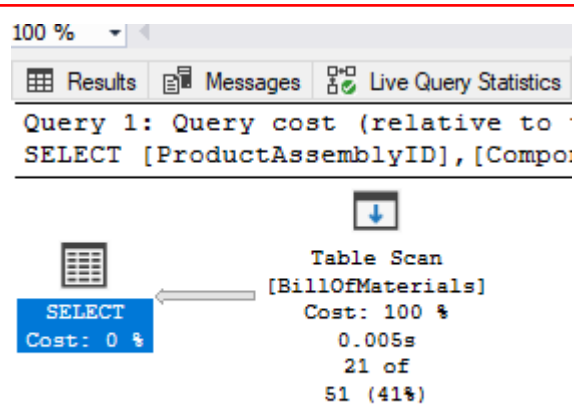
Zastosuj indeks:

```
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?



Indeks nie został użyty. Jest to dość oczywiste, gdyż filtracja w zapytaniu jest dość nieodpowiednia.

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

The image shows a screenshot of the SQL Server Messages window. The message text is: 'Msg 8622, Level 16, State 1, Line 1 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.' The completion time is 2022-05-05T11:23:05.7709371+02:00.

Pełna wiadomość: 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.

Silnik zapytań SQL sam generuje plan wykonywania podzapytań. Tutaj najzwyczajniej, ten silnik nie umiał stworzyć takiego planu, żeby wykorzystać podany przez nas stricte indeks.