Indeksy, optymalizator
Lab 5

Imię i nazwisko:

Przemysław Spyra, Piotr Urbańczyk

Celem ćwiczenia jest zapoznanie się z planami wykonania zapytań (execution plans), oraz z budową i możliwością wykorzystaniem indeksów (cz. 2.)

Swoje odpowiedzi wpisuj w miejsca oznaczone jako:

> Wyniki:

```
-- ...
```

Ważne/wymagane są komentarze.

Zamieść kod rozwiązania oraz zrzuty ekranu pokazujące wyniki, (dołącz kod rozwiązania w formie tekstowej/źródłowej)

Zwróć uwagę na formatowanie kodu

Oprogramowanie - co jest potrzebne?

Do wykonania ćwiczenia potrzebne jest następujące oprogramowanie

- MS SQL Server,
- SSMS - SQL Server Management Studio
- przykładowa baza danych AdventureWorks2017.

Oprogramowanie dostępne jest na przygotowanej maszynie wirtualnej

Przygotowanie

Uruchom Microsoft SQL Management Studio.

Stwórz swoją bazę danych o nazwie XYZ.

```
create database lab5
go

use lab5
go
```

Dokumentacja/Literatura

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

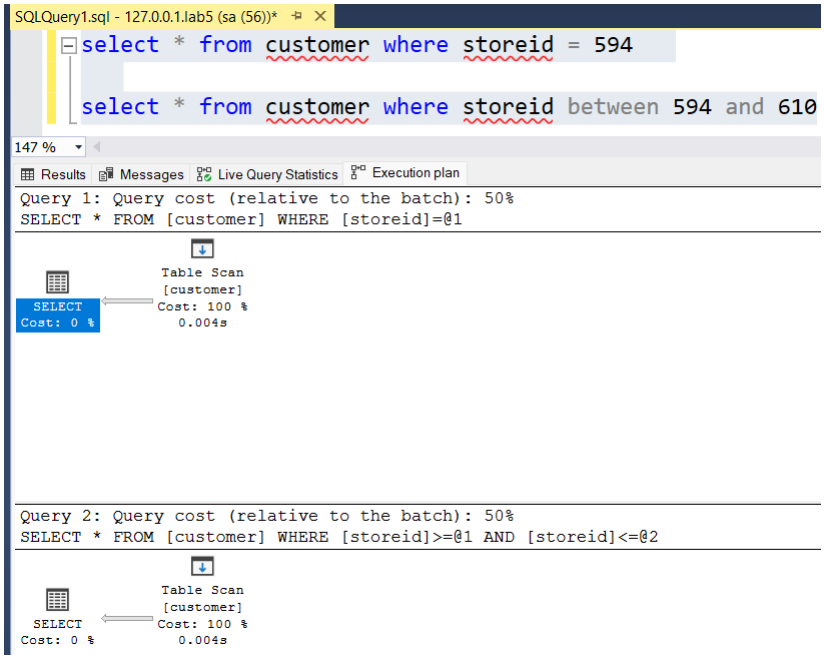
Skopiuj tabelę `Customer` do swojej bazy danych:

```
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 oraz jego koszt koszt:

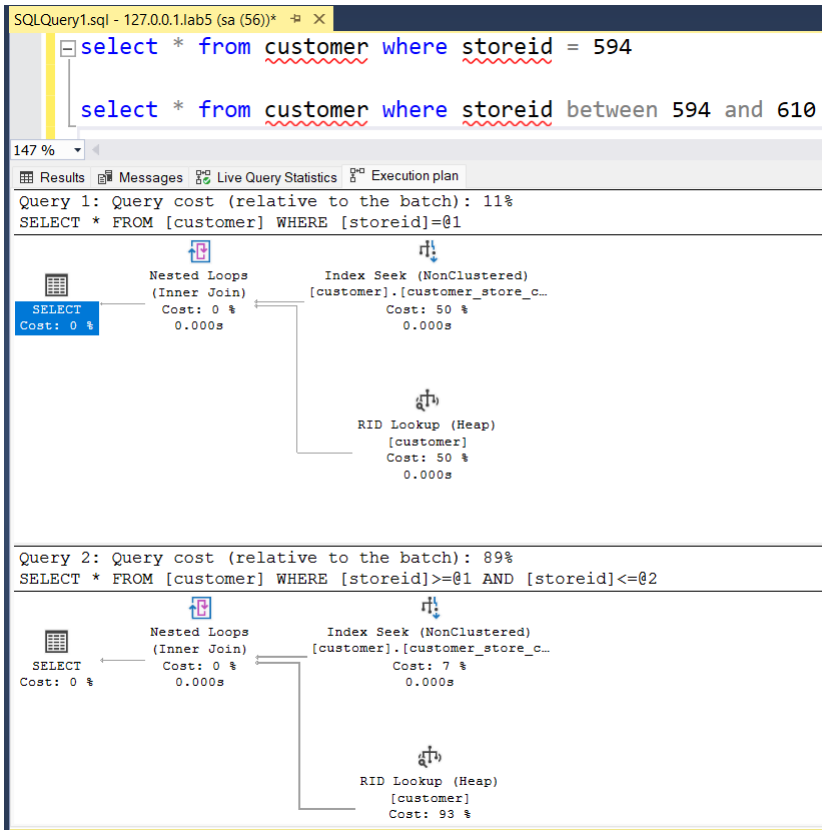


Czasy wykonania zapytań pierwszego oraz drugiego to 0.004s, a 100% ich kosztów to przeszukiwanie tabeli.

Dodaj indeks:

```
create  index customer_store_cls_idx on customer(storeid)
```

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



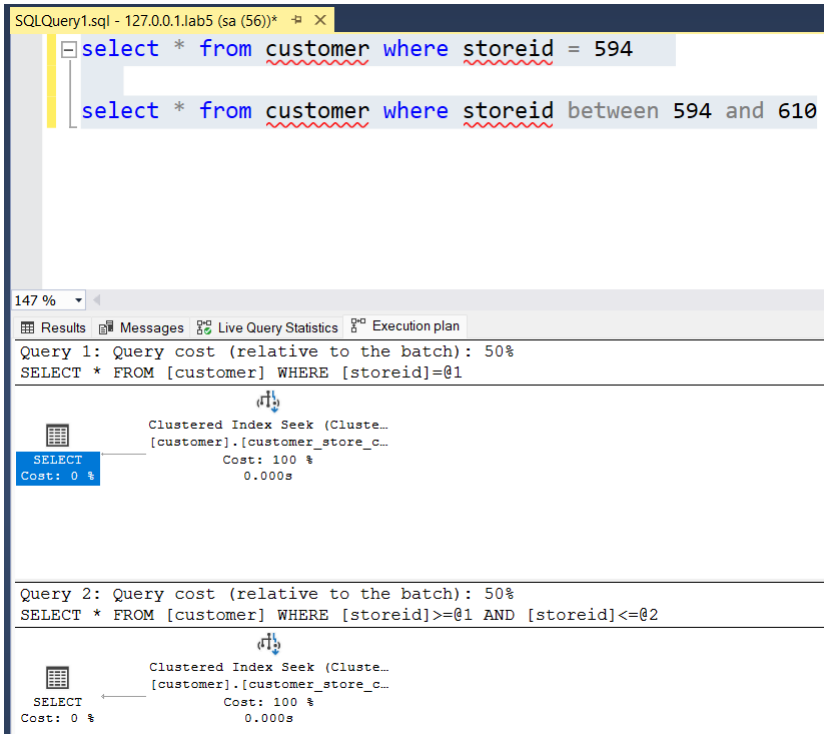
Plan wykonania zmienił się z prostego skanu tabeli (w obu przypadkach) na wykonanie Inner Joinów, a następnie wyszukanie wszystkich nieklastrowanych indeksów oraz RID Lookupa. Chodzi o to, że każdy nieklastrowany indeks zawiera ROW ID, aby móc potem szybko znaleźć pozostałą część tabeli w heap table. W taki właśnie sposób RID Lookup może przeglądać heap table używając Row ID.

- Czas wykonania zapytania 1 to 0.000s, a 50% jego kosztu to szukanie indeksu, a drugie 50% kosztu generuje RID Lookup.
- Czas wykonania zapytania 2 to 0.000s, a 7% jego kosztu to szukanie indeksu, a 93% kosztu generuje RID Lookup.

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.



Po sklastrowaniu indeksu nasze plan składają się tylko z jego wyszukiwania, które stanowi 100% kosztu w obu zapytaniach. Zapytania wykonuję się natychmiast, czyli w 0.000s

Zadanie 2 – Indeksy zawierające dodatkowe atrybuty (dane z kolumn)

Celem zadania jest poznanie indeksów z przechowujących dodatkowe atrybuty (dane z kolumn)

Skopiuj tabelę `Person` do swojej bazy danych:

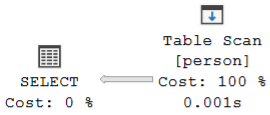
```
select businessentityid
      ,persontype
      ,namestyle
      ,title
      ,firstname
      ,middlename
      ,lastname
      ,suffix
      ,emailpromotion
      ,rowguid
      ,modifieddate
into person
from adventureworks2017.person.person
```

Wykonaj analizę planu dla trzech zapytań:

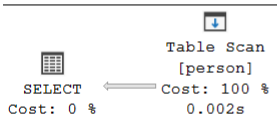
```
select * from [person] where lastname = 'Agbonile'

select * from [person] where lastname = 'Agbonile' and firstname = 'Osarumwense'

select * from [person] where firstname = 'Osarumwense'
```

Co można o nich powiedzieć?

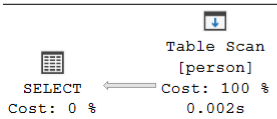
```
Query 1: Query cost (relative to the batch): 33%
SELECT * FROM [person] WHERE [lastname]=@1
```




```
Query 2: Query cost (relative to the batch): 33%
SELECT * FROM [person] WHERE [lastname]=@1 AND [firstname]=@2
```




```
Query 3: Query cost (relative to the batch): 33%
SELECT * FROM [person] WHERE [firstname]=@1
```



	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost
1	1	1	SELECT * FROM [person] WHERE [lastname]=@1	1	1	0	NULL	NULL	NULL	NULL	1.866667	NULL	NULL	NULL	0.1778438
2	1	1	|--Table Scan(OBJECT:([lab5].[dbo].[person]), W...	1	2	1	Table Scan	Table Scan	OBJECT:([l...	[lab5].[dbo].[person].[businessentityid], [lab5...	1.866667	0.1557961	0.0220477	186	0.1778438

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argu...	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost
1	1	1	SELECT * FROM [person] WHERE [lastname]=@1 AND ...	3	1	0	NULL	NULL	NULL	NULL	1	NULL	NULL	NULL	0.1778438
2	1	1	|--Table Scan(OBJECT:([lab5].[dbo].[person]), WHERE:(...	3	2	1	Table Scan	Table Scan	OBJ...	[lab5].[dbo].[person].[businessentityid], [la...	1	0.1557961	0.0220477	147	0.1778438

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost
1	1	1	SELECT * FROM [person] WHERE [firstname]=@1	5	1	0	NULL	NULL	NULL	NULL	13.57143	NULL	NULL	NULL	0.1778438
2	1	1	|--Table Scan(OBJECT:([lab5].[dbo].[person]), W...	5	2	1	Table Scan	Table Scan	OBJECT:...	[lab5].[dbo].[person].[businessentityid], [lab5...	13.57143	0.1557961	0.0220477	186	0.1778438

Odpowiedź:

Zapytania mają zwrócić wszystkie wiersze (`*`) tabeli `person` spełniające różną postać klauzuli `WHERE` (zadeklarowanie pewnej tożsamości kolumny `lastname`, `firstname` oraz obu tych kolumn na raz). Pierwsze zapytanie szuka wierszy z kolumną `lastname` równą `'Agbonile'`. Drugie zapytanie szuka wierszy z kolumnami `lastname` równą `'Agbonile'` i kolumną `firstname` równą `'Osarumwense'`. Trzecie zapytanie szuka wierszy z kolumną `firstname` równą `'Osarumwense'`.

Klauzule są tak dobrane, że w efekcie wszystkie zapytania dają ten sam rezultat (wyświetlają jeden i ten sam wiersz). Co ciekawe mają nawet taki sam koszt wykonania (0.1778438), choć różnią się estymowaną liczbą wierszy do na wykonanie zapytania (odpowiednio, ok. 2, 1, oraz ok. 14). Te różnice w liczbie estymowanych wierszy dla zapytania drugiego łatwo wytłumaczyć obecnością operatora `and`. Różnice dla pierwszej i trzeciej tabeli mogą być wynikiem statystyk gromadzonych przez system zarządzania bazą danych dla poszczególnych kolumn (np. większa selektywność jednej z kolumn).

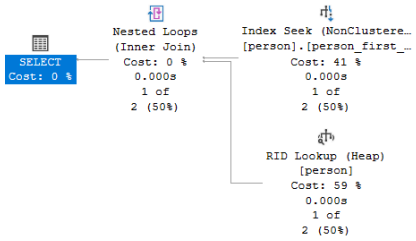
Wygląda na to, że przeszukiwane w zapytaniach kolumny nie mają indeksu, ponieważ w planie wykonania znajduje się pełne przeszukiwanie tabeli (Table Scan), a nie przeszukiwanie indeksu (Index Seek/Scan). Zapytania są na tyle proste, że nawet bez indeksów ich koszt wykonania jest stosunkowo niski. Dodanie ich mogłoby jednak znacząco przyspieszyć podobne zapytania, zwłaszcza dla dużych zestawów danych.

Przygotuj indeks obejmujący te zapytania:

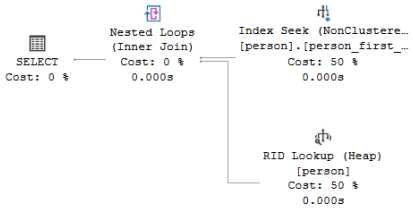
```
create index person_first_last_name_idx
on person(lastname, firstname)
```

Sprawdź plan zapytania. Co się zmieniło?

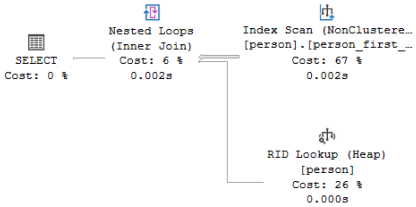
```
Query 1: Query cost (relative to the batch): 5%
SELECT * FROM [person] WHERE [lastname]=@1
```




```
Query 2: Query cost (relative to the batch): 4%
SELECT * FROM [person] WHERE [lastname]=@1 AND [firstname]=@2
```




```
Query 3: Query cost (relative to the batch): 91%
SELECT * FROM [person] WHERE [firstname]=@1
Missing Index (Impact 97.4859): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[person] ([firstname])
```



	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList
1	1	1	SELECT * FROM [person] WHERE [lastname]=@1	1	1	0	NULL	NULL	NULL	NULL	1.866667	NULL	NULL	NULL	0.008022903	NULL
2	1	1	|--Nested Loops(Inner Join, OUTER REFERENCES:([...	1	2	1	Nested Loops	Inner Join	OUTER REFER...	NULL	1.866667	0	7.802667E-06	186	0.008022903	[lab5].[dbo].[person].[businesser
3	1	1	|--Index Seek(OBJECT:([lab5].[dbo].[person].[perso...	1	3	2	Index Seek	Index Seek	OBJECT:([lab5].[...	[Bmk1000], [lab5].[...	1.866667	0.003125	0.0001590533	82	0.003284053	[Bmk1000], [lab5].[dbo].[person]
4	1	1	|--RID Lookup(OBJECT:([lab5].[dbo].[person]), SEE...	1	5	2	RID Lookup	RID Lookup	OBJECT:([lab5].[...	[lab5].[dbo].[perso...	1	0.003125	0.0001581	121	0.004731047	[lab5].[dbo].[person].[businesser

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList
1	1	1	SELECT * FROM [person] WHERE [lastname]=@1 AND ...	3	1	0	NULL	NULL	NULL	NULL	1.023365	NULL	NULL	NULL	0.006580354	NULL
2	1	1	|--Nested Loops(Inner Join, OUTER REFERENCES:([B...	3	2	1	Nested Loops	Inner Join	OUTER RE...	NULL	1.023365	0	4.277667E-06	147	0.006580354	[lab5].[dbo].[person].[businessent
3	1	1	|--Index Seek(OBJECT:([lab5].[dbo].[person].[person_...	3	3	2	Index Seek	Index Seek	OBJECT:([la...	[Bmk1000], [lab5].[...	1.023365	0.003125	0.0001581257	43	0.003283126	[Bmk1000], [lab5].[dbo].[person].[
4	1	1	|--RID Lookup(OBJECT:([lab5].[dbo].[person]), SEEK:...	3	5	2	RID Lookup	RID Lookup	OBJECT:([la...	[lab5].[dbo].[person]...	1	0.003125	0.0001581	121	0.003292951	[lab5].[dbo].[person].[businessent

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	4388	IN	0	NULL	Osarumwense	Uwaifiokun	Agbonile	NULL	0	3309A53F-3020-495A-A423-C1DBCCCAC66C	2013-11-30 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList
1	1	1	SELECT * FROM [person] WHERE [firstname]=@1	5	1	0	NULL	NULL	NULL	NULL	13.57143	NULL	NULL	NULL	0.1537648	NULL
2	1	1	|--Nested Loops(Inner Join, OUTER REFERENCES:([...	5	2	1	Nested Loops	Inner Join	OUTER RE...	NULL	13.57143	0	5.672857E-05	186	0.1537648	[lab5].[dbo].[person].[businessen
3	1	1	|--Index Scan(OBJECT:([lab5].[dbo].[person].[perso...	5	3	2	Index Scan	Index Scan	OBJECT:([l...	[Bmk1000], [lab5].[dbo...	13.57143	0.08164352	0.0221262	82	0.1037697	[Bmk1000], [lab5].[dbo].[person].
4	1	1	|--RID Lookup(OBJECT:([lab5].[dbo].[person]), SEE...	5	5	2	RID Lookup	RID Lookup	OBJECT:([l...	[lab5].[dbo].[person].[b...	1	0.003125	0.0001581	121	0.04035179	[lab5].[dbo].[person].[businessen

Odpowiedź:

Przeszukiwanie rekordów przy pomocy indeksu poprawiło plan wykonania zapytań a w przypadku pierwszych dwóch znacząco wpłynęło na koszty wykonania (odpowiednio 0.008022903 oraz 0.006580354).

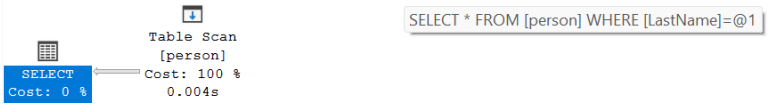
Oprócz przeszukiwania indeksów plany zawierają także operację RID Lookup, co oznacza, że SZBD po przeszukaniu indeksu musi także odwołać się do tabli by uzyskać dane z pozostałych kolumn. Dodatkowym sposobem optymalizacji, o którym mówiliśmy na zajęciach, byłoby zapytanie o kolumny, na których założony jest indeks, a nie `SELECT *`.

Trzecie zapytanie jest najbardziej kosztowne (0.1537648). Pojawia się przy nim sugestia utworzenia nieklastrowanego indeksu na kolumnie `firstname`. Oznacza to, że SZBD zidentyfikował, że takie działanie mogłoby dodatkowo poprawić wydajność dla zapytań wyszukujących w oparciu o tę kolumnę.

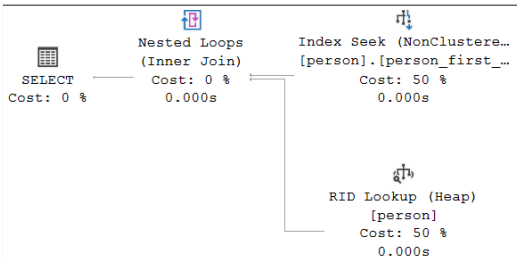
Przeprowadź ponownie analizę zapytań tym razem dla parametrów: `FirstName = 'Angela' LastName = 'Price'`. (Trzy zapytania, różna kombinacja parametrów).

Czym różni się ten plan od zapytania od `'Osarumwense Agbonile'` . Dlaczego tak jest?

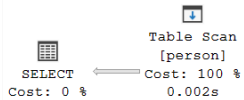
```
Query 1: Query cost (relative to the batch): 49%
SELECT * FROM [person] WHERE [LastName]=@1
```




```
Query 2: Query cost (relative to the batch): 2%
SELECT * FROM [person] WHERE [LastName]=@1 AND [FirstName]=@2
```




```
Query 3: Query cost (relative to the batch): 49%
SELECT * FROM [person] WHERE [FirstName]=@1
Missing Index (Impact 97.9375): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[person] ([firstname])
```



Results Messages Live Query Statistics Execution plan

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	3407	IN	0	NULL	Haley	A	Price	NULL	1	E65C7A6F-81FA-4BB7-935A-4EEA9297D104	2014-02-06 00:00:00.000
2	3578	IN	0	NULL	Taylor	NULL	Price	NULL	0	FF74286D-8B3E-499F-AB5C-357766D1384F	2013-06-02 00:00:00.000
3	3023	IN	0	NULL	Amanda	S	Price	NULL	0	0B9EC02B-2B57-4591-827F-5D4E3FCDE8EA	2013-11-20 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	Tot
1	84	1	SELECT * FROM [person] WHERE [LastName]=@1	1	1	0	NULL	NULL	NULL	NULL	84	NULL	NULL	NULL	0.1
2	84	1	|--Table Scan(OBJECT:([lab5].[dbo].[person]), WH...	1	2	1	Table Scan	Table Scan	OBJECT:([lab5].[dbo].[person]), WHERE:([lab5].[d...	[lab5].[dbo].[person].[businessentityid], [lab5...	84	0.1557176	0.0221262	186	0.1

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	7642	IN	0	NULL	Angela	D	Price	NULL	0	FBB7CF01-97B4-4315-B080-0DE770316BB9	2013-12-14 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	A
1	1	1	SELECT * FROM [person] WHERE [LastName]=@1 AND [...	3	1	0	NULL	NULL	NULL	NULL	1.023365	NULL	NULL	N
2	1	1	|--Nested Loops(Inner Join, OUTER REFERENCES:([Bm...	3	2	1	Nested Loops	Inner Join	OUTER REFERENCES:([Bmk1000])	NULL	1.023365	0	4.277667E-06	1
3	1	1	|--Index Seek(OBJECT:([lab5].[dbo].[person].[person_fir...	3	3	2	Index Seek	Index Seek	OBJECT:([lab5].[dbo].[person].[person_first_last_na...	[Bmk1000], [lab5].[dbo].[person].[firstname], [l...	1.023365	0.003125	0.0001581257	4
4	1	1	|--RID Lookup(OBJECT:([lab5].[dbo].[person]), SEEK:([...	3	5	2	RID Lookup	RID Lookup	OBJECT:([lab5].[dbo].[person]), SEEK:([Bmk1000]=[...	[lab5].[dbo].[person].[businessentityid], [lab5].[...	1	0.003125	0.0001581	1

	businessentityid	persontype	namestyle	title	firstname	middlename	lastname	suffix	emailpromotion	rowguid	modifieddate
1	8426	IN	0	NULL	Angela	NULL	Cooper	NULL	2	EA7F5ECA-230D-46AB-A431-A4D2CFA403DF	2013-11-03 00:00:00.000
2	8540	IN	0	NULL	Angela	NULL	Cox	NULL	1	ACE23885-36C0-400C-BE0F-19CBC9E390B3	2011-11-21 00:00:00.000
3	8151	IN	0	NULL	Angela	G	Torres	NULL	1	98DE8633-765A-4382-91B4-585C7E35C836	2013-10-22 00:00:00.000
4	8215	IN	0	NULL	Angela	NULL	Peterson	NULL	1	E520C52E-D84F-475F-A6C6-1B1149331D09	2014-02-22 00:00:00.000
5	8274	IN	0	NULL	Angela	T	Gray	NULL	0	D7AEA030-20DC-4807-BA8F-9D4468F8F314	2014-01-16 00:00:00.000
6	115	EM	0	NULL	Angela	W	Barbariol	NULL	2	8E6ADFD0-2FE0-4EB9-93F2-A04FD877F6DC	2009-01-13 00:00:00.000
7	401	SC	0	Ms.	Angela	NULL	Barbariol	NULL	0	E25E7C6E-F378-42E1-8F0C-CC4E79E35F1C	2012-05-30 00:00:00.000
8	8767	IN	0	NULL	Angela	E	Bell	NULL	2	9416BC5A-4CAE-48FF-A572-72F966C73743	2013-08-31 00:00:00.000

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList	Warnings	Type	Parallel	EstimateE
1	50	1	SELECT * FROM [person] WHERE [FirstName]=@1	5	1	0	NULL	NULL	NULL	NULL	50	NULL	NULL	NULL	0.1778438	NULL	NULL	SELECT	0	NULL
2	50	1	|--Table Scan(OBJECT:([lab5].[dbo].[person]), WH...	5	2	1	Table Sc...	Table S...	OBJEC...	[lab5].[dbo].[p...	50	0.1557176	0.0221262	186	0.1778438	[lab5].[d...	NULL	PLAN_...	0	1

Odpowiedź:

Plany zapytania pierwszego i trzeciego ponownie zawierają operację Table Scan (zamiast Index Seek) - mimo założonego indeksu. Zapytania, które nie wykorzystały indeksowania mają, zgodnie z oczekiwaniem wysoki koszt (0.1778438), zapytanie drugie ma koszt znacząco niższy (0.006580354).

Najprawdopodobniej jest to spowodowane selektywnością tych zapytań (czyli jak wiele wierszy spełnia warunki zapytania) i pozwala to sądzić, że SZBD przewiduje inny koszt wykonania zapytań w zależności od rozkładu danych w tabeli, co może wynikać z większej powszechności imienia 'Angela' i nazwiska 'Price' w bazie danych lub z różnic w rozkładzie danych.

Zadanie 3

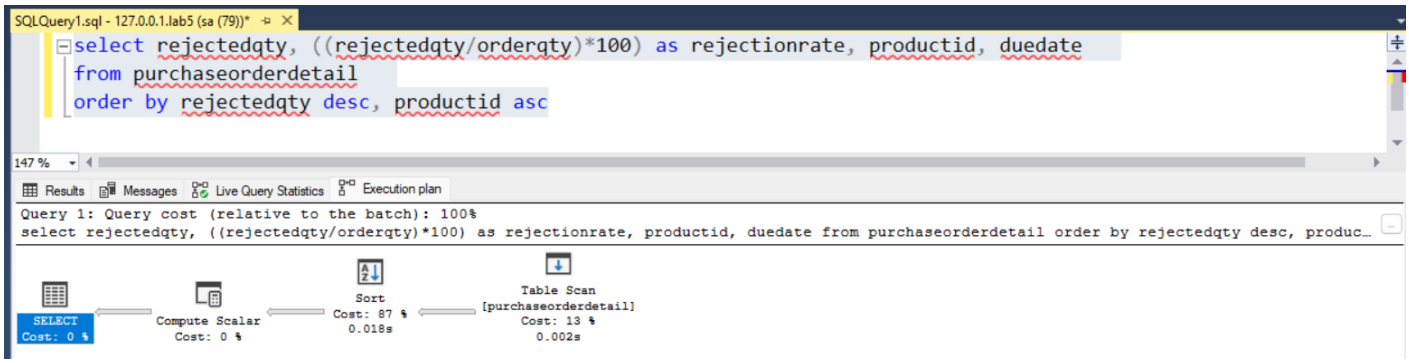
Skopiuj tabelę `PurchaseOrderDetail` do swojej bazy danych:

```
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?



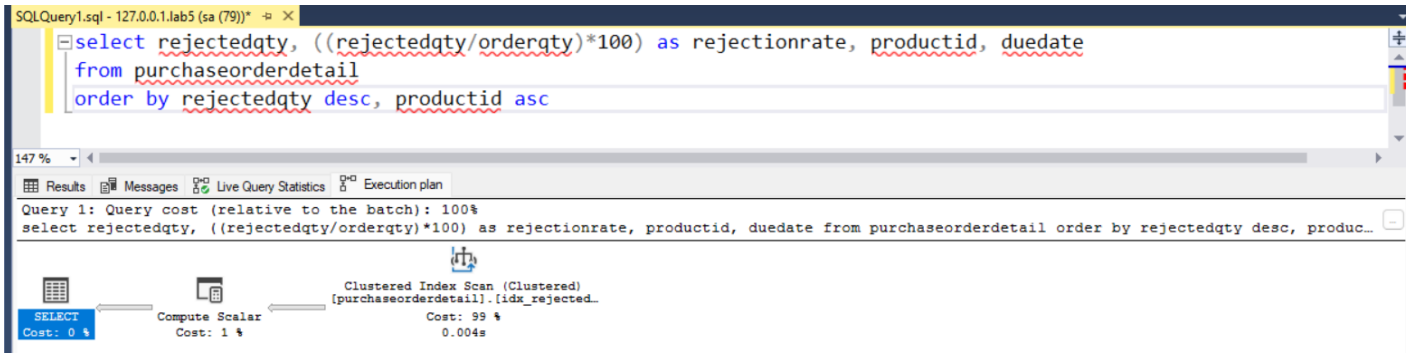
Sortowanie stanowi największy, bo aż 87 procentowy koszt zapytania i pochłania największą część czasu jego wykonania.

Jaki indeks można zastosować aby zoptymalizować koszt zapytania? Przygotuj polecenie tworzące index.

Zakładamy sklastrowany indeks obejmujący to na czym robimy `order by`, czyli de facto sortowanie.

```
CREATE CLUSTERED INDEX idx_rejectedqty_productid ON purchaseorderdetail (rejectedqty DESC, productid ASC);
```

Ponownie wykonaj analizę zapytania:



- Koszt zapytania przeniósł się prawie w całości na przeszukiwanie sklustrowanego indeksu, które jest ponad 4 razy szybsze (0.004s vs 0.018s) niż sortowanie
- Czas wykonania zapytania zmniejszył się pięciokrotnie (z 0.020s do 0.004s)

Zadanie 4

Celem zadania jest porównanie indeksów zawierających wszystkie kolumny oraz indeksów przechowujących dodatkowe dane (dane z kolumn).

Skopiuj tabelę `Address` do swojej bazy danych:

```
select * into address from  adventureworks2017.person.address
```

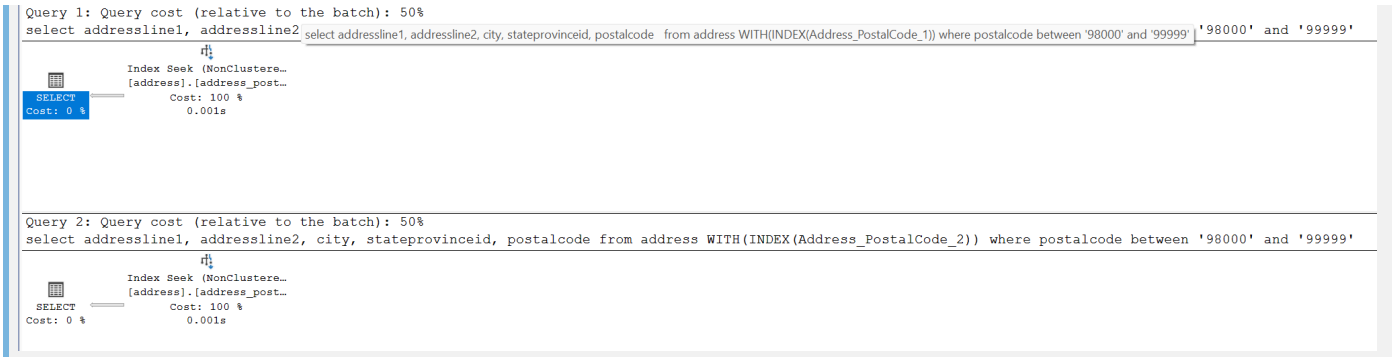
W tej części będziemy analizować następujące zapytanie:

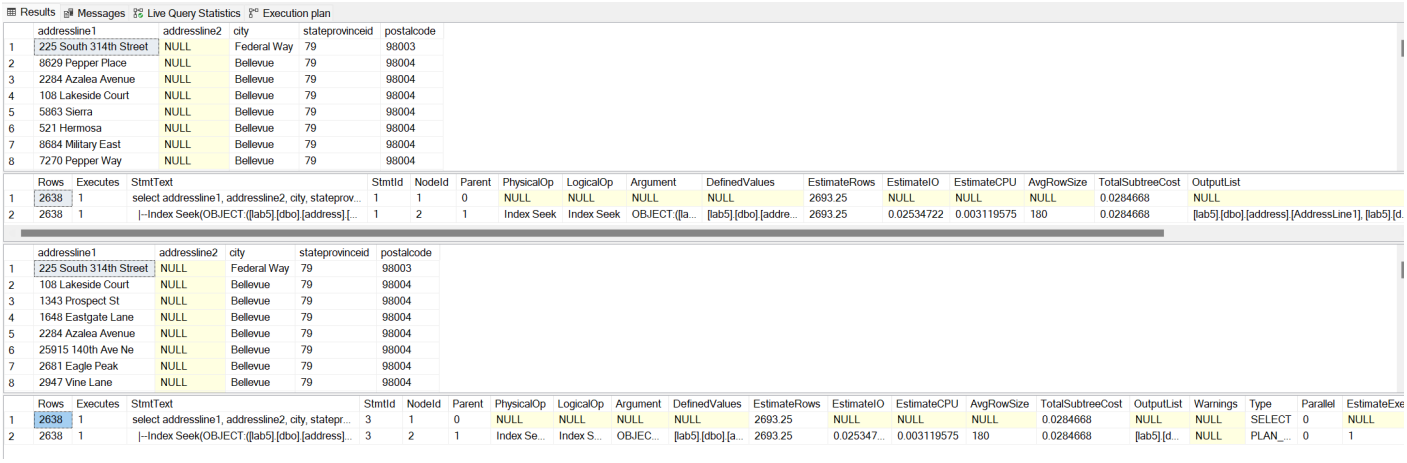
```
select addressline1, addressline2, city, stateprovinceid, postalcode
from address
where postalcode between n'98000' and n'99999'
```

```
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`:

Wyniki:





Results Messages Live Query Statistics Execution plan

	addressline1	addressline2	city	stateprovinceid	postalcode
1	225 South 314th Street	NULL	Federal Way	79	98003
2	8629 Pepper Place	NULL	Bellevue	79	98004
3	2284 Azalea Avenue	NULL	Bellevue	79	98004
4	108 Lakeside Court	NULL	Bellevue	79	98004
5	5863 Sierra	NULL	Bellevue	79	98004
6	521 Hermosa	NULL	Bellevue	79	98004
7	8684 Military East	NULL	Bellevue	79	98004
8	7270 Pepper Way	NULL	Bellevue	79	98004

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList
1	2638	1	select addressline1, addressline2, city, stateprov...	1	1	0	NULL	NULL	NULL	NULL	2693.25	NULL	NULL	NULL	0.0284668	NULL
2	2638	1	|--Index Seek(OBJECT:([lab5].[dbo].[address].[...	1	2	1	Index Seek	Index Seek	OBJECT:([la...	[lab5].[dbo].[addre...	2693.25	0.02534722	0.003119575	180	0.0284668	[lab5].[dbo].[address].[AddressLine1], [lab5].[d...

	addressline1	addressline2	city	stateprovinceid	postalcode
1	225 South 314th Street	NULL	Federal Way	79	98003
2	108 Lakeside Court	NULL	Bellevue	79	98004
3	1343 Prospect St	NULL	Bellevue	79	98004
4	1648 Eastgate Lane	NULL	Bellevue	79	98004
5	2284 Azalea Avenue	NULL	Bellevue	79	98004
6	25915 140th Ave Ne	NULL	Bellevue	79	98004
7	2681 Eagle Peak	NULL	Bellevue	79	98004
8	2947 Vine Lane	NULL	Bellevue	79	98004

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument	DefinedValues	EstimateRows	EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList	Warnings	Type	Parallel	EstimateExe
1	2638	1	select addressline1, addressline2, city, statepr...	3	1	0	NULL	NULL	NULL	NULL	2693.25	NULL	NULL	NULL	0.0284668	NULL	NULL	SELECT	0	NULL
2	2638	1	|--Index Seek(OBJECT:([lab5].[dbo].[address]...	3	2	1	Index Se...	Index S...	OBJEC...	[lab5].[dbo].[a...	2693.25	0.025347...	0.003119575	180	0.0284668	[lab5].[d...	NULL	PLAN_...	0	1

Odpowiedź:

Wygląda na to, że nie ma żadnej istotnej różnicy dla tego konkretnego zapytania przy wykorzystaniu `indeksu Address_PostalCode_1` i `Address_PostalCode_2`. Prawdopodobnie istnieją zapytania, w których różnice między tymi indeksami się będą pojawiać (np. zapytania nie tylko przeszukujące, ale i filtrujace po wszystkich kolumnach).

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 do indeksowania? Które kolumny na to wpływają?

> Wyniki:

	indexname	indexsizekb
1	address_postalcode_1	1784
2	address_postalcode_2	1808

Indeks `address_postalcode_2` jest nieznacznie większy niż `address_postalcode_1` (1808 KB w porównaniu do 1784 KB). To różnica w rozmiarze może wynikać z faktu, że `address_postalcode_2` jest indeksem, który zawiera wszystkie kolumny w kluczu indeksu, podczas gdy `address_postalcode_1` jest indeksem na `postalcode` przechowujących dodatkowe dane (dane z kolumn `addressline1, addressline2, city, stateprovinceid`). W tym ostatnim przypadku w strukturze drzewa indeksu te dodatkowe dane mogę znajdować się wyłącznie na liściach. Wprzypadku `address_postalcode_2` dane z dodatkowych kolumn mogą znajdować się na każdym poziomie drzewa.

Zadanie 5 – Indeksy z filtrami

Celem zadania jest poznanie indeksów z filtrami.

Skopiuj tabelę `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:

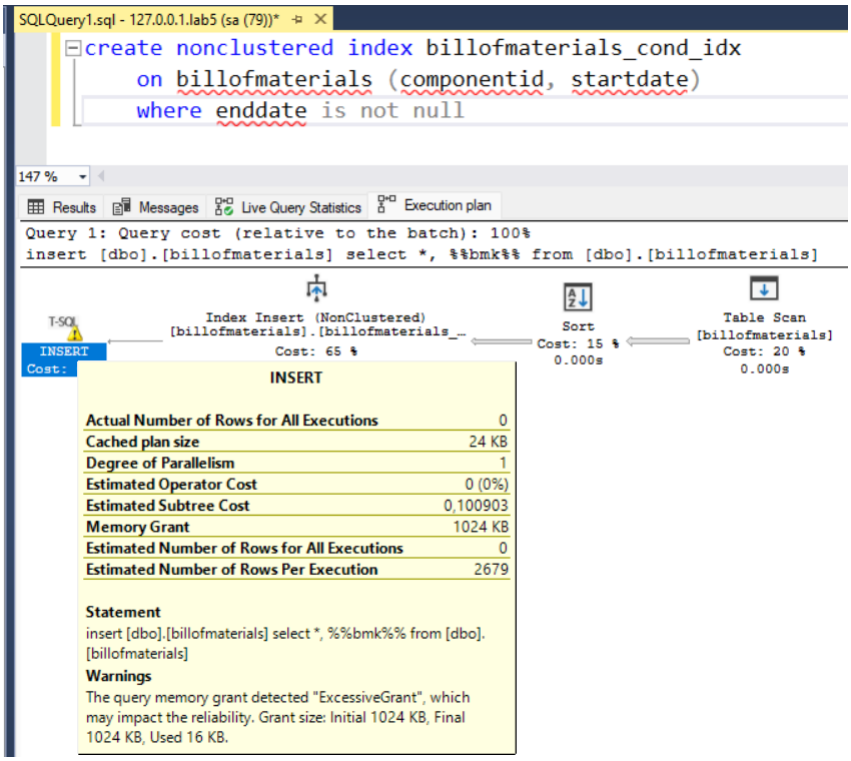
```
create nonclustered index billofmaterials_cond_idx
    on billofmaterials (componentid, startdate)
    where enddate is not null
```

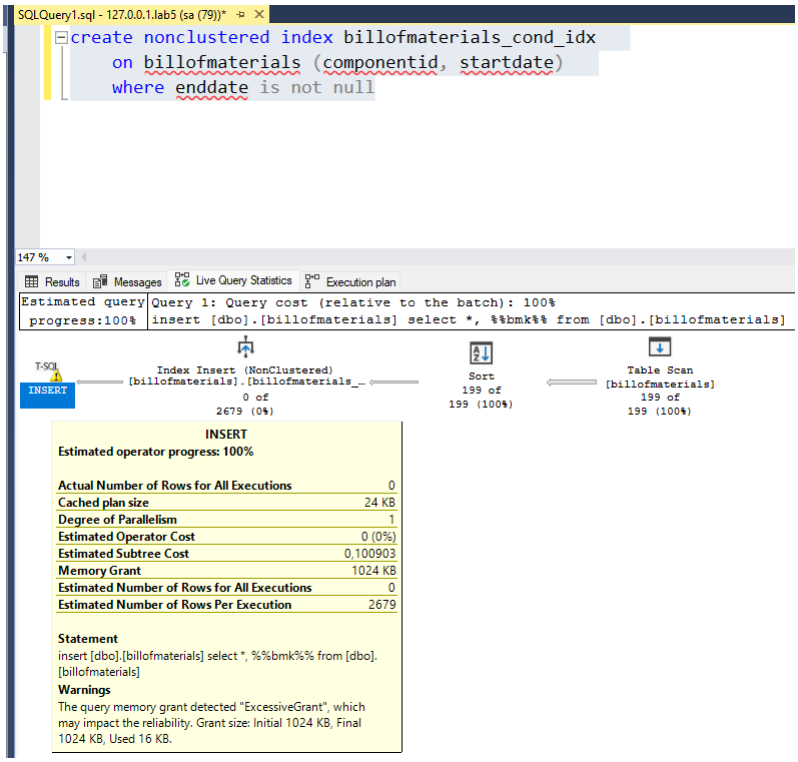
Sprawdź czy działa.

Przeanalizuj plan dla poniższego zapytania:

Czy indeks został użyty? Dlaczego?

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





Otrzymaliśmy komunikat ostrzegawczy dotyczący alokacji pamięci w związku z operacją `INSERT`. Okazuje się, że indeks `billofmaterials_cond_idx` nie jest wystarczająco selektywny dla operacji `INSERT`.

Innymi słowy, liczba wierszy spełniających warunek `WHERE enddate IS NOT NULL` jest wystarczająco duża (100%), aby silnik bazy danych uznał skanowanie tabeli za bardziej opłacalne niż korzystanie z tego indeksu.

Punktacja:

zadanie	pkt
1	2
2	2
3	2
4	2
5	2
razem	10