Indeksy, optymalizator – lab4

Imię i Nazwisko: Bartłomiej Jamiołkowski, Ada Bodziony

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

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 Managment Studio.

Stwórz swoją bazę danych o nazwie **XYZ**.

```
CREATE DATABASE XYZ
GO

USE XYZ
GO
```

Wykonaj poniższy skrypt, aby przygotować dane:

```
SELECT * INTO [SalesOrderHeader]
FROM [AdventureWorks2017].Sales.[SalesOrderHeader]
GO

SELECT * INTO [SalesOrderDetail]
FROM [AdventureWorks2017].Sales.[SalesOrderDetail]
GO
```

Dokumentacja/Literatura

Celem tej części ćwiczenia jest zapoznanie się z planami wykonania zapytań (execution plans) oraz narzędziem do automatycznego generowania indeksów.

Przydatne materiały/dokumentacja. Proszę zapoznać się z dokumentacją:

- https://docs.microsoft.com/en-us/sql/tools/dta/tutorial-database-engine-tuning-advisor
- https://docs.microsoft.com/en-us/sql/relational-databases/performance/start-and-use-the-database-engine-tuning-advisor
- https://www.simple-talk.com/sql/performance/index-selection-and-the-query-optimizer

Ikonki używane w graficznej prezentacji planu zapytania opisane są tutaj:

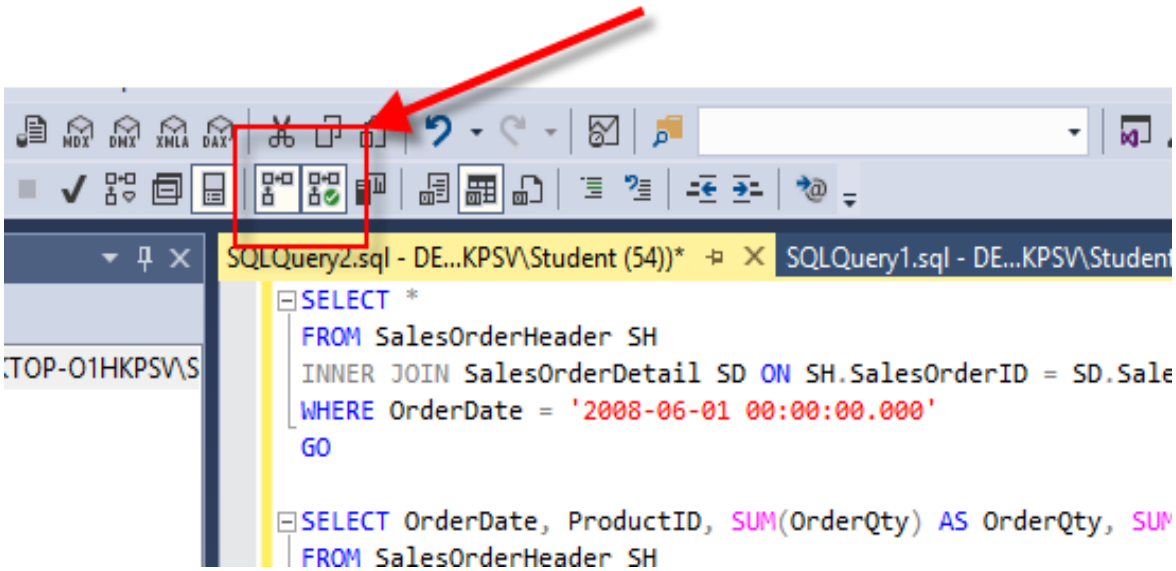
- https://docs.microsoft.com/en-us/sql/relational-databases/showplan-logical-and-physical-operators-reference

Zadanie 1 - Obserwacja

Wpisz do MSSQL Managment Studio (na razie nie wykonuj tych zapytań):

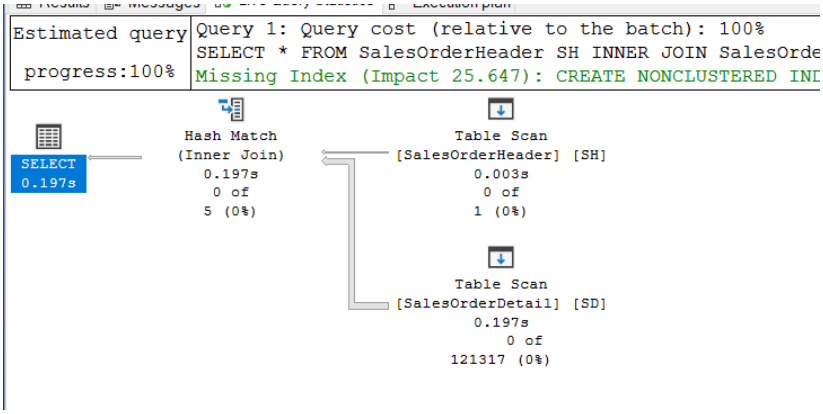
```
-- zapytanie 1
SELECT *
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
WHERE OrderDate = '2008-06-01 00:00:00.000'
GO

-- zapytanie 2
SELECT OrderDate, ProductID, SUM(OrderQty) AS OrderQty,
SUM(UnitPriceDiscount) AS UnitPriceDiscount, SUM(LineTotal)
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
GROUP BY OrderDate, ProductID
HAVING SUM(OrderQty) >= 100
GO

-- zapytanie 3
SELECT SalesOrderNumber, PurchaseOrderNumber, DueDate, ShipDate
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
WHERE OrderDate IN ('2008-06-01','2008-06-02', '2008-06-03', '2008-06-
04', '2008-06-05')
GO

-- zapytanie 4
SELECT SH.SalesOrderID, SalesOrderNumber, PurchaseOrderNumber, DueDate,
ShipDate
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
WHERE CarrierTrackingNumber IN ('EF67-4713-BD', '6C08-4C4C-B8')
ORDER BY SH.SalesOrderID
GO
```

Włącz dwie opcje: Include **Actual Execution Plan** oraz **Include Live Query Statistics**:



Teraz wykonaj poszczególne zapytania (najlepiej każde analizuj oddzielnie). Co można o nich powiedzieć? Co sprawdzają? Jak można je zoptymalizować?
(Hint: aby wykonać tylko fragment kodu SQL znajdującego się w edytorze, zaznacz go i naciśnij F5)

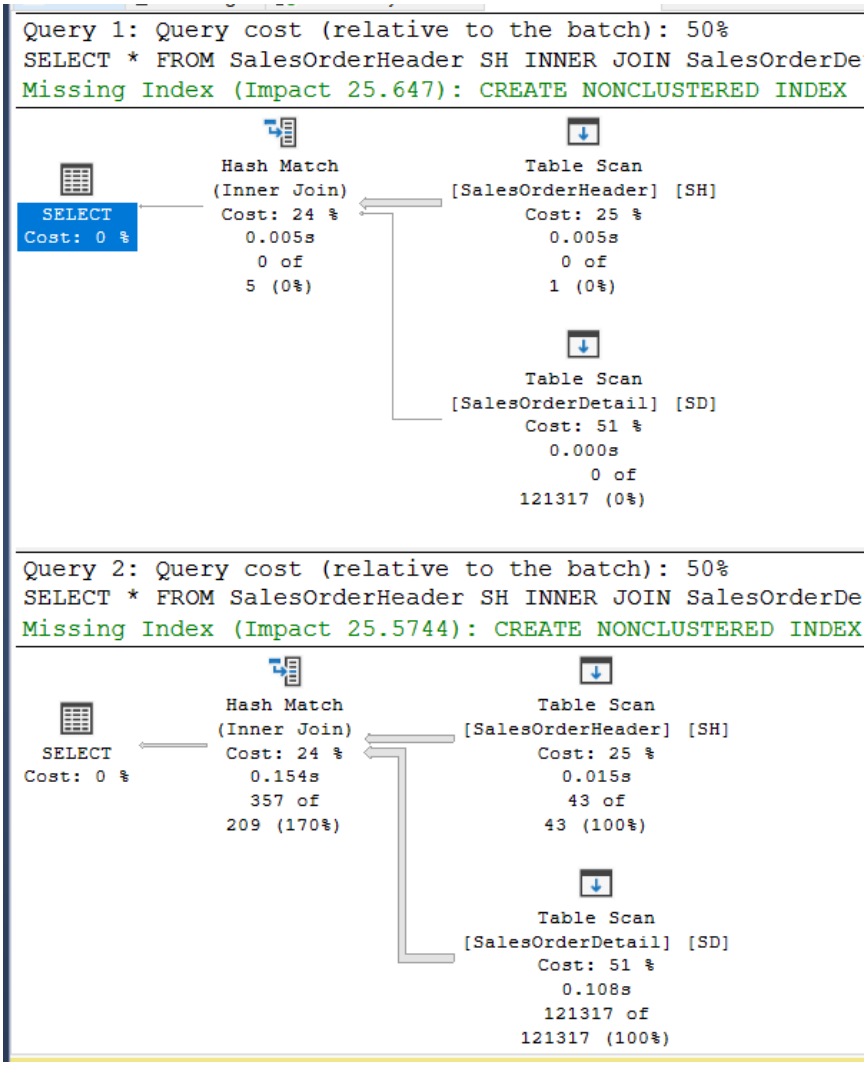
Polecenie 1)



Wykonanie ze zmienionym warunkiem:

```
WHERE OrderDate = '2011-05-31 00:00:00.000'
```

Porównanie z pierwotną wersją:

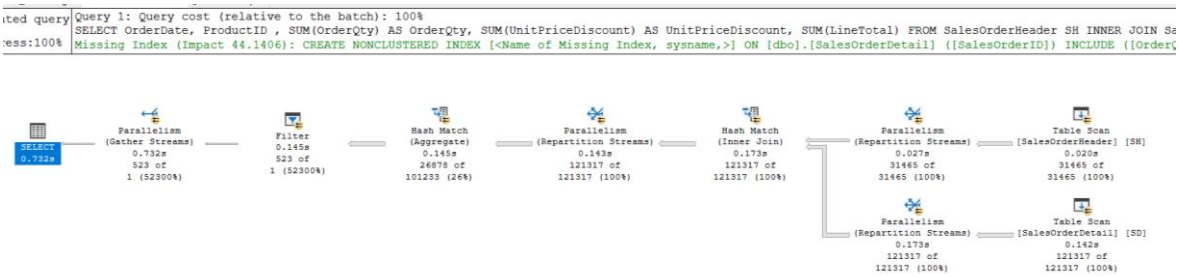


Pierwszy wykres dotyczy 1 zapytania, drugi wykres zapytania z podmienionym warunkiem. Widać że ogólna struktura wykonania zapytania nie uległa zmianie, jednak to że skan tabeli SalesOrderHeader zwraca więcej rekordów (a nie jak w przypadku pierwotnej wersji 0) automatycznie wykonanie HashMatch (inner join) zajmuje stosunkowo więcej czasu. Optymalizacją, sugerowaną przez SMSS byłoby dodanie nieklastrowego indeksu na kolumnie OrderDate w tabeli SalesOrderHeader.

Actual Number of Rows for All Exec	0
Cached plan size	120 KB
CardinalityEstimationModelVersion	150
CompileCPU	15
CompileMemory	352
CompileTime	23
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number of Rows for All E	0
Estimated Number of Rows Per Exec	4,85501
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2,45304
Memory Grant	1288 KB

Polecenie 2)

Estimated SubTree Cost = 4,02849

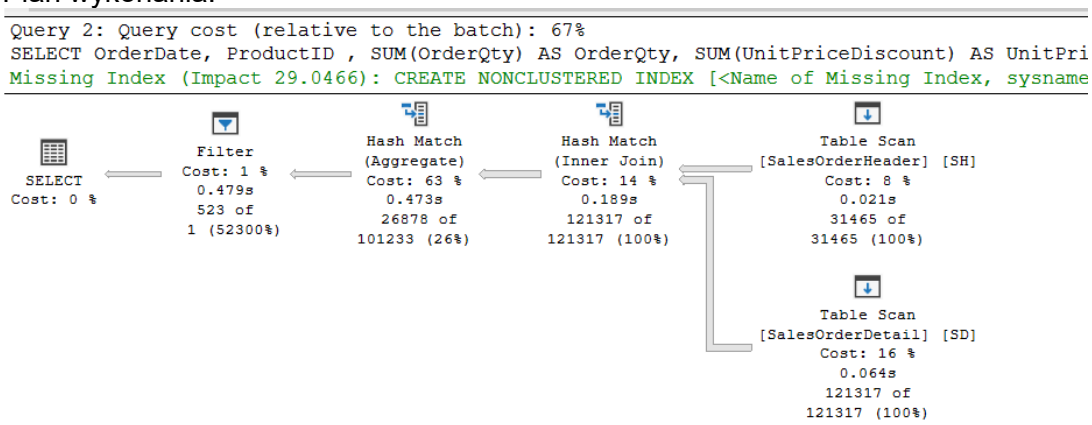


Zapytanie wykonało się z wykorzystaniem zrównoleglenia w celu poprawy zapytań, jest to związane z parametrem konfigurowalnym na poziomie ustawień serwera – cost threshold for parallelism, który domyślnie jest ustawiony na 5. W przypadku pierwszego zapytania Estimated SubTree Cost jest < 5 więc zapytanie zostało wykonane bez zrównoleglenia.

W celu porównania do warunku czy koszt zapytania jest na tyle duży musimy porównywać sekwencyjne wykonanie zapytania i jego koszt. Dla przykładu zapytanie nr 2) zmodyfikowane tak by wykonało się sekwencyjnie:

```
SELECT OrderDate, ProductID , SUM(OrderQty) AS OrderQty, SUM(UnitPriceDiscount) AS
UnitPriceDiscount, SUM(LineTotal)
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
GROUP BY OrderDate, ProductID
HAVING SUM(OrderQty) >= 100
OPTION (MAXDOP 1)
```

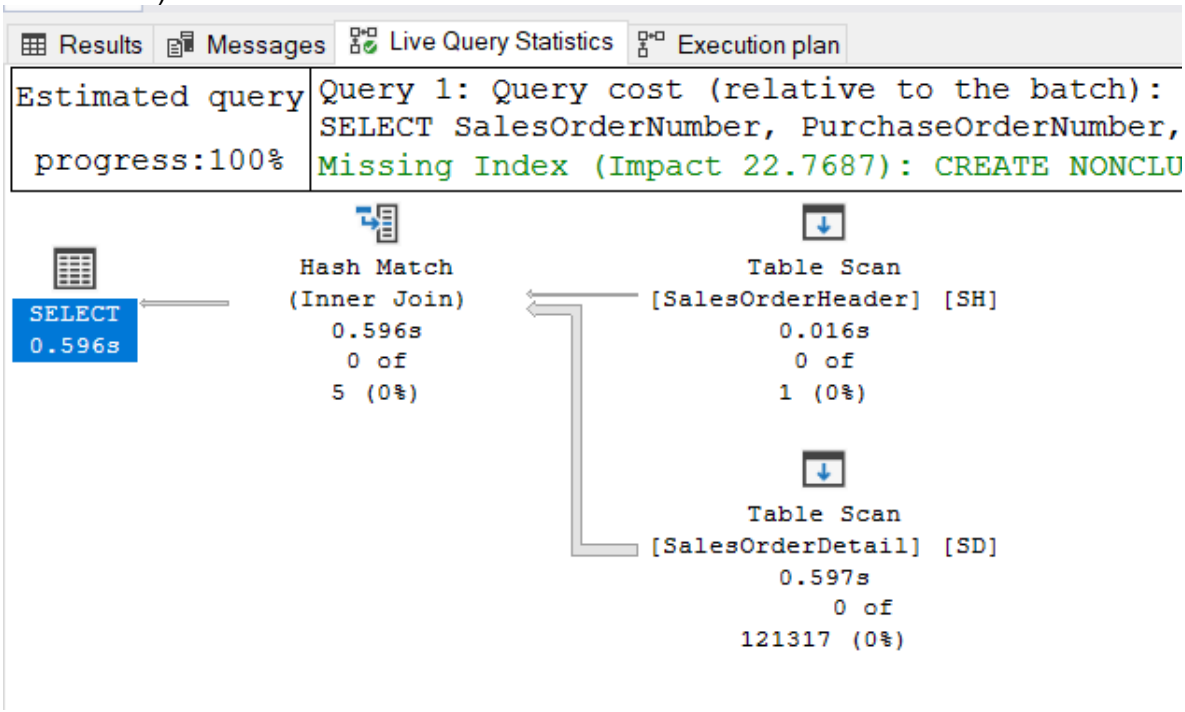
Plan wykonania:



Estimated SubTree Cost: 8,03294
Czyli widzimy już, że początkowo był on większy niż 5. A więc serwer zdecydował o zrównolegleniu operacji, a więc i obniżeniu kosztu o połowę.

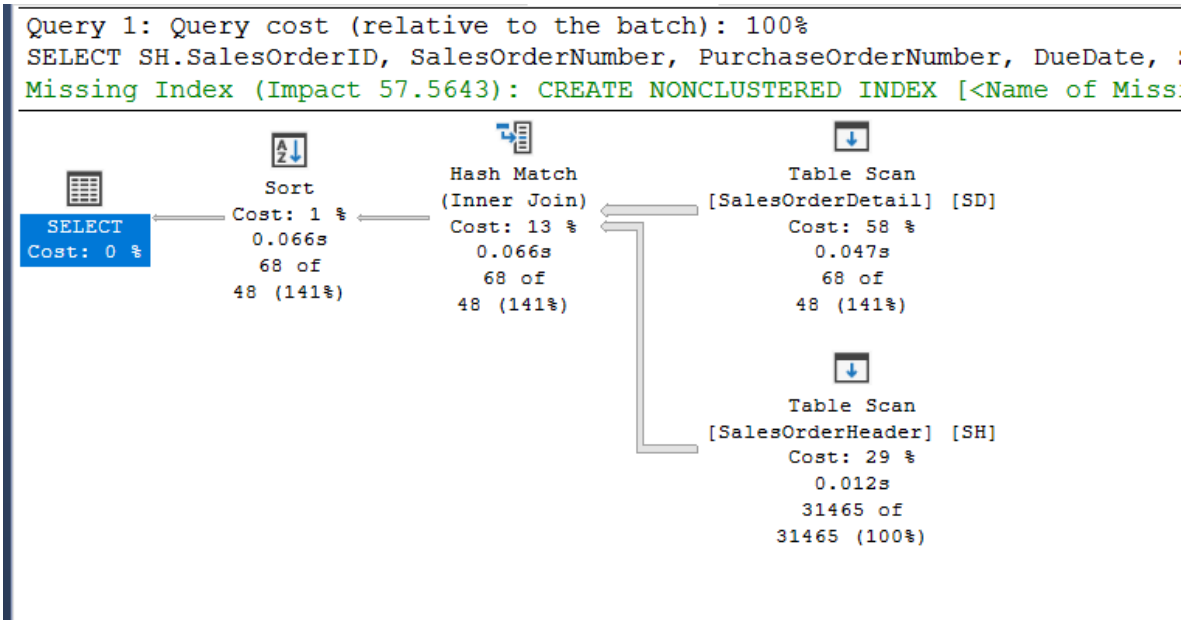
Dodatkowo SMSS sugeruje stworzenie indeksu zawierającego oprócz pierwszego poziomu – SalesOrderId, także po INCLUDED: OrderQty, ProductId, UnitPriceDiscount, LineTotal – drugi poziom . Dzięki , że oprócz kolumn które są wykorzystywane w oprecjach np. join, agregacja, w indeksie uwzględnić również kolumny, są zawarte w select w wyniku.

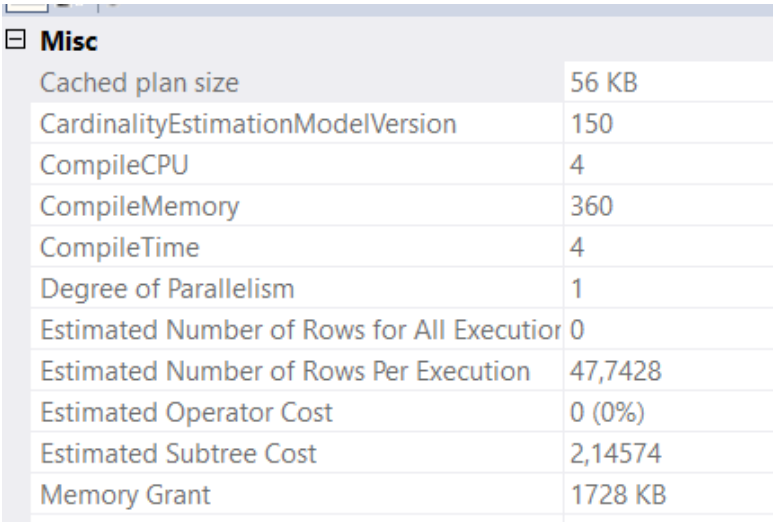
Polecenie 3)



Estimated SubTree Cost: 2,49383
Tutaj mamy podobne zapytanie jak 1), jedynie zwiększamy zakres dat branych pod uwagę i wyniku zwracamy wartośći z większej ilości kolumn. Znów 1 skanowanie tabeli zwraca nam zbiór pusty, a więc join przechodzi dość szybko.
Znów mamy również sugestię założenia nieklastrowego indeksu opartego o OrderDate i mającego zawarte w INCLUDE:
`SalesOrderNumber, PurchaseOrderNumber, DueDate, ShipDate, SalesOrderId ( również kolumny występujące jedynie w select są zawarte w propozycji – możliwe że chodzi o to że pozwoli nam to na sięgnięcie jedynie do indeksu, bez konieczności przechodzenia po tabeli)`
Co pozwoliło by serwerowi na szybsze wykonanie zapytania.

Polecenie 4)





Misc	
Cached plan size	56 KB
CardinalityEstimationModelVersion	150
CompileCPU	4
CompileMemory	360
CompileTime	4
Degree of Parallelism	1
Estimated Number of Rows for All Executior	0
Estimated Number of Rows Per Execution	47,7428
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2,14574
Memory Grant	1728 KB

Estimated SubTree Cost: 2,14574 (całe zapytanie)
Koszt zapytania jest na tyle mały że nie inicjuje zrównoleglenia wykonania zapytania.

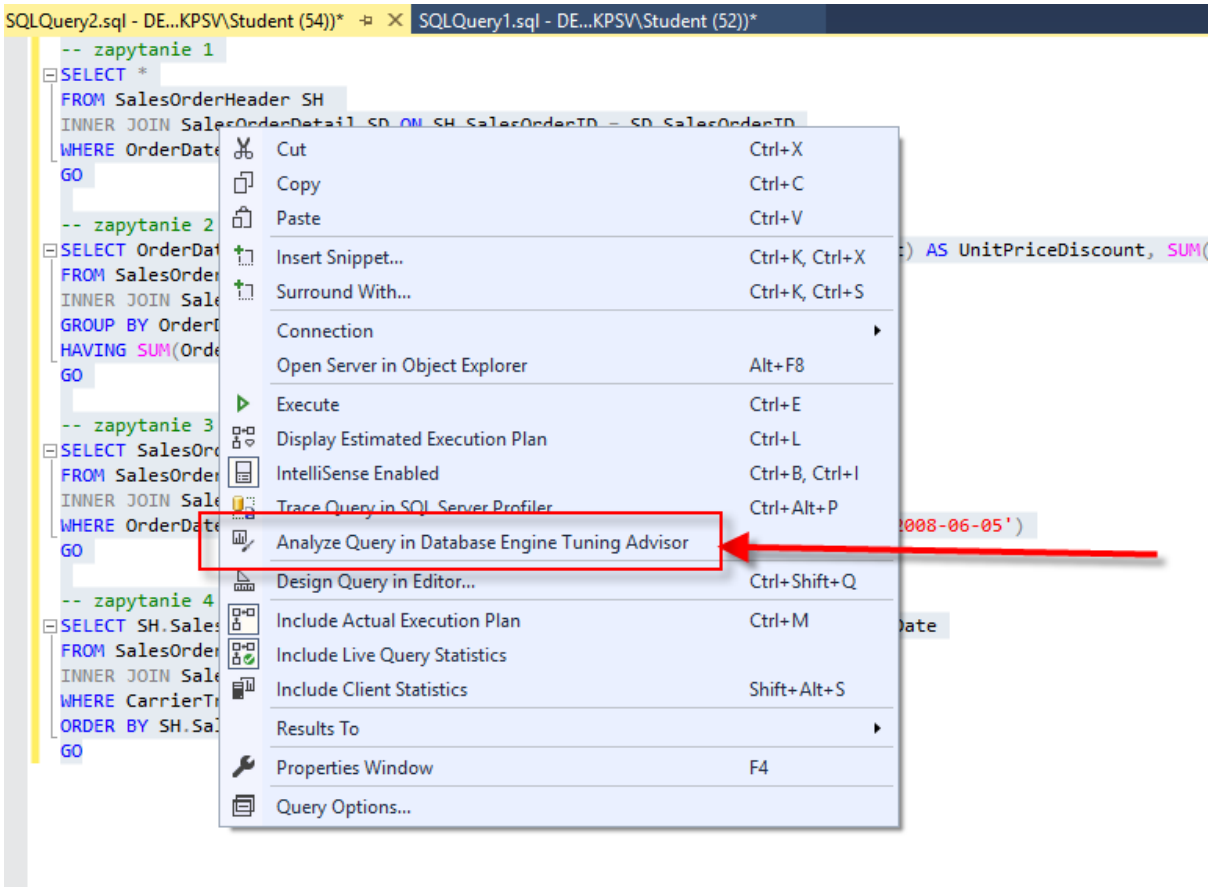
Większą część zapytania zajmuje wykonanie skanowania tabeli SalesOrderDetails z uwzględnieniem warunku w where który właśnie dotyczy tej tabeli – koszt : 1,24636 równy ponad połowie kosztu całego zapytania.

Proponowany indeks nieklastrowy:

```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[SalesOrderDetail] ([CarrierTrackingNumber])
INCLUDE ([SalesOrderID])
```

Zadanie 2 - Optymalizacja

Zaznacz **wszystkie zapytania**, i uruchom je w **Database Engine Tuning Advisor**:



Sprawdź zakładkę **Tuning Options**, co tam można skonfigurować?

Po wybraniu zakładki Tuning Options ukazują się następujące sekcje:

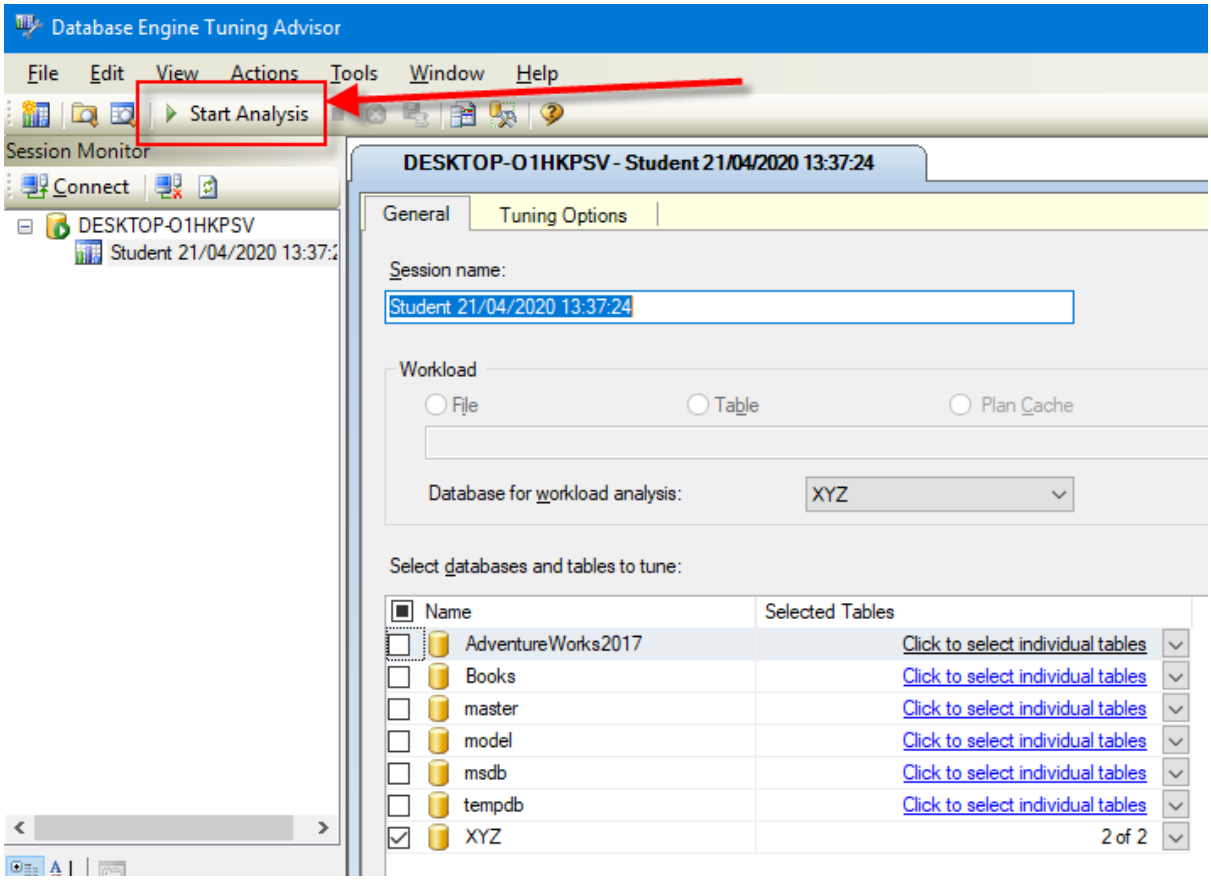
a) Limit tuning time – konfiguracja maksymalnego czasu, jaki Database Engine Tuning Advisor potrzebuje na dostrojenie obciążenia pracą. Domyślnie ten czas jest ustawiony na 1 godzinę.

b) Physical Design Structures (PDS) in to use in database – konfiguracja rodzaju fizycznej struktury projektu, którą można wykorzystać podczas strojenia. Dokładniej chodzi o indeksy lub widoki indeksowane. Dostępne opcje jednokrotnego wyboru to: Indexes and indexed views, Indexed views, Include filtered indexes, Indexes, Nonclustered indexes, Recommend columnstore indexes i Evaluate utilization of existing PDS only.

c) Partitioning strategy to employ – konfiguracja strategii partycjonowania tabeli, którą Database Engine Tuning Advisor ma uwzględnić podczas analizy. Dostępne opcje jednokrotnego wyboru to: No partitioning, Aligned partitioning i Full partitioning.

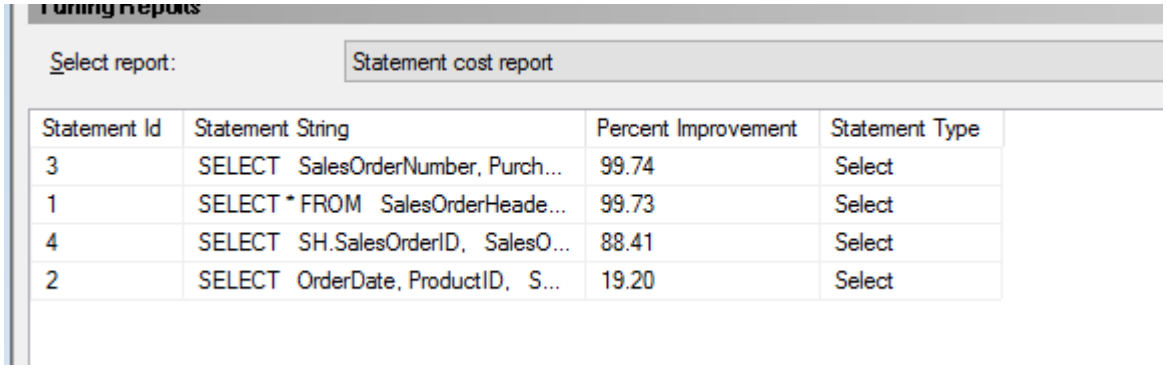
d) Physical Design Structures (PDS) to keep in database – określenie, które obiekty mają być przechowywane w bazie danych. Ta opcja może pomóc zapewnić, że zalecenia dotyczące dostrajania nie wpłyną negatywnie na dostrajanie, które zostało wcześniej przetestowane i wdrożone. Dostępne opcje przechowywania elementów PDS w bazie danych to: Do not keep any existing PDSs, Keep all existing PDSs (default option), Keep aligned partitioning, Keep indexes only, Keep clustered indexes only.

Użyj **Start Analysis**:

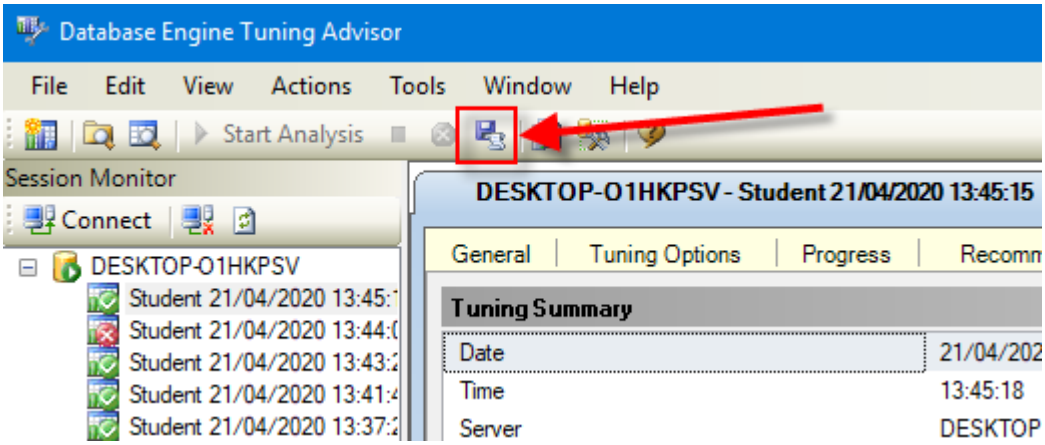


Zaobserwuj wyniki w **Recommendations**.

Przejdź do zakładki **Reports**. Sprawdź poszczególne raporty. Główną uwagę zwróć na koszty i ich poprawę:



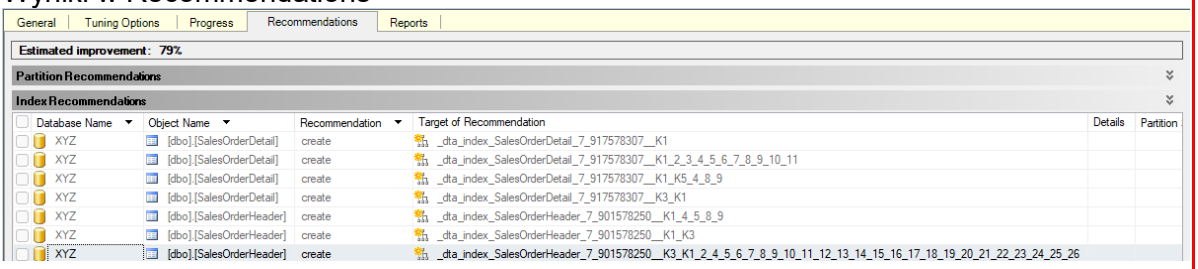
Zapisz poszczególne rekomendacje:



Uruchom zapisany skrypt w Management Studio.

Opisz, dlaczego dane indeksy zostały zaproponowane do zapytań:

Wyniki w Recommendations



Raporty w zakładce Reports

Database(s) to tune	[XYZ]
Workload	Inline
Maximum tuning time	17 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	79.65
Maximum space for recommendation (MB)	54
Space used currently (MB)	20
Space used by recommendation (MB)	49
Number of events in workload	4
Number of events tuned	4
Number of statements tuned	4
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	7

Tuning Reports

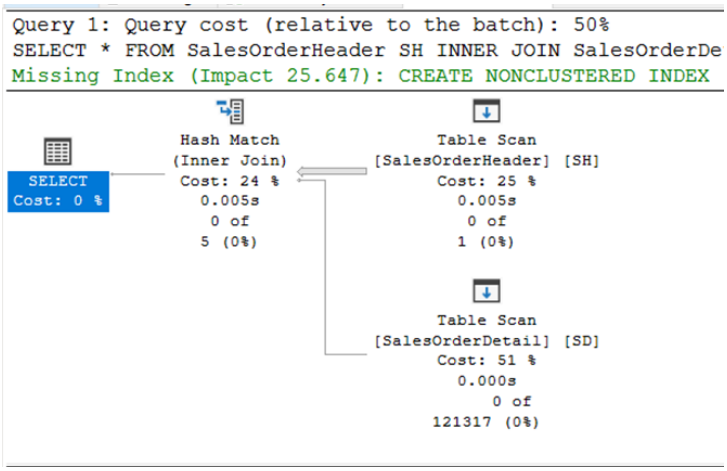
Select report: Statement cost report

Statement Id	Statement String	Percent Improvement	Statement Type
3	-- zapytanie 3 SELECT SalesOrderNu...	99.74	Select
1	-- zapytanie 1 SELECT * FROM Sales...	99.73	Select
4	-- zapytanie 4 SELECT SH.SalesOrde...	92.79	Select
2	-- zapytanie 2 SELECT OrderDate, Pr...	37.81	Select

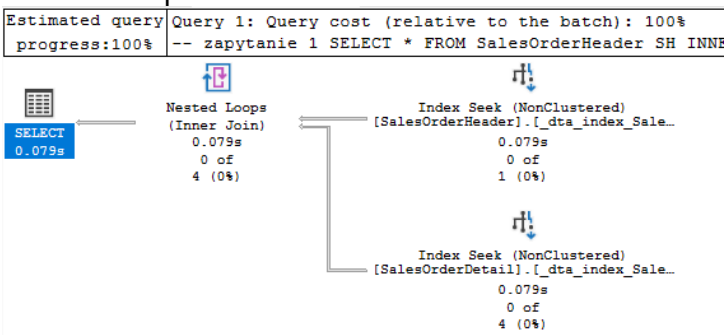
Indeksy zostały zaproponowane do zapytań w celu przyspieszania wykonywania zapytań poprzez szybkie znajdowanie rekordów bez konieczności skanowania tabel. Nie są to convering indexes, tylko nonclustered indexes, ponieważ zawierają tylko podzbiór kolumn tabeli.

Sprawdź jak zmieniły się Execution Plany. Opisz zmiany:

Polecenie 1 przed zmianą

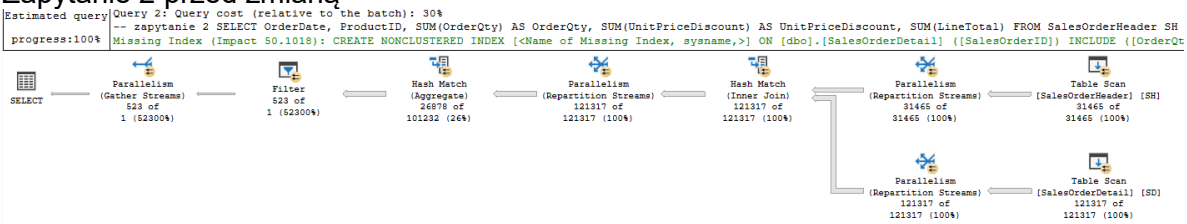


Polecenie 1 po zmianie

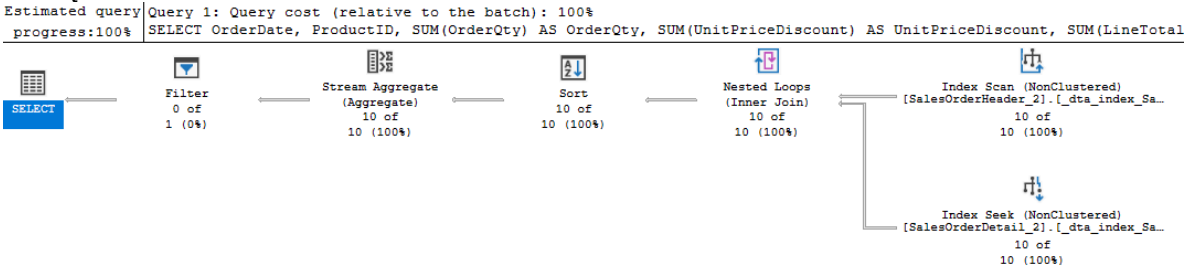


Opis zmian:
Po zastosowaniu zmian, w Execution Plan zniknęła informacja Missing Index (Impact 25.647). Oznacza to, że dodano brakujący indeks, dzięki czemu zapytanie uzyskało 25.647 średniej procentowej korzyści. Oprócz tego operator Hash Match został zastąpiony operatorem Nested Loops. Podobnie operatory Table Scan zostały zastąpione operatorami Index Seek. Pierwotnie większą część zapytania zajmowało wykonanie skanowania tabeli SalesOrderDetail (51 %). Od teraz wykonanie wyszukiwania indeksów w tabelach SalesOrderDetail i SalesOrderHeader zajmuje po 50% zapytania.

Zapytanie 2 przed zmianą



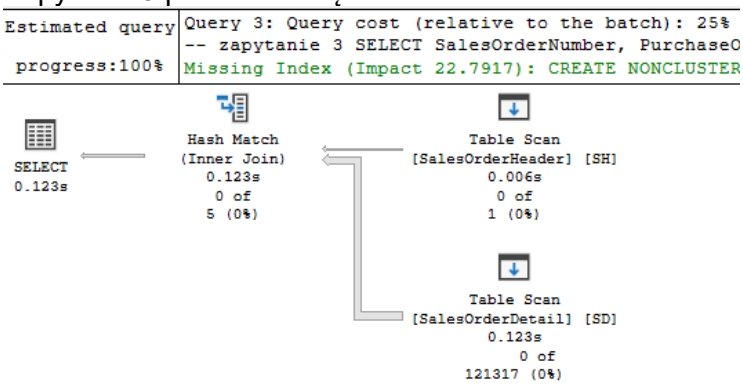
Polecenie 2 po zmianie – generowało się zbyt długo dlatego utworzono tabelę z mniejszą liczbą rekordów i uruchomiono Execution Plan



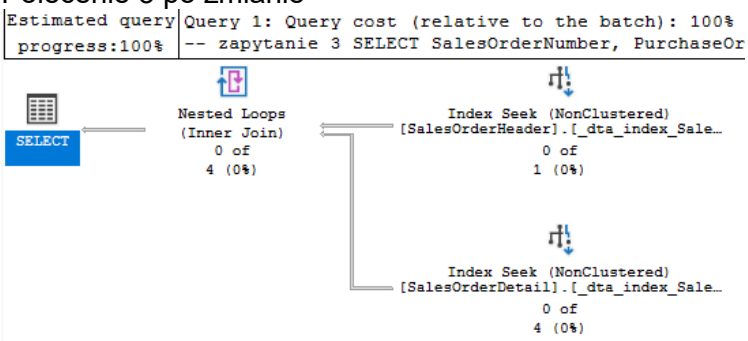
Opis zmian:
Po zastosowaniu zmian, w Execution Plan zniknęła informacja Missing Index (Impact

50.1018). Oznacza to, że dodano brakujący indeks, dzięki czemu zapytanie uzyskało 50.1018 średniej procentowej korzyści. Oprócz tego operator Hash Match został zastąpiony operatorem Nested Loops. Podobnie operatory Table Scan zostały zastąpione operatorami Index Seek oraz Index Scan. Od teraz wyszukiwanie indeksów w tabeli SalesOrderDetail zajmuje 24% zapytania, sortowanie wyników Nested Loops zajmuje 59% zapytania, a wyszukiwanie indeksów w tabeli SalesOrderHeader zajmuje 17% zapytania.

Zapytanie 3 przed zmianą

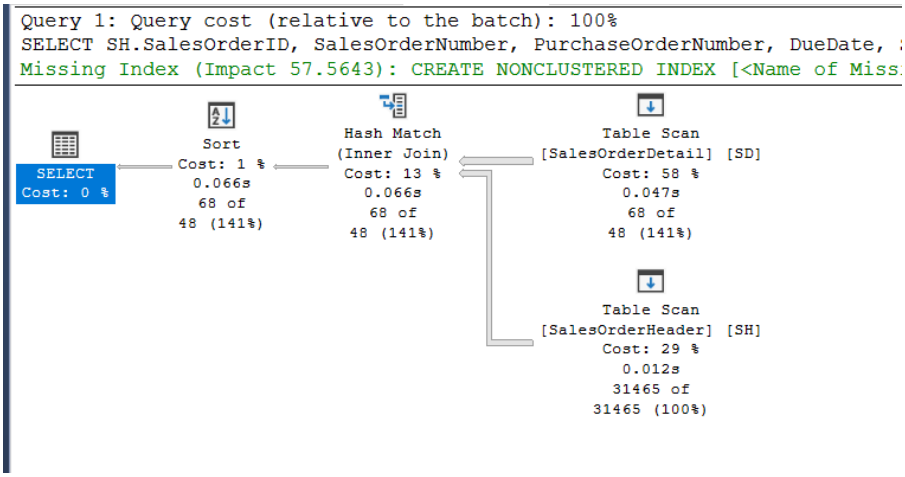


Polecenie 3 po zmianie

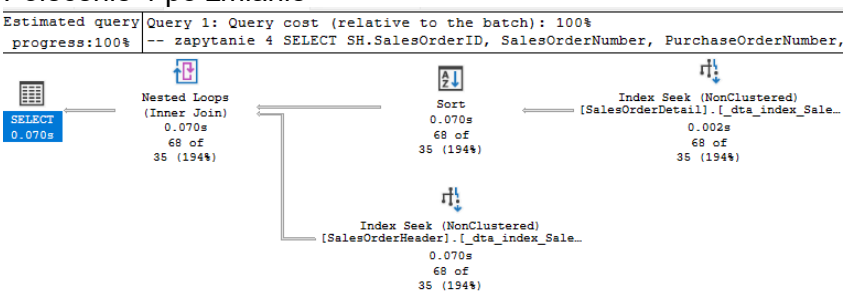


Opis zmian:
Po zastosowaniu zmian, w Execution Plan zniknęła informacja Missing Index (Impact 22.7917). Oznacza to, że dodano brakujący indeks, dzięki czemu zapytanie uzyskało 22.7917 średniej procentowej korzyści. Oprócz tego operator Hash Match został zastąpiony operatorem Nested Loops. Podobnie operatory Table Scan zostały zastąpione operatorami Index Seek. Od teraz wykonanie wyszukiwania indeksów w tabelach SalesOrderDetail i SalesOrderHeader zajmuje po 50% zapytania.

Zapytanie 4 przed zmianą



Polecenie 4 po zmianie



Opis zmian:
Po zastosowaniu zmian, w Execution Plan zniknęła informacja Missing Index (Impact 57.5643). Oznacza to, że dodano brakujący indeks, dzięki czemu zapytanie uzyskało 57.5643 średniej procentowej korzyści. Oprócz tego operator Hash Match został zastąpiony operatorem Nested Loops. Operator Sort nie jest już stosowany dla wyników Hash Match, tylko dla wyników Index Seek tabeli SalesOrderDetail. Operatory Table Scan zostały zastąpione operatorami Index Seek. Pierwotnie skanowanie tabeli SalesOrderDetail zajmowało 58% zapytania, a skanowanie tabeli SalesOrderHeader zajmowało 29% zapytania. Od teraz wyszukiwanie indeksów w tabeli SalesOrderDetail zajmuje 3% zapytania, sortowanie wyników tego wyszukiwania zajmuje 10% zapytania, a wyszukiwanie indeksów w tabeli SalesOrderHeader zajmuje 87% zapytania.

Zadanie 3 - Kontrola “zdrowia” indeksu

Dokumentacja/Literatura

Celem kolejnego zadania jest zapoznanie się z możliwością administracji i kontroli indeksów. Na temat wewnętrznej struktury indeksów można przeczytać tutaj:

- https://technet.microsoft.com/en-us/library/2007.03.sqlindex.aspx
- https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql
- https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql
- https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-indexes-transact-sql

Sprawdź jakie informacje można wyczytać ze statystyk indeksu:

```
SELECT *
FROM sys.dm_db_index_physical_stats (DB_ID('AdventureWorks2017')
,OBJECT_ID('HumanResources.Employee')
,NULL -- NULL to view all indexes; otherwise, input index number
,NULL -- NULL to view all partitions of an index
,'DETAILED') -- We want all information
```

Jakie są według Ciebie najważniejsze pola?

index_id: Identyfikator indeksu, dla którego zostały pobrane statystyki.
index_type_desc: Opisuje typ indeksu (np. 'CLUSTERED', 'NONCLUSTERED').
avg_fragmentation_in_percent: Procentowa fragmentacja indeksu.
page_count: Liczba stron zajętych przez indeks.
avg_page_space_used_in_percent: Średnie wykorzystanie miejsca na stronie przez indeks.

Sprawdź, które indeksy w bazie danych wymagają reorganizacji:

```
USE AdventureWorks2017

SELECT OBJECT_NAME([object_id]) AS 'Table Name',
index_id AS 'Index ID'
FROM sys.dm_db_index_physical_stats (DB_ID('AdventureWorks2017')
,NULL -- NULL to view all tables
,NULL -- NULL to view all indexes; otherwise, input index number
,NULL -- NULL to view all partitions of an index
,'DETAILED') --We want all information
WHERE ((avg_fragmentation_in_percent > 10
AND avg_fragmentation_in_percent < 15) -- Logical fragmentation
OR (avg_page_space_used_in_percent < 75
AND avg_page_space_used_in_percent > 60)) --Page density
AND page_count > 8 -- We do not want indexes less than 1 extent in size
AND index_id NOT IN (0) --Only clustered and nonclustered indexes
```

Zrzut ekranu/komentarz:

	Table Name	Index ID
1	JobCandidate	1
2	ProductModel	1
3	BillOfMaterials	2
4	WorkOrder	3
5	WorkOrderRouting	2

Zapytanie sprawdza wartość:
- avg_fragmentation_in_percent – zbyt duża wartość sugeruje zbytnie pofragmentowanie indeksu. Tutaj bierzemy z zakresu 10-15%
- avg_page_space_used_in_percent – wartości przekraczające zakres mogą sugerować przeciążenie indeksu. A zbyt małe wartości wskazują na nieefektywne wykorzystanie pamięci. Tutaj bierzemy z zakresu 60-75%
- page_count – teoretycznie im mniej tym lepiej, indeks bardziej zwarty, jednak trzeba wziąć pod uwagę ilość danych w danej tabeli. Tutaj > 8.

Sprawdź, które indeksy w bazie danych wymagają przebudowy:

```
USE AdventureWorks2017

SELECT OBJECT_NAME([object_id]) AS 'Table Name',
index_id AS 'Index ID'
FROM sys.dm_db_index_physical_stats (DB_ID('AdventureWorks2017')
,NULL -- NULL to view all tables
,NULL -- NULL to view all indexes; otherwise, input index number
,NULL -- NULL to view all partitions of an index
,'DETAILED') --We want all information
WHERE ((avg_fragmentation_in_percent > 15) -- Logical fragmentation
OR (avg_page_space_used_in_percent < 60)) --Page density
AND page_count > 8 -- We do not want indexes less than 1 extent in size
AND index_id NOT IN (0) --Only clustered and nonclustered indexes
```

Zrzut ekranu/komentarz:

	Table Name	Index ID
1	Person	256002
2	Person	256003
3	Person	256004

Zapytanie sprawdza wartość:
- avg_fragmentation_in_percent Tutaj bierzemy wartości większe niż 15% - czyli bardziej pofragmentowane niż w poprzednim przypadku
- avg_page_space_used_in_percent – wartości przekraczające zakres mogą sugerować przeciążenie indeksu. Tutaj bierzemy z zakresu > 60% - czyli mniej efektywne wykorzystanie niż w poprzednim przypadku
- page_count –tak samo jak poprzednio > 8.

Czym się różni przebudowa indeksu od reorganizacji?
(Podpowiedź: http://blog.plik.pl/2014/12/defragmentacja-indeksow-ms-sql.html)

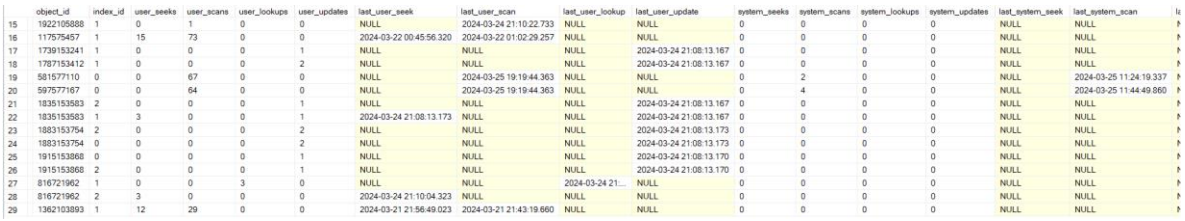
Przebudowa indeksu polega na posortowaniu danych na nowo, a struktura indeksu jest odtwarzana. Dzięki temu usuwamy fragmentację. A w przypadku reorganizacji jest to optymalizacja, która mniej inwazyjnie ingeruje w strukturę indeksu. Nie musi odświeżać

informacji zbieranych przez dbsm, służących do optymalizacji zapytań – w przypadku przebudowy są one odświeżane.

Sprawdź co przechowuje tabela sys.dm_db_index_usage_stats:

Wynik zapytania:

```
select *
from sys.dm_db_index_usage_stats
```



	object_id	index_id	user_seeks	user_scans	user_lookups	user_updates	last_user_seek	last_user_scan	last_user_lookup	last_user_update	system_seeks	system_scans	system_lookups	system_updates	last_system_seek	last_system_scan	l
15	1922105888	1	0	1	0	0	NULL	2024-03-24 21:10:22.733	NULL	NULL	0	0	0	0	NULL	NULL	N
16	117575457	1	15	73	0	0	2024-03-22 00:45:56.320	2024-03-22 01:02:29.257	NULL	NULL	0	0	0	0	NULL	NULL	N
17	1739153241	1	0	0	0	1	NULL	NULL	NULL	2024-03-24 21:08:13.167	0	0	0	0	NULL	NULL	N
18	1787153412	1	0	0	0	2	NULL	NULL	NULL	2024-03-24 21:08:13.167	0	0	0	0	NULL	NULL	N
19	581577110	0	0	67	0	0	NULL	2024-03-25 19:19:44.363	NULL	NULL	0	2	0	0	NULL	2024-03-25 11:24:19.337	N
20	597577167	0	0	64	0	0	NULL	2024-03-25 19:19:44.363	NULL	NULL	0	4	0	0	NULL	2024-03-25 11:44:49.860	N
21	1835153583	2	0	0	0	1	NULL	NULL	NULL	2024-03-24 21:08:13.167	0	0	0	0	NULL	NULL	N
22	1835153583	1	3	0	0	1	2024-03-24 21:08:13.173	NULL	NULL	2024-03-24 21:08:13.167	0	0	0	0	NULL	NULL	N
23	1883153754	2	0	0	0	2	NULL	NULL	NULL	2024-03-24 21:08:13.173	0	0	0	0	NULL	NULL	N
24	1883153754	0	0	0	0	2	NULL	NULL	NULL	2024-03-24 21:08:13.173	0	0	0	0	NULL	NULL	N
25	1915153868	0	0	0	0	1	NULL	NULL	NULL	2024-03-24 21:08:13.170	0	0	0	0	NULL	NULL	N
26	1915153868	2	0	0	0	1	NULL	NULL	NULL	2024-03-24 21:08:13.170	0	0	0	0	NULL	NULL	N
27	816721962	1	0	0	3	0	NULL	NULL	2024-03-24 21:...	NULL	0	0	0	0	NULL	NULL	N
28	816721962	2	3	0	0	0	2024-03-24 21:10:04.323	NULL	NULL	NULL	0	0	0	0	NULL	NULL	N
29	1362103893	1	12	29	0	0	2024-03-21 21:56:49.023	2024-03-21 21:43:19.660	NULL	NULL	0	0	0	0	NULL	NULL	N

Przechowuje informacje na temat liczb operacji wykonywanych na danym indeksie/ danych z indeksu np. wyszukiwanie, skanowanie, aktualizacja danych. Oraz daty ostatniego skanowania i ostatniej operacji wykonanej przez użytkownika na tym indeksie (mamy oczywiście też kolumny database_id, index_id i object_id – które jednoznacznie wskazują nam, którego indeksu dotyczą dane).

Napraw wykryte błędy z indeksami ze wcześniejszych zapytań. Możesz użyć do tego przykładowego skryptu:

```
USE AdventureWorks2017

--Table to hold results
DECLARE @tablevar TABLE(lngid INT IDENTITY(1,1), objectid INT,
index_id INT)

INSERT INTO @tablevar (objectid, index_id)
SELECT [object_id],index_id
FROM sys.dm_db_index_physical_stats (DB_ID('AdventureWorks2017')
,NULL -- NULL to view all tables
,NULL -- NULL to view all indexes; otherwise, input index number
,NULL -- NULL to view all partitions of an index
,'DETAILED') --We want all information
WHERE ((avg_fragmentation_in_percent > 15) -- Logical fragmentation
OR (avg_page_space_used_in_percent < 60)) --Page density
AND page_count > 8 -- We do not want indexes less than 1 extent in size
AND index_id NOT IN (0) --Only clustered and nonclustered indexes

SELECT 'ALTER INDEX ' + ind.[name] + ' ON ' + sc.[name] + '.'
+ OBJECT_NAME(objectid) + ' REBUILD'
FROM @tablevar tv
INNER JOIN sys.indexes ind
ON tv.objectid = ind.[object_id]
AND tv.index_id = ind.index_id
INNER JOIN sys.objects ob
ON tv.objectid = ob.[object_id]
INNER JOIN sys.schemas sc
ON sc.schema_id = ob.schema_id
```

Napisz przygotowane komendy SQL do naprawy indeksów:

Przebudowa:

```
ALTER INDEX XMLPATH_Person_Demographics
ON Person.Person REBUILD

ALTER INDEX XMLPROPERTY_Person_Demographics
ON Person.Person REBUILD

ALTER INDEX XMLVALUE_Person_Demographics
ON Person.Person REBUILD
```

Reorganizacja:

```
ALTER INDEX PK_JobCandidate_JobCandidateID
ON HumanResources.JobCandidate REORGANIZE

ALTER INDEX PK_ProductModel_ProductModelID
ON Production.ProductModel REORGANIZE

ALTER INDEX PK_BillOfMaterials_BillOfMaterialsID
ON Production.BillOfMaterials REORGANIZE

ALTER INDEX IX_WorkOrder_ProductID
ON Production.WorkOrder REORGANIZE

ALTER INDEX IX_WorkOrderRouting_ProductID
ON Production.WorkOrderRouting REORGANIZE
```

Zadanie 4 - Budowa strony indeksu

Dokumentacja

Celem kolejnego zadania jest zapoznanie się z fizyczną budową strony indeksu

- https://www.mssqltips.com/sqlservertip/1578/using-dbcc-page-to-examine-sql-server-table-and-index-data/
- https://www.mssqltips.com/sqlservertip/2082/understanding-and-examining-the-uniquifier-in-sql-server/
- http://www.sqlskills.com/blogs/paul/inside-the-storage-engine-using-dbcc-page-and-dbcc-ind-to-find-out-if-page-splits-ever-roll-back/

Wypisz wszystkie strony które są zaalokowane dla indeksu w tabeli. Użyj do tego komendy np.:

```
DBCC IND ('AdventureWorks2017', 'Person.Address', 1)
-- '1' oznacza nr indeksu
```

Zapisz sobie kilka różnych typów stron, dla różnych indeksów:

```
DBCC IND ('AdventureWorks2017', 'Person.Address', 1)
-- '1' oznacza nr indeksu

DBCC IND ('AdventureWorks2017', 'Person.Address', 2)
-- '2' oznacza nr indeksu

DBCC IND ('AdventureWorks2017', 'Person.Address', 3)
-- 3' oznacza nr indeksu
```

Polecenie 1

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10474	NULL	NULL	1029578706	1	1	72057594047889408	In-row data	10	NULL	0	0	0	0
2	1	11712	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11713	1	12010
3	1	11713	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11714	1	11712
4	1	11714	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11715	1	11713
5	1	11715	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11716	1	11714
6	1	11716	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11717	1	11715
7	1	11717	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11718	1	11716
8	1	11718	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11719	1	11717
9	1	11719	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11720	1	11718
10	1	11720	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11721	1	11719

Polecenie 2

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10472	NULL	NULL	1029578706	2	1	72057594052542464	In-row data	10	NULL	0	0	0	0
2	1	5872	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5873	0	0
3	1	5873	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5874	1	5872
4	1	5874	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5875	1	5873
5	1	5875	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5876	1	5874
6	1	5876	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5877	1	5875
7	1	5877	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5878	1	5876
8	1	5878	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5879	1	5877
9	1	5879	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5880	1	5878
10	1	5880	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5881	1	5879

Polecenie 3

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10473	NULL	NULL	1029578706	3	1	72057594052608000	In-row data	10	NULL	0	0	0	0
2	1	5920	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5921	0	0
3	1	5921	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5922	1	5920
4	1	5922	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5923	1	5921
5	1	5923	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5924	1	5922
6	1	5924	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5925	1	5923
7	1	5925	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5926	1	5924
8	1	5926	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5927	1	5925
9	1	5927	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5928	1	5926
10	1	5928	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5929	1	5927

Włącz flagę 3604 zanim zaczniesz przeglądać strony:

```
DBCC TRACEON (3604);
```

Sprawdź poszczególne strony komendą DBCC PAGE. Np.:

```
DBCC PAGE('AdventureWorks2017', 1, 13720, 3);
```

Zapisz obserwacje ze stron. Co ciekawego udało się zaobserwować?

Strona 11714 dla indeksu 1

	ParentObject	Object	Field	VALUE
1	BUFFER:	BUF @0x00000196360EA5C0	bpage	0x000001913E71A000
2	BUFFER:	BUF @0x00000196360EA5C0	bPmmpage	0x0000000000000000
3	BUFFER:	BUF @0x00000196360EA5C0	bsort_r_nextbP	0x00000196360EA510
4	BUFFER:	BUF @0x00000196360EA5C0	bsort_r_prevbP	0x0000000000000000
5	BUFFER:	BUF @0x00000196360EA5C0	bhash	0x0000000000000000
6	BUFFER:	BUF @0x00000196360EA5C0	bpageno	(1:11714)
7	BUFFER:	BUF @0x00000196360EA5C0	bpart	8
8	BUFFER:	BUF @0x00000196360EA5C0	bstat	0x9
9	BUFFER:	BUF @0x00000196360EA5C0	breferences	0
10	BUFFER:	BUF @0x00000196360EA5C0	berrcode	0

Strona 9573 dla indeksu 2
Dla indeksu2 strona 5880

	ParentObject	Object	Field	VALUE
1	BUFFER:	BUF @0x000001E5972F1BC0	bpage	0x000001E579704000
2	BUFFER:	BUF @0x000001E5972F1BC0	bPmmpage	0x0000000000000000
3	BUFFER:	BUF @0x000001E5972F1BC0	bsort_r_n...	0x0000000000000000
4	BUFFER:	BUF @0x000001E5972F1BC0	bsort_r_pr...	0x0000000000000000
5	BUFFER:	BUF @0x000001E5972F1BC0	bhash	0x0000000000000000
6	BUFFER:	BUF @0x000001E5972F1BC0	bpageno	(2:5880)
7	BUFFER:	BUF @0x000001E5972F1BC0	bpart	3
8	BUFFER:	BUF @0x000001E5972F1BC0	ckptGen	0x0000000000000000
9	BUFFER:	BUF @0x000001E5972F1BC0	bDirtyRef...	0
10	BUFFER:	BUF @0x000001E5972F1BC0	bstat	0x809
11	BUFFER:	BUF @0x000001E5972F1BC0	breferences	3
12	BUFFER:	BUF @0x000001E5972F1BC0	berrcode	-3
13	BUFFER:	BUF @0x000001E5972F1BC0	bUse1	3233
14	BUFFER:	BUF @0x000001E5972F1BC0	bstat2	0x0
15	BUFFER:	BUF @0x000001E5972F1BC0	blog	0x15a
16	BUFFER:	BUF @0x000001E5972F1BC0	bsampleC...	0
17	BUFFER:	BUF @0x000001E5972F1BC0	bloCount	0

Strona 8745 dla indeksu 3

	ParentObject	Object	Field	VALUE
1	BUFFER:	BUF @0x000001E5972EC600	bpage	0x000001E5845FA000
2	BUFFER:	BUF @0x000001E5972EC600	bPmmpage	0x0000000000000000
3	BUFFER:	BUF @0x000001E5972EC600	bsort_r_nextbP	0x000001E5972EC550
4	BUFFER:	BUF @0x000001E5972EC600	bsort_r_prevbP	0x0000000000000000
5	BUFFER:	BUF @0x000001E5972EC600	bhash	0x0000000000000000
6	BUFFER:	BUF @0x000001E5972EC600	bpageno	(2:8745)
7	BUFFER:	BUF @0x000001E5972EC600	bpart	2
8	BUFFER:	BUF @0x000001E5972EC600	ckptGen	0x0000000000000000
9	BUFFER:	BUF @0x000001E5972EC600	bDirtyRefCount	0
10	BUFFER:	BUF @0x000001E5972EC600	bstat	0x809
11	BUFFER:	BUF @0x000001E5972EC600	breferences	1
12	BUFFER:	BUF @0x000001E5972EC600	berrcode	-3
13	BUFFER:	BUF @0x000001E5972EC600	bUse1	63646
14	BUFFER:	BUF @0x000001E5972EC600	bstat2	0x0
15	BUFFER:	BUF @0x000001E5972EC600	blog	0x15ab215a
16	BUFFER:	BUF @0x000001E5972EC600	bsampleCount	0
17	BUFFER:	BUF @0x000001E5972EC600	bloCount	0
18	BUFFER:	BUF @0x000001E5972EC600	resPoolId	0
19	BUFFER:	BUF @0x000001E5972EC600	bcputicks	0
20	BUFFER:	BUF @0x000001E5972EC600	bReadMicroSec	0
21	BUFFER:	BUF @0x000001E5972EC600	bDirtyContext	0x0000000000000000

Można zauważyć obiekty postaci Memory Dump @0x00000002698F8852. Oznaczają one zrzut pamięci bazy danych, gdzie dane od @ oznaczają fizyczny adres w pamięci. A strony postaci BUF @0x000001E5971DD200 zapisywane są w buforze pamięci co pozwala na szybszy dostęp do danych.

Punktacja

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