

Indeksy, optymalizator

Lab 4

Imię i nazwisko: Mateusz Skowron, Bartłomiej Wiśniewski, Karol Wrona

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

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 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):

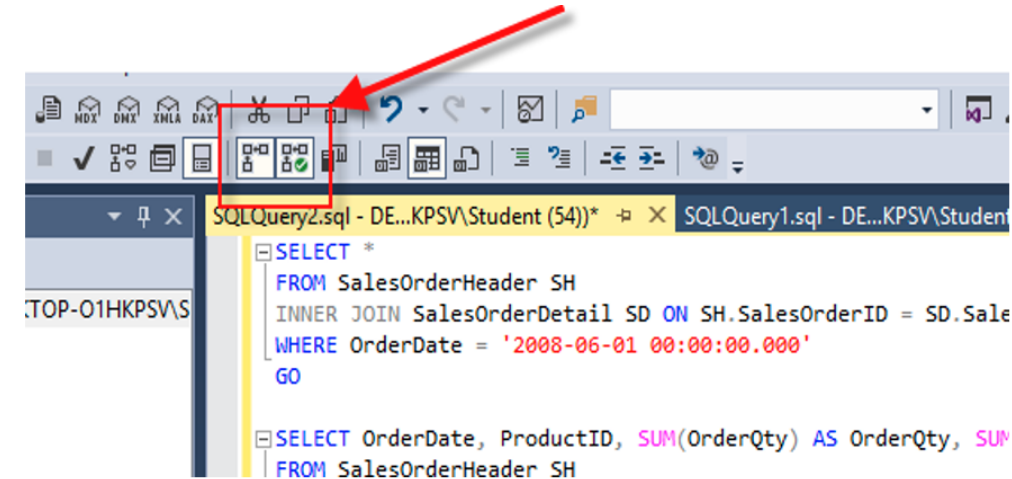
```
-- zapytanie 1
select *
from salesorderheader sh
inner join salesorderdetail sd on sh.salesorderid = sd.salesorderid
where orderdate = '2008-06-01 00:00:00.000'
go

-- 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)

Wyniki

Zapytanie 1

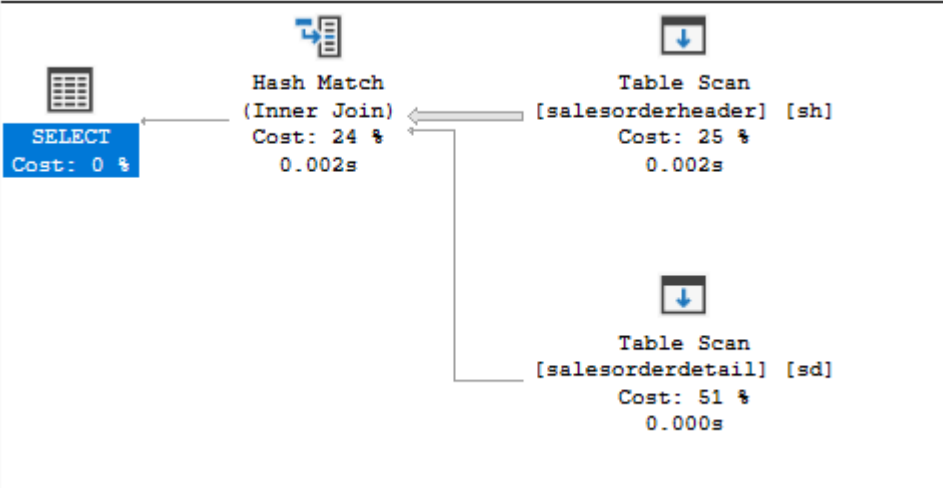
Opis

Zapytanie pobiera wszystkie kolumny z tabeli `salesorderheader` oraz `salesorderdetail`, łącząc je na podstawie kolumny `salesorderid`. Następnie filtruje wyniki, zwracając tylko te rekordy, gdzie wartość kolumny `orderdate` wynosi '2008-06-01 00:00:00.000'.

Wynik

SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID	SalesPersonID	TerritoryID	BillToAddressID	ShipToAddressID	ShipMethodID	CreditCardID	CreditCardApprovalCode	CurrencyRateID
--------------	----------------	-----------	---------	----------	--------	-----------------	------------------	---------------------	---------------	------------	---------------	-------------	-----------------	-----------------	--------------	--------------	------------------------	----------------

Execution Plan:



Zapytanie zwraca pusty zbiór wyników, jednakże proces jego wykonania generuje znaczący koszt związany z przeszukiwaniem całej drugiej tabeli, pomimo braku wyników w pierwszej.

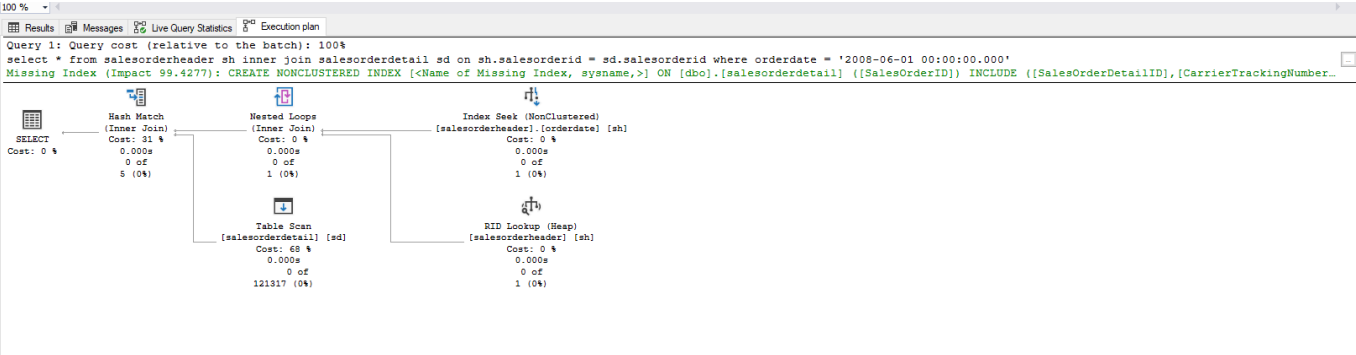
Optymalizacja

Można zoptymalizować to zapytanie poprzez stworzenie indeksu na kolumnie `orderdate` w tabeli `salesorderheader`. Pozwoli to na szybsze określenie, że zbiór wynikowy jest pusty bez konieczności przeszukiwania całej drugiej tabeli.

Możemy utworzyć indeks na kolumnę `orderdate` w tabeli `salesorderheader` poniższą komendą:

```
CREATE NONCLUSTERED INDEX [orderdate] ON [dbo].[salesorderheader] ([OrderDate])
```

Po utworzeniu indeksu koszt i czas zapytania spadają do zera.



Zapytanie 2

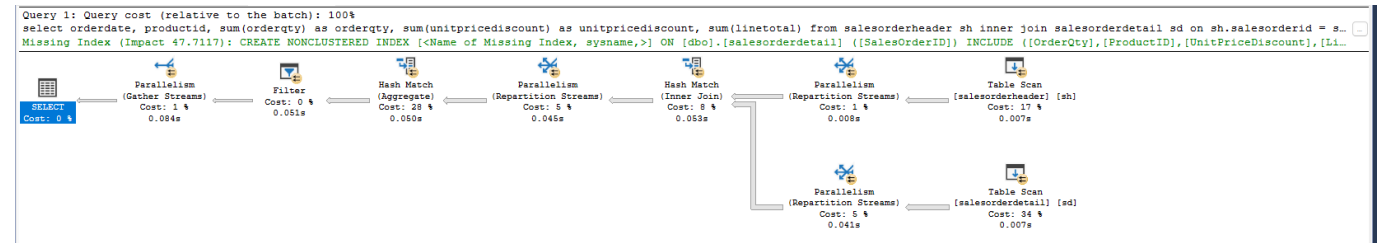
Opis

Zapytanie pobiera `orderdate`, `productid` oraz sumy `orderqty`, `unitpricediscount` i `linetotal`. Łączy tabele `salesorderheader` i `salesorderdetail` na podstawie kolumny `salesorderid`, grupując wyniki według `orderdate` i `productid`, a następnie filtrując grupy, gdzie suma `orderqty` jest większa lub równa 100.

Wynik

	orderdate	productid	orderqty	unitpricediscount	(No column name)
1	2013-03-30 00:00:00.000	863	358	0,68	7090.168675
2	2013-05-30 00:00:00.000	716	138	0,02	4121.795476
3	2012-08-30 00:00:00.000	765	107	0,02	50284.244192
4	2013-03-30 00:00:00.000	763	121	0,00	57158.270000
5	2013-10-30 00:00:00.000	712	251	0,16	1345.391258
6	2012-06-30 00:00:00.000	832	133	0,00	27831.048000
7	2013-06-30 00:00:00.000	877	339	0,32	1546.566765
8	2012-06-30 00:00:00.000	729	101	0,00	20435.532000
9	2012-07-31 00:00:00.000	832	119	0,00	24901.464000
10	2012-07-31 00:00:00.000	729	107	0,00	21649.524000

Execution Plan:



Zapytanie zwraca poprawny zbiór wyników, jednak proces sumowania kolumn może być zoptymalizowany.

Optymalizacja

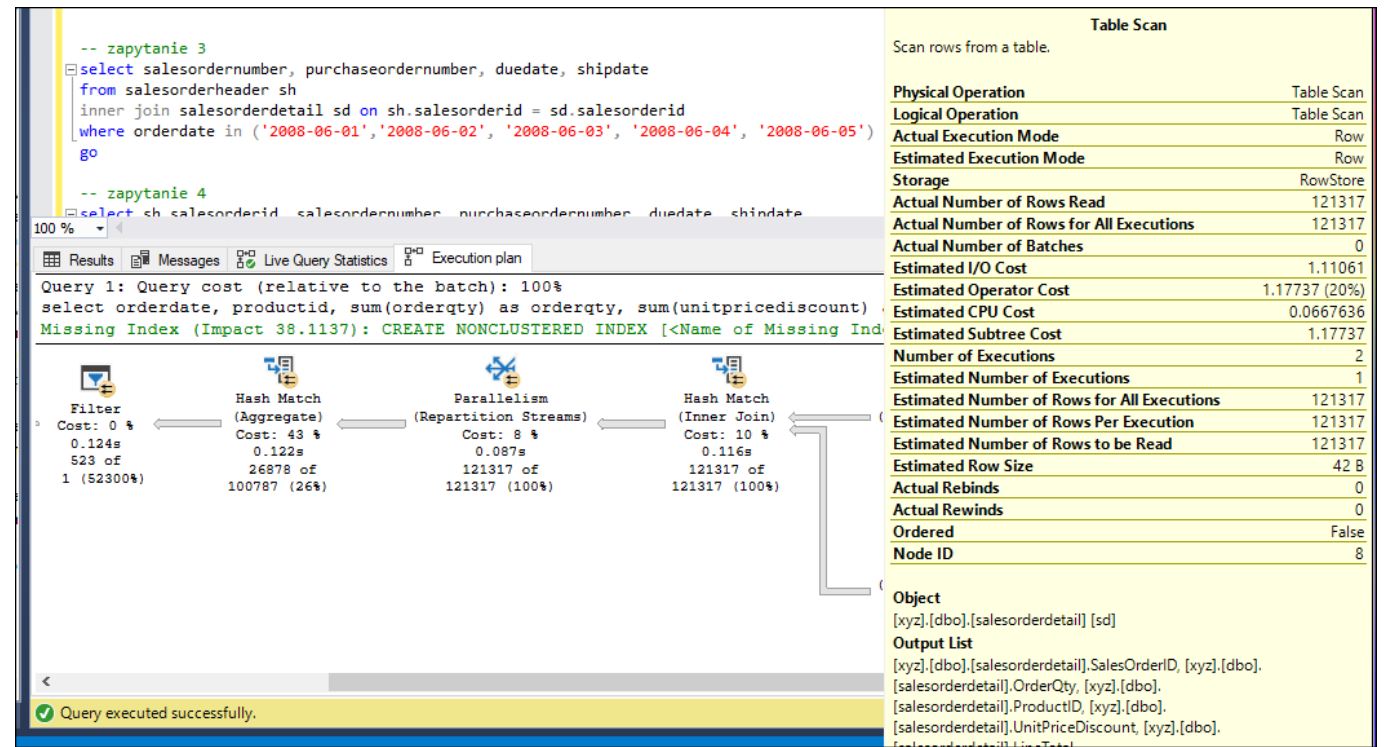
Aby zoptymalizować to zapytanie, można utworzyć indeksy na kolumnach **orderdate** w tabeli **salesorderheader** i **productid** w tabeli **salesorderdetail**. Dodatkowo, można skorzystać z indeksu z opcją **include**, aby szybciej uzyskać dostęp do kolumn sumowanych podczas agregacji.

Potrzebny indeks można utworzyć poniższą komendą:

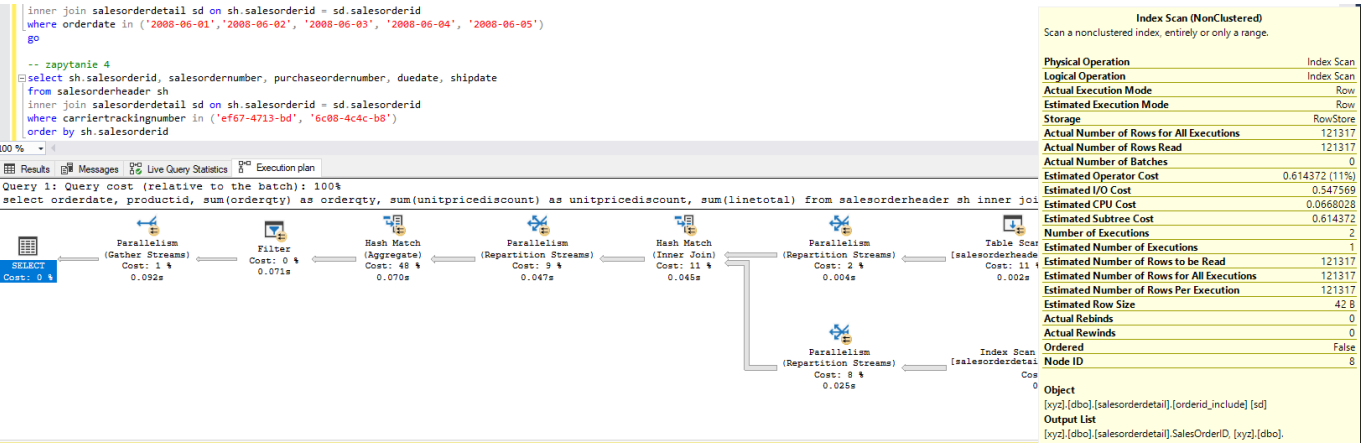
```
CREATE NONCLUSTERED INDEX [orderid_include] ON [dbo].[salesorderdetail]
([SalesOrderID]) INCLUDE ([OrderQty], [ProductID], [UnitPriceDiscount],
[LineTotal])
```

Po utworzeniu indeksu koszt i czas zapytania znacznie się zmniejszają.

Przed:



Po:

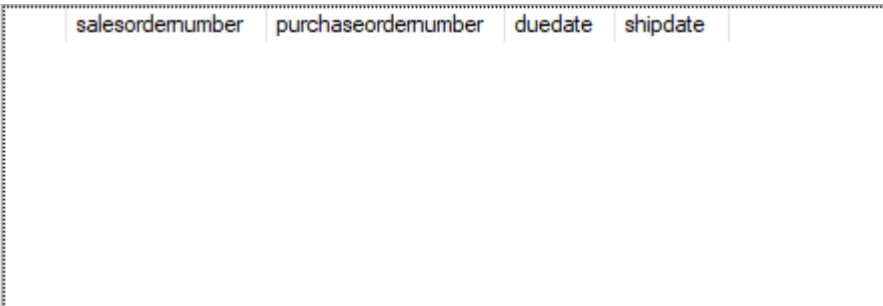


Zapytanie 3

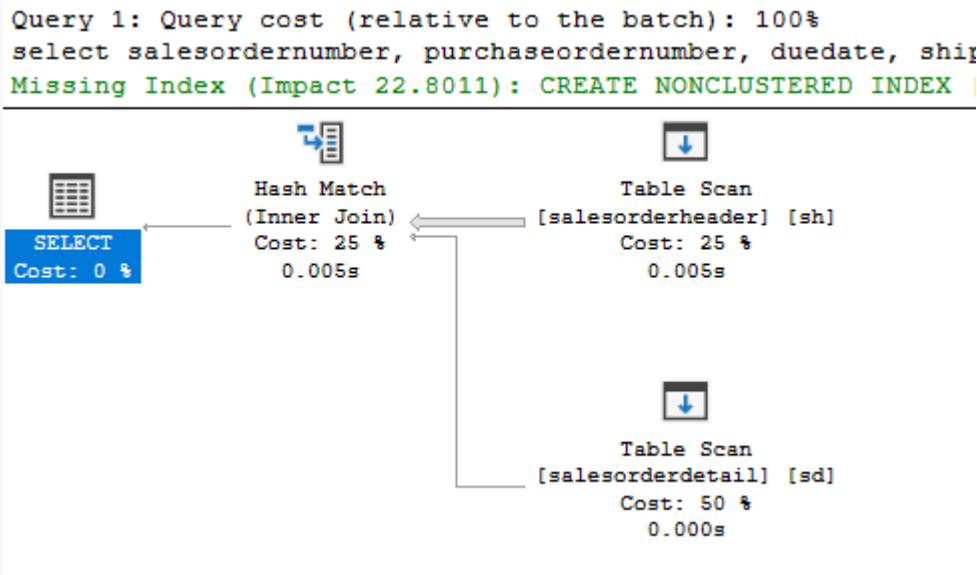
Opis

Zapytanie pobiera kolumny salesordernumber, purchaseordernumber, due date, shipdate, łącząc tabele salesorderheader i salesorderdetail na podstawie kolumny salesorderid. Następnie filtruje wyniki, zwracając tylko te rekordy, gdzie orderdate należy do określonych dat.

Wynik



Execution Plan:



Sytuacja bardzo podobna do tej z zapytania 1. Zbiór wynikowy jest pusty, ale odkrycie tego było kosztowne.

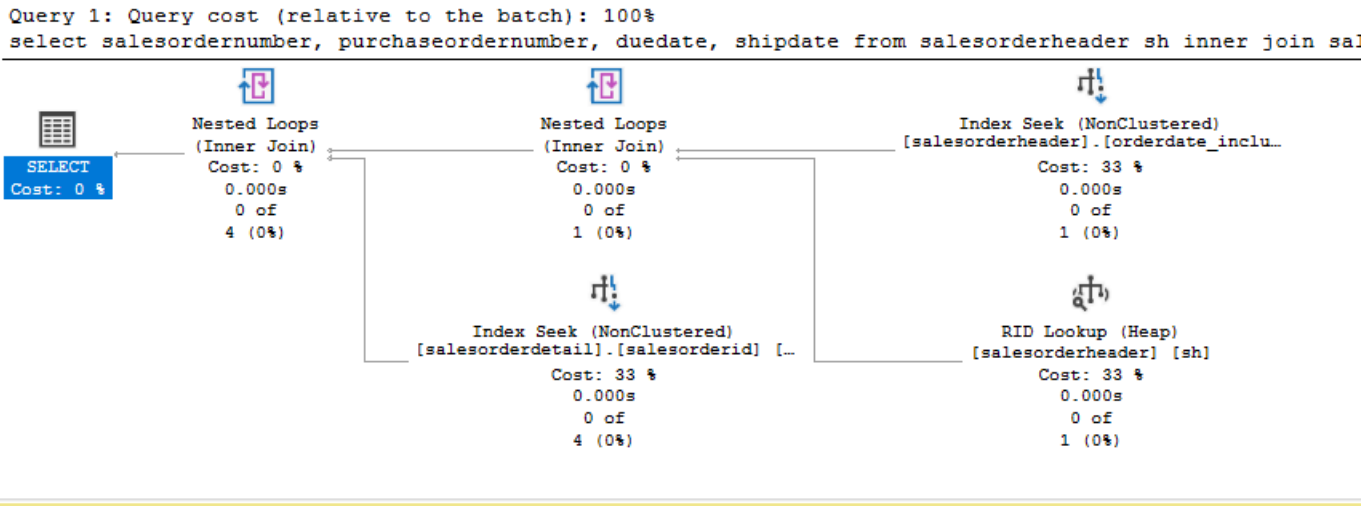
Optymalizacja

Stworzenie indeksu na kolumnie `orderdate` w tabeli `salesorderheader` przyspieszy zapytanie, eliminując zbędne obliczenia.

Potrzebny indeks można utworzyć poniższa komendą:

```
CREATE NONCLUSTERED INDEX [orderdate] ON [dbo].[salesorderheader] ([OrderDate])
```

Po wykonaniu zapytania po utworzeniu indeksu, koszt i czas zapytania spadają do zera.



Zapytanie 4

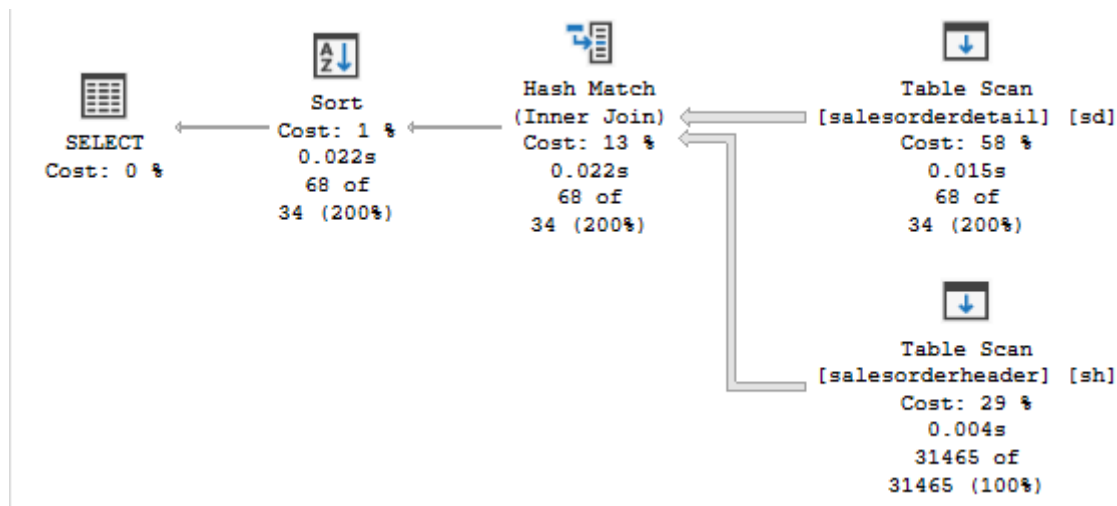
Opis

Zapytanie pobiera kolumny `sh.salesorderid`, `salesordernumber`, `purchaseordernumber`, `due date`, `shipdate`, łącząc tabelę `salesorderheader` i `salesorderdetail` na podstawie kolumny `salesorderid`. Następnie filtruje wyniki, zwracając tylko te rekordy, gdzie `carriertrackingnumber` należy do określonych wartości.

Wynik

	salesorderid	salesordernumber	purchaseordernumber	due date	shipdate
1	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
2	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
3	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
4	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
5	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
6	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
7	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
8	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000

Execution Plan:



Widzimy, że znaczną część kosztu zapytania powoduje zapytanie pochodzące z tabeli `salesorderdetail`, a to w niej wyszukiwany jest `carriertrackingnumber` z warunku `WHERE`.

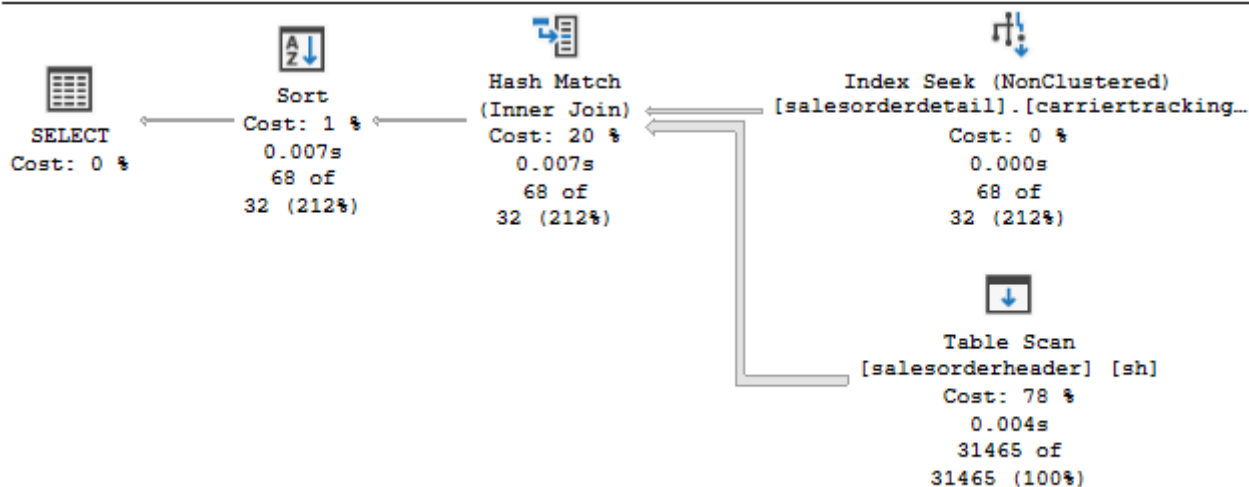
Optymalizacja

Pierwszym krokiem do optymalizacji tego zapytania jest utworzenie indeksu na kolumnie `carriertrackingnumber` w tabeli `salesorderdetail`. Indeks ten przyspieszy wyszukiwanie rekordów na podstawie wartości w tej kolumnie, co znacznie zwiększy wydajność zapytania. Można także skorzystać z `include`, aby szybciej uzyskać wartość kolumny `salesorderid`, co również pozwoli nam przyspieszyć to zapytanie.

Potrzebny indeks można utworzyć poniższą komendą:

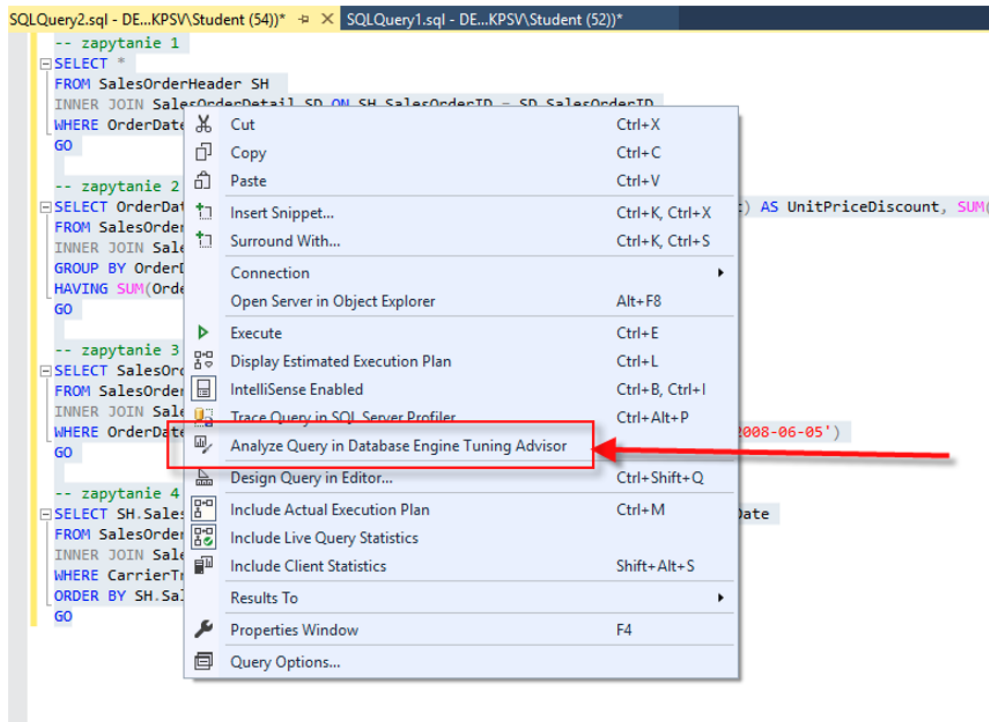
```
CREATE NONCLUSTERED INDEX [carriertrackingnumber] ON [dbo].[salesorderdetail]
([carriertrackingnumber]) INCLUDE ([salesorderid])
```

Po wykonaniu zapytania po utworzeniu indeksu, czas i koszt każdego z etapów oraz całego zapytania znacznie spadają.



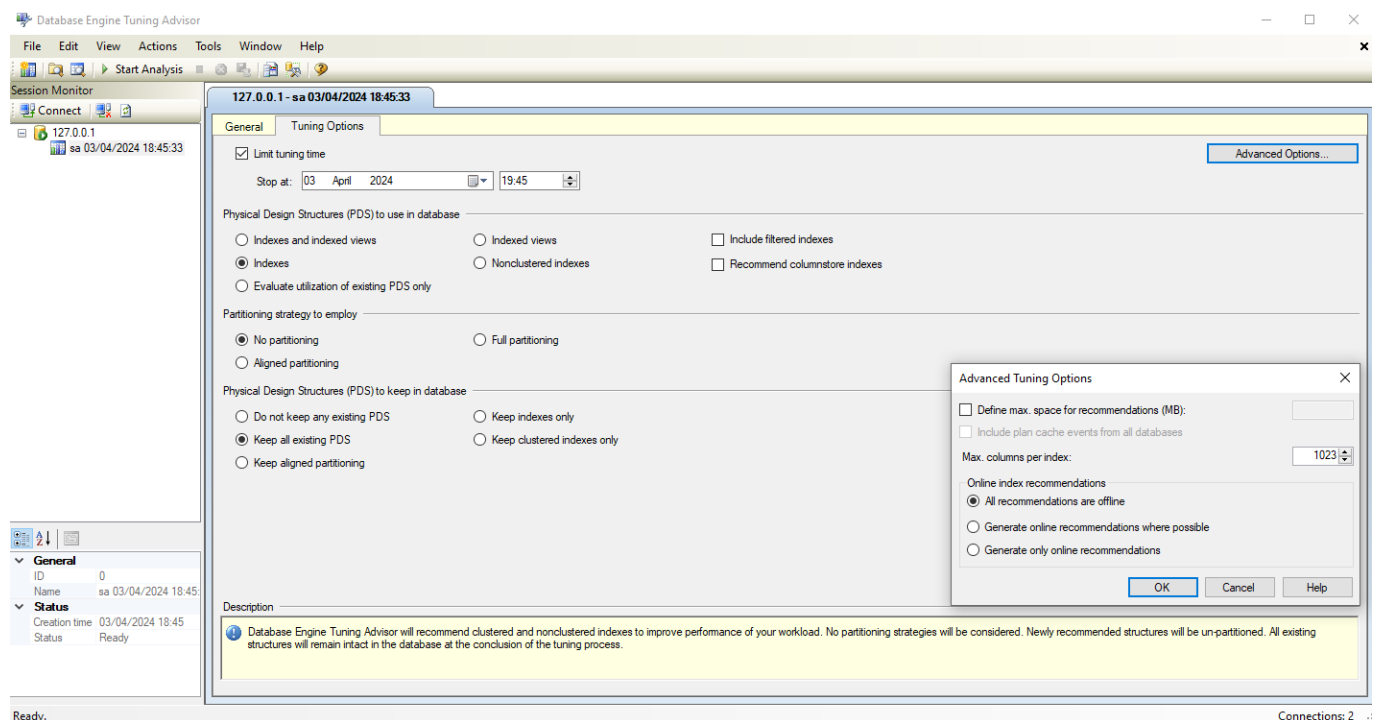
Zadanie 2 - Optymalizacja

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



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

Wyniki



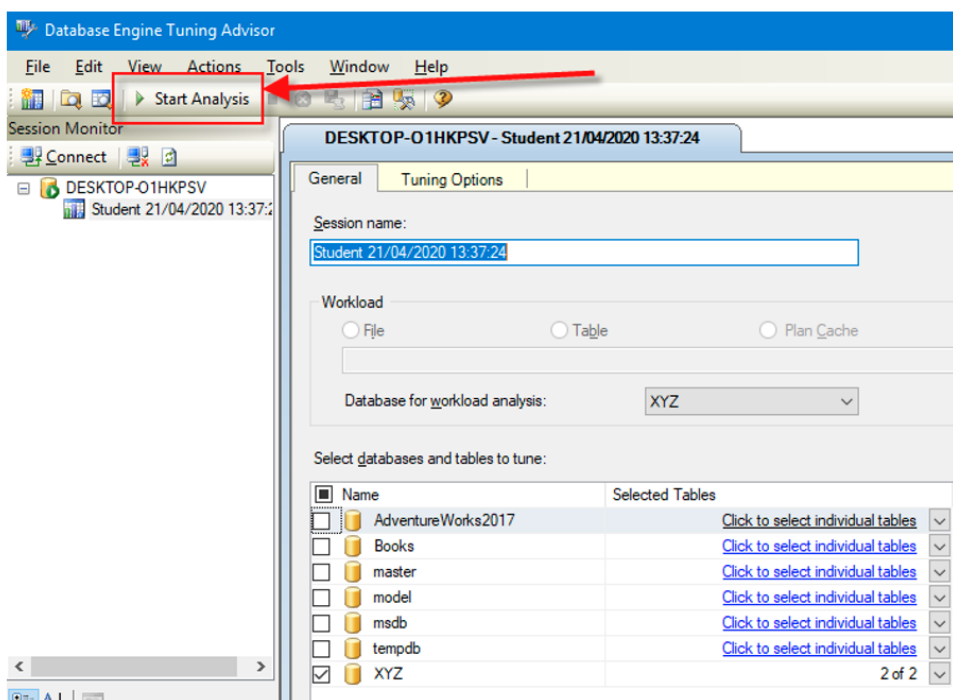
Mamy możliwość konfiguracji następujących parametrów:

- **Ograniczenie czasu tuningu:** Określ, ile czasu narzędzie ma przeznaczyć na analizę i generowanie rekomendacji. Możesz wybrać konkretny czas zakończenia procesu.
- **Fizyczne struktury przechowujące dane do użycia w bazie danych:** Wybierz struktury, takie jak indeksy lub indeksowane widoki, które mają być brane pod uwagę w rekomendacjach. Możesz również zdecydować, czy uwzględnić indeksy filtrowane i indeksy kolumnowe.
- **Strategia partycjonowania:** Wybierz preferowaną strategię partycjonowania, taką jak partycjonowanie pełne czy równomierne.
- **Zachowanie fizycznych struktur przechowywania danych:** Określ, które struktury zachować w bazie danych, np. wszystkie, żadne, tylko indeksy itp.

Oprócz tych podstawowych opcji, istnieją także zaawansowane ustawienia, takie jak:

- **Maksymalna przestrzeń rekomendacji:** Określ maksymalną ilość miejsca, jaką narzędzie może zarezerwować na rekomendacje (wyrażoną w megabajtach).
- **Uwzględnienie zdarzeń z bufora planów:** Decyduj, czy uwzględnić zdarzenia z bufora planów z wszystkich baz danych w analizie.
- **Maksymalna liczba kolumn na indeks:** Określ maksymalną liczbę kolumn, które mogą być zawarte w pojedynczym indeksie.
- **Praca bazy danych podczas tuningu:** Wybierz, czy dopuszczasz przerywanie pracy bazy danych podczas procesu tuningu.

Użyj **Start Analysis**:



Zaobserwuj wyniki w **Recommendations**.

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

Tuning reports			
Select report:		Statement cost report	
Statement Id	Statement String	Percent Improvement	Statement Type
3	SELECT SalesOrderNumber, Purch...	99.74	Select
1	SELECT * FROM SalesOrderHeade...	99.73	Select
4	SELECT SH.SalesOrderID, SalesO...	88.41	Select
2	SELECT OrderDate, ProductID, S...	19.20	Select

Zapisz poszczególne rekomendacje

Uruchom zapisany skrypt w Management Studio.

Opisz, dlaczego dane indeksy zostały zaproponowane do zapytań:

Wyniki

Recommendations

Database Engine Tuning Advisor

File Edit View Actions Tools Window Help

Session Monitor

127.0.0.1 - sa 03/04/2024 18:45:33

127.0.0.1

sa 03/04/2024 18:45:33

General Tuning Options Progress Recommendations Reports

Estimated improvement: 62%

Partition Recommendations

Database Name Recommendation Target of Recommendation Details No. of Partitions Definition

Index Recommendations

Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
xyz	[dbo].[salesorderdetail]	create	_idx_index_salesorderdetail_9_917578307_K1			544	([SalesOrderID] asc)
xyz	[dbo].[salesorderdetail]	create	_idx_index_salesorderdetail_9_917578307_K3_1			2312	([CenterTrackingNumber] asc) include ([SalesOrderID])
xyz	[dbo].[salesorderdetail]	create	_idx_index_salesorderdetail_9_917578307_K1_K5_4_8_9			4760	([SalesOrderID] asc, [ProductID] asc) include ([OrderQty], [UnitPriceDesc])
xyz	[dbo].[salesorderdetail]	create	_idx_index_salesorderdetail_9_917578307_K1_2_3_4_5_6_7_8_9_10_11			11976	([SalesOrderID] asc) include ([SalesOrderDetailID], [CenterTrackingNumber])
xyz	[dbo].[salesorderheader]	create	_idx_index_salesorderheader_9_901578250_K3_1			416	([OrderDate] asc) include ([SalesOrderID])
xyz	[dbo].[salesorderheader]	create	_idx_index_salesorderheader_9_901578250_K1_4_5_8_9			1336	([SalesOrderID] asc) include ([DueDate], [ShipDate], [SalesOrderNumber])
xyz	[dbo].[salesorderheader]	create	_idx_index_salesorderheader_9_901578250_K3_K1_2_4_5_6_7_8_9_10_11_12_13_14_15_16_17_18_19_20_21_22_23_24_25_26			6264	([OrderDate] asc, [SalesOrderID] asc) include ([RevisionNumber], [DueDate])

General

ID 1

Name sa 03/04/2024 18:4

Status

Creation time 03/04/2024 18:45

Status Finished

Tuning session completed successfully.

Connections: 2

Raporty

Database Engine Tuning Advisor

File Edit View Actions Tools Window Help

Start Analysis

Session Monitor

Connect

127.0.0.1

sa 03/04/2024 18:45:33

127.0.0.1 - sa 03/04/2024 18:45:33

General Tuning Options Progress Recommendations Reports

Tuning Summary

Date	03/04/2024
Time	18:54:24
Server	127.0.0.1
Database(s) to tune	[xyz]
Workload	Inline
Maximum tuning time	51 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	62.03
Maximum space for recommendation (MB)	54
Space used currently (MB)	21
Space used by recommendation (MB)	50
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
Number of statistics recommended to be created	1

Tuning Reports

Select report: Statement cost report

Statement Id	Statement String	Percent Improvement	Statement Type
3	-- zapytanie 3 select salesordernumb...	99.74	Select
1	select * from salesorderheader sh i...	99.73	Select
4	-- zapytanie 4 select sh.salesorderid,...	94.59	Select
2	-- zapytanie 2 select orderdate, prod...	19.22	Select

General

ID 1

Name sa 03/04/2024 18:4

Status

Creation time 03/04/2024 18:45

Status Finished

Tuning session completed successfully.

Connections: 3

Zapisany skrypt

```
USE [xyz]
GO

CREATE NONCLUSTERED INDEX [IX_SalesOrderDetail_SalesOrderID] ON [dbo].
[salesorderdetail]
(
    [SalesOrderID] ASC
)
INCLUDE([SalesOrderDetailID],[CarrierTrackingNumber],[OrderQty],[ProductID],
[SpecialOfferID],[UnitPrice],[UnitPriceDiscount],[LineTotal],[rowguid],
[ModifiedDate])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO

CREATE NONCLUSTERED INDEX [IX_SalesOrderDetail_SalesOrderID_ProductID] ON [dbo].
[salesorderdetail]
(
    [SalesOrderID] ASC,
    [ProductID] ASC
)
INCLUDE([OrderQty],[UnitPriceDiscount],[LineTotal])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IX_SalesOrderDetail_CarrierTrackingNumber] ON [dbo].
[salesorderdetail]
(
    [CarrierTrackingNumber] ASC
)
INCLUDE([SalesOrderID])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO

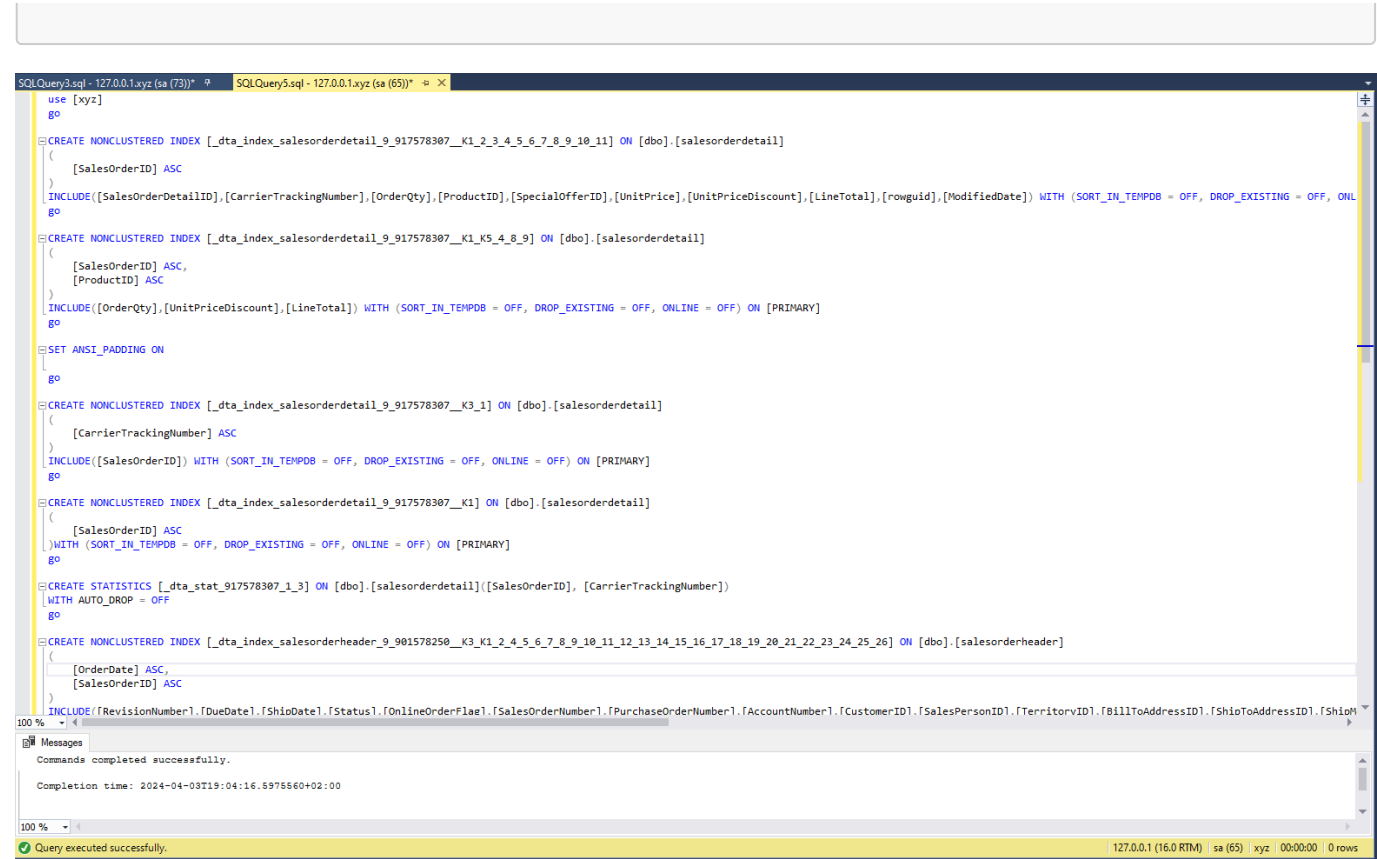
CREATE NONCLUSTERED INDEX [IX_SalesOrderDetail_SalesOrderID] ON [dbo].
[salesorderdetail]
(
    [SalesOrderID] ASC
)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO

CREATE STATISTICS [stats_SalesOrderDetail_SalesOrderID_CarrierTrackingNumber] ON
[dbo].[salesorderdetail]([SalesOrderID], [CarrierTrackingNumber])
WITH AUTO_DROP = OFF
GO

CREATE NONCLUSTERED INDEX [IX_SalesOrderHeader_OrderDate_SalesOrderID] ON [dbo].
[salesorderheader]
(
    [OrderDate] ASC,
    [SalesOrderID] ASC
)
INCLUDE([RevisionNumber],[DueDate],[ShipDate],[Status],[OnlineOrderFlag],
[SalesOrderNumber],[PurchaseOrderNumber],[AccountNumber],[CustomerID],
[SalesPersonID],[TerritoryID],[BillToAddressID],[ShipToAddressID],[ShipMethodID],
[CreditCardID],[CreditCardApprovalCode],[CurrencyRateID],[SubTotal],[TaxAmt],
[Freight],[TotalDue],[Comment],[rowguid],[ModifiedDate])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO

CREATE NONCLUSTERED INDEX [IX_SalesOrderHeader_SalesOrderID] ON [dbo].
[salesorderheader]
(
    [SalesOrderID] ASC
)
INCLUDE([DueDate],[ShipDate],[SalesOrderNumber],[PurchaseOrderNumber])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO

CREATE NONCLUSTERED INDEX [IX_SalesOrderHeader_OrderDate] ON [dbo].
[salesorderheader]
(
    [OrderDate] ASC
)
INCLUDE([SalesOrderID])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
GO
```



Dlaczego dane indeksy zostały zaproponowane do zapytań?

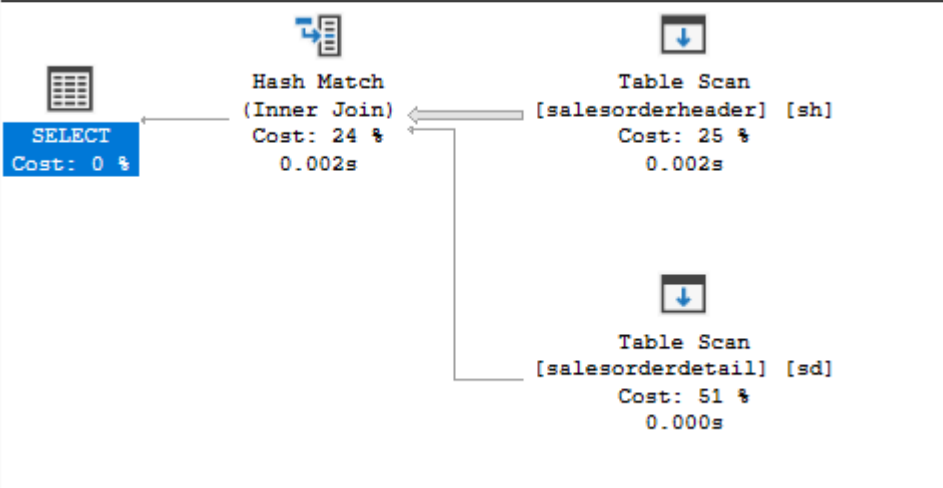
Indeksy zostały zaproponowane na podstawie analizy wykonywanych zapytań. Ponieważ zapytania korzystają głównie z identyfikatorów (np. "salesorderid"), dat zamówień ("orderdate") oraz numerów śledzenia przewoźnika ("carriertrackingnumber"), analizator analizując zapytania wskazał właśnie te kolumny i zaproponował indeksy dla tych kolumn, aby usprawnić wyszukiwanie danych.

Sprawdź jak zmieniły się Execution Plany. Opisz zmiany.

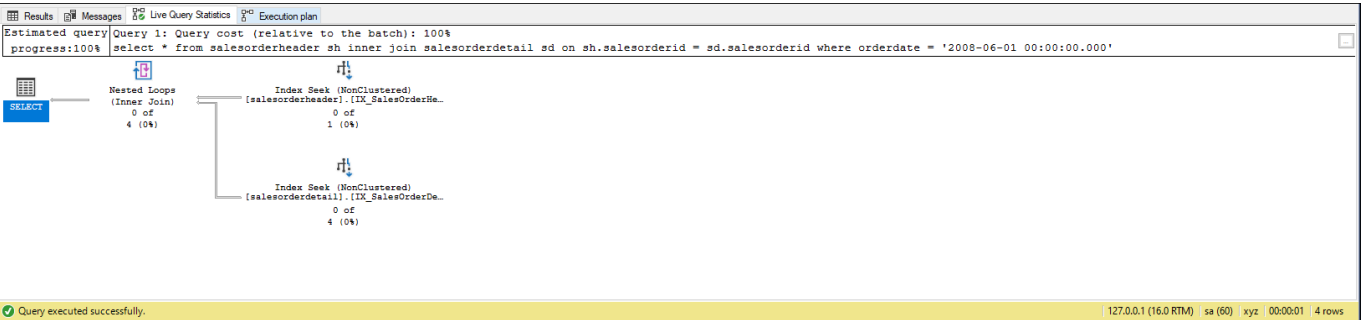
Wyniki

Zapytanie 1

Stary Execution Plan

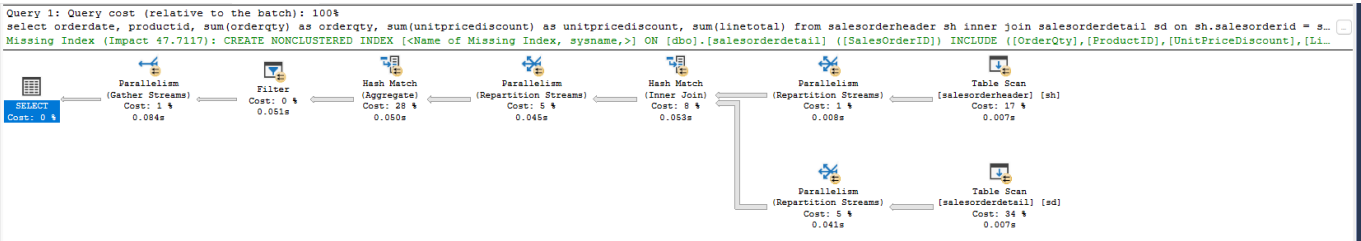


Nowy Execution Plan

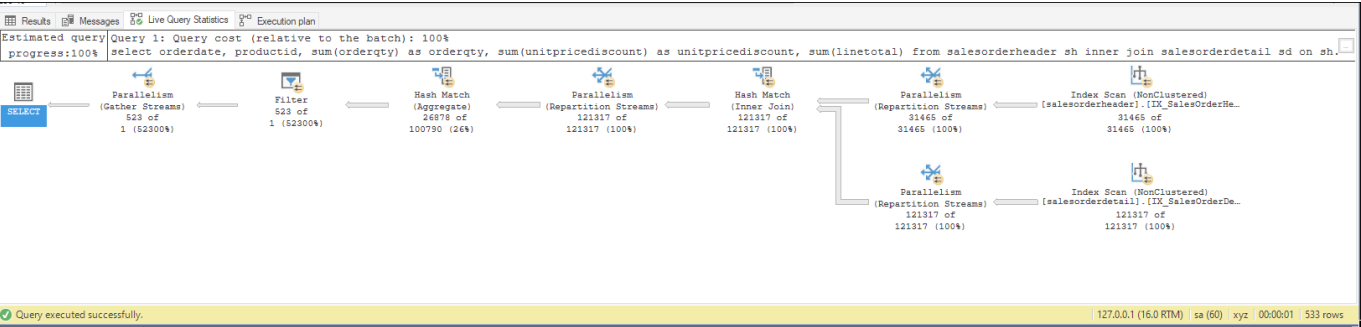


Zapytanie 2

Stary Execution Plan



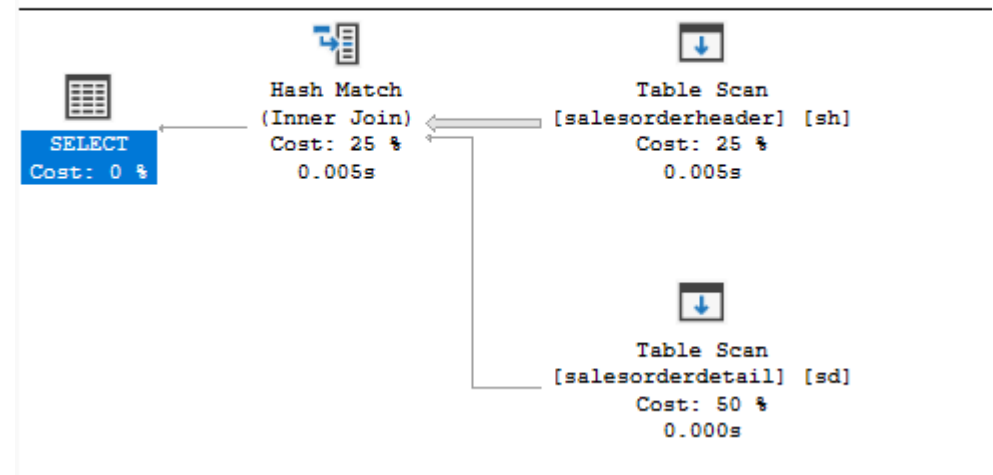
Nowy Execution Plan



Zapytanie 3

Stary Execution Plan

Query 1: Query cost (relative to the batch): 100%
select salesordernumber, purchaseordernumber, duedate, ship
Missing Index (Impact 22.8011): CREATE NONCLUSTERED INDEX



Nowy Execution Plan

100 %

ResultsMessagesLive Query StatisticsExecution plan

Estimated query progress:100%

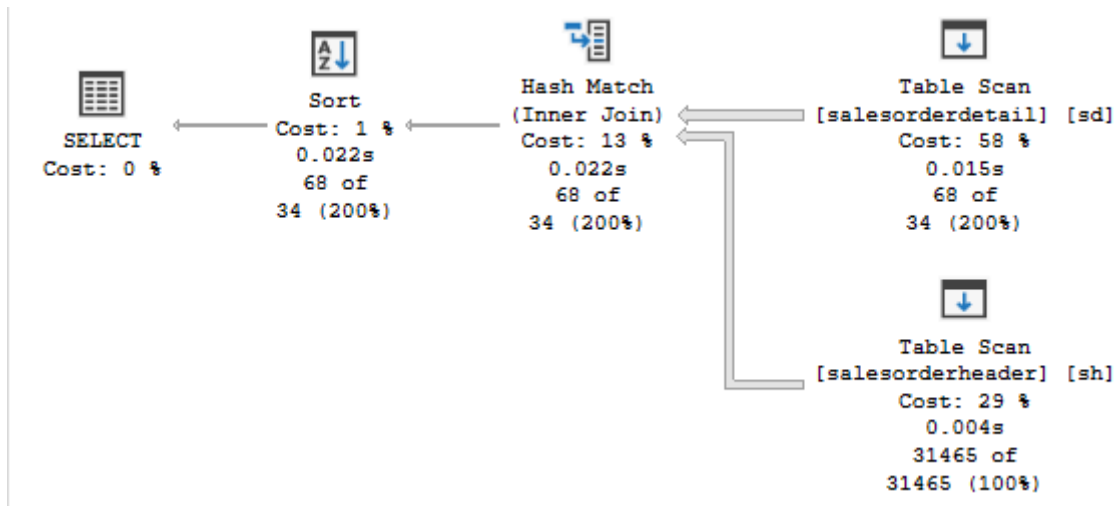
Query 1: Query cost (relative to the batch): 100%
select salesordernumber, purchaseordernumber, duedate, ship

The execution plan for Query 1 shows a Nested Loops (Inner Join) operation. The left input is an Index Seek (NonClustered) on [salesorderheader].[IX_SalesOrderHe...] with a cost of 0 of 1 (0%). The right input is an Index Seek (NonClustered) on [salesorderdetail].[IX_SalesOrderDe...] with a cost of 0 of 4 (0%). The final output is a SELECT statement.

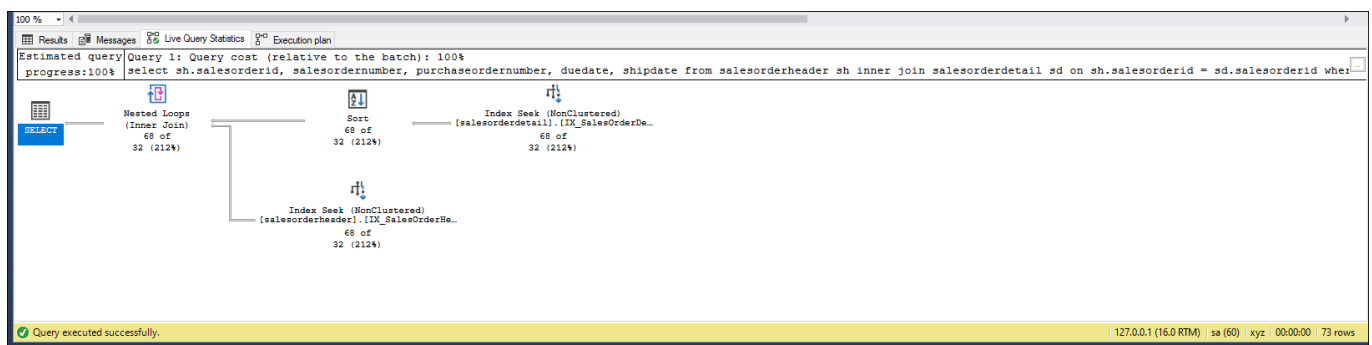
Query executed successfully.

Zapytanie 4

Stary Execution Plan



Nowy Execution Plan



Jak widać główną różnicą jest zastąpienie *table scans* przez *index scan* które jest dużo szybsze i efektywniejsze. Widzimy także że zamiast *hash join* korzystamy z *nested loop*. Na ogół ten pierwszy jest szybszy od tego drugiego, natomiast *hash join* nie korzysta z indeksów w przeciwieństwie do *nested loops*. Dlatego po stworzeniu indeksów optymalizator zmienił sposób wykonywania joinów.

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?

Wyniki

SQLQuery2.sql - 12...Works2017 (sa (66)) * ~vs31F9.sql - 127.0.0.1xyz (sa (58)) * R X

```

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

```

100 % Results Messages

	database_id	object_id	index_id	partition_number	index_type_desc	alloc_unit_type_desc	index_depth	index_level	avg_fragmentation_in_percent	fragment_count	avg_fragment_size_in_pages	page_count	avg_page_space_used_in_percent	record_count	ghost_n
1	8	2059048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	50	2	1	2	74.1536940943909	316	0
2	8	2059048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.494193229552755	2	0
3	8	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	6	14.3333333333333	86	99.2700642451198	27647	0
4	8	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	18.0380528796756	86	0
5	8	62623266	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	16.3577958881962	17	0
6	8	62623266	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	5.43612552508031	17	0
7	8	66099276	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	1	1	1	4.78131949592291	5	0
8	8	66099276	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	28	86.6444279713368	27	0
9	8	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	9.09090909090909	2	5.5	11	70.0103409933284	13	0
10	8	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	1.74203113417346	11	0
11	8	98099390	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	4	55.6214479861626	4	0
12	8	98099390	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	2.2238695329874	13	0
13	8	130099504	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	10.3039288361749	14	0
14	8	130099504	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	6.84457622930566	14	0
15	8	226099846	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	3	78.3333333333333	235	99.7255744996293	19972	0
16	8	226099846	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	37.719298245614	235	0
17	8	254623950	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	50	2	1	2	53.3419817148505	163	0
18	8	254623950	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.494193229552755	2	0
19	8	254623950	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	68.4457622930566	163	0
20	8	274100017	1	1	CLUSTERED INDEX	IN_ROW_DATA	3	0	0.183678824455523	161	23.6708074534161	3811	85.6245737583395	19972	0
21	8	274100017	1	1	CLUSTERED INDEX	IN_ROW_DATA	3	1	87.5	8	1	8	76.4872127501853	3811	0
22	8	274100017	1	1	CLUSTERED INDEX	IN_ROW_DATA	3	2	0	1	1	1	1.26019273535953	8	0
23	8	274100017	1	1	CLUSTERED INDEX	ROW_OVERFLOW_DATA	1	0	0	NULL	NULL	1	0	0	0
24	8	274100017	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	1	0	0	0
25	8	274100017	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	0	6.66666666666667	8	13.125	105	97.2379416851989	19972	0
26	8	274100017	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	62.3424759808001	105	0
27	8	274100017	3	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	0	0	7	9.28571428571429	65	98.6755621447986	19972	0
28	8	274100017	3	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	20.051890289103	65	0
29	8	274100017	256000	1	PRIMARY XML INDEX	IN_ROW_DATA	2	0	0	1	3	3	68.2151470224858	195	0
30	8	274100017	256000	1	PRIMARY XML INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.617741536940944	3	0
31	8	274100017	256001	1	PRIMARY XML INDEX	IN_ROW_DATA	3	0	0	2	1076	2152	99.6210155670867	301696	0
32	8	274100017	256001	1	PRIMARY XML INDEX	IN_ROW_DATA	3	1	0	2	3	6	84.848863355572	2152	0
33	8	274100017	256001	1	PRIMARY XML INDEX	IN_ROW_DATA	3	2	0	1	1	1	1.3096120583148	6	0
34	8	274100017	256002	1	XML INDEX	IN_ROW_DATA	3	0	0.503959683225342	52	26.7115384615385	1389	99.4253521126761	301696	0

Query executed successfully. 127.0.0.1 (16.0 RTM) sa (58) xyz 00:00:00 312 rows

Do najważniejszych pól należą:

- **index_id / index_type-desc** - to pole mówi nam z jakim typem indexu (lub sterty) mamy do czynienia
- **alloc_unity_type_desc** - czy mamy doczynienia z *IN_ROW_DATA* lub *LOB_DATA*, ewentualnie *OVERFLOW_ROW_DATA*. W większości przypadków mamy do czynienia z *IN_ROW_DATA*, jeśli dany wiersz przekracza ustaloną wartość (zwykle 8060 bajtów) to część pól jest kopiowana do *OVERFLOW_ROW_DATA*, z ostatnim typem mamy do czynienia jeśli pole jest zdefiniowane jako LOB.
- **index_depth** - głębokość indexu, w przypadku sterty = 1
- **index_level** - aktualny poziom w indexie (wiersze w tej komendzie odpowiadają pojedynczemu poziomowi w B-drzewie)
- **avg_fragmentation_in_percent** - logiczna defragmentacja w przypadku indexów oraz fragmentacja extentów w przypadku sterty
- **page_count** - liczba stron używanych 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
```

Wyniki

The screenshot shows a SQL query executed in SQL Server Enterprise Manager. The query is designed to find indexes with high fragmentation or low page density. The results table shows 5 rows of data.

	table name	index id
1	JobCandidate	1
2	ProductModel	1
3	BillOfMaterials	2
4	WorkOrder	3
5	WorkOrderRouting	2

W bazie Adventure Works mamy 5 tabel z umiarkowaną fragmentacją lub z nieoptymalną gęstością strony. W sytuacji gdy mamy małą gęstość strony, to zwiększamy ilość operacji I/O ale mamy nie musimy się martwić tym że wiersz przestanie się mieścić na stronie. W przypadku wysokiej gęstości, operacje I/O są bardzo optymalne, ale w przypadku gdy wiersz przestanie się mieścić na stronie, musimy taki wiersz splitować na dwie strony co jest nieoptymalne w przypadku dużej ilości operacji insert/update.

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
```

Wyniki

```
SQLQuery2.sql - 12...Works2017 (sa (66))* ~vs31F9.sql - 127.0.0.1.xyz (sa (58))*
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
```

100 %

Results Messages

	table name	index id
1	Person	256002
2	Person	256003
3	Person	256004

Wyniki wskazują, że trzy indeksy na tabeli Person wymagają przebudowy. Może to być spowodowane częstymi modyfikacjami rekordów w tabeli Person, szczególnie z uwagi na obecność kolumny **ModifiedDate**. Dodatkowo, rozmiar tej tabeli jest relatywnie duży, co również może przyczynić się do częstych modyfikacji i potencjalnej fragmentacji indeksów.

Czym się różni przebudowa indeksu od reorganizacji?

(Podpowiedź: <http://blog.plik.pl/2014/12/defragmentacja-indeksow-ms-sql.html>)

Wyniki

Obie operacje - reorganizacja i przebudowa indeksów - mają na celu zmniejszenie poziomu fragmentacji indeksu, jednak różnią się sposobem działania oraz skutkami.

Przebudowa polega na całkowitym zrzuceniu i ponownym zbudowaniu indeksu. Jest to bardziej radykalne podejście, które eliminuje całkowicie fragmentację, jednak jest znacznie bardziej czasochłonne i wymaga większej ilości zasobów. Może być wykonywany online lub offline, z opcją ONLINE pozwalającą na wykonywanie tej operacji w trakcie działania bazy danych, jednakże zaleca się, aby operacja przebudowy odbywała się offline, gdy baza danych nie jest wykorzystywana.


Reorganizacja operuje na poziomie liści B-drzewa. Polega na uporządkowaniu fizycznej kolejności stron, co redukuje fragmentację zewnętrzną, oraz na modyfikacji stron w celu dopasowania gęstości do określonych parametrów fill dla danego indeksu. Jest to operacja wykonywana online, zużywająca mniej zasobów niż przebudowa, jednak może być mniej skuteczna w przypadku silnie zfragmentowanych indeksów.

Źródło: <https://learn.microsoft.com/en-us/sql/relational-databases/indexes/reorganize-and-rebuild-indexes?view=sql-server-ver16&redirectedfrom=MSDN#rebuild-an-index>


Sprawdź co przechowuje tabela sys.dm_db_index_usage_stats:



Wyniki

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

SQLQuery2.sql - 12...Works2017 (sa (66)) * ~vs31F9.sql - 127.0...Works2017 (sa (58)) * 

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

100 % 

 Results  Messages

	database_id	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
1	4	64719283	3	0	4	0	0	NULL	2024-04-03 17:42:26.100	NULL	NULL	0	0	0	0	NULL	NULL
2	8	274100017	1	0	1	0	0	NULL	2024-04-03 17:48:18.813	NULL	NULL	0	0	0	0	NULL	NULL
3	4	1627152842	1	1	0	0	0	2024-04-03 17:37:25.567	NULL	NULL	NULL	0	0	0	0	NULL	NULL
4	4	528720936	1	14	0	0	0	2024-04-03 17:42:26.100	NULL	NULL	NULL	0	0	0	0	NULL	NULL
5	4	1691153070	1	0	1	0	0	NULL	2024-04-03 17:37:25.567	NULL	NULL	0	0	0	0	NULL	NULL
6	9	901578250	0	0	1	0	0	NULL	2024-04-03 17:35:18.390	NULL	NULL	0	0	0	0	NULL	NULL
7	9	917578307	0	0	1	0	0	NULL	2024-04-03 17:35:18.390	NULL	NULL	0	0	0	0	NULL	NULL
8	8	1733581214	1	1	0	0	0	2024-04-03 17:33:21.670	NULL	NULL	NULL	0	0	0	0	NULL	NULL
9	8	98099390	1	1	0	0	0	2024-04-03 17:33:20.877	NULL	NULL	NULL	0	0	0	0	NULL	NULL
10	8	1266103551	1	1	0	0	0	2024-04-03 17:33:20.877	NULL	NULL	NULL	0	0	0	0	NULL	NULL


 Query executed successfully. 127.0.0.1 (16.0 RTM) sa (58) AdventureWorks2017 00:00:00 10 rows

Tabela przechowuje informacje o użyciu indeksów bazy danych. Zawiera statystyki dotyczące operacji odczytu i zapisu dla indeksów w bazie danych. Dane tej tabeli mogą być przydatne do analizy i optymalizacji wydajności bazy danych poprzez identyfikację rzadko używanych indeksów lub takich, które wymagają aktualizacji statystyk.

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
```

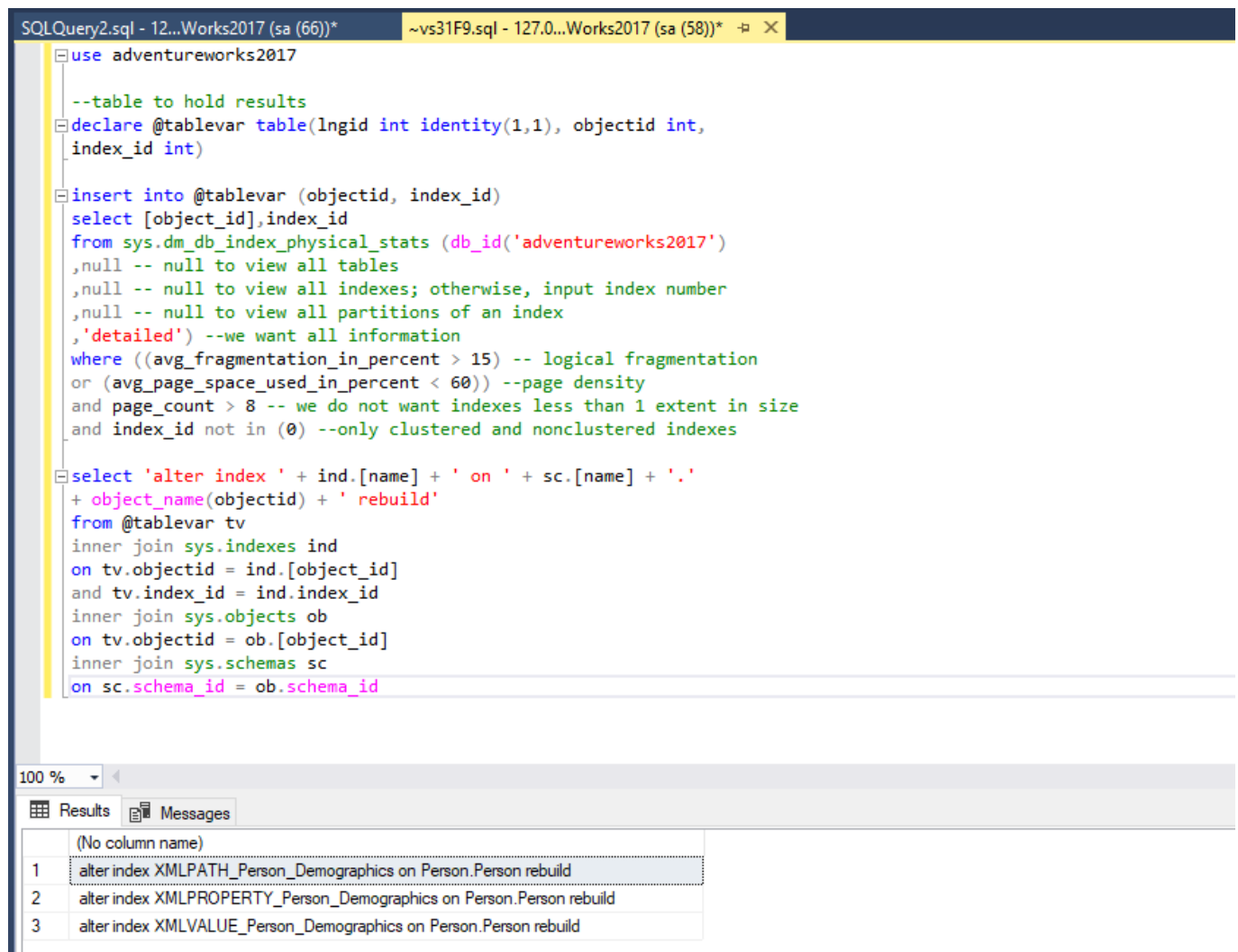
```
and page_count > 8 -- we do not want indexes less than 1 extent in size
and index_id not in (0) --only clustered and nonclustered indexes

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:

Wyniki

Przebudowa



The screenshot shows a SQL query in SQL Server Enterprise Manager. The query is designed to identify and rebuild indexes that are fragmented, have low page density, or are small. It uses a temporary table to store the results of a query that filters indexes based on these criteria. The final step is to generate and execute a series of 'ALTER INDEX ... REBUILD' statements for each identified index.

```
SQLQuery2.sql - 12...Works2017 (sa (66))* ~vs31F9.sql - 127.0...Works2017 (sa (58))* -P X
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
```

100 %

Results Messages

	(No column name)
1	alter index XMLPATH_Person_Demographics on Person.Person rebuild
2	alter index XMLPROPERTY_Person_Demographics on Person.Person rebuild
3	alter index XMLVALUE_Person_Demographics on Person.Person rebuild


```
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

```
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 > 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

select 'alter index ' + ind.[name] + ' on ' + sc.[name] + '.'
+ object_name(objectid) + ' reorganize'
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
```

~vsCCA7.sql - 127.0.0.1.master (sa (73)) ~vs8A53.sql - 127.0...Works2017 (sa (59))

```

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 > 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

select 'alter index ' + ind.[name] + ' on ' + sc.[name] + ','
+ object_name(objectid) + ' reorganize'
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

```

100 %

Results Messages

	(No column name)
1	alter index PK_JobCandidate_JobCandidateID on HumanResources.JobCandidate reorganize
2	alter index PK_ProductModel_ProductModelID on Production.ProductModel reorganize
3	alter index PK_BillOfMaterials_BillOfMaterialsID on Production.BillOfMaterials reorganize
4	alter index IX_WorkOrder_ProductID on Production.WorkOrder reorganize
5	alter index IX_WorkOrderRouting_ProductID on Production.WorkOrderRouting reorganize

```

alter index IX_WorkOrderRouting_ProductID on Production.WorkOrderRouting
reorganize
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

```

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:

Wyniki

Index 1

SQLQuery2.sql - 12...Works2017 (sa (66))*

~vs31F9.sql - 127.0...Works2017 (sa (58))*

dbcc ind ('adventureworks2017', 'person.address', 1)

100 %

Results Messages

	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
11	1	11721	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11722	1	11720
12	1	11722	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11723	1	11721
13	1	11723	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11724	1	11722
14	1	11724	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11725	1	11723
15	1	11725	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11726	1	11724
16	1	11726	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11727	1	11725
17	1	11727	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11728	1	11726
18	1	11728	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11729	1	11727
19	1	11729	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11730	1	11728
20	1	11730	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11731	1	11729
21	1	11731	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11732	1	11730
22	1	11732	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11733	1	11731
23	1	11733	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11734	1	11732
24	1	11734	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11735	1	11733
25	1	11735	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11736	1	11734
26	1	11736	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11737	1	11735
27	1	11737	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11738	1	11736
28	1	11738	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11739	1	11737
29	1	11739	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11740	1	11738
30	1	11740	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11741	1	11739
31	1	11741	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11742	1	11740
32	1	11742	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11743	1	11741
33	1	11743	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11968	1	11742
34	1	11744	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11745	1	12074
35	1	11745	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11746	1	11744
36	1	11746	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11747	1	11745
37	1	11747	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11748	1	11746
38	1	11748	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0	1	11749	1	11747

Query executed successfully.

127.0.0.1 (16.0 RTM) sa (58) AdventureWorks2017 00:00:00 350 rows

Index 2

SQLQuery2.sql - 12...Works2017 (sa (66))*

~vs31F9.sql - 127.0...Works2017 (sa (58))*✕

dbcc ind ('adventureworks2017', 'person.address', 2)

PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10472	NULL	1029578706	2	1	72057594052542464	Inrow data	10	NULL	0	0	0	0
2	1	5872	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5873	0
3	1	5873	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5874	1
4	1	5874	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5875	1
5	1	5875	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5876	1
6	1	5876	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5877	1
7	1	5877	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5878	1
8	1	5878	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5879	1
9	1	5879	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5880	1
10	1	5880	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5881	1
11	1	5881	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5882	1
12	1	5882	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5883	1
13	1	5883	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5884	1
14	1	5884	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5885	1
15	1	5885	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5886	1
16	1	5886	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5887	1
17	1	5887	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5888	1
18	1	5888	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5889	1
19	1	5889	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5890	1
20	1	5890	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5891	1
21	1	5891	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5892	1
22	1	5892	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5893	1
23	1	5893	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5894	1
24	1	5894	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5895	1
25	1	5895	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5896	1
26	1	5896	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5897	1
27	1	5897	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5898	1
28	1	5898	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5899	1
29	1	5899	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5900	1
30	1	5900	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5901	1
31	1	5901	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5902	1
32	1	5902	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	5903	1
33	1	5903	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	8472	1
34	1	5904	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	1	0	0	0
35	1	8472	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	8473	1
36	1	8473	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	8474	1
37	1	8474	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	8475	1
38	1	8475	1	10472	1029578706	2	1	72057594052542464	Inrow data	2	0	1	8476	1

Query executed successfully.127.0.0.1 (16.0 RTM) | sa (58) | AdventureWorks2017 | 00:00:00 | 66 rows

Index 3

SQLQuery2.sql - 12...Works2017 (sa (66))*

~vs31F9.sql - 127.0...Works2017 (sa (58))*✕

dbcc ind ('adventureworks2017', 'person.address', 3)

PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10473	NULL	1029578706	3	1	72057594052608000	Inrow data	10	NULL	0	0	0	0
2	1	5920	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5921	0
3	1	5921	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5922	1
4	1	5922	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5923	1
5	1	5923	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5924	1
6	1	5924	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5925	1
7	1	5925	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5926	1
8	1	5926	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5927	1
9	1	5927	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5928	1
10	1	5928	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5929	1
11	1	5929	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5930	1
12	1	5930	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5931	1
13	1	5931	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5932	1
14	1	5932	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5933	1
15	1	5933	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5934	1
16	1	5934	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	5935	1
17	1	5935	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8464	1
18	1	8464	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8465	1
19	1	8465	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8466	1
20	1	8466	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8467	1
21	1	8467	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8468	1
22	1	8468	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8469	1
23	1	8469	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8470	1
24	1	8470	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8471	1
25	1	8471	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8528	1
26	1	8528	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8529	1
27	1	8529	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8530	1
28	1	8530	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8531	1
29	1	8531	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8532	1
30	1	8532	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8533	1
31	1	8533	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8534	1
32	1	8534	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8535	1
33	1	8535	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8568	1
34	1	8536	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	1	1	8537	0
35	1	8537	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	1	1	8539	1
36	1	8538	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	2	0	0	0
37	1	8539	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	1	0	0	1
38	1	8568	1	10473	1029578706	3	1	72057594052608000	Inrow data	2	0	1	8569	1

Query executed successfully.127.0.0.1 (16.0 RTM) | sa (58) | AdventureWorks2017 | 00:00:00 | 216 rows

Uzyskano strony o typach: 1, 2, 3 i 10.

Strona danych (Data Page - PageType 1)

Przechowuje rzeczywiste dane tabeli, takie jak wiersze. Wysokość strony wynosi zazwyczaj 8 KB. Te strony są również znane jako strony danych wierszy.

Strona indeksu (Index Page - PageType 2)

Przechowuje dane indeksu dla danej tabeli. Każdy indeks posiada swoje własne strony indeksu. Strony indeksów zawierają odwołania do rzeczywistych danych lub do innych stron indeksu.

Strona zaalokowana dla indeksu (Index Allocation Map (IAM) Page - PageType 3)

Przechowuje informacje na temat innych stron, które są zaalokowane dla danego indeksu. Umożliwia systemowi zarządzania bazą danych śledzenie używanych i dostępnych stron w indeksie.

Strona danych lub indeksu (Data or Index Page - PageType 10)

Jest to specjalny typ strony, który może być zarówno stroną danych, jak i stroną indeksu. Zwykle używany w przypadku, gdy strona może zawierać zarówno dane, jak i indeksy, co jest częstym przypadkiem w przypadku strony liścia klastra.

Włącz flagę 3604 zanim zaczniesz przeglądać strony:

```
dbcc traceon (3604);
```

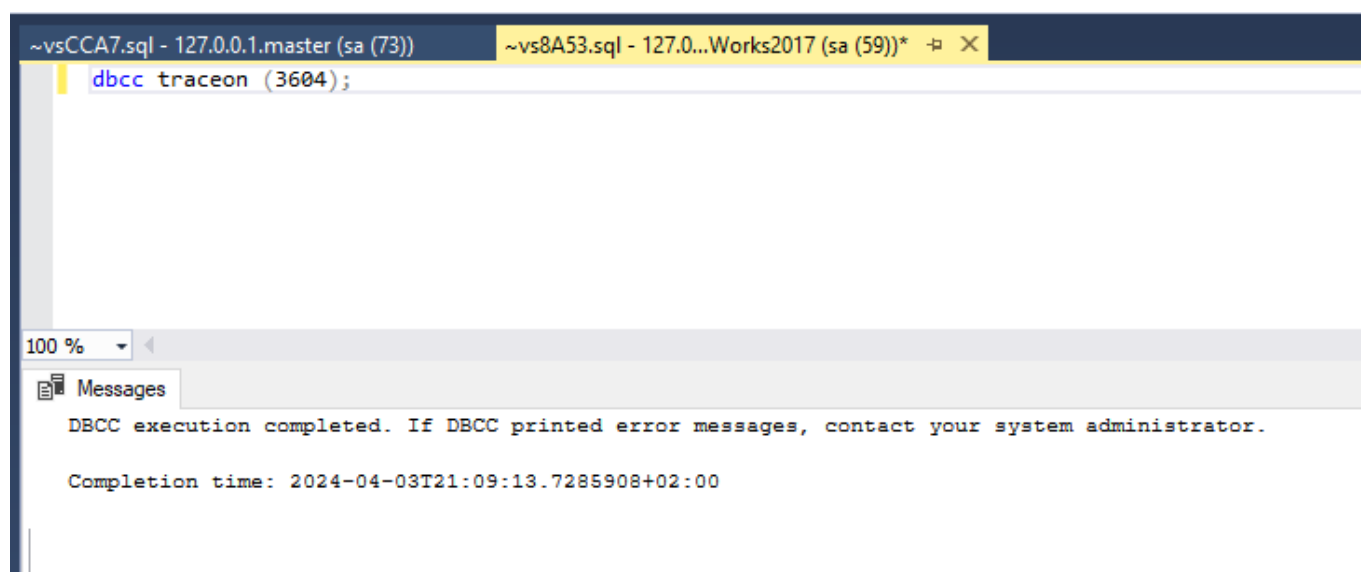
Sprawdź poszczególne strony komendą DBCC PAGE. np.:

```
dbcc page('adventureworks2017', 1, 13720, 3);
```

Zapisz obserwacje ze stron. Co ciekawego udało się zaobserwować?

Wyniki

Włączenie flagi



```
~vsCCA7.sql - 127.0.0.1.master (sa (73))  ~vs8A53.sql - 127.0...Works2017 (sa (59))*  dbcc traceon (3604);
```

100 %

Messages

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

Completion time: 2024-04-03T21:09:13.7285908+02:00

Sprawdzanie stron

```
SQLQuery4.sql - 12...Works2017 (sa (75))* x SQLQuery3.sql - 12...Works2017 (sa (73))* SQLQuery2.sql - 12...Works2017 (sa (66))* --vs31F9.sql - 127.0...Works2017 (sa (58))*
dbcc page('adventureworks2017', 1, 13720, 3);

100 %
Messages

PAGE: (1:13720)

BUFFER:

BUF 0x0000001A58E168F40

bpage = 0x0000001A1B57C2000 bPmpage = 0x00000000000000000 bsort_r_nextbP = 0x00000000000000000
bsort_r_prevbP = 0x00000000000000000 bhash = 0x00000000000000000 bpageNo = (1:13720)
bpart = 1 bstat = 0x9 bstat2 = 0x0 bstat3 = 0x0
berrcode = 0 blog = 0x15a bsampleCount = 0 bIoCount = 0
resPoolId = 0 bDirtyPendingCount = 0 bDirtyContext = 0x00000000000000000 bReadMicroSec = 1272
bDbid = 8 bpru = 0x0000001A18F670040 bDbPageBroker = 0x00000000000000000

PAGE HEADER:

Page 0x0000001A1B57C2000

m_pageId = (1:13720) m_headerVersion = 1 m_type = 1
m_typeFlagBits = 0x0 m_level = 0 m_flagBits = 0x220
m_objId (AllocUnitId.idObj) = 297 m_indexId (AllocUnitId.idInd) = 256 Metadata: AllocUnitId = 72057594057392128
Metadata: PartitionId = 72057594049658880 Metadata: IndexId = 1
Metadata: ObjectId = 274100017 m_prevPage = (1:13719) m_nextPage = (1:13721)
pminlen = 41 m_slotCnt = 5 m_freeCnt = 1188
m_freeData = 6994 m_reservedCnt = 0 m_lsn = (37:2554:337)
m_xactReserved = 0 m_xdesId = (0:1586) m_ghostRecCnt = 0
m_tornBits = -1132865352 DB Frag ID = 1

Allocation Status

GAM (1:2) = ALLOCATED SGAM (1:3) = NOT ALLOCATED PFS (1:8088) = 0x40 ALLOCATED 0_PCT_FULL
DIFF (1:6) = NOT CHANGED ML (1:7) = NOT MIN_LOGGED

Slot 0 Offset 0x60 Length 1378

Record Type = PRIMARY_RECORD Record Attributes = NULL_BITMAP VARIABLE_COLUMNS
Record Size = 1378
Memory Dump 0x00000006A568F600

0000000000000000: 30003900 471a0000 49004e00 00000000 00905c98 0..I.N.....\
0000000000000014: 5d1cd556 428c77bf 62247c0c 5c000000 0056a300 1.0Vbwjbf\.....UL
0000000000000028: 00d00088 0207003d 00450047 00550055 00550062 .....=..E.G.U.U.b
000000000000003C: 054e0065 0069006c 004d0043 00610072 006c0073 ..N.e.i.l.M.C.a.z.i.s
0000000000000050: 006f006e 0dfff202 b04ea05 002a0001 00f01049 ..n.Bf".a.....8.I
0000000000000064: 006e0064 00690076 00690064 00760061 006c0063 ..n.d.i.v.i.d.u.a.i.s
0000000000000078: 00750072 00760065 00790060 42680074 00740070 ..u.r.e.y..80b.t.e.p

100 %
Query executed successfully. 127.0.0.1 (16.0 RTM) sa (75) AdventureWorks2017 00:00:00 0 rows
```

```
SQLQuery4.sql - 12...Works2017 (sa (75))* x SQLQuery3.sql - 12...Works2017 (sa (73))* SQLQuery2.sql - 12...Works2017 (sa (66))* --vs31F9.sql - 127.0...Works2017 (sa (58))*
dbcc page('adventureworks2017', 1, 13720, 2);

100 %
Messages

PAGE: (1:13720)

BUFFER:

BUF 0x0000001A58E168F40

bpage = 0x0000001A1B57C2000 bPmpage = 0x00000000000000000 bsort_r_nextbP = 0x00000000000000000
bsort_r_prevbP = 0x00000000000000000 bhash = 0x00000000000000000 bpageNo = (1:13720)
bpart = 1 bstat = 0x9 bstat2 = 0x0 bstat3 = 0x0
berrcode = 0 blog = 0x15a bsampleCount = 0 bIoCount = 0
resPoolId = 0 bDirtyPendingCount = 0 bDirtyContext = 0x00000000000000000 bReadMicroSec = 1272
bDbid = 8 bpru = 0x0000001A18F670040 bDbPageBroker = 0x00000000000000000

PAGE HEADER:

Page 0x0000001A1B57C2000

m_pageId = (1:13720) m_headerVersion = 1 m_type = 1
m_typeFlagBits = 0x0 m_level = 0 m_flagBits = 0x220
m_objId (AllocUnitId.idObj) = 297 m_indexId (AllocUnitId.idInd) = 256 Metadata: AllocUnitId = 72057594057392128
Metadata: PartitionId = 72057594049658880 Metadata: IndexId = 1
Metadata: ObjectId = 274100017 m_prevPage = (1:13719) m_nextPage = (1:13721)
pminlen = 41 m_slotCnt = 5 m_freeCnt = 1188
m_freeData = 6994 m_reservedCnt = 0 m_lsn = (37:2554:337)
m_xactReserved = 0 m_xdesId = (0:1586) m_ghostRecCnt = 0
m_tornBits = -1132865352 DB Frag ID = 1

Allocation Status

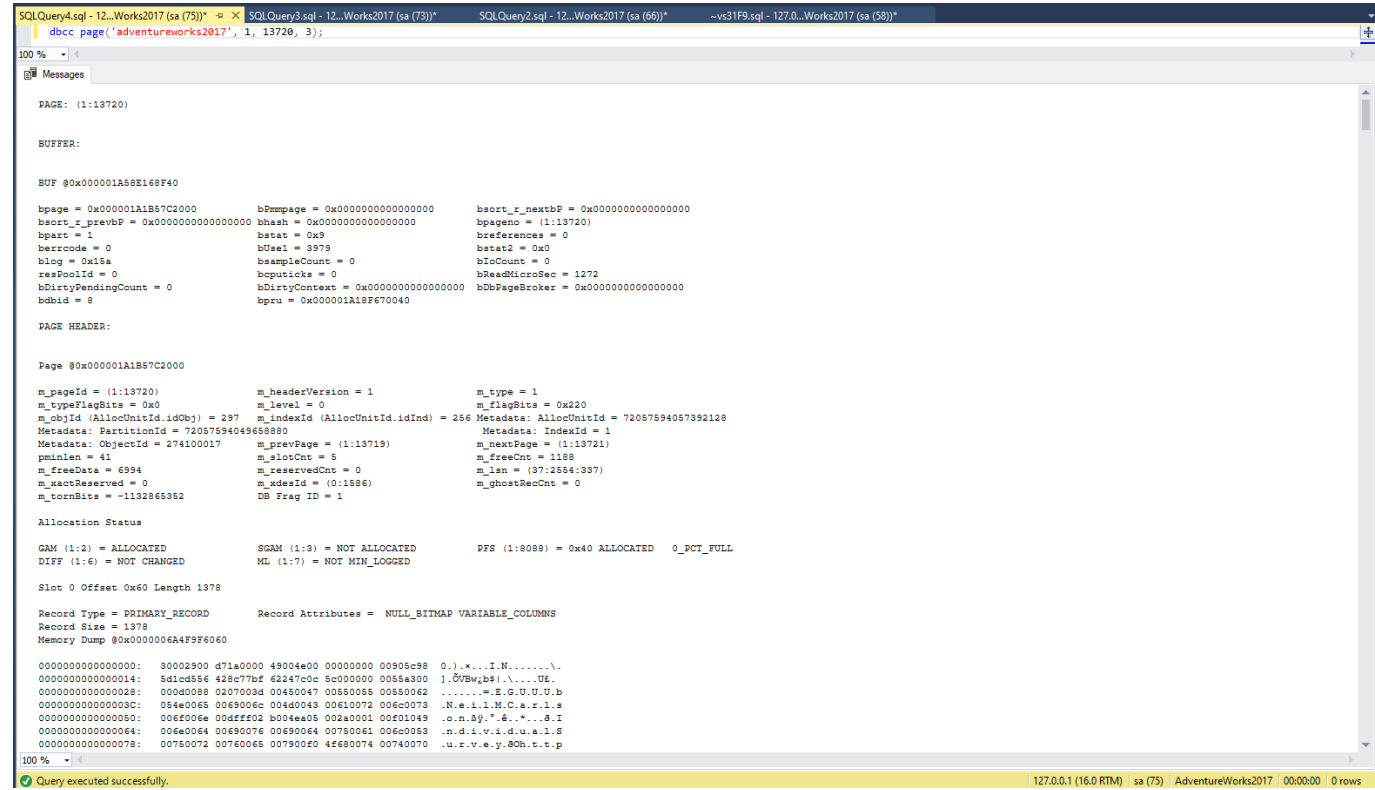
GAM (1:2) = ALLOCATED SGAM (1:3) = NOT ALLOCATED PFS (1:8088) = 0x40 ALLOCATED 0_PCT_FULL
DIFF (1:6) = NOT CHANGED ML (1:7) = NOT MIN_LOGGED

DATA:

Memory Dump 0x00000006A4E9F600

00000006A4E9F600: 01010000 20020001 97350000 01002900 99350000 ....5....).5..
00000006A4E9F6014: 01000500 29010000 404521b 98350000 01000000 .....M.R..S.....
00000006A4E9F6028: 25000000 f0900000 51010000 32060000 00000000 .....Q...2.....
00000006A4E9F603C: b0d879bc 00000000 00000000 00000000 00000000 ..9%.....
00000006A4E9F6050: 00000000 00000000 00000000 00000000 00002900 .....0.....
00000006A4E9F6064: 471a0000 49004e00 00000000 00905c98 5d1cd556 ....I.N.....\0V
00000006A4E9F6078: 428c77bf 62247c0c 5c000000 0056a300 00d00088 Bvjbf\.....UL.....
00000006A4E9F608C: 0207003d 00450047 00550055 00550062 054e0065 .....E.G.U.U.b.N.e

100 %
Query executed successfully. 127.0.0.1 (16.0 RTM) sa (75) AdventureWorks2017 00:00:00 0 rows
```



Podczas analizy zauważyliśmy cztery ciekawe obserwacje:

Informacje o stronie: Na początku raportu uzyskaliśmy ogólne informacje o stronie, takie jak jej identyfikator (PAGE: (1:13720)) oraz dane bufora (BUFFER). Informacje o buforze zawierają szczegóły dotyczące sposobu przechowywania strony w pamięci podręcznej, takie jak adresy, statystyki odczytu/zapisu oraz stan błędu.

Nagłówek strony: Zawiera metadane dotyczące strony, takie jak identyfikator strony, wersja nagłówka, typ strony, poziom w hierarchii, flagi, informacje o obiekcie, indeksie, poprzedniej i następnej stronie, ilość slotów, ilość danych wolnych oraz zarezerwowane dane.

Status alokacji: W sekcji "Allocation Status" znajdują się informacje o alokacji miejsca w różnych strukturach bazy danych, takich jak GAM (Global Allocation Map), SGAM (Shared Global Allocation Map), PFS (Page Free Space), DIFF (Differential Changed Map) oraz ML (Minimum Log Map). Te informacje są przydatne do zarządzania przestrzenią w plikach bazy danych.

Dane: Widzimy, że na stronie przechowywane są rzeczywiste dane (coś w rodzaju zrzutu pamięci). Te dane mają format szesnastkowy.

Punktacja:

zadanie	pkt
1	3
2	3
3	3
4	1

razem	10
-------	----