Indeksy, optymalizator
Lab 4

Imię i nazwisko:

- **Szymon Budziak**
- **Piotr Ludynia**

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):

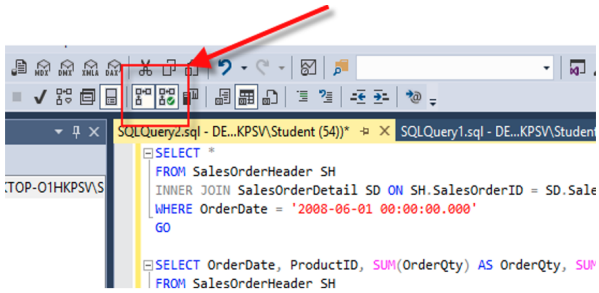
```
-- zapytanie 1
select *
from salesorderheader sh
inner join salesorderdetail sd on sh.salesorderid = sd.salesorderid
where orderdate = '2008-06-01 00:00:00.000'
go

-- 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)

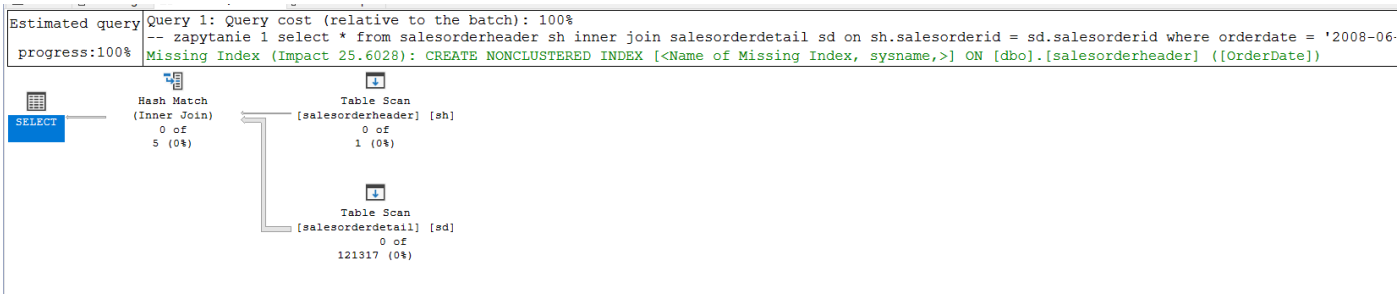
> Wyniki:

Zapytanie 1.

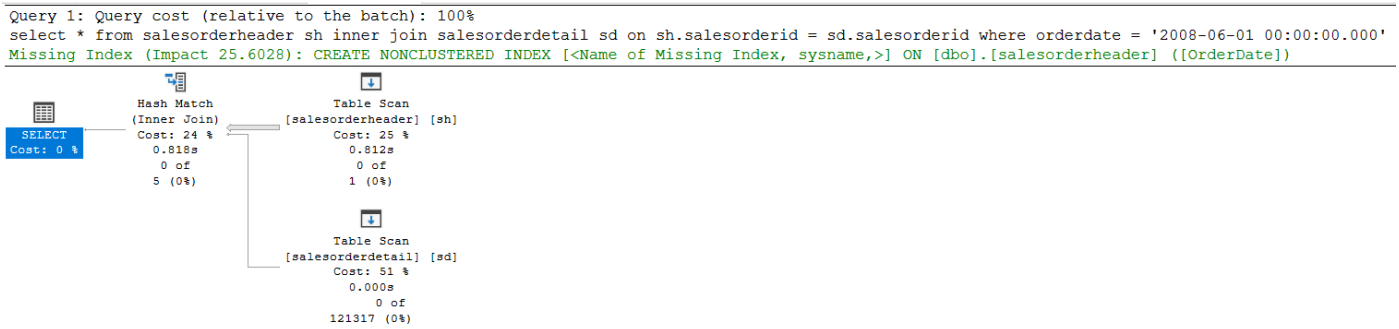
```
-- zapytanie 1
select *
from salesorderheader sh
inner join salesorderdetail sd on sh.salesorderid = sd.salesorderid
where orderdate = '2008-06-01 00:00:00.000'
go
```

Zapytanie 1 nie zwraca żadnych wyników przez filtrację po dacie. Celem zapytania jest zwrócenie wszystkich danych o sprzedaży z tabeli salesorderheader i salesorderdetail o sprecyzowanej wartości pola daty - `2008-06-01 00:00:00.000`, jednak żaden rekord nie odpowiada kryteriom. Ale dalej możemy zbudować plan wykonania zapytania. Widzimy, że pomimo braku odpowiednich rekordów zapytanie dalej wykonuje pełny skan tabeli salesorderdetail (121317 elementów)!

Statystyki



Plan i czas wykonania



Optymalizacja

Potencjalną optymalizacją takiego zapytania byłoby stworzenie indeksu na tabeli salesorderdetail lub nawet na obu tabelach by uniknąć skanów. Oba indeksy powinny zawierać klucz użyty w operacji join - `salesorderid`.

Zapytanie 2.

```
-- 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 drugie wyciąga datę zamówienia, id produktu, liczbę zamówionych produktów, zniżkę i zsumowaną wartość linetotal. Zapytanie jest wykonane z joinem, który łączy je tabelą salesorderdetail by pogrupować zamówienia według daty oraz id produktu, a także by dostarczyć informacji o liczbie zamówionych jednostek.

Czas wykonania zapytania jest dość długi, bo wynosi 30 sekund. Analizując plan zapytania możemy zobaczyć dwa pełne skany tabel, które przyczyniają się do kosztowności zapytania.

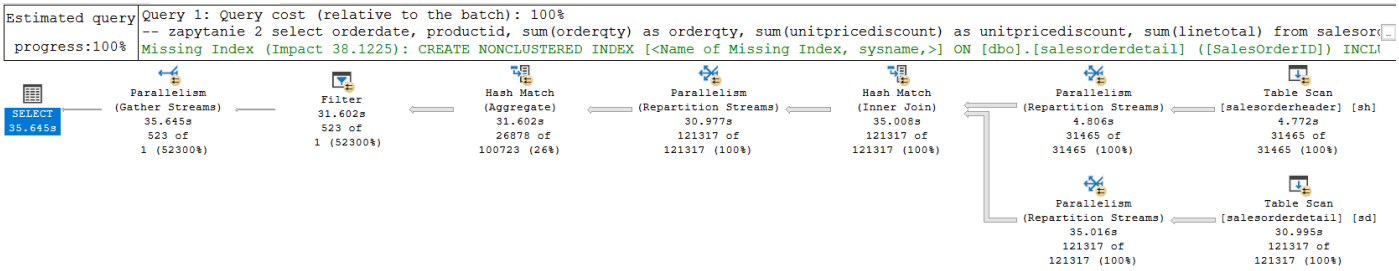
Wynik zapytania

Results | Messages | Live Query Statistics | Execution plan

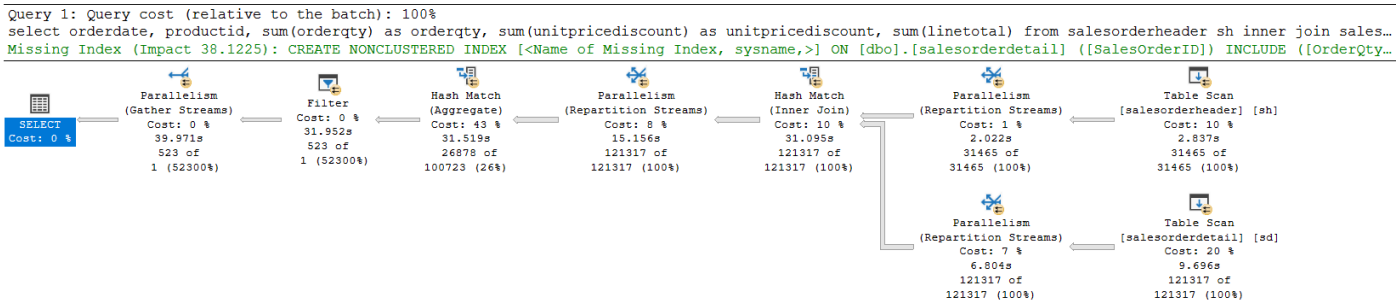
	orderdate	productid	orderqty	unitpricediscount	(No column name)
1	2012-06-30 00:00:00.000	763	144	0.02	67378.168676
2	2013-03-30 00:00:00.000	826	103	0.00	6956.517000
3	2013-07-31 00:00:00.000	937	116	0.00	5636.904000
4	2013-10-30 00:00:00.000	998	106	0.05	33889.772400
5	2013-10-30 00:00:00.000	875	180	0.19	932.733177
6	2013-04-30 00:00:00.000	849	113	0.00	4067.322000
7	2013-09-30 00:00:00.000	876	294	0.17	20668.512000
8	2013-03-30 00:00:00.000	760	140	0.02	65762.703708
9	2013-12-31 00:00:00.000	715	123	0.00	3689.262000
10	2012-08-30 00:00:00.000	765	107	0.02	50284.244192
11	2014-05-01 00:00:00.000	881	132	0.02	4300.433076
12	2013-03-30 00:00:00.000	763	121	0.00	57158.270000
13	2014-03-31 00:00:00.000	998	112	0.00	36719.320000
14	2014-01-29 00:00:00.000	870	125	0.00	406.186000
15	2013-03-30 00:00:00.000	863	358	0.68	7090.168675
16	2012-05-30 00:00:00.000	714	128	0.04	3653.596700
17	2013-12-31 00:00:00.000	712	127	0.00	717.402000
18	2013-08-30 00:00:00.000	870	207	0.11	631.979508

Query executed successfully.

Statystyki



Plan i czas wykonania



Optymalizacja

Takie zapytanie można by zoptymalizować indeksem na salesorderheader. Ważne by taki indeks zawierał pozycje użyte w zapytaniu - `orderdate`, `productid`, `orderqty`, `unitpricediscount`, `linetotal`.

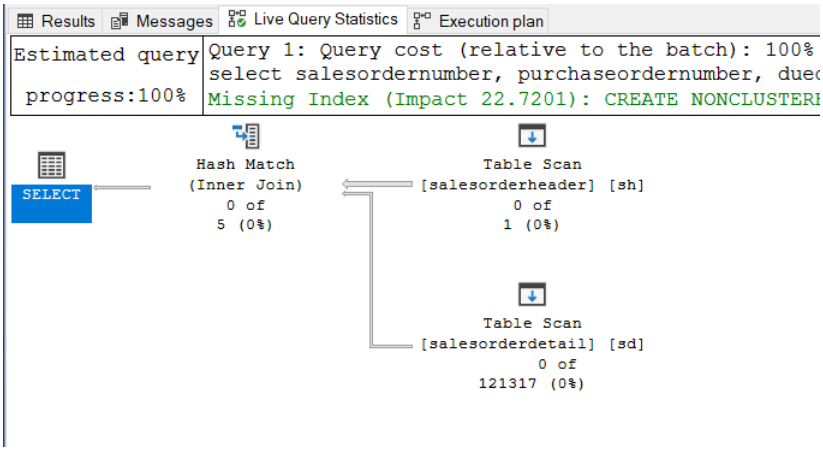
Zapytanie 3.

```
-- 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
```

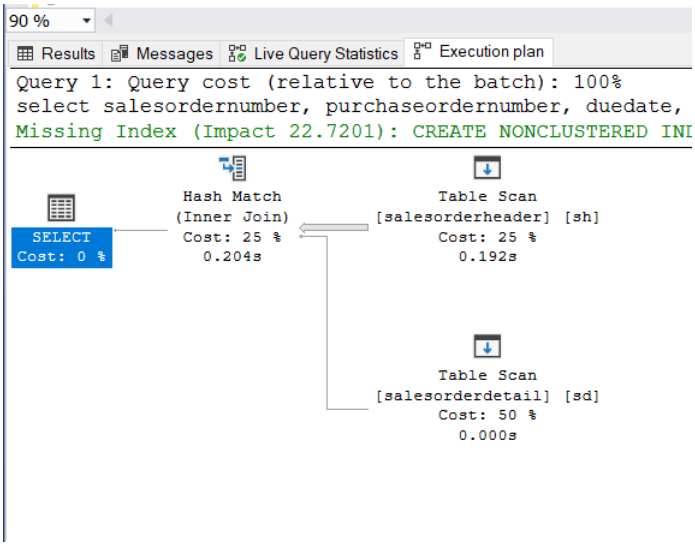
Kolejne zapytanie również filtruje datę tak, że w wyniku nie dostajemy żadnych rekordów. Wynik jest podobny do wyniku z zapytania pierwszego. Tym razem dostępne 4 daty zamówienia oraz zwracamy tylko numer zamówienia i kupna, oraz pola duedate i shipdate.

Ma bardzo prosty plan wykonania, jednak tak jak w 1 zapytaniu wykonuje ono kosztowny skan całej tabel:

Statystyki



Plan i czas wykonania



Optymalizacja

Przez analogię do zapytania pierwszego moglibyśmy zoptymalizować zapytanie dodając do niego indeks na obu tabelach jednak należy uważać by uwzględnił on użyte kolumny dla salesorderheader - `salesordernumber`, `purchaseordernumber`, `duedate`, `shipdate`.

Zapytanie 4.

```
-- 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
```

Zapytanie 4 jest podobne do 3. Różnice między nimi są takie, że 4 nie filtruje po dacie, a zamist tego wyświetla rekordy, które odpowiadają numerom śledzenia (carriertrackingnumber). Kolejną różnicą jest sortowanie po nowym, pierwszym wierszu tabeli czyli id zamówienia.

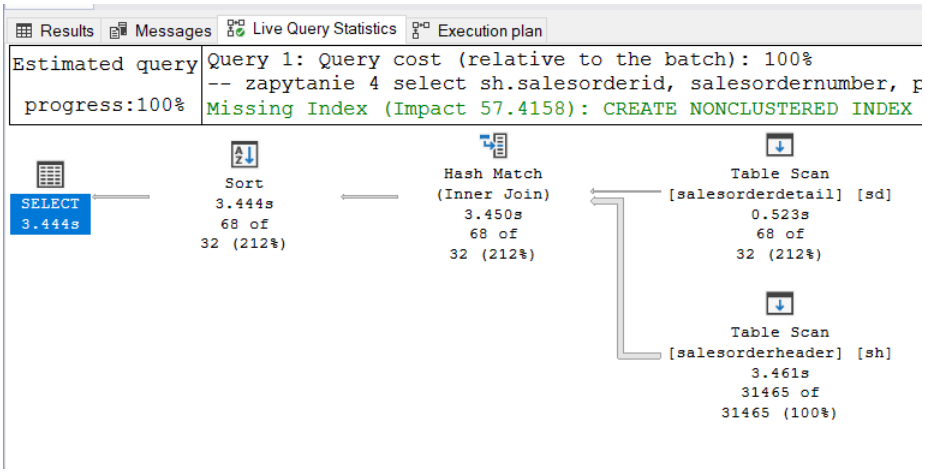
Wynik zapytania

Results | Messages | Live Query Statistics | Execution plan

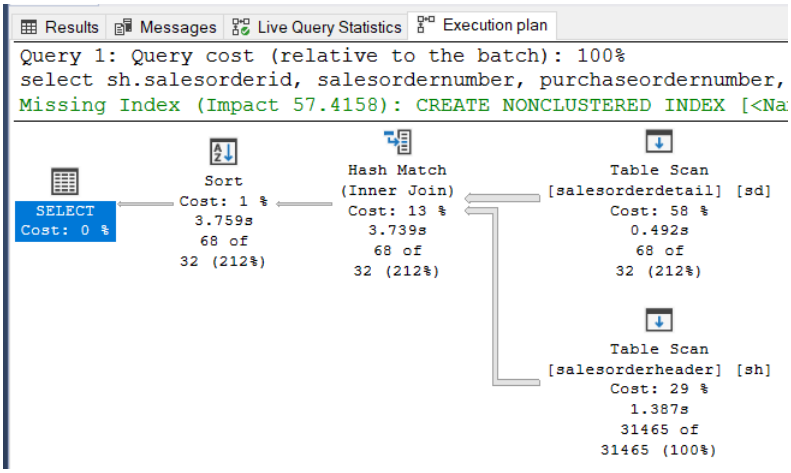
	salesorderid	salesordernumber	purchaseordernumber	duedate	shipdate
1	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
2	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
3	49501	SO49501	PO17574111786	2013-02-09 00:00:00.000	2013-02-04 00:00:00.000
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

Plan wykonania zapytania wygląda następująco

Statystyki



Plan i czas wykonania



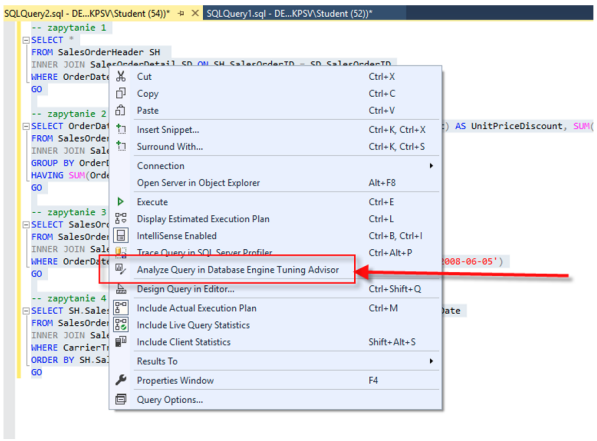
Optymalizacja

To zapytanie mogłoby zostać zoptymalizowane przez dodanie indeksów które trzymają już posortowane wartości `salesorderid`. Pozwoliłoby to na uniknięcie sortowania widocznego na planie wykonania zapytania.

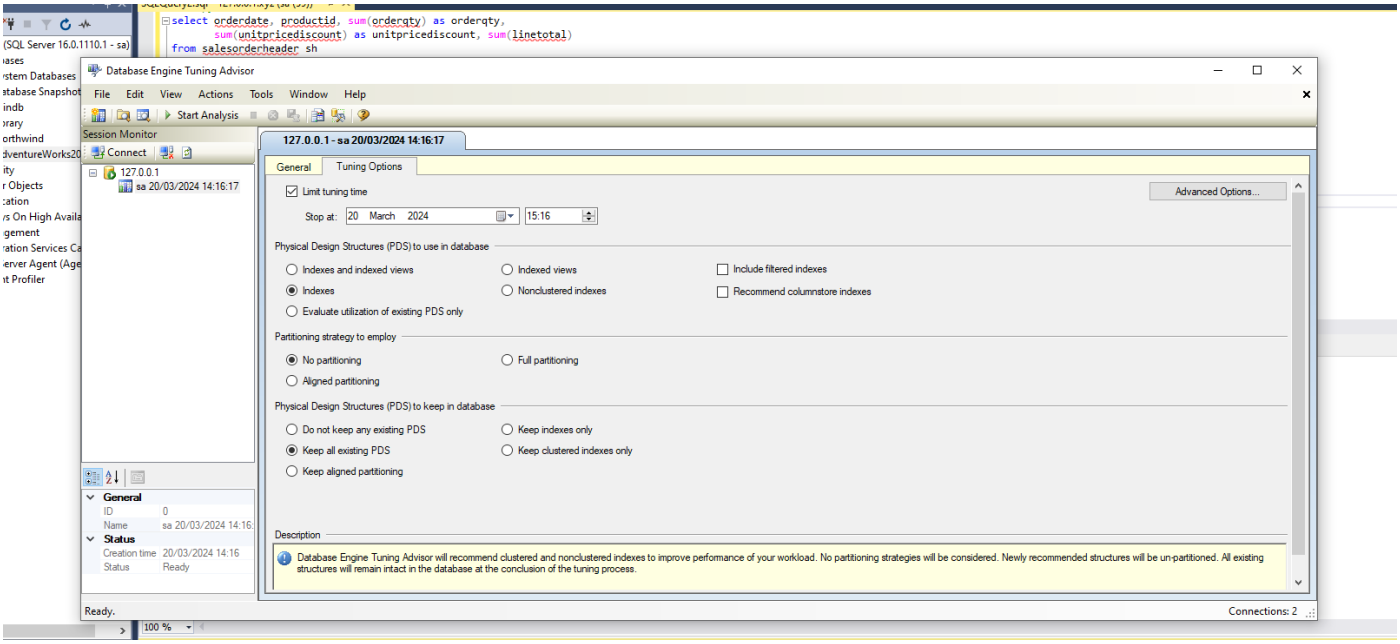
> SSMS sam sugeruje dodanie indeksów nieklastrowych. Plany zapytań zawierają dużo pełnych skanów tabel i sortowań. Takie indeksy mogą pozwolić na optymalizację. W zadaniu drugim zobaczymy jak wyglądają zaporoponowane indeksy i czy pokrywają się z naszymi pomysłami.

Zadanie 2 - Optymalizacja

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

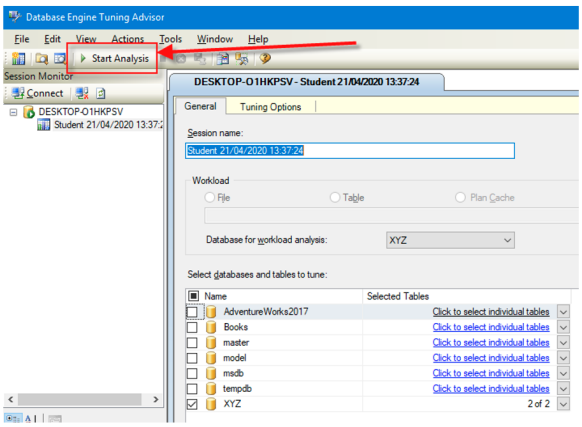


Sprawdź zakładkę **Tuning Options**, co tam można skonfigurować?

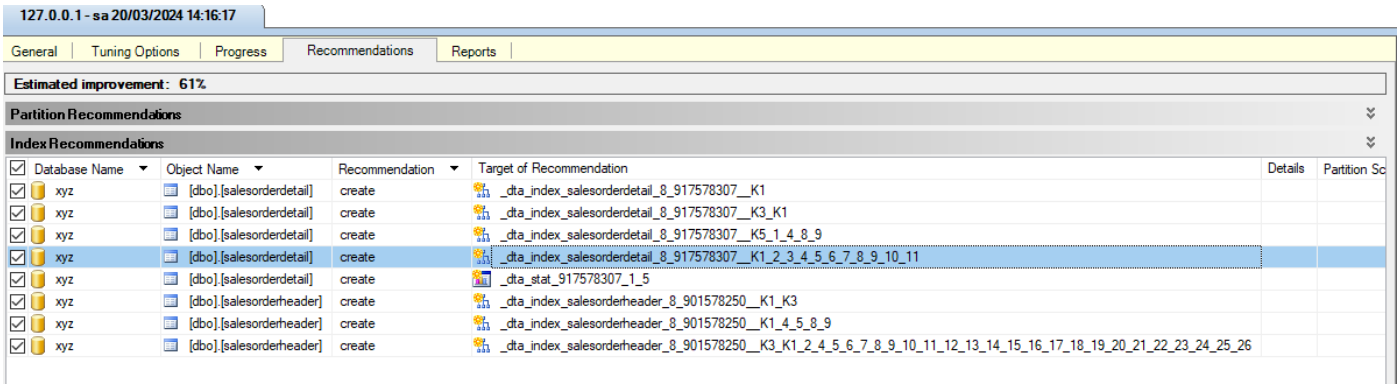


> Wyniki: W zakładce tuning options możemy ustawić struktury PDS, strategię partycji i te struktury PDS któe mają być zachowane. Struktury fizyczne które są dostępne, to indeksy, ich widoki, indeksy bezklastrowe, filtrowane oraz typu columnstore. Zaznaczona konfiguracja używa indeksów jako fizycznych struktur, nie używa partycjonowania oraz zachowuje struktury fizyczne.

Użyj **Start Analysis**:



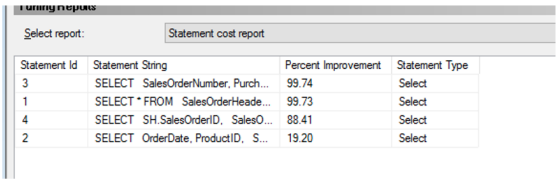
Zaobserwuj wyniki w **Recommendations**.



Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Sc
xyz	[dbo].[salesorderdetail]	create	_dta_index_salesorderdetail_8_917578307__K1		
xyz	[dbo].[salesorderdetail]	create	_dta_index_salesorderdetail_8_917578307__K3_K1		
xyz	[dbo].[salesorderdetail]	create	_dta_index_salesorderdetail_8_917578307__K5_1_4_8_9		
xyz	[dbo].[salesorderdetail]	create	_dta_index_salesorderdetail_8_917578307__K1_2_3_4_5_6_7_8_9_10_11		
xyz	[dbo].[salesorderdetail]	create	_dta_stat_917578307_1_5		
xyz	[dbo].[salesorderheader]	create	_dta_index_salesorderheader_8_901578250__K1_K3		
xyz	[dbo].[salesorderheader]	create	_dta_index_salesorderheader_8_901578250__K1_4_5_8_9		
xyz	[dbo].[salesorderheader]	create	_dta_index_salesorderheader_8_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		

W recomendations, możemy zaobserwować rekomendacje, które podaje nam SSMS.

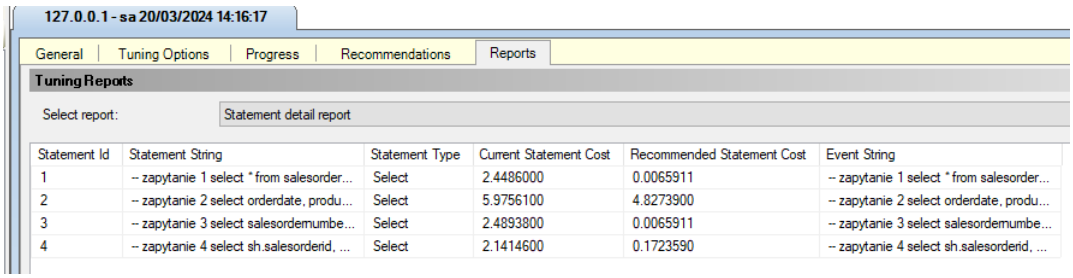
Przejdź do zakładki **Reports**. Sprawdź poszczególne raporty. Główną uwagę zwróć na koszty i ich poprawę:



Select report: Statement cost report

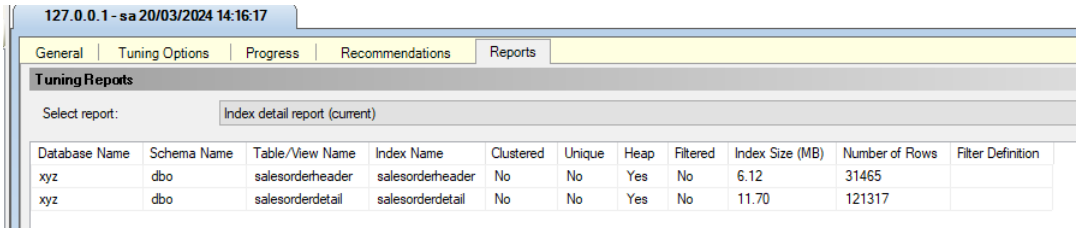
Statement Id	Statement String	Percent Improvement	Statement Type
3	SELECT SalesOrderNumber, Purch...	99.74	Select
1	SELECT * FROM SalesOrderHeade...	99.73	Select
4	SELECT SH.SalesOrderID, SalesO...	88.41	Select
2	SELECT OrderDate, ProductID, S...	19.20	Select

Zapisz poszczególne rekomendacje:



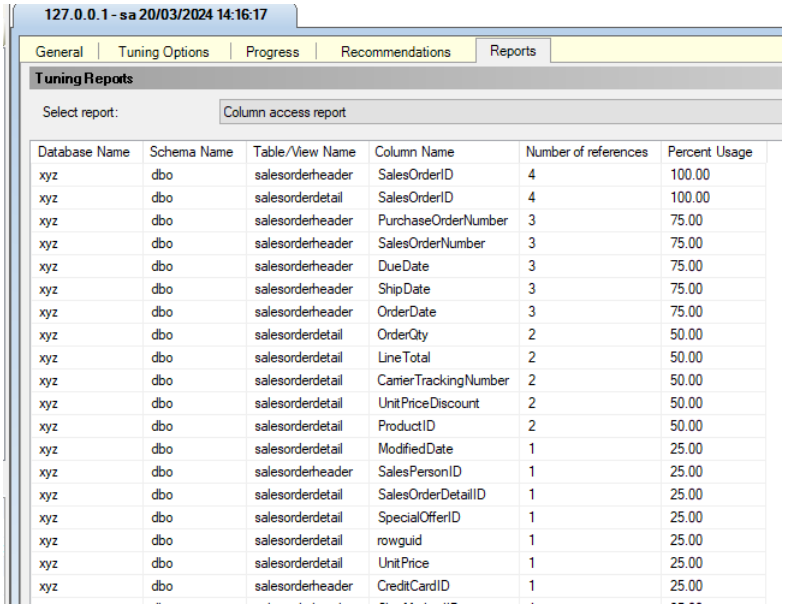
Select report: Statement detail report

Statement Id	Statement String	Statement Type	Current Statement Cost	Recommended Statement Cost	Event String
1	-- zapytanie 1 select * from salesorder...	Select	2.4486000	0.0065911	-- zapytanie 1 select * from salesorder...
2	-- zapytanie 2 select orderdate, produ...	Select	5.9756100	4.8273900	-- zapytanie 2 select orderdate, produ...
3	-- zapytanie 3 select salesordernumbe...	Select	2.4893800	0.0065911	-- zapytanie 3 select salesordernumbe...
4	-- zapytanie 4 select sh.salesorderid, ...	Select	2.1414600	0.1723590	-- zapytanie 4 select sh.salesorderid, ...



Select report: Index detail report (current)

Database Name	Schema Name	Table/View Name	Index Name	Clustered	Unique	Heap	Filtered	Index Size (MB)	Number of Rows	Filter Definition
xyz	dbo	salesorderheader	salesorderheader	No	No	Yes	No	6.12	31465	
xyz	dbo	salesorderdetail	salesorderdetail	No	No	Yes	No	11.70	121317	



Select report: Column access report

Database Name	Schema Name	Table/View Name	Column Name	Number of references	Percent Usage
xyz	dbo	salesorderheader	SalesOrderID	4	100.00
xyz	dbo	salesorderdetail	SalesOrderID	4	100.00
xyz	dbo	salesorderheader	PurchaseOrderNumber	3	75.00
xyz	dbo	salesorderheader	SalesOrderNumber	3	75.00
xyz	dbo	salesorderheader	DueDate	3	75.00
xyz	dbo	salesorderheader	ShipDate	3	75.00
xyz	dbo	salesorderheader	OrderDate	3	75.00
xyz	dbo	salesorderdetail	OrderQty	2	50.00
xyz	dbo	salesorderdetail	LineTotal	2	50.00
xyz	dbo	salesorderdetail	CarrierTrackingNumber	2	50.00
xyz	dbo	salesorderdetail	UnitPriceDiscount	2	50.00
xyz	dbo	salesorderdetail	ProductID	2	50.00
xyz	dbo	salesorderdetail	ModifiedDate	1	25.00
xyz	dbo	salesorderheader	SalesPersonID	1	25.00
xyz	dbo	salesorderdetail	SalesOrderDetailID	1	25.00
xyz	dbo	salesorderdetail	SpecialOfferID	1	25.00
xyz	dbo	salesorderdetail	rowguid	1	25.00
xyz	dbo	salesorderdetail	UnitPrice	1	25.00
xyz	dbo	salesorderheader	CreditCardID	1	25.00

Zaproponowany przez SSMS skrypt wygląda następująco

```
use [xyz]
go
```

```
CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K1_2_3_4_5_6_7_8_9_10_11] 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 [_dta_index_salesorderdetail_8_917578307__K5_1_4_8_9] ON [dbo].[salesorderdetail]
([ProductID] ASC)
INCLUDE(
       [SalesOrderID],
       [OrderQty],
       [UnitPriceDiscount],
       [LineTotal])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go



SET ANSI_PADDING ON
go



CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K3_K1] ON [dbo].[salesorderdetail]
([CarrierTrackingNumber] ASC, [SalesOrderID] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go



CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K1] ON [dbo].[salesorderdetail]
([SalesOrderID] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go



CREATE STATISTICS [_dta_stat_917578307_1_5]
ON [dbo].[salesorderdetail]([SalesOrderID], [ProductID])
WITH AUTO_DROP = OFF
go



CREATE NONCLUSTERED INDEX
[_dta_index_salesorderheader_8_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]
       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 [_dta_index_salesorderheader_8_901578250__K1_4_5_8_9] 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 [_dta_index_salesorderheader_8_901578250__K1_K3] ON [dbo].[salesorderheader]
([SalesOrderID] ASC,[OrderDate] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go
```

Uruchom zapisany skrypt w Management Studio.

Opisz, dlaczego dane indeksy zostały zaproponowane do zapytań:

> Wyniki: Indeksy zostały zaproponowane by zoptymalizować czas wykonywania zapytań. Użycie indeksu pozwala na ograniczenie liczby operacji do wykonania przy przeszukiwaniu danych w tabelach.

> Na tabeli salesorderdetail tworzone są 4 indeksy: Dwa pierwsze z nich mapują ID odpowiednio sprzedaży oraz produktu na kolumny z tabeli. Pozwala to na szybkie wydobycie wartości tych kolumn na podstawie wartości ID. Kolejny indeks sortuje wartości ID sprzedaży oraz numeru przewozowego. Pozwala to na szybkie otrzymanie uszeregowanych wartości i tym samym łatwe zdobywanie informacji o poszczególnych dostawach.

```
CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K1_2_3_4_5_6_7_8_9_10_11] 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 [_dta_index_salesorderdetail_8_917578307__K5_1_4_8_9] ON [dbo].[salesorderdetail]
([ProductID] ASC)
INCLUDE(
       [SalesOrderID],
       [OrderQty],
       [UnitPriceDiscount],
       [LineTotal])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go


CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K3_K1] ON [dbo].[salesorderdetail]
([CarrierTrackingNumber] ASC, [SalesOrderID] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go


CREATE NONCLUSTERED INDEX [_dta_index_salesorderdetail_8_917578307__K1] ON [dbo].[salesorderdetail]
([SalesOrderID] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go
```

> Pozostałe trzy indeksy są tworzone dla salesorderheader. Podobnie jak w poprzednim przypadku tworzymy mapowanie sortowanej daty i ID zamówienia do pozostałych kolumn. Tworzony jest też indeks mapujący ID zamówienia na 4 kolumny osobno. Zawierają one pozostałe informacje o datach istotnych dla zamówienia oraz numery sprzedaży i kupna. Ostatni indeks to sortowane ID sprzedaży oraz data zamówienia. Te trzy indeksy optymalizują zapytania uwzględniające tabelę salesorderheader.

```
CREATE NONCLUSTERED INDEX
[_dta_index_salesorderheader_8_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] 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 [_dta_index_salesorderheader_8_901578250__K1_4_5_8_9] 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 [_dta_index_salesorderheader_8_901578250__K1_K3] ON [dbo].[salesorderheader]
([SalesOrderID] ASC,[OrderDate] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
go
```

> Zaproponowane indeksy zgadzają się z naszymi propozycjami z zadania 1!

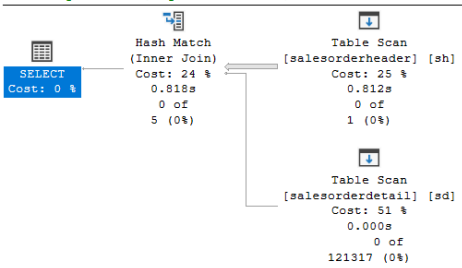
Sprawdź jak zmieniły się Execution Plany. Opisz zmiany:

Zapytanie 1.

Stary plan:

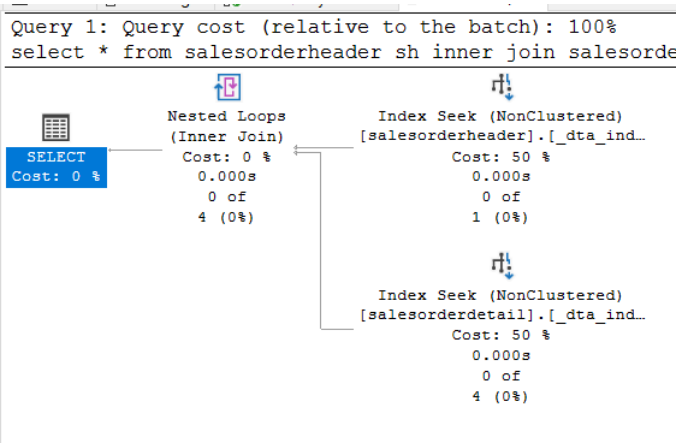
Plan i czas wykonania

```
Query 1: Query cost (relative to the batch): 100%
select * from salesorderheader sh inner join salesorderdetail sd on sh.salesorderid = sd.salesorderid where orderdate = '2008-06-01 00:00:00.000'
Missing Index (Impact 25.6028): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[salesorderheader] ([OrderDate])
```



Nowy plan:

Plan i czas wykonania

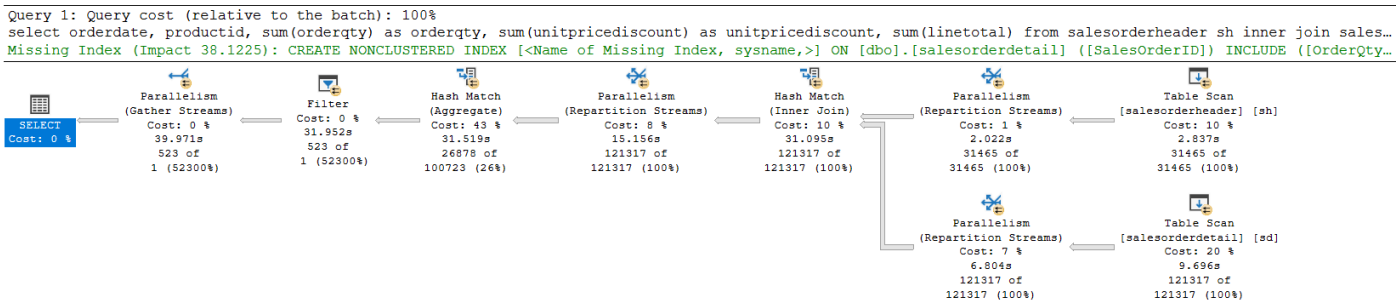


> Możemy zaobserwować, że cały koszt wykonania jest teraz rozłożony pomiędzy wyszukiwaniem w dwóch indeksach a zamaist hash match-owania, inner join jest przeprowadzany przy pomocy zagnieżdżonych pętli. Uzyskujemy potencjalną optymalizację. Index Seek sprawdza jedynie 4 elementy.

Zapytanie 2.

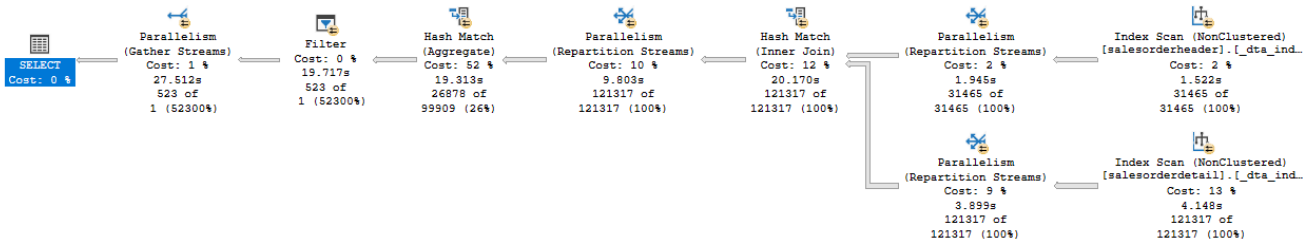
Stary plan:

Plan i czas wykonania



Nowy plan:

Plan i czas wykonania

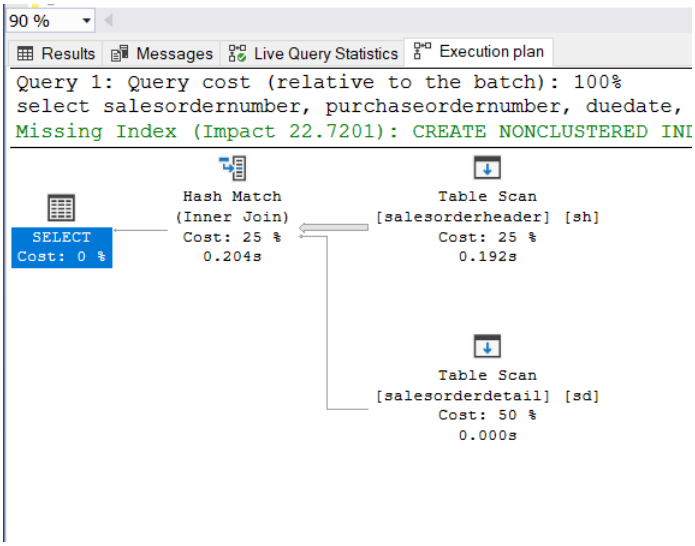


> Jedyna zasadnicza zmiana to skan indeksu zamiast skanu tabel. Działanie joinów pozostaje takie same.

Zapytanie 3.

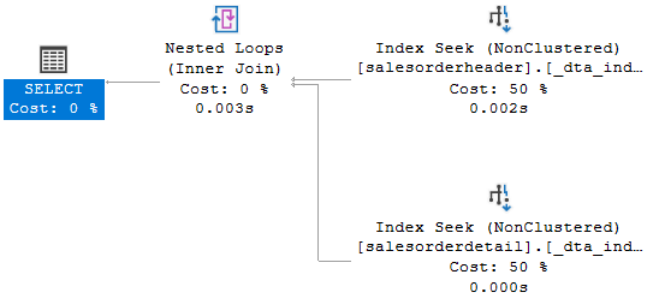
Stary plan:

Plan i czas wykonania



Nowy plan:

Plan i czas wykonania

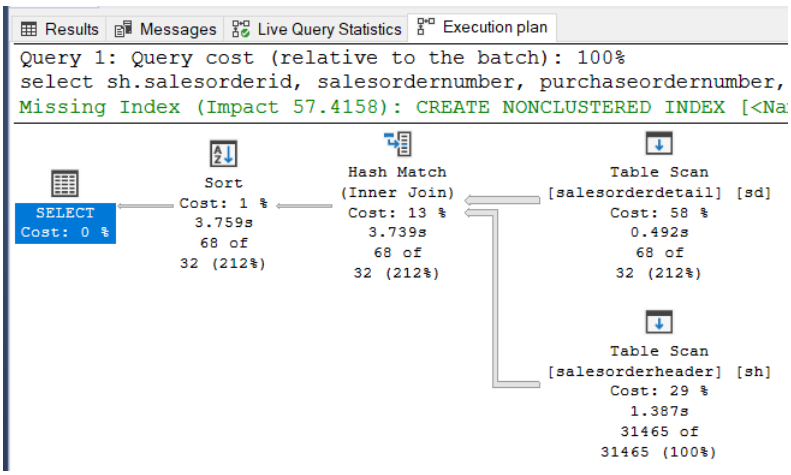


> W tym przykładzie join jest obsługiwany poprzez zagnieżdżone pętle, a odczyty z tabeli poprzez Index Seek. Wyszukiwanie, nie skanowanie jest wykonywane, tak jak w poprzednich przypadkach.

Zapytanie 4.

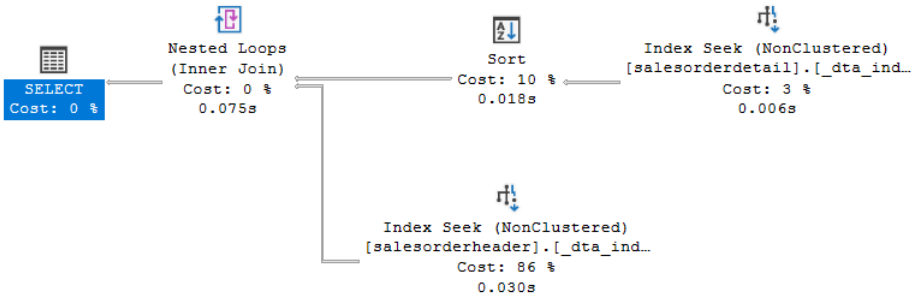
Stary plan:

Plan i czas wykonania



Nowy plan:

Plan i czas wykonania



> W ostatnim przykładzie sortowanie jest przeprowadzane przed join-em a skan tabel jest zamieniony na wyszukiwanie indeksowe. Dodatkowo join jest obsłużony przez zagnieżdżone pętle. Uzyskujemy znaczące przyspieszenie.

Zadanie 3 - Kontrola "zdrowia" indeksu

Dokumentacja/Literatura

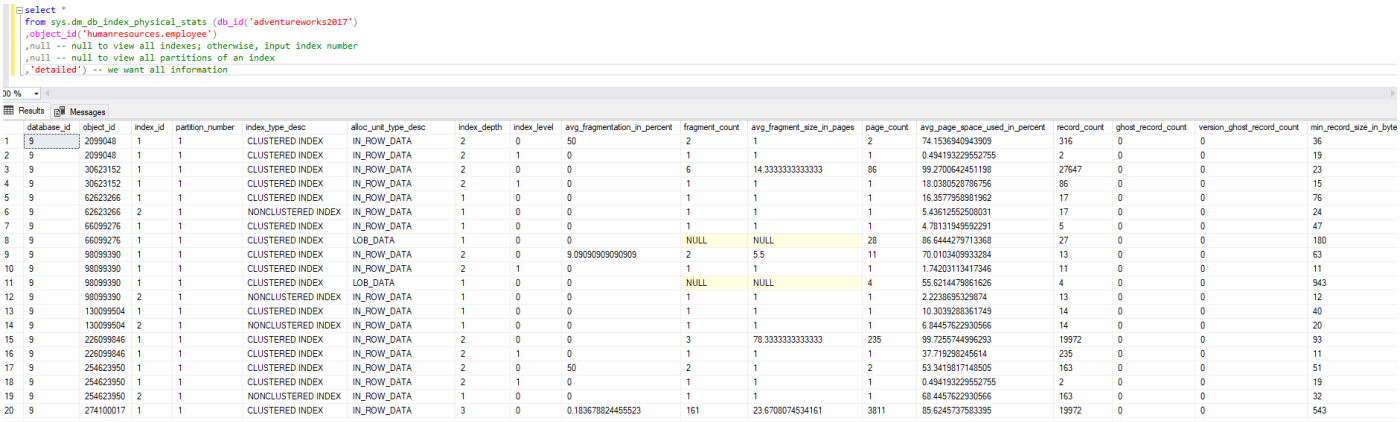
Celem kolejnego zadania jest zapoznanie się z możliwością administracji i kontroli indeksów.

Na temat wewnętrznej struktury indeksów można przeczytać tutaj:

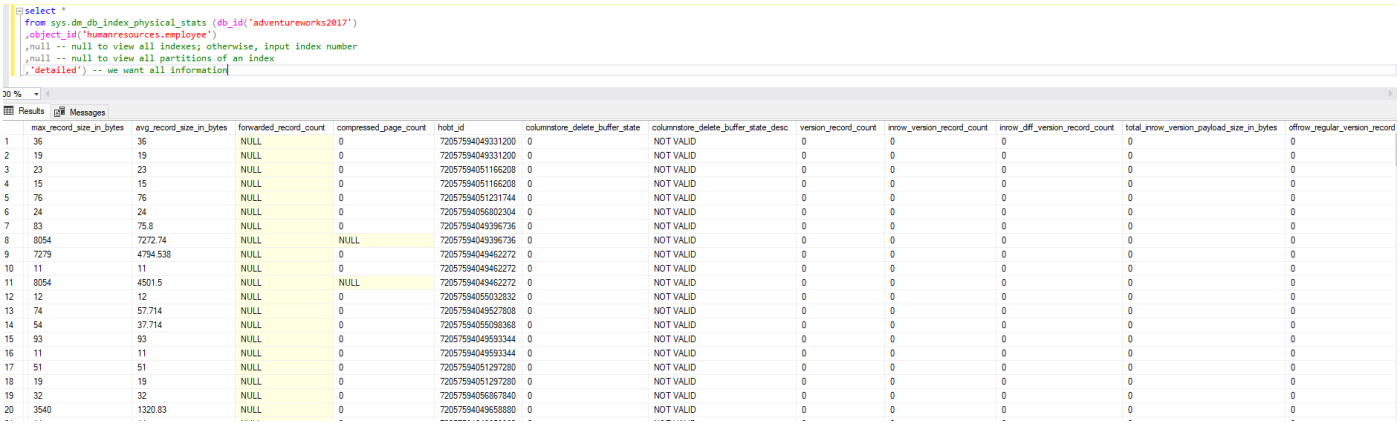
- https://technet.microsoft.com/en-us/library/2007.03.sqlindex.aspx
- https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql
- https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql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



	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_record_count	version_ghost_record_count	min_record_size_in_byte
1	9	2099048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	50	2	1	2	74.1536940943909	316	0	0	36
2	9	2099048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.49419322955275 5	2	0	0	19
3	9	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	6	14.3333333333333	86	99.2700642451198	27647	0	0	23
4	9	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	18.0380528786756	86	0	0	15
5	9	62623266	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	16.3577958981962	17	0	0	76
6	9	62623266	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	5.43612552508031	17	0	0	24
7	9	66099276	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	4.78131949592291	5	0	0	47
8	9	66099276	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	28	86.6444279713368	27	0	0	180
9	9	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	9.09090909090909	2	5.5	11	70.0103409933284	13	0	0	63
10	9	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	1.74203113417346	11	0	0	11
11	9	98099390	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	4	55.6214479861626	4	0	0	943
12	9	98099390	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	2.2238695329874	13	0	0	12
13	9	130099504	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	10.3039288361749	14	0	0	40
14	9	130099504	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	6.84457622930566	14	0	0	20
15	9	226099846	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	3	78.3333333333333	235	99.7255744996293	19972	0	0	93
16	9	226099846	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	37.719298245614	235	0	0	11
17	9	254623950	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	50	2	1	2	53.3419817148505	163	0	0	51
18	9	254623950	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.494193229552755	2	0	0	19
19	9	254623950	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	68.4457622930566	163	0	0	32
20	9	274100017	1	1	CLUSTERED INDEX	IN_ROW_DATA	3	0	0.183678824455523	161	23.6708074534161	3811	85.6245737583395	19972	0	0	543



	max_record_size_in_bytes	avg_record_size_in_bytes	forwarded_record_count	compressed_page_count	hobt_id	columnstore_delete_buffer_state	columnstore_delete_buffer_state_desc	version_record_count	inrow_version_record_count	inrow_diff_version_record_count	total_inrow_version_payload_size_in_bytes	offrow_regular_version_record
1	36	36	NULL	0	72057594049331200	0	NOT VALID	0	0	0	0	0
2	19	19	NULL	0	72057594049331200	0	NOT VALID	0	0	0	0	0
3	23	23	NULL	0	72057594051166208	0	NOT VALID	0	0	0	0	0
4	15	15	NULL	0	72057594051166208	0	NOT VALID	0	0	0	0	0
5	76	76	NULL	0	72057594051231744	0	NOT VALID	0	0	0	0	0
6	24	24	NULL	0	72057594056802304	0	NOT VALID	0	0	0	0	0
7	83	75.8	NULL	0	72057594049396736	0	NOT VALID	0	0	0	0	0
8	8054	7272.74	NULL	NULL	72057594049396736	0	NOT VALID	0	0	0	0	0
9	7279	4794.538	NULL	0	72057594049462272	0	NOT VALID	0	0	0	0	0
10	11	11	NULL	0	72057594049462272	0	NOT VALID	0	0	0	0	0
11	8054	4501.5	NULL	NULL	72057594049462272	0	NOT VALID	0	0	0	0	0
12	12	12	NULL	0	72057594055032832	0	NOT VALID	0	0	0	0	0
13	74	57.714	NULL	0	72057594049527808	0	NOT VALID	0	0	0	0	0
14	54	37.714	NULL	0	72057594055098368	0	NOT VALID	0	0	0	0	0
15	93	93	NULL	0	72057594049593344	0	NOT VALID	0	0	0	0	0
16	11	11	NULL	0	72057594049593344	0	NOT VALID	0	0	0	0	0
17	51	51	NULL	0	72057594051297280	0	NOT VALID	0	0	0	0	0
18	19	19	NULL	0	72057594051297280	0	NOT VALID	0	0	0	0	0
19	32	32	NULL	0	72057594056867840	0	NOT VALID	0	0	0	0	0
20	3540	1320.83	NULL	0	72057594049658880	0	NOT VALID	0	0	0	0	0

Jakie są według Ciebie najważniejsze pola?

> Pola które dają najwięcej informacji o indeksach to zdecydowanie `avg_page_Space_used_in_percent`, `record_count` oraz `avg_record_size_in_bytes` `avg_page_space_used_in_percent` określa efektywność zaalokowanej pamięci `avg_fragmentation_in_percent` oznacza jak bardzo fragmentowane są dane. Im bardziej tym gorzej. Fizyczna tabela ciągła jest optymalna. Dodatkowo index_type_desc opisuje rodzaj indeksu co również jest istotną informacją.

Sprawdź, które indeksy w bazie danych wymagają reorganizacji:

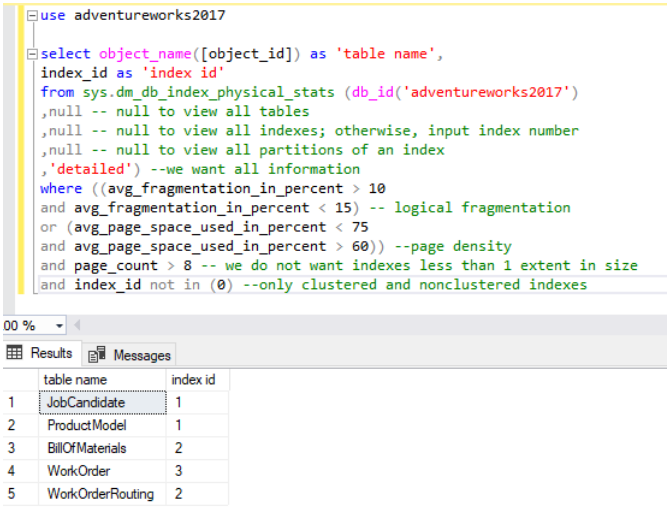
```
use adventureworks2017

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: Niektóre tabele trzeba zoptymalizować. Kryteria zmiany to indeksy o fragmentacji pomiędzy 10% a 15%, średnie wykorzystanie strony między 60 a 75%, dla indeksów o więcej niż 8 stronach. Jeśli indeksy spełniają te warunki to są odfiltrowane i zwrócone. Oznacza to, że właśnie te indeksy chcemy zmodyfikować.

> zrzut ekranu/komentarz:



```
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
```

00 %

Results | Messages

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

> Jak widzimy jedynie 5 indeksów wymaga reorganizacji.

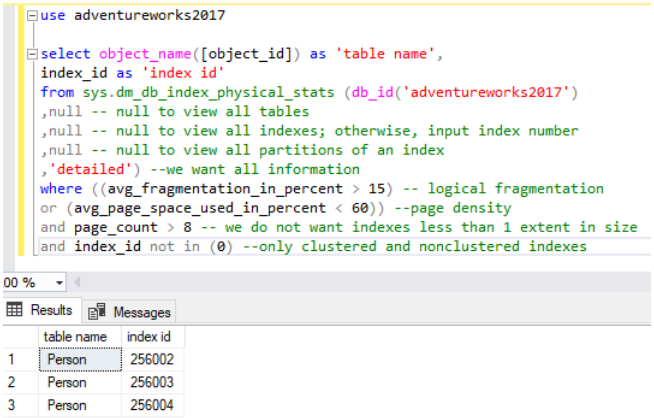
Sprawdź, które indeksy w bazie danych wymagają przebudowy:

```
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: Wcześniej odfiltrowaliśmy indeksy które warto przebudować. Teraz znajdujemy indeksy najgorsze. Te które koniecznie należy przebudować Tak jak poprzednio patrzymy tylko na indeksy o więcej niż 8 stronach. Tym razem wybieramy te indeksy których fragmentacja przekracza 15% albo wykorzystanie strony jest mniejsze niż 60%.

> zrzut ekranu/komentarz:



```
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
```

00 %

Results | Messages

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

> Jedyne indeksy wymagające przebudowy działają na tabeli Person. Ich ID to 256002, 256003, 256004

Czym się różni przebudowa indeksu od reorganizacji?

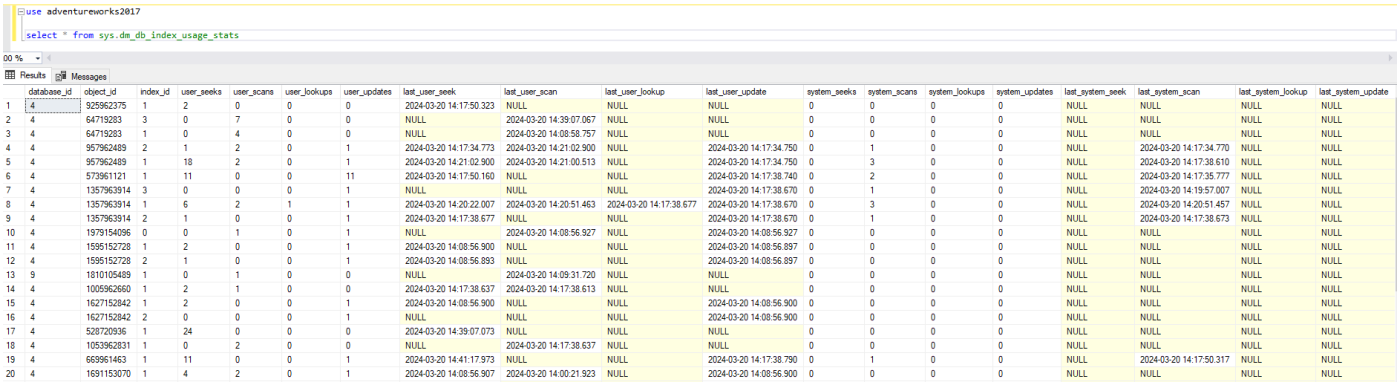
(Podpowiedź: http://blog.plik.pl/2014/12/defragmentacja-indeksow-ms-sql.html)

> Wyniki: Reorganizacja przeprowadza fragmentację indeksu w miejscu. Działa na istniejącej strukturze. Przebudowa usuwa indeks i buduje go od zera. Można wywnioskować, że przebudowa jest operacją którą powinniśmy wykonywać w krytycznych sytuacjach w których samo wykonanie reorganizacji nie wystarczy by uzyskać poprawienie jakości działania

> indeksu. W ćwiczeniu użyliśmy przebudowy kiedy wartość fragmentacji i wykorzystania strony były gorsze niż kiedy użyliśmy reorganizacji.

Sprawdź co przechowuje tabela `sys.dm_db_index_usage_stats`:

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



```
use adventureworks2017
select * from sys.dm_db_index_usage_stats
```

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	last_system_lookup	last_system_update
1	4	925962375	1	2	0	0	0	2024-03-20 14:17:50.323	NULL	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
2	4	64719283	3	0	7	0	0	NULL	2024-03-20 14:39:07.067	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
3	4	64719283	1	0	4	0	0	NULL	2024-03-20 14:08:58.757	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
4	4	957962489	2	1	2	0	1	2024-03-20 14:17:34.773	2024-03-20 14:21:02.900	NULL	2024-03-20 14:17:34.750	0	1	0	0	NULL	2024-03-20 14:17:34.770	NULL	NULL
5	4	957962489	1	18	2	0	1	2024-03-20 14:21:02.900	2024-03-20 14:21:00.513	NULL	2024-03-20 14:17:34.750	0	3	0	0	NULL	2024-03-20 14:17:38.610	NULL	NULL
6	4	573961121	1	11	0	0	11	2024-03-20 14:17:50.160	NULL	NULL	2024-03-20 14:17:38.740	0	2	0	0	NULL	2024-03-20 14:17:35.777	NULL	NULL
7	4	1357963914	3	0	0	0	1	NULL	NULL	NULL	2024-03-20 14:17:38.670	0	1	0	0	NULL	2024-03-20 14:19:57.007	NULL	NULL
8	4	1357963914	1	6	2	1	1	2024-03-20 14:20:22.007	2024-03-20 14:20:51.463	2024-03-20 14:17:38.677	2024-03-20 14:17:38.670	0	3	0	0	NULL	2024-03-20 14:20:51.457	NULL	NULL
9	4	1357963914	2	1	0	0	1	2024-03-20 14:17:38.677	NULL	NULL	2024-03-20 14:17:38.670	0	1	0	0	NULL	2024-03-20 14:17:38.673	NULL	NULL
10	4	1979154096	0	0	1	0	1	NULL	2024-03-20 14:08:56.927	NULL	2024-03-20 14:08:56.927	0	0	0	0	NULL	NULL	NULL	NULL
11	4	1595152728	1	2	0	0	1	2024-03-20 14:08:56.900	NULL	NULL	2024-03-20 14:08:56.897	0	0	0	0	NULL	NULL	NULL	NULL
12	4	1595152728	2	1	0	0	1	2024-03-20 14:08:56.893	NULL	NULL	2024-03-20 14:08:56.897	0	0	0	0	NULL	NULL	NULL	NULL
13	9	1810105489	1	0	1	0	0	NULL	2024-03-20 14:09:31.720	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
14	4	1005962660	1	2	1	0	0	2024-03-20 14:17:38.637	2024-03-20 14:17:38.613	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
15	4	1627152842	1	2	0	0	1	2024-03-20 14:08:56.900	NULL	NULL	2024-03-20 14:08:56.900	0	0	0	0	NULL	NULL	NULL	NULL
16	4	1627152842	2	0	0	0	1	NULL	NULL	NULL	2024-03-20 14:08:56.900	0	0	0	0	NULL	NULL	NULL	NULL
17	4	528720936	1	24	0	0	0	2024-03-20 14:39:07.073	NULL	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
18	4	1053962831	1	0	2	0	0	NULL	2024-03-20 14:17:38.637	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
19	4	669961463	1	11	0	0	1	2024-03-20 14:41:17.973	NULL	NULL	2024-03-20 14:17:38.790	0	1	0	0	NULL	2024-03-20 14:17:50.317	NULL	NULL
20	4	1691153070	1	4	2	0	1	2024-03-20 14:08:56.907	2024-03-20 14:00:21.923	NULL	2024-03-20 14:08:56.900	0	0	0	0	NULL	NULL	NULL	NULL

> Wyniki: Tabela zawiera historię użycia indeksów. Możemy sprawdzić jak często używane są indeksy i jak są przydatne. Może być tak, że będzie stworzony niepotrzebny indeks. W takim przypakdu nie będzie on pewnie używany. Dowiemy się o tym z tej tabeli.

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
```

```
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
```

00 %

Results Messages

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

Napisz przygotowane komendy SQL do naprawy indeksów:

Wyniki:

```
--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

alter index IX_WorkOrderRouting_ProductID on Production.WorkOrderRouting reorganize

--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;
```

Zadanie 4 - Budowa strony indeksu

Dokumentacja

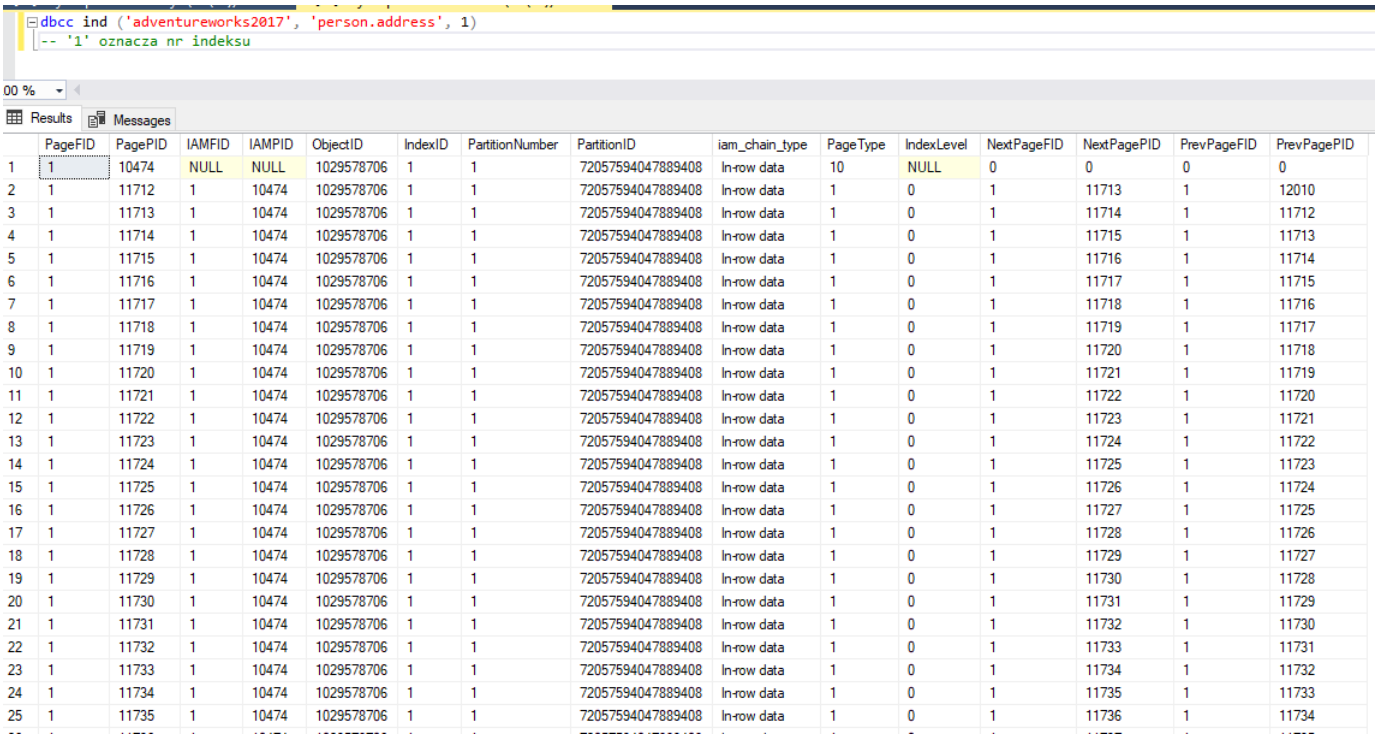
Celem kolejnego zadania jest zapoznanie się z fizyczną budową strony indeksu

- 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.:

Indeks 1

```
dbcc ind ('adventureworks2017', 'person.address', 1)
-- '1' oznacza nr indeksu
```

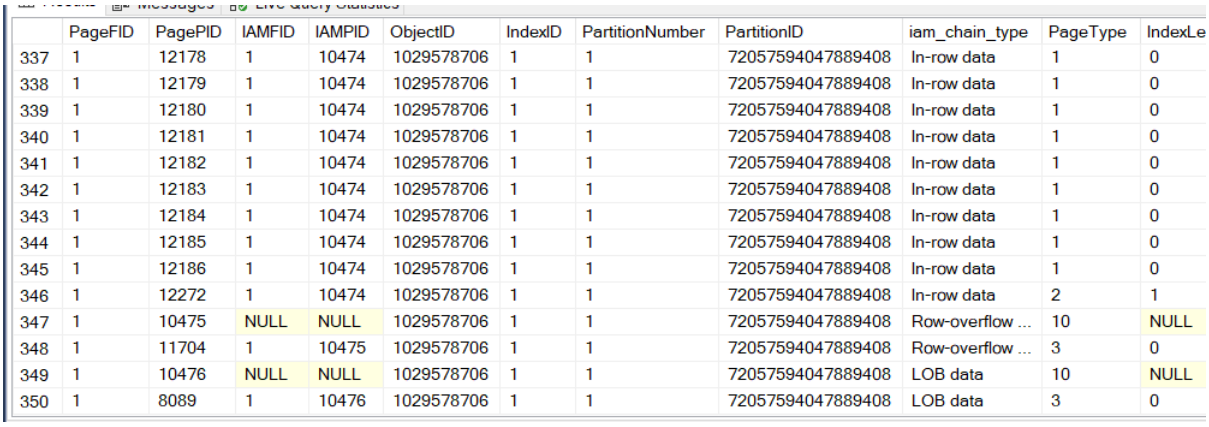


dbcc ind ('adventureworks2017', 'person.address', 1)
-- '1' oznacza nr indeksu

00 %

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



	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLe
337	1	12178	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
338	1	12179	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
339	1	12180	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
340	1	12181	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
341	1	12182	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
342	1	12183	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
343	1	12184	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
344	1	12185	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
345	1	12186	1	10474	1029578706	1	1	72057594047889408	In-row data	1	0
346	1	12272	1	10474	1029578706	1	1	72057594047889408	In-row data	2	1
347	1	10475	NULL	NULL	1029578706	1	1	72057594047889408	Row-overflow ...	10	NULL
348	1	11704	1	10475	1029578706	1	1	72057594047889408	Row-overflow ...	3	0
349	1	10476	NULL	NULL	1029578706	1	1	72057594047889408	LOB data	10	NULL
350	1	8089	1	10476	1029578706	1	1	72057594047889408	LOB data	3	0

Zapisz sobie kilka różnych typów stron, dla różnych indeksów:

> Wyniki: Dla indeksu 1 zapisaliśmy 2 strony: 13720, 8089 o typach odpowiednio 10 i 3. Widoczne w dalszej części

---

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?

- Dla strony `13720` dostaliśmy wyłącznie wynik "Messages". Wygląda on następująco

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

```
00 %
Messages

PAGE: (1:13720)


BUFFER:


BUF @0x0000017F5859E400

bpage = 0x0000017F3F1D8000          bPmmpage = 0x0000000000000000       bsort_r_nextbP = 0x0000017F5859E350
bsort_r_prevbP = 0x0000017F5859E340 bhash = 0x0000000000000000          bpageno = (1:13720)
bpart = 0                           bstat = 0x9                         breferences = 3
berrcode = 0                        bUse1 = 8323                        bstat2 = 0x0
blog = 0x15ab215a                   bsampleCount = 1                    bIoCount = 0
resPoolId = 0                       bcputicks = 474                     bReadMicroSec = 212
bDirtyPendingCount = 0              bDirtyContext = 0x0000000000000000  bDbPageBroker = 0x0000000000000000
bdbid = 9                           bpru = 0x0000017F289D0040

PAGE HEADER:


Page @0x0000017F3F1D8000

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
```

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

```
00 %
Messages
00000000000003FC:   0000f7ea 09010f00 000f1a00 0000f009 45006400  ..÷ê  .........ð  E.d.
0000000000000410:   75006300 61007400 69006f00 6e00ef02 000cf80b  u.c.a.t.i.o.n.ï...ø.
0000000000000424:   ea05000f 00000f11 0f500061 00720074 00690061  ê........P.a.r.t.i.a
0000000000000438:   006c0020 0043006f 006c006c 00650067 006500f7  .l. .C.o.l.l.e.g.e.÷
000000000000044C:   ea09010f 00000f1c 000000f0 0a4f0063 00630075  ê .........ð.O.c.c.u
0000000000000460:   00700061 00740069 006f006e 00ef0200 0df80cea  .p.a.t.i.o.n.ï...ø.ê
0000000000000474:   05000f00 000f1108 43006c00 65007200 69006300  ........C.l.e.r.i.c.
0000000000000488:   61006c00 f7ea0901 0f00000f 22000000 f00d4800  a.l.÷ê     ....."...ð.H.
000000000000049C:   6f006d00 65004f00 77006e00 65007200 46006c00  o.m.e.O.w.n.e.r.F.l.
00000000000004B0:   61006700 ef02000e f80dea05 000f0000 0f110131  a.g.ï...ø.ê........1
00000000000004C4:   00f7ea09 01710000 13260000 00f00f4e 0075006d  .÷ê   .q...&...ð.N.u.m
00000000000004D8:   00620065 00720043 00610072 0073004f 0077006e  .b.e.r.C.a.r.s.O.w.n
00000000000004EC:   00650064 00ef0200 0ff80eea 05007100 00138713  .e.d.ï...ø.ê..q....
0000000000000500:   260a0100 e40b5402 00000000 00000000 000000f7  &...ä.T.............÷
0000000000000514:   ea090128 00010f26 000000f0 0f43006f 006d006d  ê .(...&...ð.C.o.m.m
0000000000000528:   00750074 00650044 00690073 00740061 006e0063  .u.t.e.D.i.s.t.a.n.c
000000000000053C:   006500ef 020010f8 0fea0500 2800010f 11093000  .e.ï...ø.ê..(.... 0.
0000000000000550:   2d003100 20004d00 69006c00 65007300 f7f7      -.1. .M.i.l.e.s.÷÷

Slot 0 Column 1 Offset 0x4 Length 4 Length (physical) 4

BusinessEntityID = 6871

Slot 0 Column 2 Offset 0x8 Length 4 Length (physical) 4

PersonType = IN

Slot 0 Column 3 Offset 0xc Length 1 (Bit position 0)

NameStyle = 0

Slot 0 Column 4 Offset 0x0 Length 0 Length (physical) 0

Title = [NULL]

Slot 0 Column 5 Offset 0x3d Length 8 Length (physical) 8

FirstName = Neil

Slot 0 Column 6 Offset 0x45 Length 2 Length (physical) 2
```

W wyniku dostajemy identyfikator stron oraz informacje o buforze i nagłówku.

- Dla strony `8089` dostajemy dużo mniej obszerną informację niż dla 13720. Brakuje na przykład wypisywanych informacji o slotach

Messages | Live Query Statistics

```
BUFFER:


BUF @0x000001FFBAB66D80

bpage = 0x000001FE0A1D6000          bPmmpage = 0x0000000000000000       bsort_r_nextbP = 0x000001FFBAB66E50
bsort_r_prevbP = 0x000001FFBAB66E40 bhash = 0x0000000000000000         bpageno = (1:8089)
bpart = 2                           bstat = 0x9                         breferences = 0
berrcode = 0                        bUse1 = 16533                       bstat2 = 0x0
blog = 0x1215a                      bsampleCount = 0                    bIoCount = 0
resPoolId = 0                       bcputicks = 0                       bReadMicroSec = 1108
bDirtyPendingCount = 0              bDirtyContext = 0x0000000000000000  bDbPageBroker = 0x0000000000000000
bdbid = 8                           bpru = 0x000001FE0EC08040

PAGE HEADER:


Page @0x000001FE0A1D6000

m_pageId = (1:8089)                 m_headerVersion = 1                 m_type = 3
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x200
m_objId (AllocUnitId.idObj) = 186   m_indexId (AllocUnitId.idInd) = 256 Metadata: AllocUnitId = 72057594050117632
Metadata: PartitionId = 72057594047889408                                Metadata: IndexId = 1
Metadata: ObjectId = 1029578706     m_prevPage = (0:0)                  m_nextPage = (0:0)
pminlen = 0                         m_slotCnt = 0                       m_freeCnt = 8096
m_freeData = 96                     m_reservedCnt = 0                   m_lsn = (37:1108:68)
m_xactReserved = 0                  m_xdesId = (0:0)                    m_ghostRecCnt = 0
m_tornBits = 47906992               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


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

Completion time: 2024-03-25T16:36:51.1127764+01:00
```

Informacje stron ewidentnie różnią się w zależności od typu. Dla strony `13720` pojawiły się kolumny danych i opis slotów i kolumn a na dwóch pozostałych opis bufora i nagłówka strony.

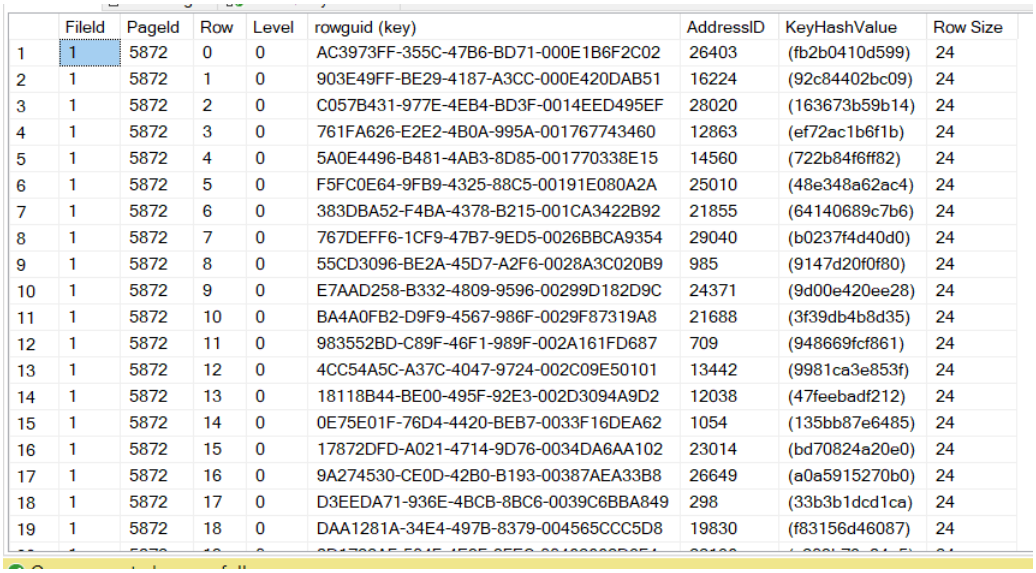
Indeks 2

Sprawdziliśmy jeszcze stronę dla indeksu 2. Strona o numerze `5872` i typie 2.

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

dbcc page('adventureworks2017', 1, 5872, 3);
```

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	Ne
1	1	10472	NULL	NULL	1029578706	2	1	72057594052542464	In-row data	10	NULL	0
2	1	5872	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
3	1	5873	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
4	1	5874	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
5	1	5875	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
6	1	5876	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
7	1	5877	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
8	1	5878	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
9	1	5879	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
10	1	5880	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
11	1	5881	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
12	1	5882	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
13	1	5883	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
14	1	5884	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
15	1	5885	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
16	1	5886	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
17	1	5887	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
18	1	5888	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1
19	1	5889	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1



	FileId	PageId	Row	Level	rowguid (key)	AddressID	KeyHashValue	Row Size
1	1	5872	0	0	AC3973FF-355C-47B6-BD71-000E1B6F2C02	26403	(fb2b0410d599)	24
2	1	5872	1	0	903E49FF-BE29-4187-A3CC-000E420DAB51	16224	(92c84402bc09)	24
3	1	5872	2	0	C057B431-977E-4EB4-BD3F-0014EED495EF	28020	(163673b59b14)	24
4	1	5872	3	0	761FA626-E2E2-4B0A-995A-001767743460	12863	(ef72ac1b6f1b)	24
5	1	5872	4	0	5A0E4496-B481-4AB3-8D85-001770338E15	14560	(722b84f6ff82)	24
6	1	5872	5	0	F5FC0E64-9FB9-4325-88C5-00191E080A2A	25010	(48e348a62ac4)	24
7	1	5872	6	0	383DBA52-F4BA-4378-B215-001CA3422B92	21855	(64140689c7b6)	24
8	1	5872	7	0	767DEFF6-1CF9-47B7-9ED5-0026BBCA9354	29040	(b0237f4d40d0)	24
9	1	5872	8	0	55CD3096-BE2A-45D7-A2F6-0028A3C020B9	985	(9147d20f0f80)	24
10	1	5872	9	0	E7AAD258-B332-4809-9596-00299D182D9C	24371	(9d00e420ee28)	24
11	1	5872	10	0	BA4A0FB2-D9F9-4567-986F-0029F87319A8	21688	(3f39db4b8d35)	24
12	1	5872	11	0	983552BD-C89F-46F1-989F-002A161FD687	709	(948669fcf861)	24
13	1	5872	12	0	4CC54A5C-A37C-4047-9724-002C09E50101	13442	(9981ca3e853f)	24
14	1	5872	13	0	18118B44-BE00-495F-92E3-002D3094A9D2	12038	(47feebadf212)	24
15	1	5872	14	0	0E75E01F-76D4-4420-BEB7-0033F16DEA62	1054	(135bb87e6485)	24
16	1	5872	15	0	17872DFD-A021-4714-9D76-0034DA6AA102	23014	(bd70824a20e0)	24
17	1	5872	16	0	9A274530-CE0D-42B0-B193-00387AEA33B8	26649	(a0a5915270b0)	24
18	1	5872	17	0	D3EEDA71-936E-4BCB-8BC6-0039C6BBA849	298	(33b3b1dcd1ca)	24
19	1	5872	18	0	DAA1281A-34E4-497B-8379-004565CCC5D8	19830	(f83156d46087)	24

w odróżnieniu od dwóch poprzednich typów stron tutaj dostajemy tabelkę. Tabela opisuje indeks - wskaźniki do odpowiednich wierszy w tabeli, na której indeks jest utworzony. Strona 5872 jest właśnie taką składową indeksu 2.

Punktacja:

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