

Indeksy, optymalizator

Lab 4

Imię i nazwisko:

Jacek Budny, Mateusz Kleszcz

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):

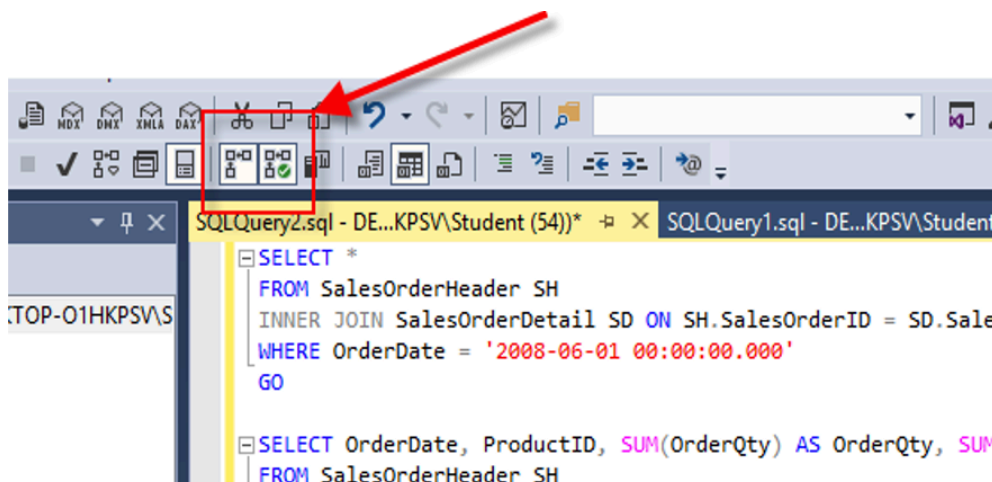
```
-- zapytanie 1
select *
from salesorderheader sh
inner join salesorderdetail sd on sh.salesorderid = sd.salesorderid
where orderdate = '2008-06-01 00:00:00.000'
go

-- 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**:



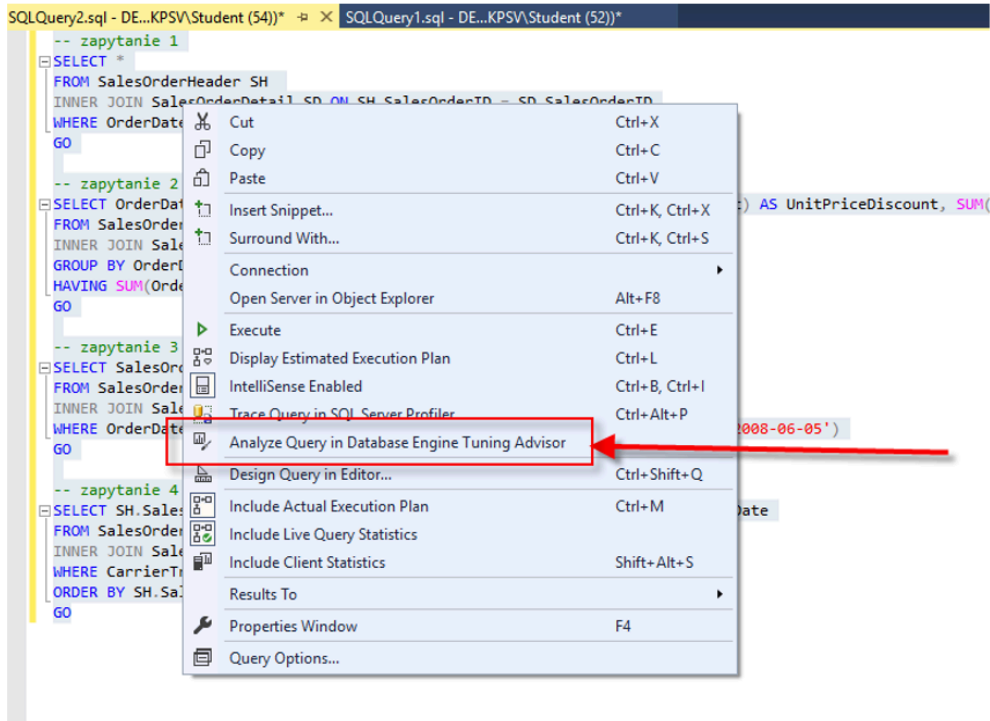
Teraz wykonaj poszczególne zapytania (najlepiej każde analizuj oddzielnie). Co można o nich powiedzieć? Co sprawdzają? Jak można je zoptymalizować?

(Hint: aby wykonać tylko fragment kodu SQL znajdującego się w edytorze, zaznacz go i naciśnij F5)

Wyniki: Za każdym razem otrzymywaliśmy ostrzeżenie o brakującym indeksie, ze względu na wykonywanie inner join, w tym przypadku serwer musi przeszukiwać tabelę zamiast wykorzystać indeks. Dpbrym pomysłem jest również użycie indeksów do kolumn wykorzystywanych w klauzuli where np OrderDate, co może przyspieszyć wyszukiwanie.

Zadanie 2 - Optymalizacja

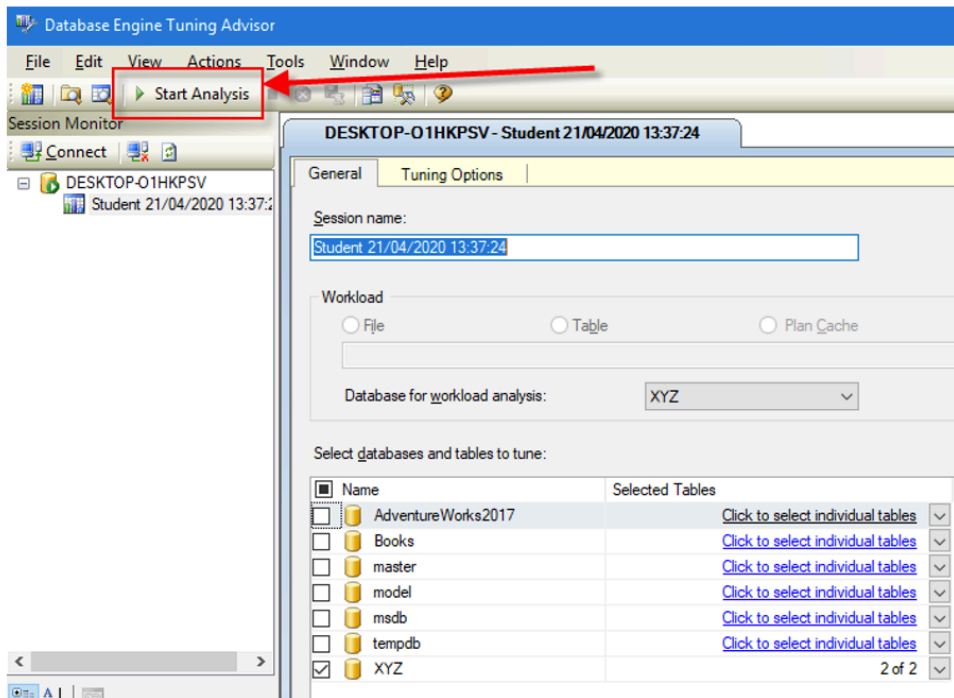
Zaznacz wszystkie zapytania, i uruchom je w **Database Engine Tuning Advisor**:



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

Wyniki: można skonfigurować PDS (indeksy, indeksowane widoki, indeksy klastrowe i nieklastrowe), można włączyć partycjonowanie pełne albo aligned, czy chcemy te wszystkie indeksy i partycje zachować

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:

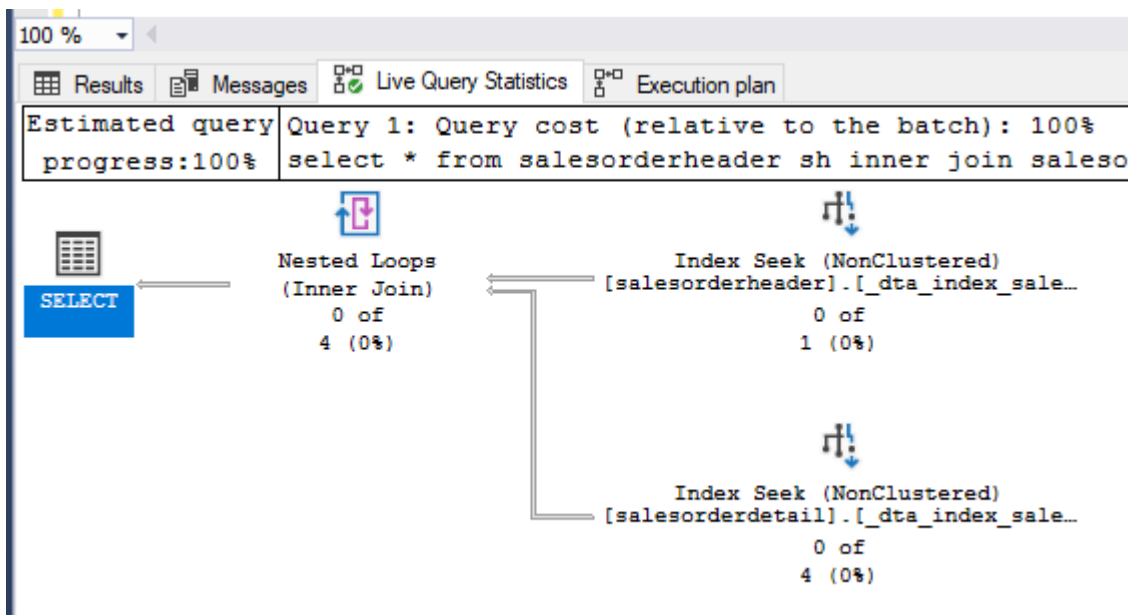
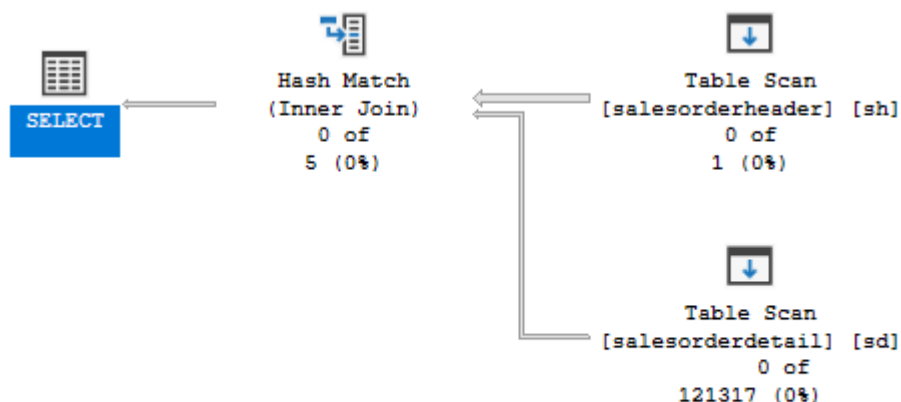
Wyniki: w rekomendacjach jest w reports jest: zostały zaproponowane następujące indeksy: ID - na przykład: SalesOrderID, ProductID, ponieważ są wykorzystywane i inner join'ach w każdym zapytaniu. Również OrderDate, który jest wykorzystywany 1 i 3 zapytaniu w klauzuli where oraz CarrierTrackingNumber wykorzystywany w klauzuli where w 4 zapytaniu.

Po przeanalizowaniu raportu poprawy poszczególnych kosztów, uzyskaliśmy wynik że dla 1 i 3 zapytania jesteśmy w stanie uzyskać poprawę o około 99.7% a dla zapytania 4. o około 93.8%.

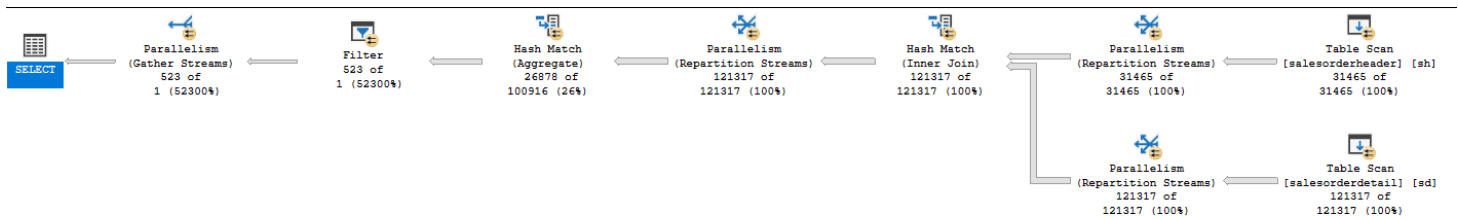
Dla zapytania 2. szacowana poprawa wynosi około 19%. W drugim zapytaniu nie występuje klauzula where, więc poprawę jesteśmy w stanie uzyskać przez indeksowanie kolumn wykorzystywanych w inner join.

Sprawdź jak zmieniły się Execution Plany. Opisz zmiany:

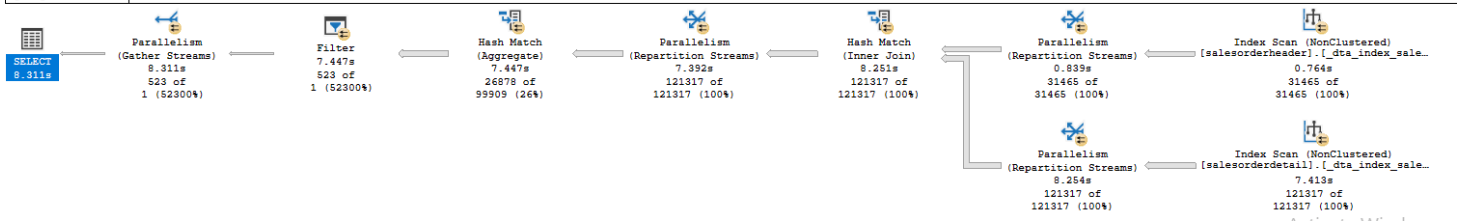
1:



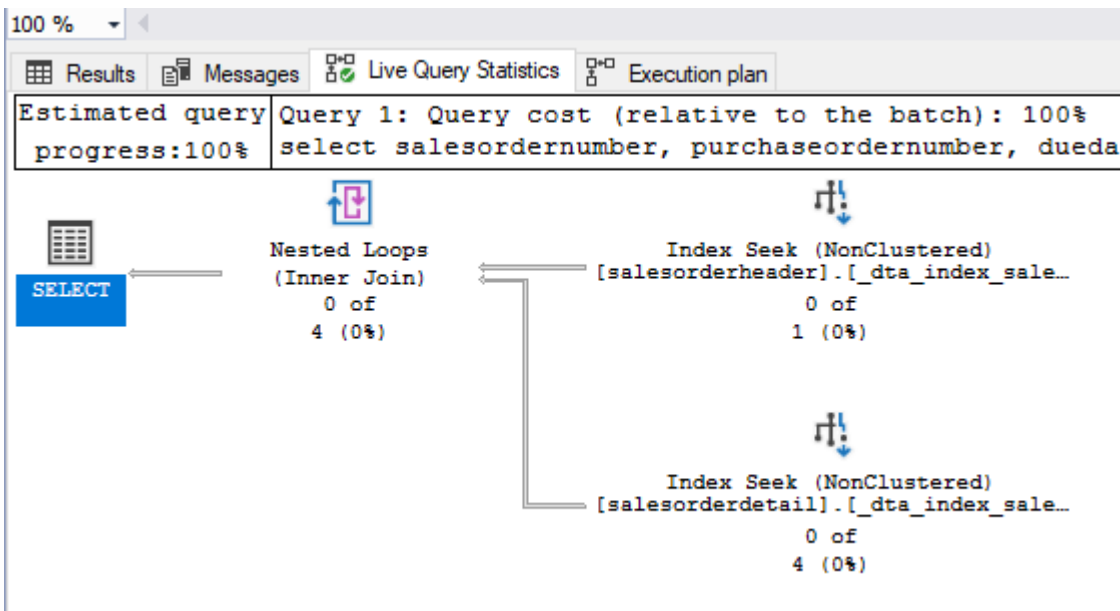
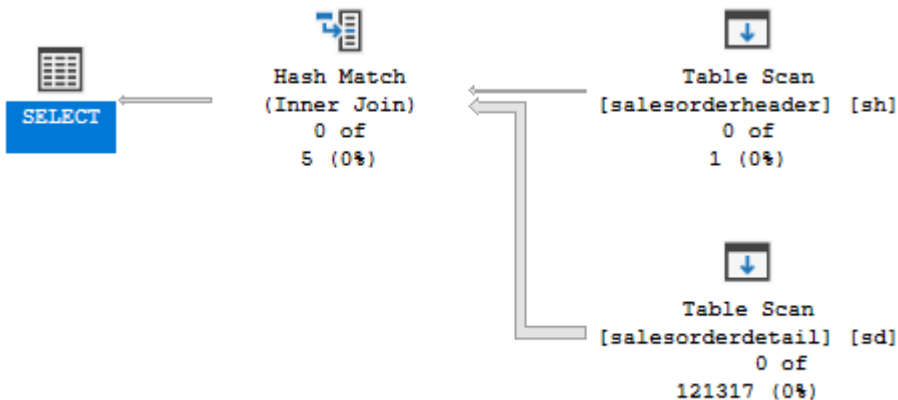
2:



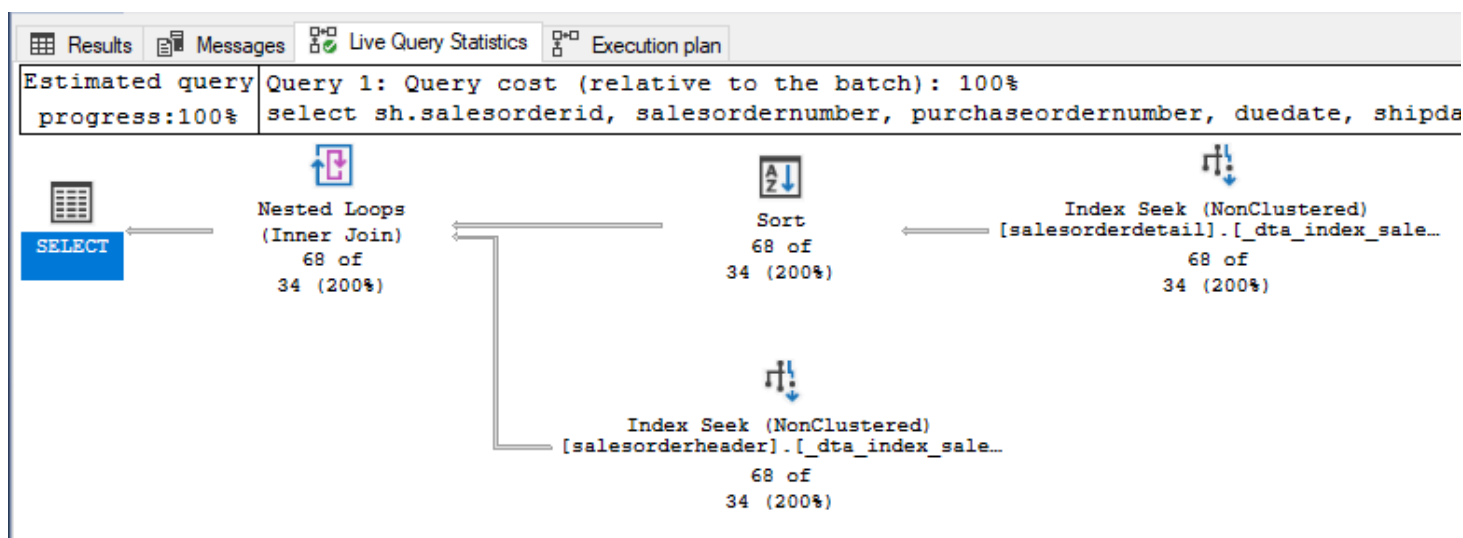
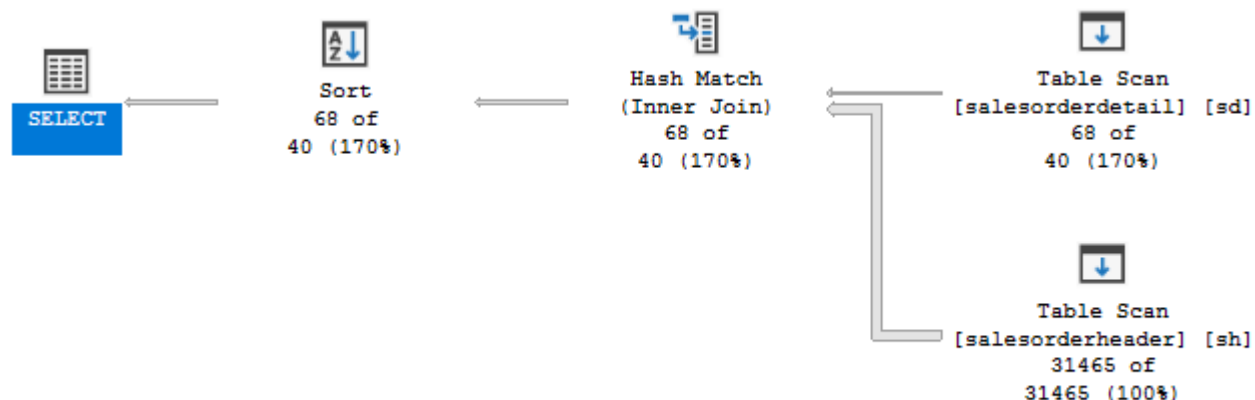
Results Messages Live Query Statistics Execution plan
 Estimated query progress:100% Query 1: Query cost (relative to the batch): 100%
 select orderdate, productid, sum(orderqty) as orderqty, sum(unitpricediscount) as unitpricediscount, sum(linetotal) from salesorderheader sh inner join salesorderdetail



3:



4:



Wyniki: Table scan zamienia się na index seek, hash match zamienia się na nested loop.

W szczególności:

W zapytaniu 1 i 3 dzięki dodaniu indeksów, system nie musiał skanować tabel w całości co ma niebagatelny wpływ na wydajność.

W 2. zapytaniu Table Scan zmienia się na Index Scan

W zapytaniu 4 dzięki indeksom zmniejszyła się ilość skanowanych rekordów

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:

	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
1	8	2099048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	50	2	1	2	74.1536340943909	316	0	0
2	8	2099048	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	0.494193228552755	2	0	0
3	8	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	6	14.3333333333333	86	99.2700642451196	27647	0	0
4	8	30623152	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	18.6380526786756	86	0	0
5	8	62623266	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	16.3577958861962	17	0	0
6	8	62623266	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	5.42612552608031	17	0	0
7	8	66099276	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	1	1	1	4.78131949592291	5	0	0
8	8	66099276	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	28	86.6444279713268	27	0	0
9	8	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	9.09090909090909	2	5.5	11	70.0103409933284	13	0	0
10	8	98099390	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0	1	1	1	1.74203113417346	11	0	0
11	8	98099390	1	1	CLUSTERED INDEX	LOB_DATA	1	0	0	NULL	NULL	4	55.6214479861626	4	0	0
12	8	98099390	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	2.2238695329874	13	0	0
13	8	130099504	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	10.3039288361749	14	0	0
14	8	130099504	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	6.84457622930566	14	0	0
15	8	226099846	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	0	3	78.3333333333333	235	99.7255744996293	19972	0	0

Ważne pola:

index_type_desc - typ indeksu (klastrowany czy nie);
index_depth - ilość poziomów indeksu;
avg_fragmentation_in_percent - średnia procentowa fragmentacja zewnętrzna dla indeksów
avg_page_space_used_in_percent - średnia procentowa fragmentacja wewnętrzna dla indeksu
page_count - ilość stron zajętych przez dany 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:

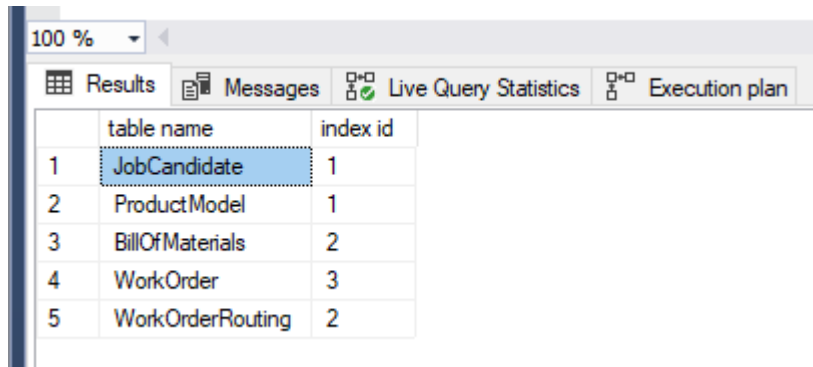


	table name	index id
1	JobCandidate	1
2	ProductModel	1
3	BillOfMaterials	2
4	WorkOrder	3
5	WorkOrderRouting	2

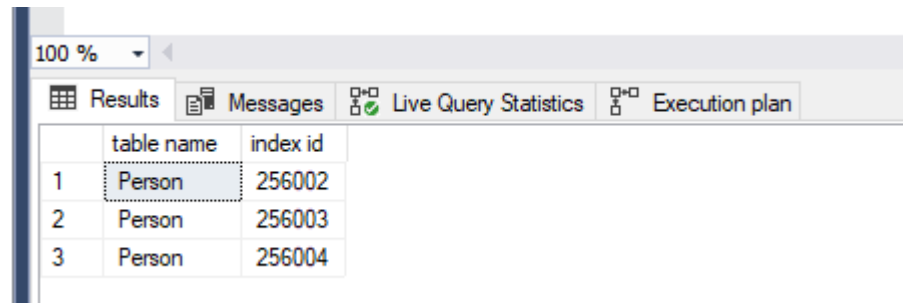
Wskazane indeksy tabel wymagają reorganizacji.

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:



The screenshot shows a SQL Server query results window. The 'Results' tab is active, displaying a table with two columns: 'table name' and 'index id'. There are three rows of data, all for the 'Person' table. The first row has index id 256002, the second 256003, and the third 256004. The window also shows tabs for 'Messages', 'Live Query Statistics', and 'Execution plan'.

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

Wskazane indeksy wymagają przebudowy.

Czym się różni przebudowa indeksu od reorganizacji?

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

Wyniki:

Reorganizacja indeksów polega na optymalizacji istniejących indeksów bez zmiany ich struktury. Przebudowa indeksów jest bardziej radykalną operacją, która polega na całkowitym usunięciu i ponownym utworzeniu indeksu. Reorganizacji dokonujemy gdy wartość wskaźnika fragmentacji

zewnętrznej znajduje się między 10 a 15 i wartość wskaźnika fragmentacji wewnętrznej znajduje się między 60 a 75 procent.

Przebudowę przeprowadzamy gdy wskaźnik fragmentacji zewnętrznej jest większy niż 15 procent a wskaźnik fragmentacji wewnętrznej jest mniejszy niż 60 procent.

Fragmentacja zewnętrzna określa rozproszenie bloków danych indeksu na dysku.

Fragmentacja wewnętrzna określa liczbę pustych miejsc wewnątrz bloków danych w strukturze indeksu.

Sprawdź co przechowuje tabela sys.dm_db_index_usage_stats:

Wyniki:

	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	2024-04-04 21:43:51.710	NULL	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
2	4	64719283	3	0	2	0	0	NULL	2024-04-04 21:41:20.853	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
3	4	957962489	2	1	0	0	1	2024-04-04 21:43:36.507	NULL	NULL	2024-04-04 21:43:36.487	0	0	0	0	NULL	NULL	NULL	NULL
4	4	957962489	1	6	1	0	1	2024-04-04 21:43:39.577	2024-04-04 21:43:39.517	NULL	2024-04-04 21:43:36.487	0	0	0	0	NULL	NULL	NULL	NULL
5	4	573961121	1	11	0	0	11	2024-04-04 21:43:51.627	NULL	NULL	2024-04-04 21:43:39.623	0	0	0	0	NULL	NULL	NULL	NULL
6	4	1357963914	3	0	0	0	1	NULL	NULL	NULL	2024-04-04 21:43:39.570	0	0	0	0	NULL	NULL	NULL	NULL
7	4	1357963914	1	0	0	1	1	NULL	2024-04-04 21:43:39.577	2024-04-04 21:43:39.570	2024-04-04 21:43:39.570	0	0	0	0	NULL	NULL	NULL	NULL
8	4	1357963914	2	1	0	0	1	2024-04-04 21:43:39.577	NULL	NULL	2024-04-04 21:43:39.570	0	0	0	0	NULL	NULL	NULL	NULL
9	4	1005962660	1	2	1	0	0	2024-04-04 21:43:39.543	2024-04-04 21:43:39.520	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
10	4	1627152842	1	1	0	0	0	2024-04-04 20:22:51.637	NULL	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
11	4	528720936	1	10	0	0	0	2024-04-04 21:41:20.857	NULL	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
12	4	1053962831	1	0	2	0	0	NULL	2024-04-04 21:43:39.543	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
13	9	1221579390	2	0	1	0	0	NULL	2024-04-04 21:39:27.527	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL
14	4	669961463	1	6	0	0	1	2024-04-04 21:43:51.760	NULL	NULL	2024-04-04 21:43:39.650	0	0	0	0	NULL	NULL	NULL	NULL
15	4	1691153070	1	0	1	0	0	NULL	2024-04-04 20:22:51.637	NULL	NULL	0	0	0	0	NULL	NULL	NULL	NULL

Tabela przechowuje informacje o używaniu indeksów przez bazę danych. Można tam znaleźć informacje o ilości wywołań danego typu operacji na indeksach wraz z datą i czasem ich ostatniego wywołania.

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:

	(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

```

Skrypt dla reorganizacji:

```
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
```

Wyniki:

	(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 PK_JobCandidate_JobCandidateID on HumanResources.JobCandidate reorganize
alter index PK_ProductModel_ProductModelID on Production.ProductModel reorganize
alter index PK_BillOfMaterials_BillOfMaterialsID on Production.BillOfMaterials reorganize
alter index IX_WorkOrder_ProductID on Production.WorkOrder reorganize
alter index IX_WorkOrderRouting_ProductID on Production.WorkOrderRouting reorganize
```


Zadanie 4 - Budowa strony indeksu

Dokumentacja

Celem kolejnego zadania jest zapoznanie się z fizyczną budową strony indeksu

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

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

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

Zapisz sobie kilka różnych typów stron, dla różnych indeksów:

Wyniki:

1:

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

2:

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10472	NULL	NULL	1029578706	2	1	72057594052542464	In-row data	10	NULL	0	0	0	0
2	1	5872	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5873	0	0
3	1	5873	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5874	1	5872
4	1	5874	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5875	1	5873
5	1	5875	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5876	1	5874
6	1	5876	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5877	1	5875
7	1	5877	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5878	1	5876
8	1	5878	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5879	1	5877
9	1	5879	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5880	1	5878
10	1	5880	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5881	1	5879
11	1	5881	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5882	1	5880
12	1	5882	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5883	1	5881
13	1	5883	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5884	1	5882
14	1	5884	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5885	1	5883
15	1	5885	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5886	1	5884
16	1	5886	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5887	1	5885
17	1	5887	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5888	1	5886
18	1	5888	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5889	1	5887
19	1	5889	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5890	1	5888
20	1	5890	1	10472	1029578706	2	1	72057594052542464	In-row data	2	0	1	5891	1	5889

3:

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	10473	NULL	NULL	1029578706	3	1	72057594052608000	In-row data	10	NULL	0	0	0	0
2	1	5920	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5921	0	0
3	1	5921	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5922	1	5920
4	1	5922	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5923	1	5921
5	1	5923	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5924	1	5922
6	1	5924	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5925	1	5923
7	1	5925	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5926	1	5924
8	1	5926	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5927	1	5925
9	1	5927	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5928	1	5926
10	1	5928	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5929	1	5927
11	1	5929	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5930	1	5928
12	1	5930	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5931	1	5929
13	1	5931	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5932	1	5930
14	1	5932	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5933	1	5931
15	1	5933	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5934	1	5932
16	1	5934	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	5935	1	5933
17	1	5935	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	8464	1	5934
18	1	8464	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	8465	1	5935
19	1	8465	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	8466	1	8464
20	1	8466	1	10473	1029578706	3	1	72057594052608000	In-row data	2	0	1	8467	1	8465

4:

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID	PrevPageFID	PrevPagePID
1	1	2164	NULL	NULL	1029578706	4	1	72057594052673536	In-row data	10	NULL	0	0	0	0
2	1	8544	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8545	0	0
3	1	8545	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8546	1	8544
4	1	8546	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8547	1	8545
5	1	8547	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8548	1	8546
6	1	8548	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8549	1	8547
7	1	8549	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8550	1	8548
8	1	8550	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8551	1	8549
9	1	8551	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8552	1	8550
10	1	8552	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8553	1	8551
11	1	8553	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8554	1	8552
12	1	8554	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8555	1	8553
13	1	8555	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8556	1	8554
14	1	8556	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8557	1	8555
15	1	8557	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8558	1	8556
16	1	8558	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8559	1	8557
17	1	8559	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8560	1	8558
18	1	8560	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8561	1	8559
19	1	8561	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8562	1	8560
20	1	8562	1	2164	1029578706	4	1	72057594052673536	In-row data	2	0	1	8563	1	8561

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:

```
Record Type = PRIMARY_RECORD          Record Attributes =  NULL_BITMAP VARIABLE_COLUMNS
Record Size = 142
Memory Dump @0x00000007350E761FC

0000000000000000:  30002400 936c0000 07000000 d94e1306 968f1d47 0.$..l.....ÛN....G
0000000000000014:  ac391b4e 62f27c54 00000000 8ea20000 09000400 ~9.Nbò|T....e...  ...
0000000000000028:  05005800 58006a00 78008e00 31003500 31003100 ..X.X.j.x..1.5.1.1.
000000000000003C:  20005200 6f007800 62007500 72007900 20004400 .R.o.x.b.u.r.y. .D.
0000000000000050:  72006900 76006500 43006c00 69006600 66007300 r.i.v.e.C.l.i.f.f.s.
0000000000000064:  69006400 65005600 38005900 20003100 4c003100 i.d.e.V.8.Y. .1.L.1.
0000000000000078:  e6100000 010cbd7b 928b4139 4840d4ca 46a8a5d7 æ.....¼{A9H@ÔÊF"¥×
000000000000008C:  5ec0                                     ~À

Slot 3 Column 1 Offset 0x4 Length 4 Length (physical) 4

AddressID = 27795

Slot 3 Column 2 Offset 0x34 Length 36 Length (physical) 36

AddressLine1 = 1511 Roxbury Drive

Slot 3 Column 3 Offset 0x0 Length 0 Length (physical) 0

AddressLine2 = [NULL]

Slot 3 Column 4 Offset 0x58 Length 18 Length (physical) 18

City = Cliffside

Slot 3 Column 5 Offset 0x8 Length 4 Length (physical) 4

StateProvinceID = 7

Slot 3 Column 6 Offset 0x6a Length 14 Length (physical) 14

PostalCode = V8Y 1L1
```

Komenda DBCC PAGE zwraca szczegółowe informacje o strukturze wybranej strony.

Wyniki zawierają informacje o tym, jaka kolumna jest zapisana z jakim offsetem na danej stronie oraz przedstawiona jest jej wartość.

Punktacja:

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