Indeksy, optymalizator - lab3

Imię i Nazwisko: Daniel Kuc

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

Swoje odpowiedzi wpisuj w **czerwone pola**. Ważne/wymagane są komentarze. Preferowane są zrzuty ekranu, ale dołącz też polecenia formie tekstowej

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**. Jeśli jednak dzielisz z kimś serwer, to użyj swoich inicjałów:

```
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. Proszę zapoznać się z dokumentacją:

- https://docs.microsoft.com/en-us/sql/tools/dta/tutorial-database-engine-

- 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:

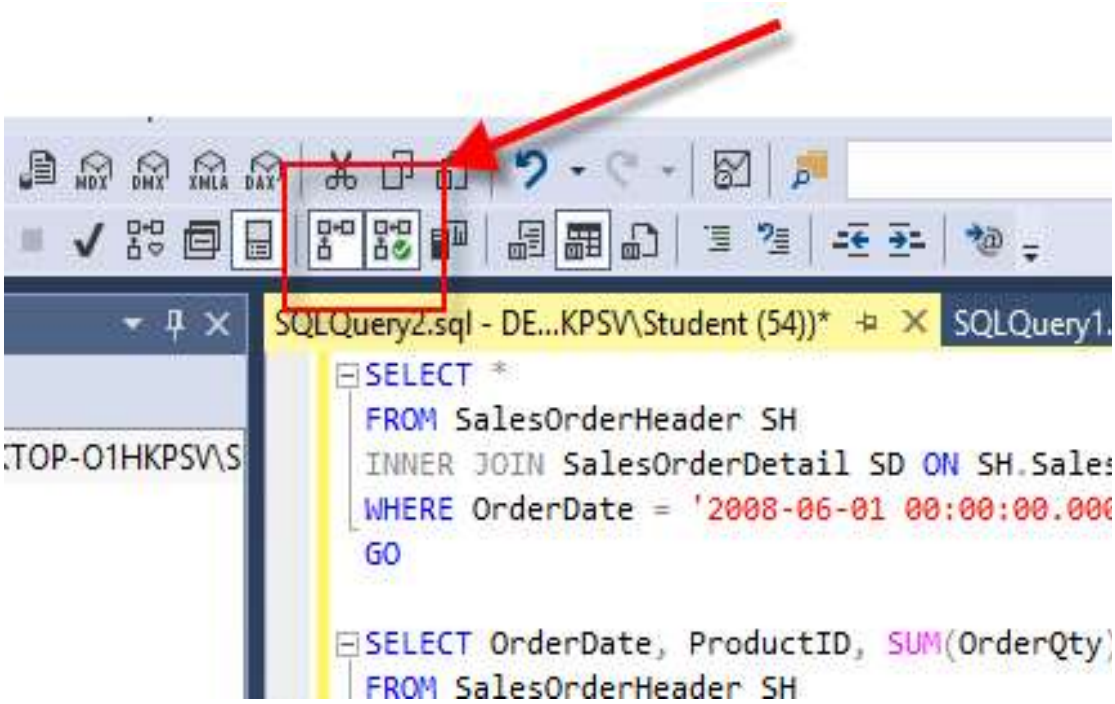
- https://docs.microsoft.com/en-us/sql/relational-databases/showplan-logical-and-physical-operators-reference

Zadanie 1 - Obserwacja

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

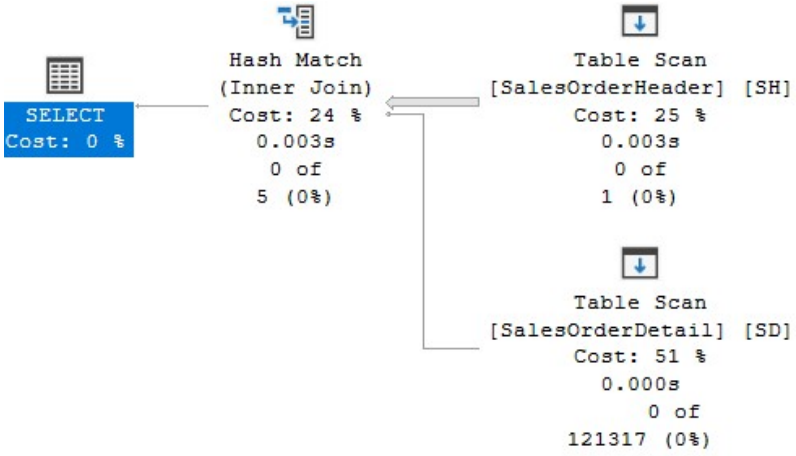
```
-- zapytanie 1
SELECT *
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
WHERE OrderDate = '2008-06-01 00:00:00.000'
GO

-- 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**:



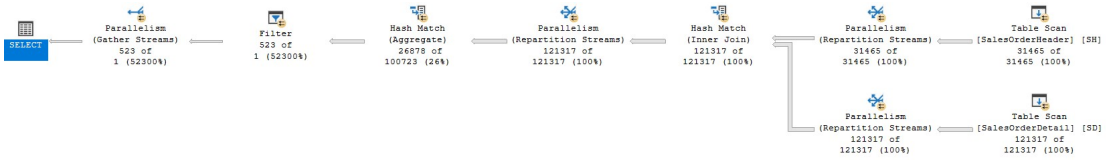
Teraz wykonaj poszczególne zapytania (najlepiej każde alalizuj oddzielnie). Co można o nich powiedzieć? Co sprawdzają? Jak można je zoptymalizować?
(Hint: aby wykonać tylko fragment kodu SQL znajdującego się w edytorze, zaznacz go i naciśnij F5)

-- zapytanie 1



Z planu wykonania wynika, że jeżeli jedna z tabel (tj. tabela SalesOrderHeader przesiana wherem), która ma trafić do joina jest pusty to join z zbiorem pusty i tak jest pusty, więc kolejne tabele nie są analizowane.

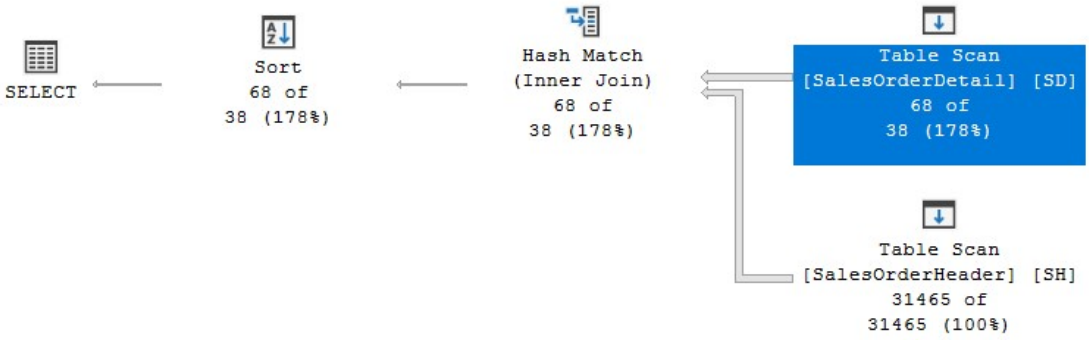
-- zapytanie 2



Zapytanie 2 pozwala nam prześledzić poszczególne etapy wykonywania. Zaczynając od łączenia tabel poprze join do późniejszej agregacji oraz obliczania sum, aż do filtrowania za pomocą klauzli having.

-- zapytanie 3
Dzieje się podobna rzecz jak w zapytaniu 1 ponieważ where tak przesiewa tabele SalesOrderHader, że zostaje zbiór pusty.

-- zapytanie 4

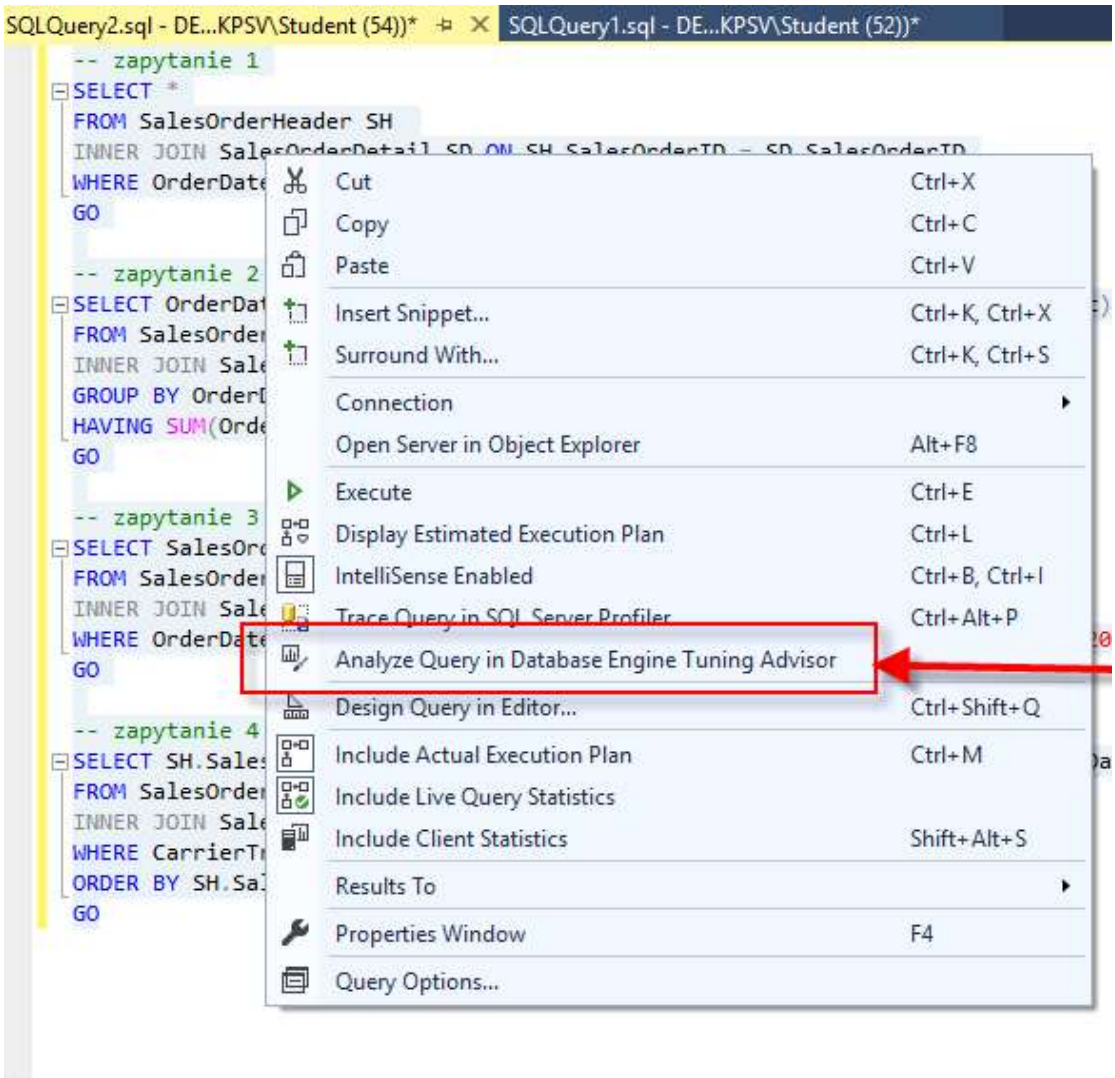


Powtarzający się wniosek z zapytania 1 jak i 3 ale warto odnotować, że serwer bazodanowy najpierw przesiewa tabele wherem a dopiero potem "zabiera" się do joina aby zbędnie nie "mielić" wierszy, które by i tak zostały odsiane ze względu na klauzule where.

Optymalizacja:
Widzimy, że każde polecenie głównie opiera się na przeszukiwaniu bazy (co raczej nie jest zaskakującym wnioskiem), więc żeby przyśpieszyc ten proces trzeba wprowadzić (zgodnie z sugestią SMSS) indeksy w bazie danych.

Zadanie 2 - Optymalizacja

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



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

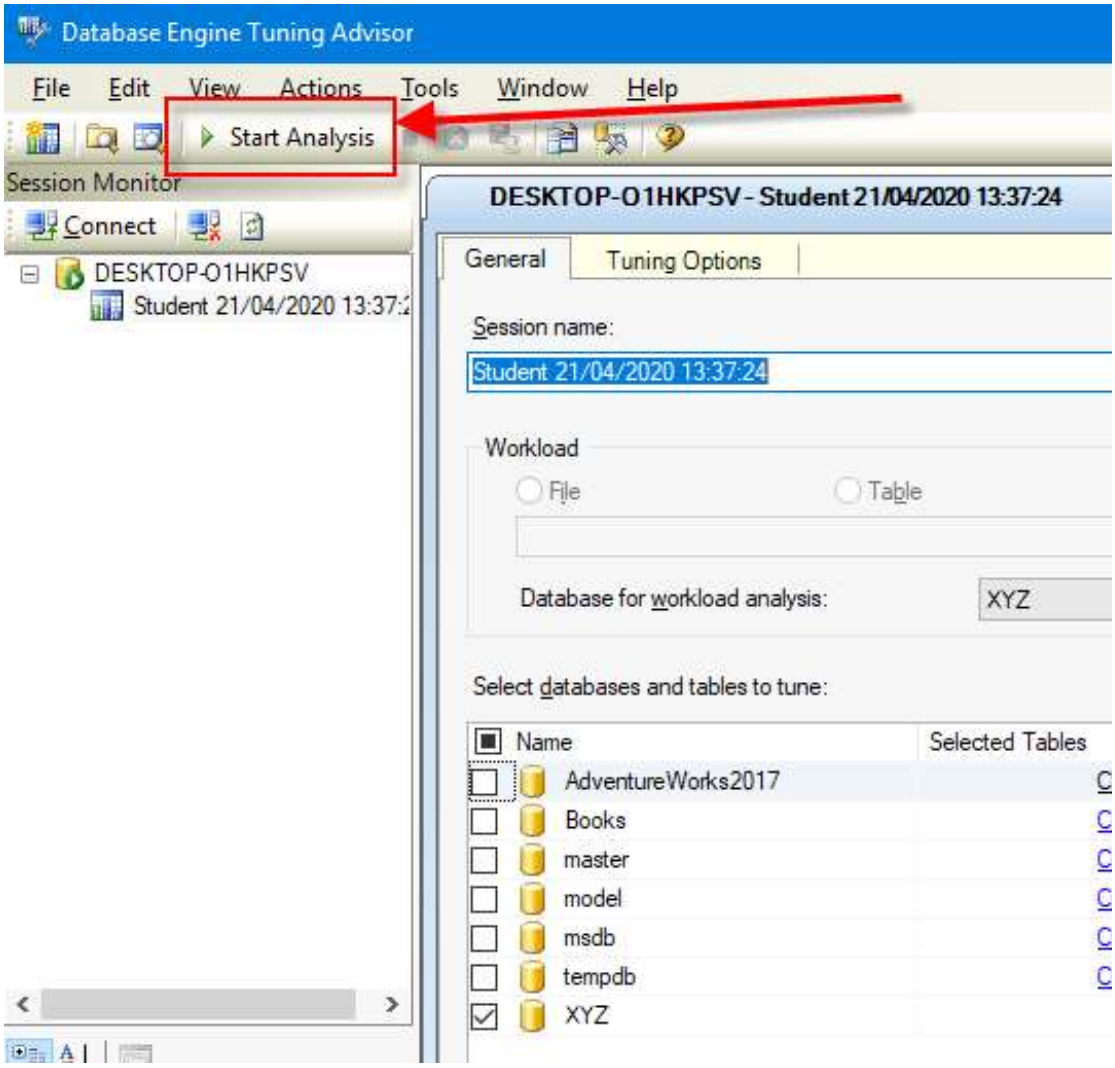
Można skonfigurować PDS, czyli strategię odkrywania partycji.
(intuicyjnie "W przypadku baz danych, partycjonowanie odnosi się do podziału dużych tabel na mniejsze części, co może poprawić wydajność zapytań." - nie jestem pewien czy to zdanie dobrze oddaje intuicje czy jest kompletnym fiaskiem)
Strategia odkrywania partycji określa, jak system ma szukać partycji w tabelach, a następnie jak ma je wykorzystać do poprawy wydajności zapytań.

Z kolei "partition strategy to employ" odnosi się do strategii partycjonowania, czyli sposobu podziału tabel na partycje.
"No Partitioning" - oznacza, że tabela nie będzie partycjonowana, a wszystkie dane będą przechowywane w jednej partycji. Jest to domyślna opcja.

"Full Partitioning" - oznacza, że tabela zostanie podzielona na wiele partycji, zwykle na podstawie jednej lub kilku kolumn. Każda partycja przechowuje różne wiersze na podstawie wartości kolumny lub kolumn, na których opiera się partycjonowanie.

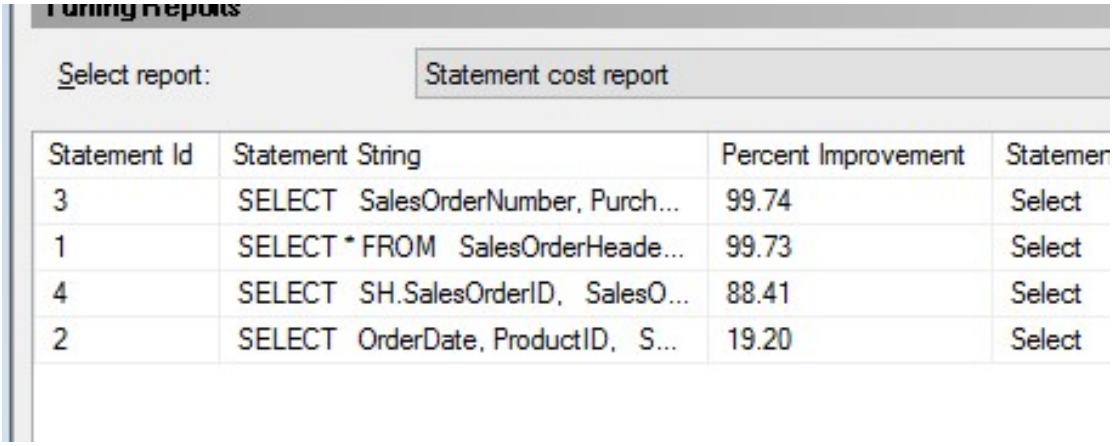
"Aligned Partitioning" - oznacza, że partycjonowanie będzie dokonywane na podstawie innej tabeli, zwaną tabelą podziału (partitioning table). Tabela podziału musi mieć kolumny o takich samych nazwach i typach danych jak tabela docelowa, a także takie same klucze główne i unikalne ograniczenia. Dzięki temu partycje w tabeli docelowej będą miały ten sam rozmiar i będą przechowywać wiersze o tych samych wartościach klucza co partycje w tabeli podziału, co pozwoli na lepsze zarządzanie partycjami i poprawi wydajność zapytań.

Użyj **Start Analysis**:

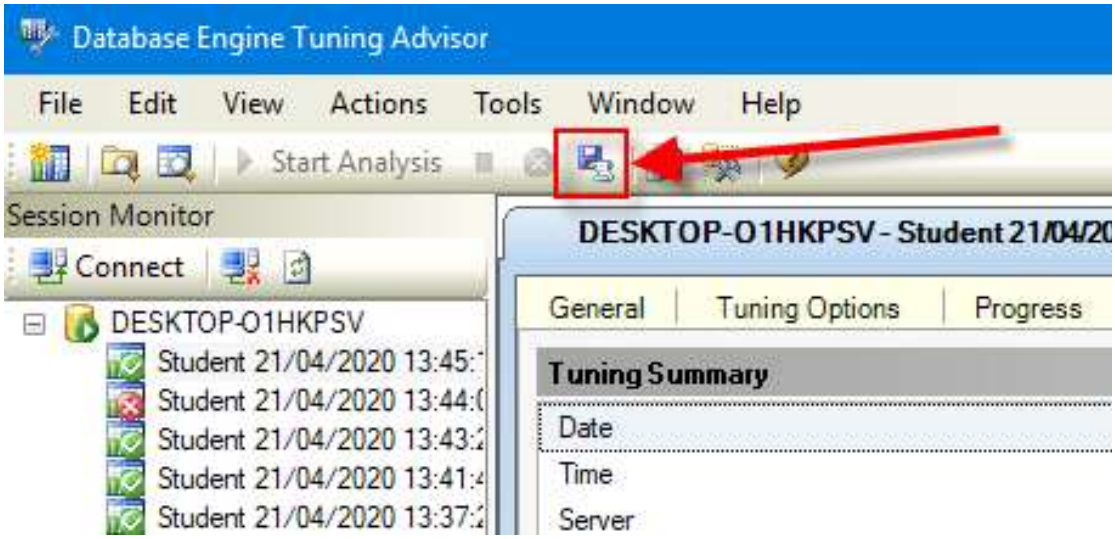


Zaobserwuj wyniki w **Recommendations**.

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



Zapisz poszczególne rekomendacje:



Uruchom zapisany skrypt w Management Studio.

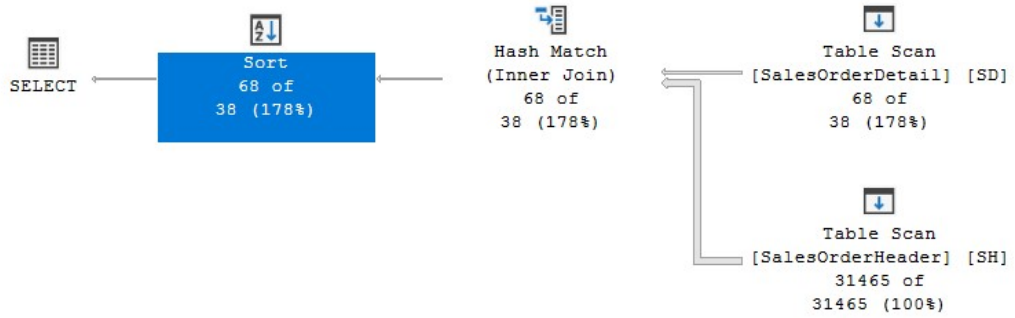
Opisz, dlaczego dane indeksy zostały zaproponowane do zapytań:

Ponieważ zaproponowane indeksy umożliwią szybsze odnajdywanie i sortowanie danych w tabelach, co poprawi wydajność zapytań.

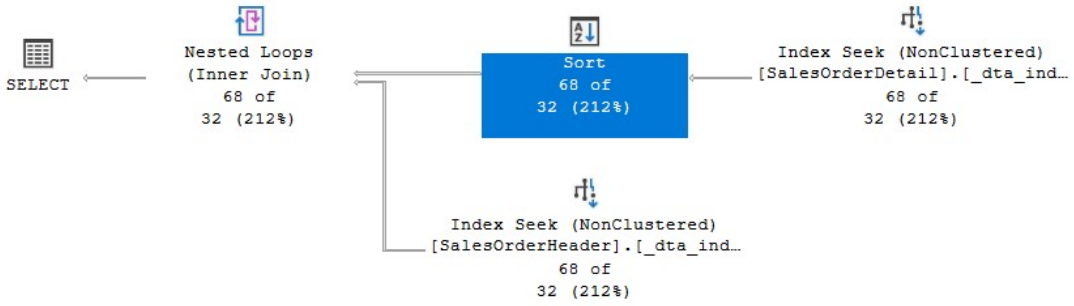
Sprawdź jak zmieniły się Execution Plany. Opisz zmiany:

Teraz na początku zamiast "Table Scan" znajduje się "Index Scan". Znacząco zmalał estimated cost ponieważ wcześniej dla zapytania 2 wynosił 0.60 a po dodaniu indeksów wynosi 0.09 czyli indeksy jak najbardziej działają i optymalizują zapytanie.

Różnice również możemy zauważyć w 4 zapytaniu gdzie sortowanie po wprowadzeniu indeksów wykonuje się wcześniej i zwraca tylko SalesOrderID, gdzie wcześniej zwracało SalesOrderID, SalesOrderNumber,... .
Bez indeksów:



Z indeksami:



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- HYPERLINK "https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-index-physical-stats-transact-sql"stats-transact-sql
- https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sy HYPERLINK "https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-indexes-transact-sql"s-indexes-transact-sql

Zadanie 3 - Kontrola “zdrowia” indeksu

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?

Według mnie, najważniejsze pola to:

index_id - identyfikator indeksu
avg_fragmentation_in_percent - średni stopień fragmentacji indeksu, który wskazuje, w jakim stopniu dane są rozproszone na stronach indeksu
page_count - liczba stron zajmowanych przez indeks, co pozwala na ocenę rozmiaru indeksu i kosztu jego utrzymywania w pamięci.
avg_page_space_used_in_percent - średni poziom wypełnienia stron, co pozwala na ocenę efektywności wykorzystania pamięci przez indeks.
record_count - liczba rekordów w indeksie, co pozwala na ocenę liczby wierszy, które indeksuje dany indeks.

Te pola pozwalają na ocenę wydajności i kosztów utrzymania indeksów, co może pomóc w optymalizacji baz danych i poprawie wydajności działania.

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
```

Screen:

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

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
```

Screen:

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

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

Reorganizacji używamy w momencie, gdy problem fragmentacji jest niewielki. Reorganizacja zużywa mniej zasobów i może zostać wykonana on-line.

Mocno pofragmentowane indeksy powinno się przebudować.

Sprawdź co przechowuje tabela sys.dm_db_index_usage_stats:

Tabela sys.dm_db_index_usage_stats przechowuje informacje o wykorzystaniu indeksów w bazie danych SQL Server.

Tabela zawiera następujące kolumny:

database_id - identyfikator bazy danych
object_id - identyfikator obiektu (np. tabeli) dla którego jest wykorzystywany indeks
index_id - identyfikator indeksu
user_seeks - liczba operacji wyszukiwania dokonanych przez użytkowników
user_scans - liczba operacji skanowania dokonanych przez użytkowników
user_lookups - liczba operacji wyszukiwania dokonanych przez użytkowników w celu znalezienia wartości, które nie znajdują się w indeksie
user_updates - liczba operacji aktualizacji dokonanych przez użytkowników na indeksie
last_user_seek - czas ostatniego wyszukiwania dokonanego przez użytkownika za pomocą tego indeksu
last_user_scan - czas ostatniego skanowania dokonanego przez użytkownika za pomocą tego indeksu
last_user_lookup - czas ostatniego wyszukiwania dokonanego przez użytkownika za pomocą tego indeksu
last_user_update - czas ostatniej aktualizacji dokonanej przez użytkownika na indeksie.

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:

```
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
```

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/

Zadanie 4 - Budowa strony indeksu

Wylistuj 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:

IndexID 1: (tylko taki występuje w tabeli wynikowej)
TYP, PID
1, 11712
2, 12272
3, 11704
10, 10476

Włącz flagę 3604 zanim zaczniesz przeglądać strony:

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

Sprawdź poszczególne strony komendą DBCC PAGE. Przykład:

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

Zapisz obserwacje ze stron. Co ciekawego udało się zaobserwować?

Typ 1 wyświetla znacznie więcej informacji włącznie z zalokowaną pamiecią.
Typ 10 nie ma zaolokowanej pamięci, podobnie jak typ 3.

Możemy zauważyć że poszczególne typy stron odpowiadają za:
1: strona danych, zawierająca wiersze danych
2: strona danych, zawierająca wiersze indentyfikatorów (tj. ROWID`s)
3: strona z indeksem zawierająca faktyczne dane wierszy
10: strona IAM (Page ID or Allocation Map), zawierająca informacje o tym, jakie strony są używane i dostępne dla alokacji w danym obiekcie bazy danych.

Punktacja

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