

Indeksy - Karta pracy nr 1

Imię i Nazwisko: Wojciech Kosztyla

Swoje odpowiedzi wpisuj w **czerwone pola**. Preferowane są zrzuty ekranu, **wymagane** komentarze.

Co jest potrzebne?

Do wykonania ćwiczenia potrzebne są:

MS SQL Server wersja co najmniej 2016,
przykładowa baza danych **AdventureWorks2017**.

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

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-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):

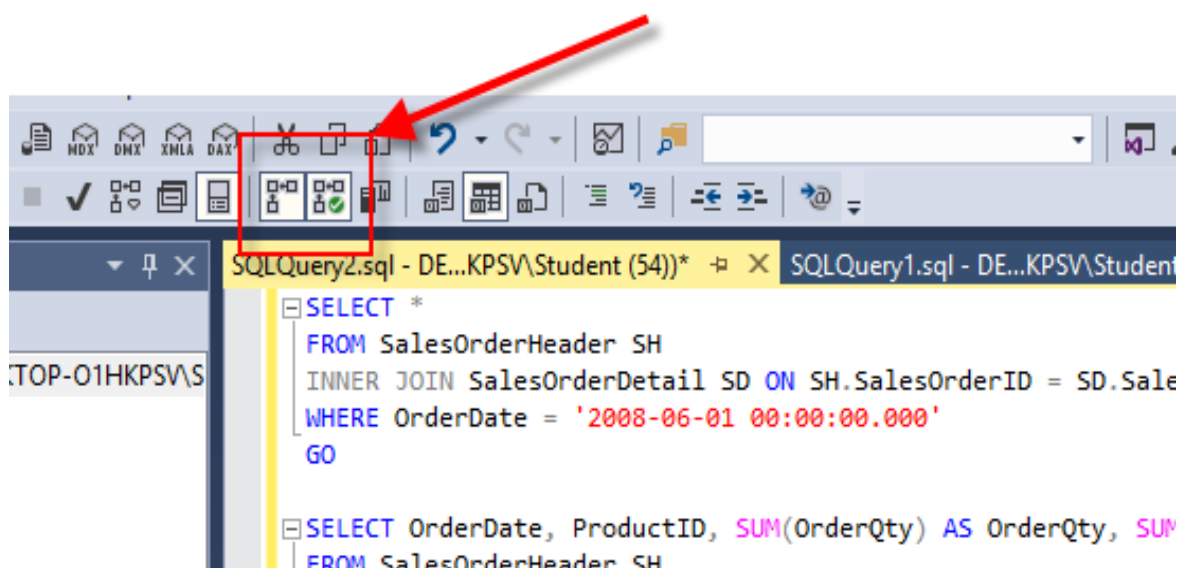
```
-- zapytanie 1
SELECT *
FROM SalesOrderHeader SH
INNER JOIN SalesOrderDetail SD ON SH.SalesOrderID = SD.SalesOrderID
WHERE OrderDate = '2008-06-01 00:00:00.000'
GO

-- 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)

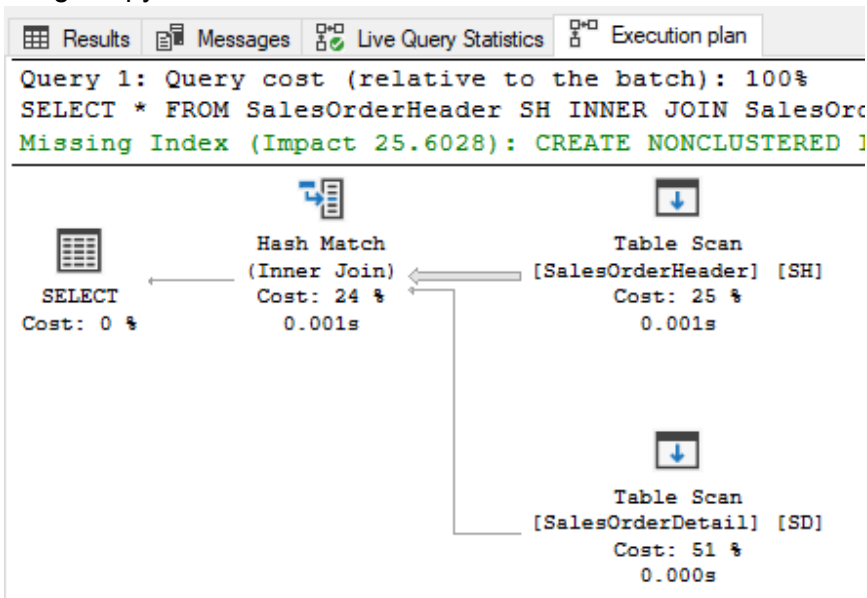
W pierwszym zapytaniu filtrujemy tablicę SalesOrderHeader przez wybór konkretnej daty

```
Predicate
[XYZ].[dbo].[SalesOrderHeader].[OrderDate] as [SH].
[OrderDate]='2008-06-01 00:00:00.000'
Object
```

po czym łączymy inner joinem z tablicą z dodatkowymi szczegółami dla zamówienia. Ta kwerenda nie znalazła żadnego wiersza

| Results | Messages | Live Query Statistics | Execution plan |
|--------------|----------------|-----------------------|----------------|
| | | | |
| SalesOrderID | RevisionNumber | OrderDate | DueDate |

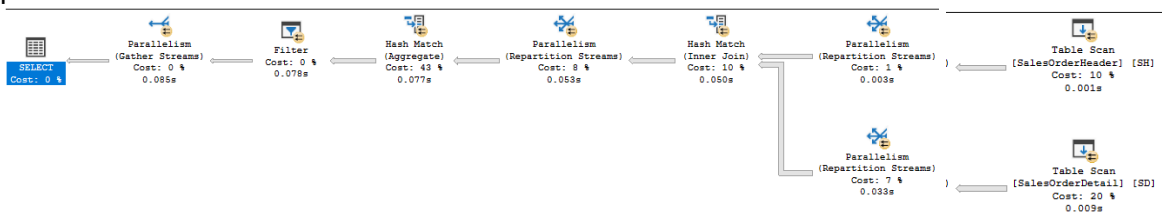
Patrząc na Execution plan widzimy, że skan SalesOrderDetail kosztuje około 51% kosztu całego zapytania



Dostajemy również wiadomość z propozycją optymalizacji - stworzeniem indeksu na kolumnie OrderDate

```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]  
ON [dbo].[SalesOrderHeader] ([OrderDate])
```

W drugiej kwerendzie agregujemy wyniki zapytania na łączonej tablicy i filtrujemy jednym parametrem



Ponownie program podpowiada nam sposób optymalizacji poprzez stworzenie indeksu

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

Trzecia kwerenda jest bardzo podobna do tej pierwszej. Jediną różnicą jest wyciąganie 4 kolumn zamiast wszystkich i filtrowanie przez listę.

Predicate

```
[XYZ].[dbo].[SalesOrderHeader].[OrderDate] as [SH].
[OrderDate]='2008-06-01 00:00:00.000' OR [XYZ].[dbo].
[SalesOrderHeader].[OrderDate] as [SH].[OrderDate]='2008-
06-02 00:00:00.000' OR [XYZ].[dbo].[SalesOrderHeader].
[OrderDate] as [SH].[OrderDate]='2008-06-03 00:00:00.000'
OR [XYZ].[dbo].[SalesOrderHeader].[OrderDate] as [SH].
[OrderDate]='2008-06-04 00:00:00.000' OR [XYZ].[dbo].
[SalesOrderHeader].[OrderDate] as [SH].[OrderDate]='2008-
06-05 00:00:00.000'
```

Object

Optymalizacja ponownie zaproponowana przez program

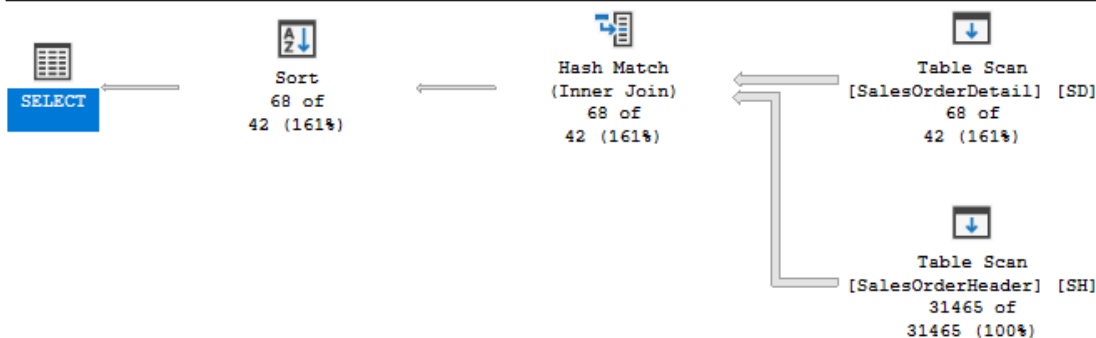
```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[SalesOrderHeader] ([OrderDate])
INCLUDE ([SalesOrderID],[DueDate],[ShipDate],[SalesOrderNumber],[PurchaseOrderNumber])
```

Czwarta kwerenda ponownie polega na połączeniu dwóch tablic, filtrowaniu poprzez listę możliwych wartości

```
WHERE CarrierTrackingNumber IN ('EF67-4713-BD', '6C08-4C4C-B8')
```

 i tym razem posortowaniu.

progress:100% Missing Index (Impact 57.4106): CREATE NONCLUSTERED INDEX [<Name of M

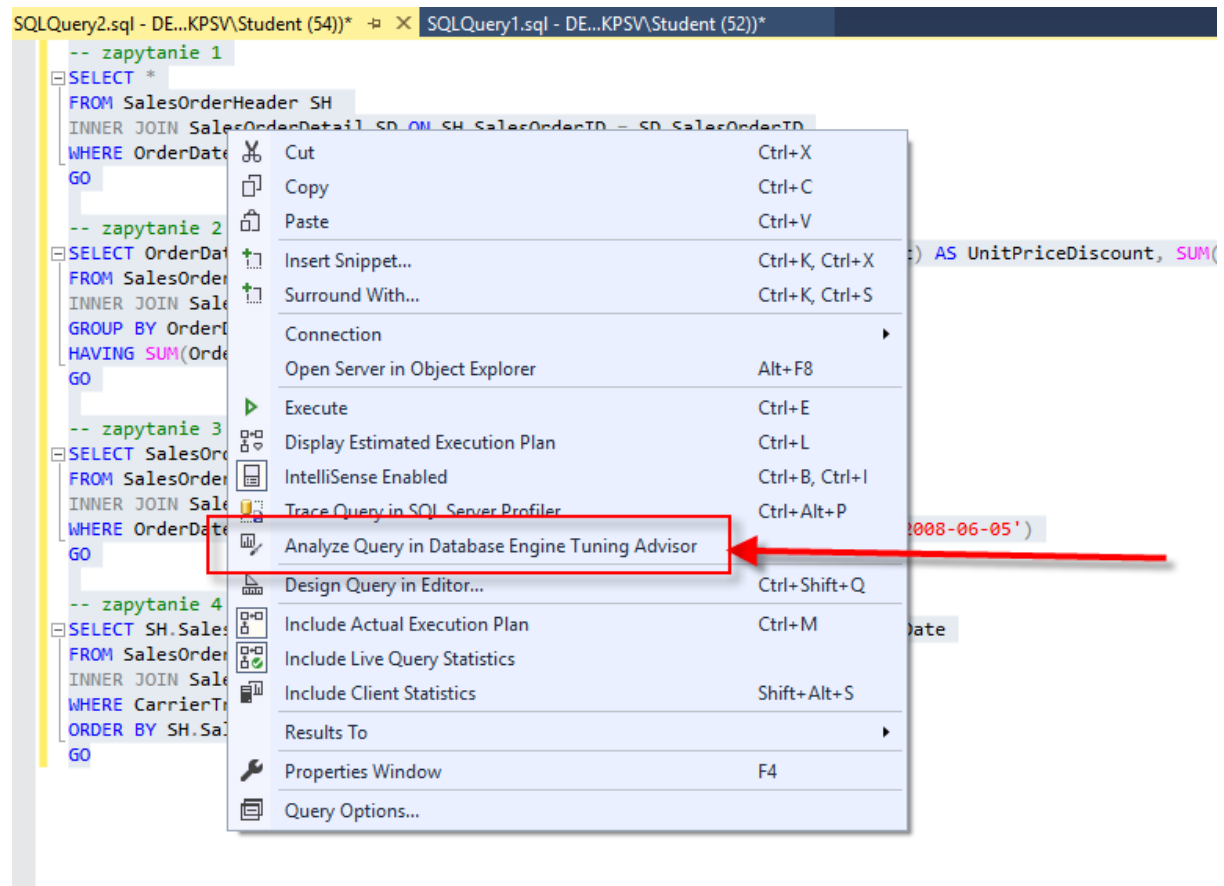


I po raz kolejny odstawiamy propozycję optymalizacji poprzez stworzenie indeksu

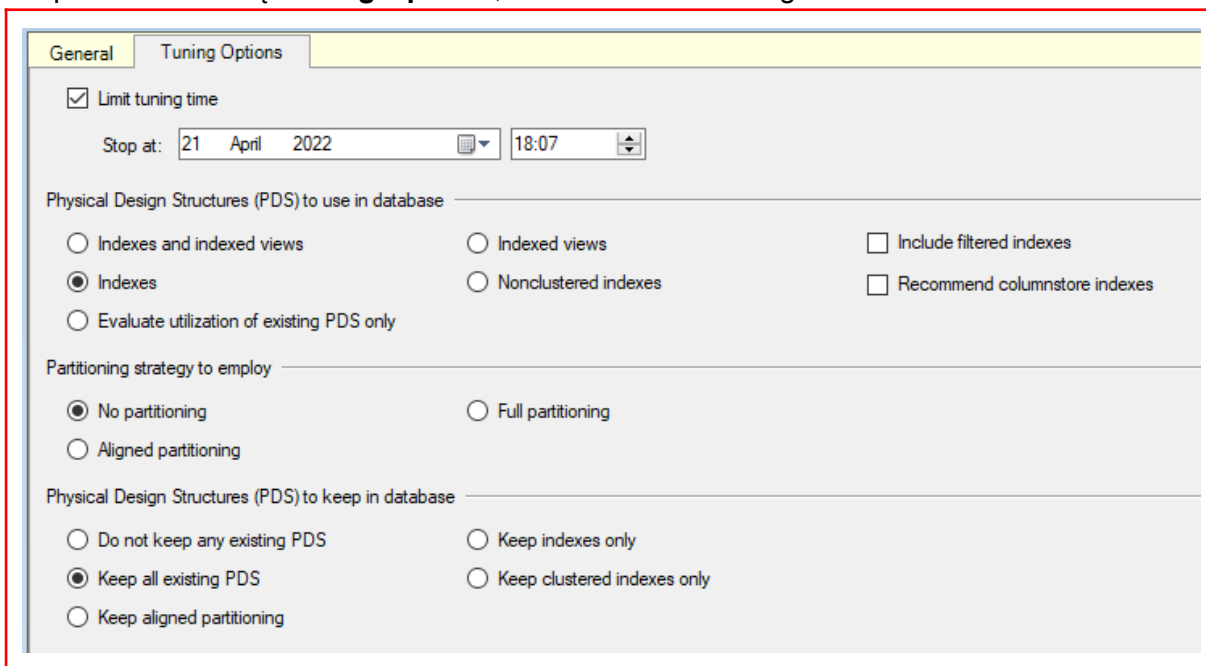
```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[SalesOrderDetail] ([CarrierTrackingNumber])
INCLUDE ([SalesOrderID])
```

Zadanie 2 - Optymalizacja

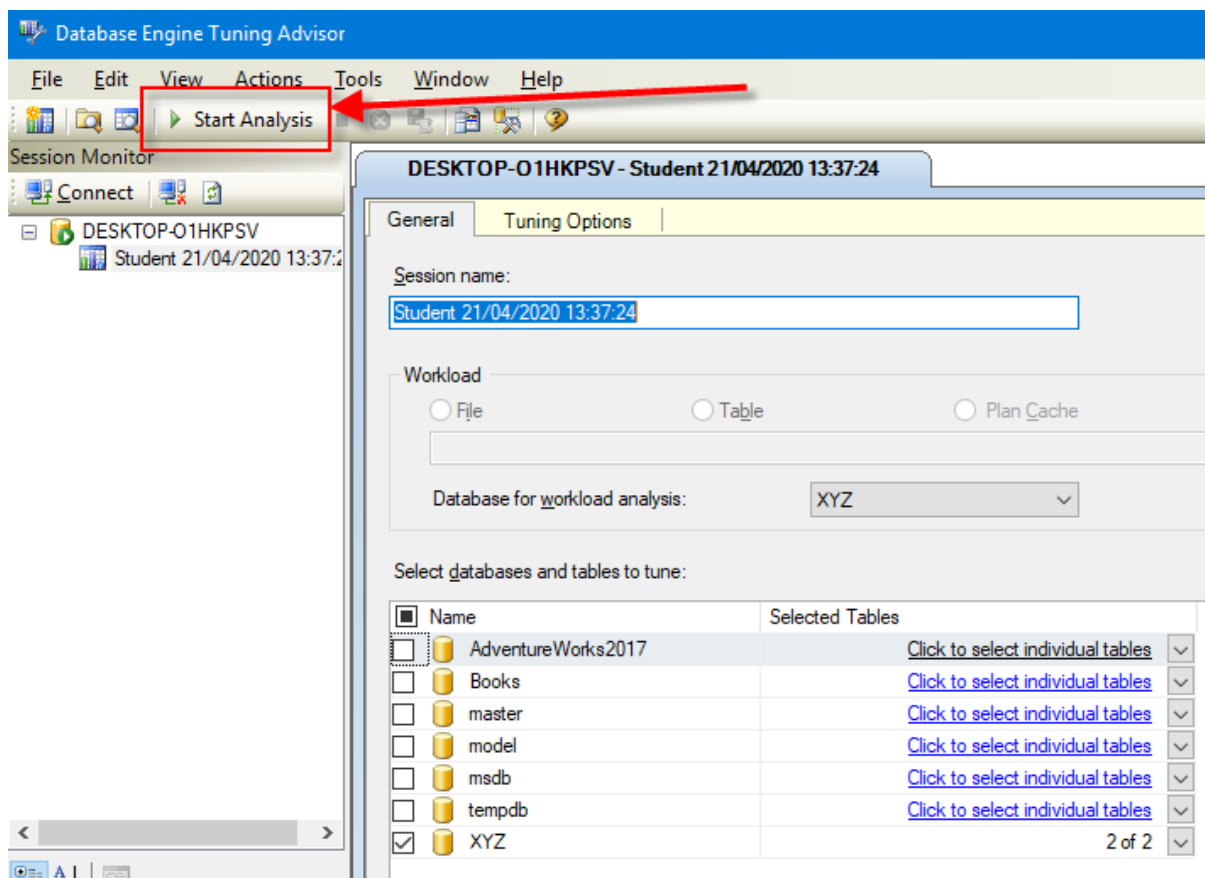
Zaznacz **wszystkie zapytania**, i uruchom je w Database Engine Tuning Advisor:



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



Użyj **Start Analysis**:



Zaobserwuj wyniki w **Recommendations**.

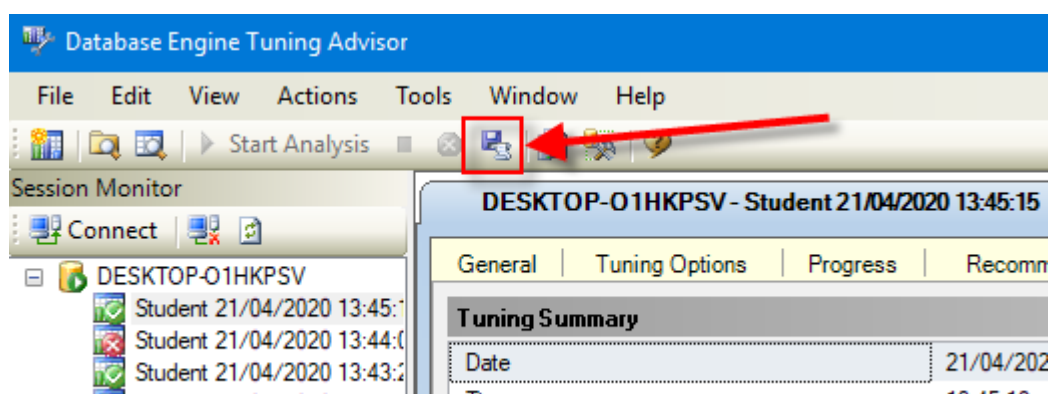
Przejdź do zakładki **Reports**. Sprawdź poszczególne raporty. Główną uwagę zwróć na koszty i ich poprawę:

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:



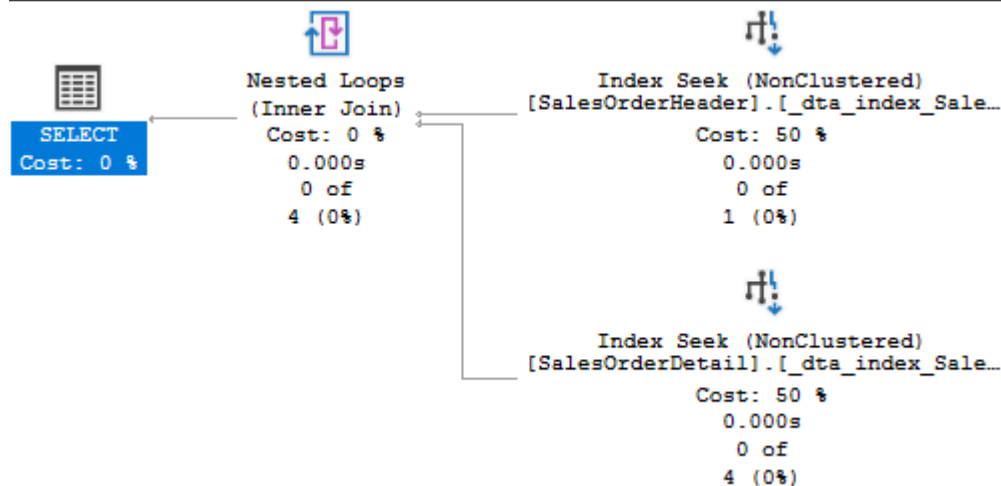
Uruchom zapisany skrypt w Management Studio.

Opisz, dlaczego dane indeksy zostały zaproponowane do zapytań:

Indeksy znacząco poprawiają prędkość zapytań. Indeksy, które zostały zaproponowane są dobrane pod kwerendy, które wykonywaliśmy wcześniej.

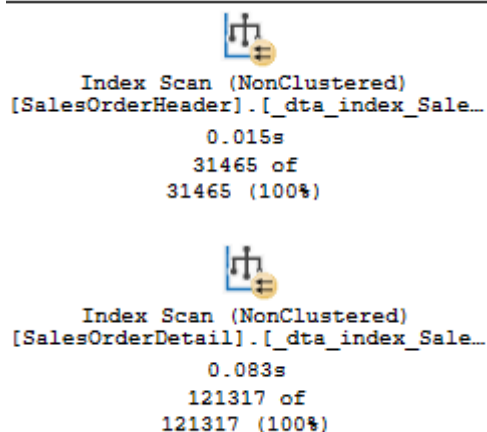
Sprawdź jak zmieniły się Execution Plany. Opisz zmiany:

W pierwszej kwerendzie szukanie po tablicy zostało zamienione na szukanie indeksu.

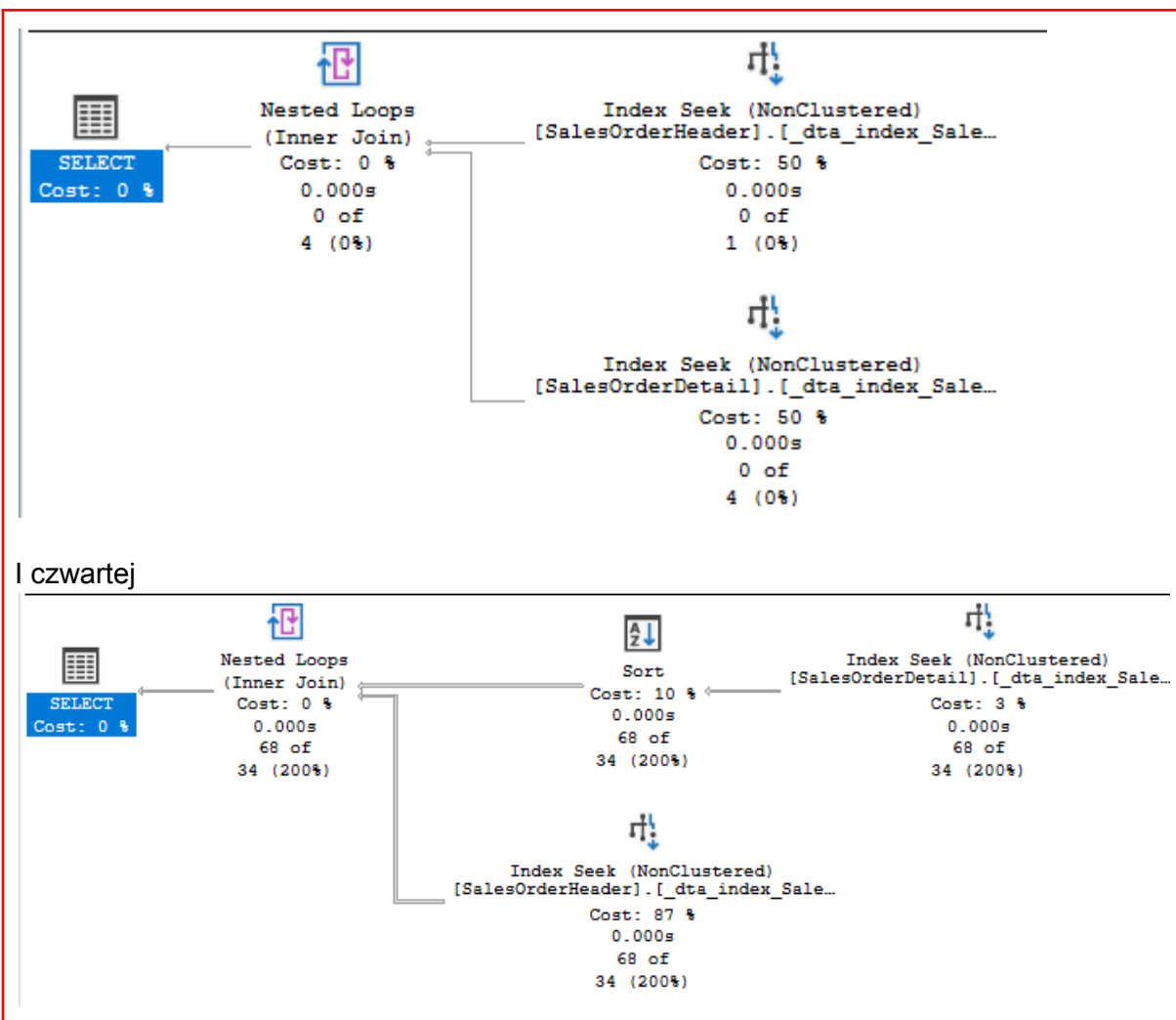


Czas wykonania spadł też do tak małej wartości, że nie została nawet zmierzona (0.000s).

W drugiej kwerendzie tak samo - każdy Table Scan zamienił się na Index Scan:



Podobnie w trzeciej



I czwartej

Dokumentacja

Celem 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>

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
```

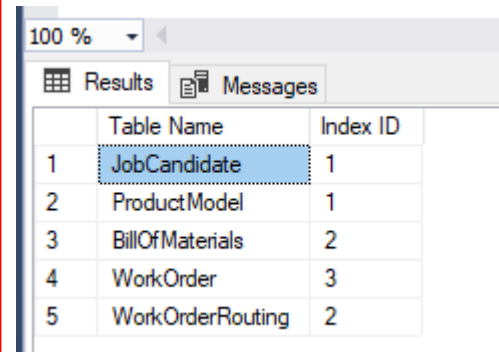
Jakie są według Ciebie najważniejsze pola?

```
avg_fragmentation_in_percent,  
record_count,  
avg_record_size_in_bytes
```

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:



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there is a zoom level of 100% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'Table Name' and 'Index ID'. The table contains five rows of data, with the first row highlighted in blue.

| | 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>)

Przebudowa indeksu to w praktyce usunięcie indeksu i stworzenie go od nowa, natomiast reorganizacja jest niczym defragmentacja - czyści strony indeksów bez usuwania ich.

Sprawdź co przechowuje tabela sys.dm_db_index_usage_stats:

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

Wynik:

| | database_id | object_id | index_id | user_seeks | user_scans | user_lookups | user_updates | last_user_seek | last_user_scan |
|---|-------------|------------|----------|------------|------------|--------------|--------------|-------------------------|-------------------------|
| 1 | 4 | 1686297067 | 2 | 0 | 0 | 0 | 1 | NULL | NULL |
| 2 | 4 | 1686297067 | 1 | 0 | 3 | 0 | 1 | NULL | 2022-04-21 17:10:37.527 |
| 3 | 4 | 1302295699 | 1 | 15 | 0 | 0 | 3 | 2022-04-21 17:25:33.470 | NULL |
| 4 | 4 | 34815186 | 1 | 0 | 0 | 0 | 1 | NULL | NULL |

| last_user_lookup | last_user_update | system_seeks | system_scans | system_lookups | system_updates | last_system_seek |
|------------------|-------------------------|--------------|--------------|----------------|----------------|------------------|
| NULL | 2022-04-21 17:10:37.503 | 0 | 1 | 0 | 0 | NULL |
| NULL | 2022-04-21 17:10:37.503 | 0 | 1 | 0 | 0 | NULL |
| NULL | 2022-04-21 17:10:37.580 | 0 | 1 | 0 | 0 | NULL |
| NULL | 2022-04-21 17:10:37.530 | 0 | 0 | 0 | 0 | NULL |

| last_system_scan | last_system_lookup | last_system_update |
|-------------------------|--------------------|--------------------|
| 2022-04-21 17:10:37.507 | NULL | NULL |
| 2022-04-21 17:10:37.517 | NULL | NULL |
| 2022-04-21 17:10:33.387 | NULL | NULL |
| NULL | NULL | NULL |

Tabela ta zawiera statystyki użycia indeksów. Widzimy np. ID bazy danych, ID indeksu, ile razy użytkownik z niego skorzystał, kiedy to zrobił.

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:

```
USE AdventureWorks2017
```

```
-- REBUILD
```

```
--Table to hold results
```

```
DECLARE @tablevar TABLE(Ingid 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
```

```
-- REORGANIZE
```

```

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

```

Dokumentacja

Celem 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-unique-index-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:

Indeks 1:

| | PageFID | PagePID | IAMFID | IAMPID | ObjectID | IndexID | PartitionNumber | PartitionID | iam_chain_type | PageType |
|---|---------|---------|--------|--------|------------|---------|-----------------|-------------------|----------------|----------|
| 1 | 1 | 10474 | NULL | NULL | 1029578706 | 1 | 1 | 72057594047889408 | In-row data | 10 |
| 2 | 1 | 11712 | 1 | 10474 | 1029578706 | 1 | 1 | 72057594047889408 | In-row data | 1 |

Indeks 2:

| | PageFID | PagePID | IAMFID | IAMPID | ObjectID | IndexID | PartitionNumber | PartitionID | iam_chain_type | PageType |
|---|---------|---------|--------|--------|------------|---------|-----------------|-------------------|----------------|----------|
| 1 | 1 | 10472 | NULL | NULL | 1029578706 | 2 | 1 | 72057594052542464 | In-row data | 10 |
| 2 | 1 | 5872 | 1 | 10472 | 1029578706 | 2 | 1 | 72057594052542464 | In-row data | 2 |

Włącz flagę 3604 zanim zaczniesz przeglądać strony:

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

Sprawdź poszczególne strony komendą DBCC PAGE. Przykład:

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

Zapisz obserwację ze stron. Co ciekawego udało się zaobserwować?

```
SQLQuery10.sql - D:\KPSV\Student (07) - SQLQuery10.sql -
DBCC TRACEON (3604);
DBCC PAGE('AdventureWorks2017', 1, 10474, 3);
GO
```

PAGE: (1:10474)

BUFFER:

BUF @0x000001A8E4F6B040

| | | |
|----------------------------|----------------------------|------------------------------------|
| bpage = 0x000001A8C7E18000 | bhash = 0x0000000000000000 | bpageno = (1:10474) |
| bdbid = 5 | breferences = 0 | bcputicks = 0 |
| bsampleCount = 0 | bUse1 = 6092 | bstat = 0x9 |
| blog = 0x15a | bnext = 0x0000000000000000 | bDirtyContext = 0x0000000000000000 |
| bstat2 = 0x0 | | |

PAGE HEADER:

Page @0x000001A8C7E18000

| | | |
|---|-------------------------------------|-----------------------|
| m_pageId = (1:10474) | m_headerVersion = 1 | m_type = 10 |
| m_typeFlagBits = 0x0 | m_level = 0 | m_flagBits = 0x200 |
| m_objId (AllocUnitId.idObj) = 268 | m_indexId (AllocUnitId.idInd) = 256 | |
| Metadata: AllocUnitId = 72057594055491584 | | |
| Metadata: PartitionId = 72057594047889408 | | Metadata: IndexId = 1 |
| Metadata: ObjectId = 1029578706 | m_prevPage = (0:0) | m_nextPage = (0:0) |
| pminlen = 90 | m_slotCnt = 2 | m_freeCnt = 6 |
| m_freeData = 8182 | m_reservedCnt = 0 | m_lsn = (37:1348:30) |
| m_xactReserved = 0 | m_xdesId = (0:0) | m_ghostRecCnt = 0 |
| m_tornBits = 2062223489 | DB Frag ID = 1 | |

Allocation Status

| | | |
|--|------------------------|--------------------------|
| GAM (1:2) = ALLOCATED | SGAM (1:3) = ALLOCATED | |
| PFS (1:8088) = 0x70 IAM_PG MIXED_EXT ALLOCATED | 0_PCT_FULL | DIFF (1:6) = NOT CHANGED |
| ML (1:7) = NOT MIN_LOGGED | | |

IAM: Header @0x0000004730FFA064 Slot 0, Offset 96

| | | |
|--------------------|----------------|------------------|
| sequenceNumber = 0 | status = 0x0 | objectId = 0 |
| indexId = 0 | page_count = 0 | start_pg = (1:0) |

IAM: Single Page Allocations @0x0000004730FFA08E

| | | |
|----------------|----------------|----------------|
| Slot 0 = (0:0) | Slot 1 = (0:0) | Slot 2 = (0:0) |
| Slot 3 = (0:0) | Slot 4 = (0:0) | Slot 5 = (0:0) |
| Slot 6 = (0:0) | Slot 7 = (0:0) | |

IAM: Extent Alloc Status Slot 1 @0x0000004730FFA0C2

| | | |
|-----------|-------------|-----------------|
| (1:0) | - (1:11704) | = NOT ALLOCATED |
| (1:11712) | - (1:11976) | = ALLOCATED |
| (1:11984) | - (1:11992) | = NOT ALLOCATED |
| (1:12000) | - (1:12008) | = ALLOCATED |
| (1:12016) | - (1:12024) | = NOT ALLOCATED |
| (1:12032) | - (1:12040) | = ALLOCATED |
| (1:12048) | - (1:12056) | = NOT ALLOCATED |
| (1:12064) | - (1:12072) | = ALLOCATED |
| (1:12080) | - (1:12088) | = NOT ALLOCATED |
| (1:12096) | - (1:12104) | = ALLOCATED |
| (1:12112) | - (1:12120) | = NOT ALLOCATED |
| (1:12128) | - (1:12136) | = ALLOCATED |
| (1:12144) | - | = NOT ALLOCATED |
| (1:12152) | - (1:12160) | = ALLOCATED |
| (1:12168) | - | = NOT ALLOCATED |
| (1:12176) | - (1:12184) | = ALLOCATED |
| (1:12192) | - (1:12264) | = NOT ALLOCATED |
| (1:12272) | - | = ALLOCATED |
| (1:12280) | - (1:33784) | = NOT ALLOCATED |

“Ciekawą” obserwacją może być przestrzeń niezaalokowana w pełni, lecz z “dziurami”, co było do przewidzenia z racji typu indeksu.

Sprawdziłem wszystkie zapisane strony. Z wyświetlonych danych jestem w stanie stwierdzić, że:

PageType 1 zawiera dane z bazy

PageType 2 zawiera dane o poszczególnych rekordach z bazy

PageType 10 zawiera metadane indeksu