Optymalizacja zapytań SQL

Na przykładzie SQL Server 2008

Agenda

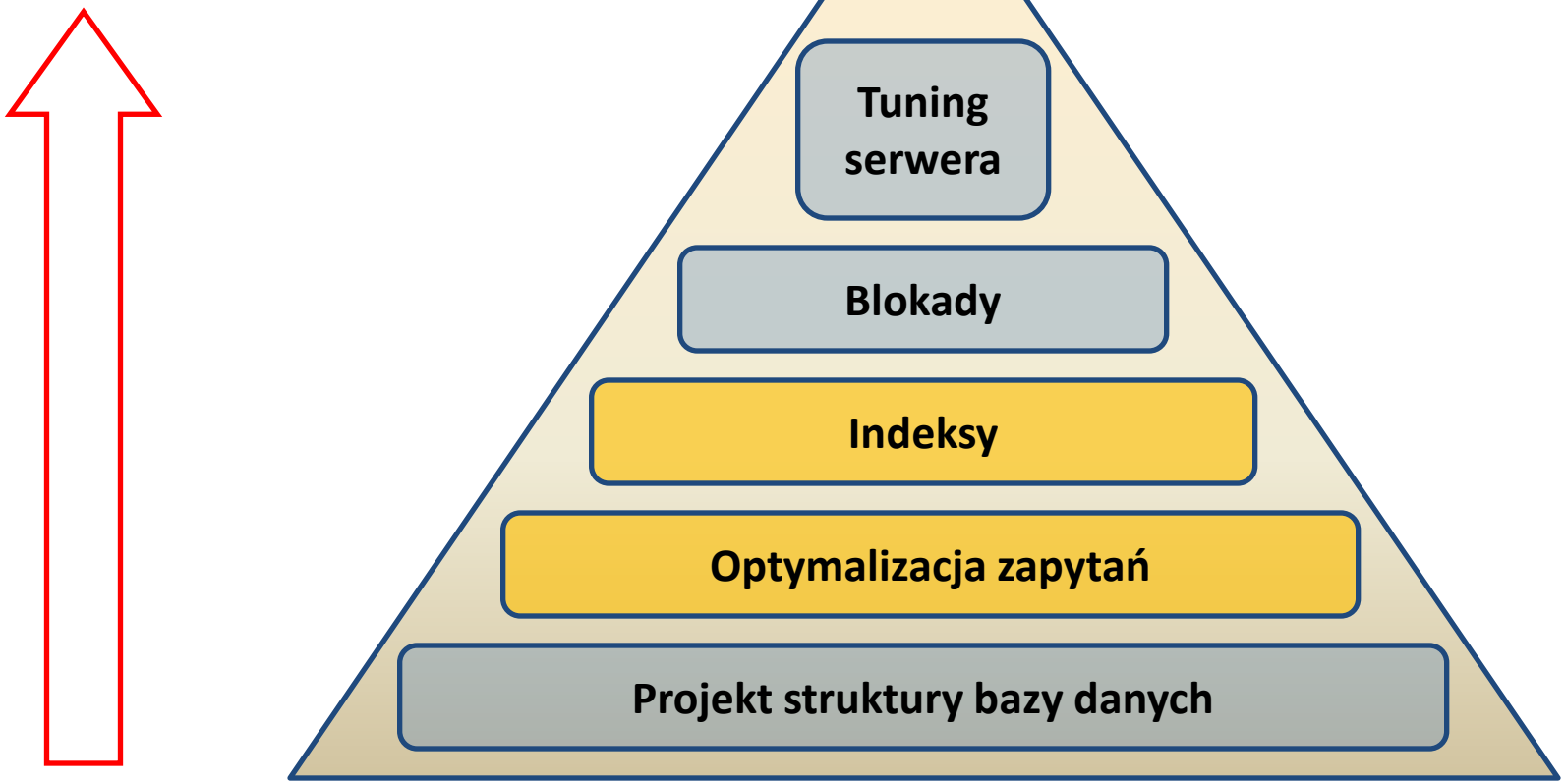
- **Po co optymalizować?**
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Po co optymalizować?

- Mamy coraz lepszy sprzęt
- Można dokupować kolejne:
  - procesory
  - pamięć
  - dyski
  - Serwery
- Pozwala to na nadążanie za spadkiem wydajności (wraz ze wzrostem ilości danych i użytkowników systemu)
- Niestety na krótką metę i nie zawsze pomaga (proporcjonalnie do kosztów)
- Taniej i (niekoniecznie) łatwiej jest odciąć się od spraw sprzętowych i skupić na samej bazie danych
- Efekty potrafią być oszałamiające!

Model optymalizacji wydajności

- Strzałka – kolejność realizacji
- Szerokość bloku – skala możliwych do uzyskania efektów



Model optymalizacji wydajności

- Projekt struktury bazy
  - Najważniejsze zadanie w procesie!
  - Ze złym projektem niewiele da się zrobić
  - Warstwa abstrakcji danych!
- Optymalizacja zapytań
  - operacje na zbiorach zamiast podejścia iteracyjnego
- Indeksy
  - Pomost pomiędzy zapytaniem a danymi
  - Mogą ogromnie przyspieszyć dostęp do danych
  - Nie zamaskują efektu źle napisanych zapytań
- Blokady
  - Baza „śmiga” z jednym użytkownikiem, przy kilku ledwo zipie..
- Tuning serwera
  - Kolejne procesory, RAM, dyski…

Agenda

- Po co optymalizować?
- **Fizyczna organizacja danych w SQL Server 2008**
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Fizyczna organizacja danych w SQL Server 2008

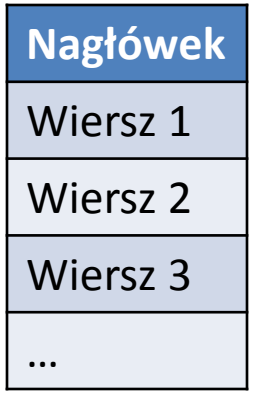
- Logicznie tabela składa się z wierszy, które składają się z kolumn.

ContactID	Title	FirstName	LastName	EmailAddress	Phone
1	Mr.	Gustavo	Achong	gustavo0@adventure-works.com	398-555-0132
2	Ms.	Catherine	Abel	catherine0@adventure-works.com	747-555-0171
3	Ms.	Kim	Abercrombie	kim2@adventure-works.com	334-555-0137
4	Sr.	Humberto	Acevedo	humberto0@adventure-works.com	599-555-0127
5	Sra	Pilar	Ackerman	pilar1@adventure-works.com	1 (11) 500 555-0132

- Jak te dane przechowywane są na dysku?
- Jakie są ograniczenia przy definiowaniu tabel?
- Jaki ma to wpływ na wydajność?

Fizyczna organizacja danych w SQL Server 2008

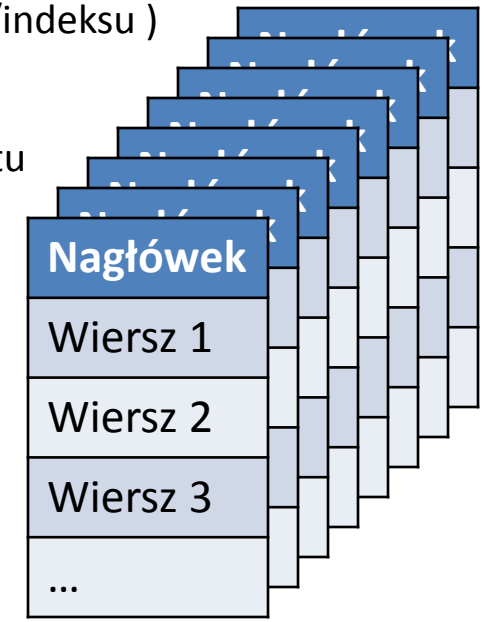
- Podstawowa jednostka – strona (page)
  - Rozmiar: 8 KB (dokładnie 8060 bajtów na dane)
  - Jest to jednocześnie maksymalna długość wiersza (nie licząc kolumn przechowywanych na osobnych stronach)
  - Wiersz nie może być podzielony pomiędzy strony.



- Rodzaje stron
  - **data** (wszystkie dane z wyjątkiem kolumn typów: **text, ntext, image, nvarchar(max), varchar(max), varbinary(max), xml**)
  - **index** (wpisy indeksów)
  - **text/image** (**text, ntext, image, nvarchar(max), varchar(max), varbinary(max), xml** oraz niemieszczące się w wierszu: **varchar, nvarchar, varbinary**)
  - **GAM**, (Global Allocation Map) **SGAM** (Shared GAM), **IAM** (Index Allocation Map) – wrócimy do nich!

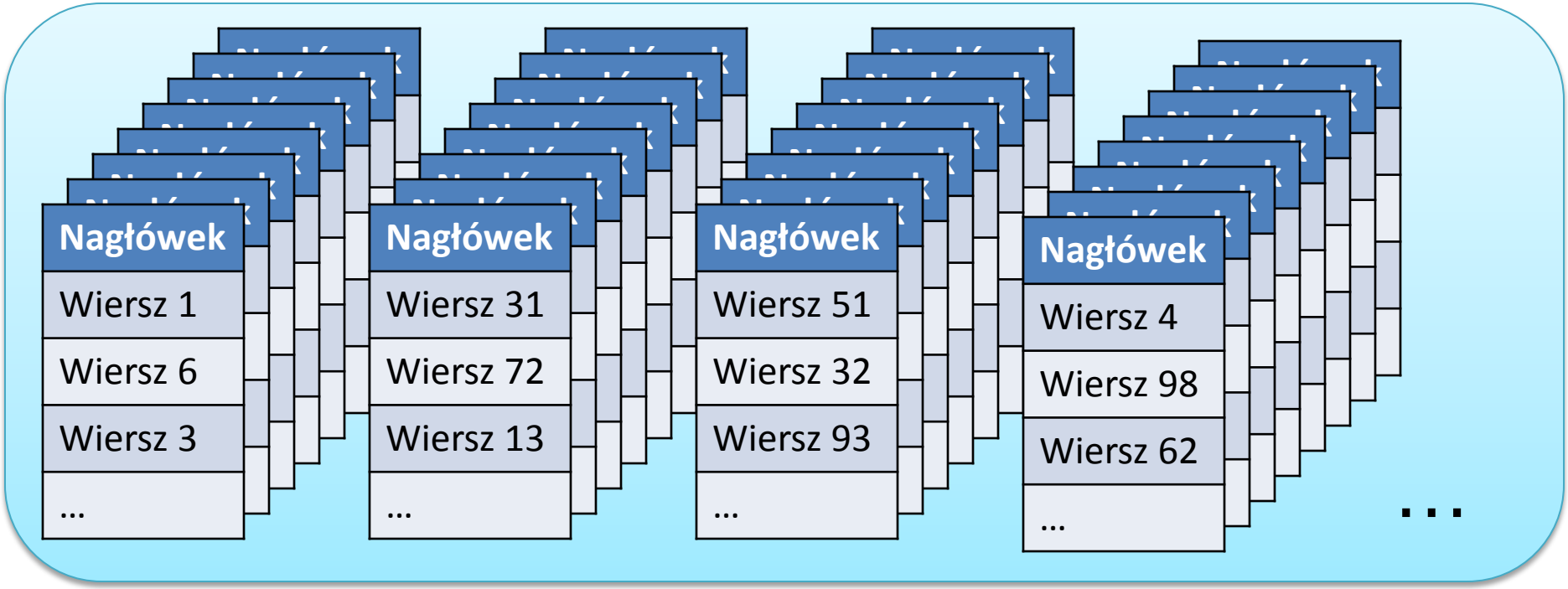
Fizyczna organizacja danych w SQL Server 2008

- 8 KB (strona) to trochę mało…
- 8 stron – 64 KB to w sam raz na jednostkę alokacji!
- Jednostka ta zwana jest obszarem (extent).
- Rodzaje obszarów
  - Jednolite (uniform extent)
    - Zawierają strony należące do jednego obiektu (tabeli /indeksu)
  - Mieszane (mixed extent)
    - Zawierają strony należące do więcej niż jednego obiektu
- Alokowane i odczytywane są zawsze całe obszary a nie pojedyncze strony



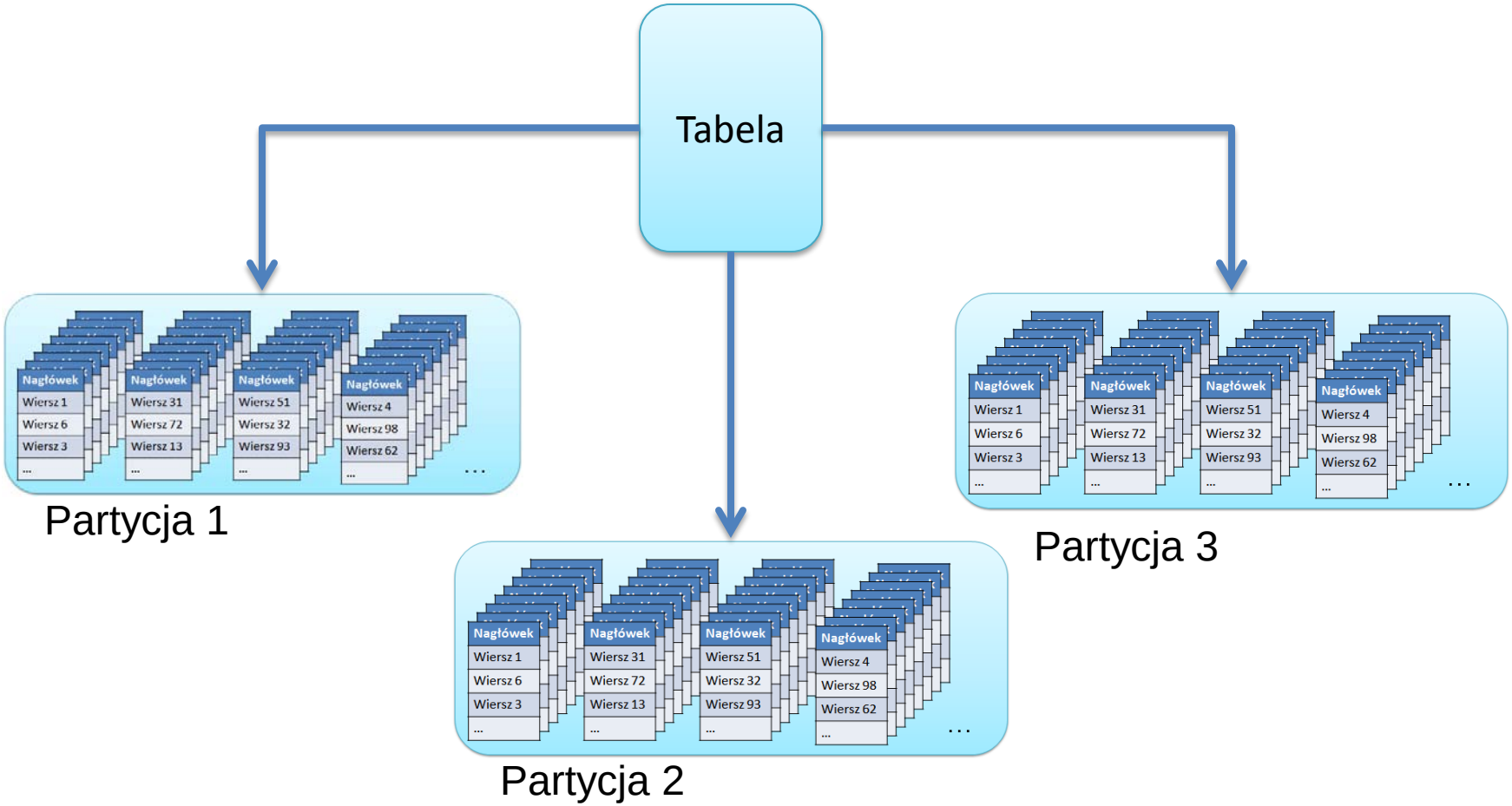
Fizyczna organizacja danych w SQL Server 2008

- Sterta (heap) – zbiór obszarów zawierających dane z jednej tabeli (lub partycji w przypadku tabel partycjonowanych)
- Dane nie są ze sobą powiązane w żaden sposób
- Wyszukiwanie wymaga przejrzenia wszystkich stron



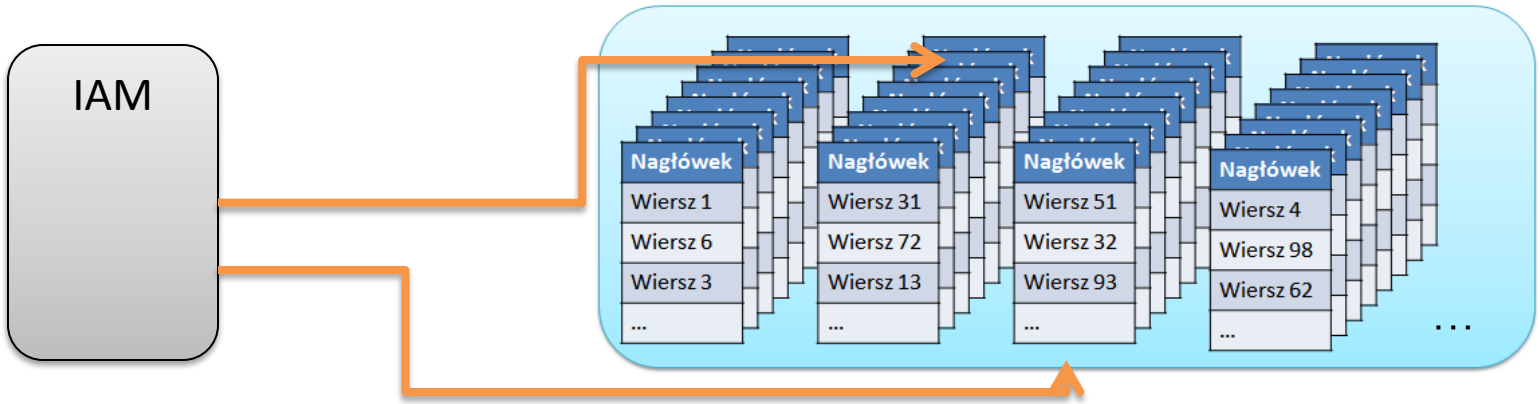
Fizyczna organizacja danych w SQL Server 2008

- Tabela może składać się z jednej lub więcej partycji
- Sterta jest tworzona osobno dla każdej partycji



Fizyczna organizacja danych w SQL Server 2008

- Skąd wiadomo które obszary są wolne, zajęte, do których obiektów należą obszary czy strony?
- Ze stron GAM, SGAM i IAM ;-)
  - GAM (Global Allocation Map) – informacje o zajętych obszarach jednolitych (uniform)
  - SGAM (Shared GAM) - informacje o zajętych obszarach mieszanych (mixed)
  - IAM (Index Allocation Map) – informacje o przynależności obszarów do obiektów

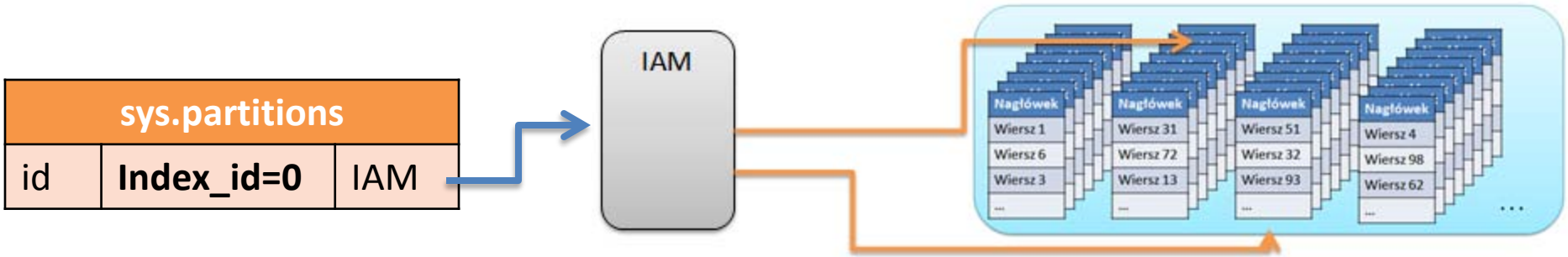


Fizyczna organizacja danych w SQL Server 2008

- No dobrze, ale jak trafić do odpowiedniej strony IAM?
- Każdy obiekt (tabela / indeks) ma wpisy w tabelach systemowych dotyczące alokacji jego danych
- Dostęp do tych informacji – widok **sys.partitions**
- Każda sterta, indeks, obszar LOB mają odpowiadający

im wpis. Wpis ten zawiera wskaźnik do IAM

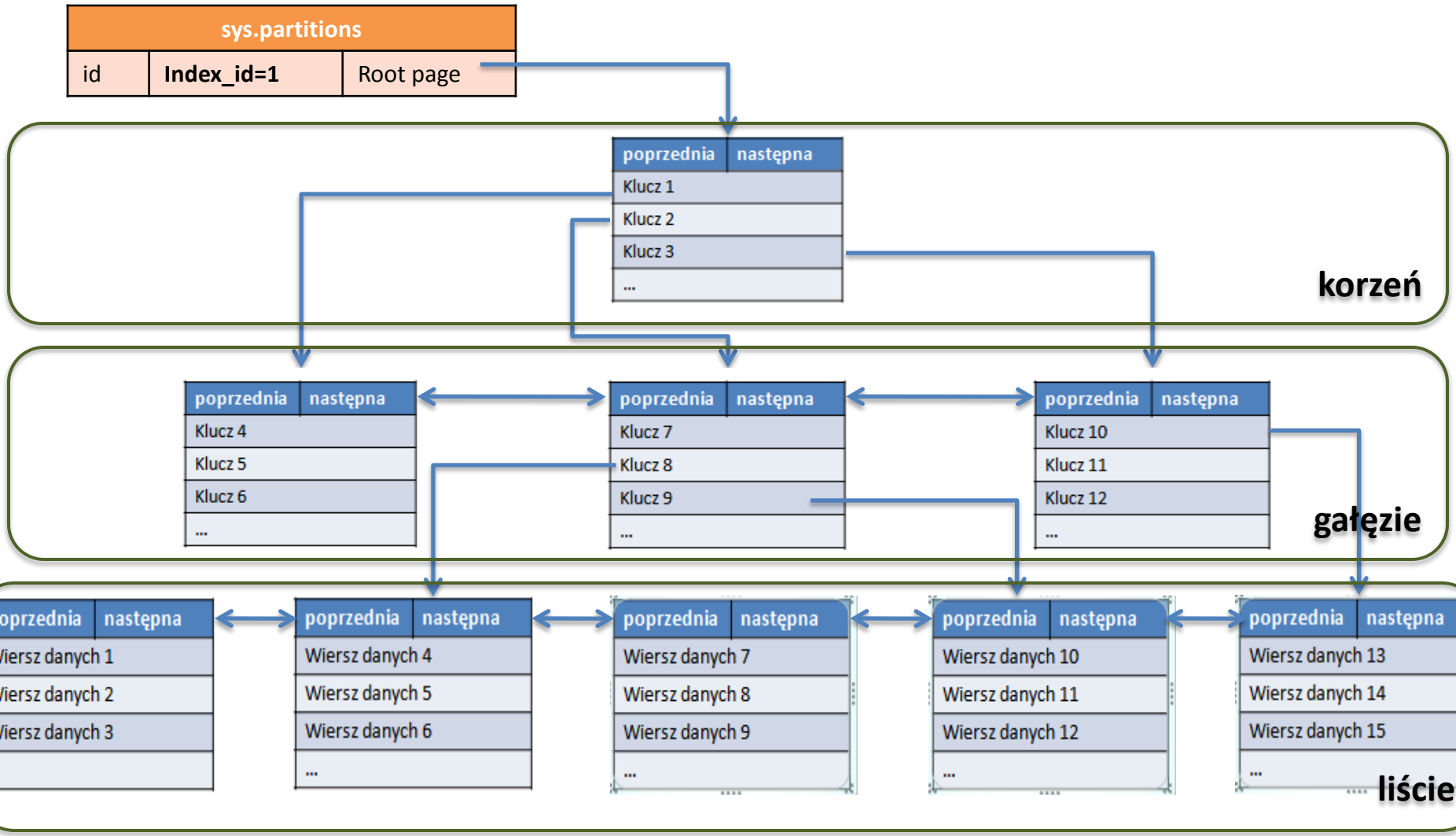
- **Wartośc kolumny index_id:**
  - 0 – sterta
  - 1 – indeks zgrupowany
  - 2..250 – indeksy niezgrupowane
  - 255 – dane LOB



Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- **Indeksy**
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Indeks zgrupowany



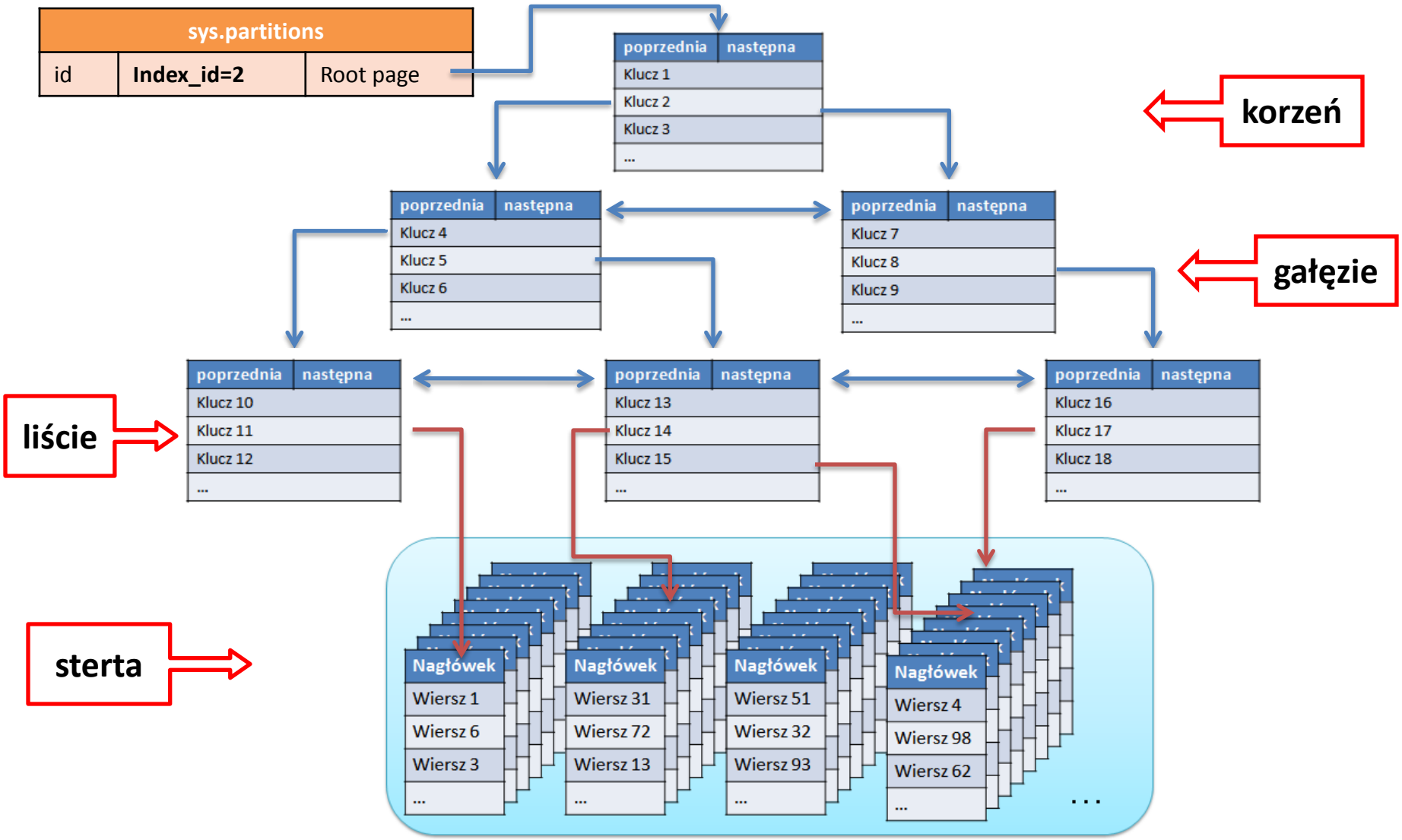
Indeks zgrupowany

- Struktura drzewiasta (B-tree) – drzewo zrównoważone
- Na poziomie korzenia i gałęzi – **strony indeksu**
- Na poziomie liści – właściwe **strony z danymi** z tabeli
- Dane fizycznie uporządkowane rosnąco wg klucza indeksu
- Tylko jeden indeks zgrupowany dla tabeli!
- Unikalność kluczy zapewniona wewnętrznie
  - Jeśli w tabeli występują dwie takie same wartości klucza, dodawana do nich jest losowa liczba i taki klucz staje się wewnętrznie rozpoznawany jako unikalny
- Kiedy stosowanie jest szczególnie uzasadnione
  - Operowanie na zakresach danych i danych grupowanych
  - Pobieranie danych w określonym porządku
  - Zapytania korzystające z wielu kolumn tabeli
  - Lepsza wydajność przy dodawaniu nowych wierszy
- Na jakich kolumnach tworzyć indeks zgrupowany?
  - Mała długość
  - Wysoka selektywność (mało powtarzających się wartości klucza indeksu)
  - Rzadko bądź wcale nie zmieniane wartości
  - Wartości klucza dla kolejno dodawanych wierszy są rosnące

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- **Indeksy**
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

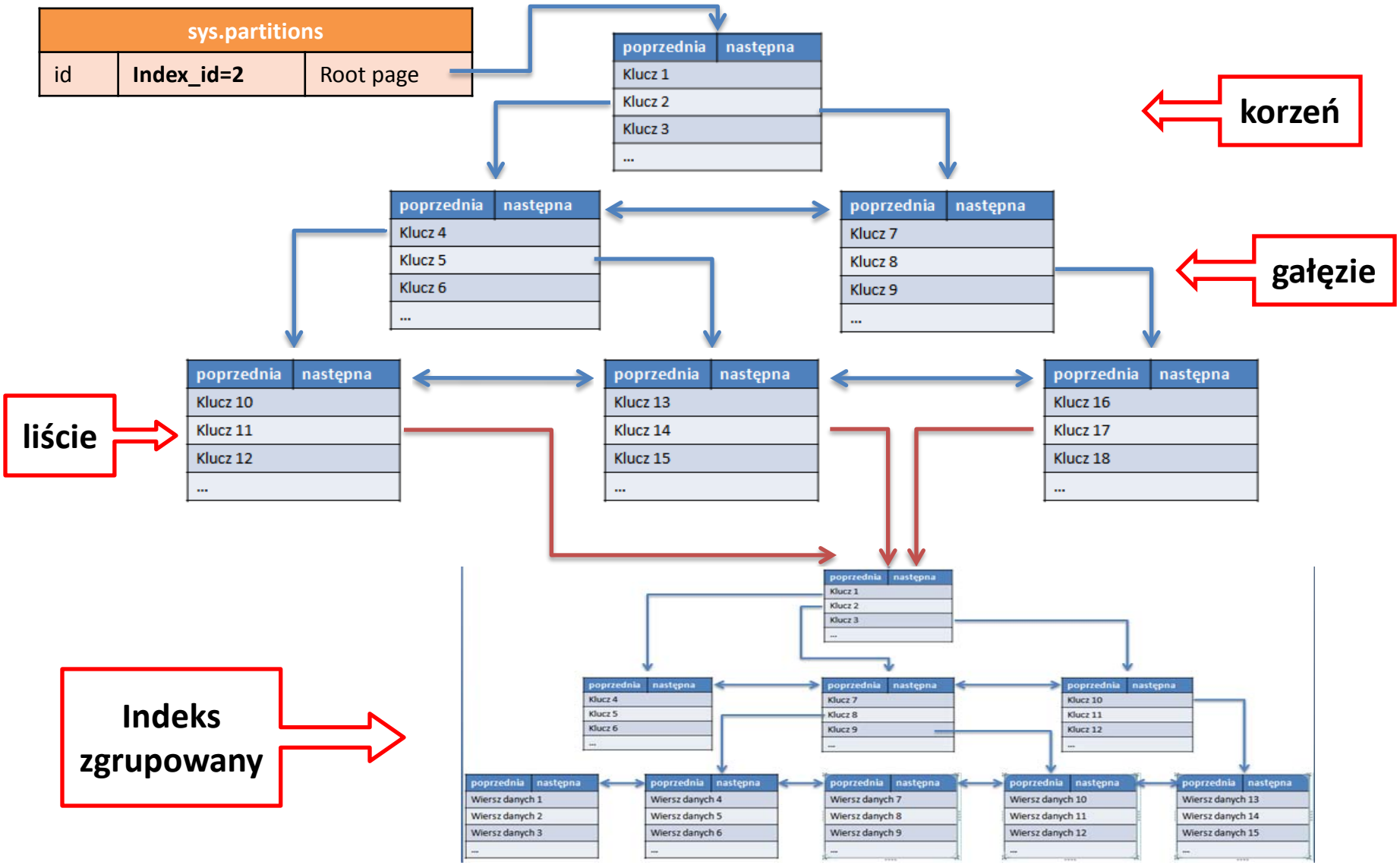
Indeks niezgrupowany (budowany na stercie)



Indeks niezgrupowany (budowany na stercie)

- Struktura drzewiasta (B-tree) – drzewo zrównoważone
- Na poziomie korzenia, gałęzi i liści – strony indeksu
- Liście zawierają wskaźniki do właściwych stron na stercie
- Można tworzyć do 248 indeksów niezgrupowanych na tabeli
- Stosowane są gdy dane wyszukiwane są według wielu kryteriów (różne zapytania)
- Maksymalna długość klucza – 900 bajtów
- Maksymalnie 16 kolumn w kluczu

Indeks niezgrupowany (budowany na zgrupowanym)



Indeks niezgrupowany (budowany na zgrupowanym)

- Praktycznie wszystko tak samo jak w budowanym na stercie.
- Z wyjątkiem dwóch rzeczy:
  - Liście zawierają wartości klucza z indeksu zgrupowanego
  - Wskaźnik zawsze ustawiony jest na korzeń indeksu zgrupowanego
- Jeśli indeks zgrupowany zostanie usunięty – niezgrupowany zostanie przebudowany (na wariant oparty o stertę)
- Jeśli indeks zgrupowany zostanie utworzony – indeksy niezgrupowane zostaną także przebudowane (ze sterty na zgrupowany)

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- **Indeksy**
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Indeksy pokrywające

- Indeksy poprawiają wydajność poprzez zmniejszenie ilości operacji wejścia/wyjścia (odczyt danych z dysku)
  - SQL Server używa mechanizmu pamięci podręcznej (cache) do przechowywania stron danych i indeksów
  - Jeżeli strona została pobrana raz, to istnieje szansa, że niedługo znów będzie potrzebna
  - W takim przypadku nie będzie pobierana z dysku tylko z pamięci cache.
- Przy korzystaniu z indeksów niezgrupowanych można dodatkowo ograniczyć sięganie do stron danych przez:
  - Umieszczanie dodatkowych kolumn w kluczu indeksu (limity!)
  - Umieszczanie kolumn „niekluczowych” w indeksie
- Jeżeli wszystkie potrzebne do realizacji zapytania dane znajdują się na stronach indeksu (liściach) to mówimy o **indeksie pokrywającym**

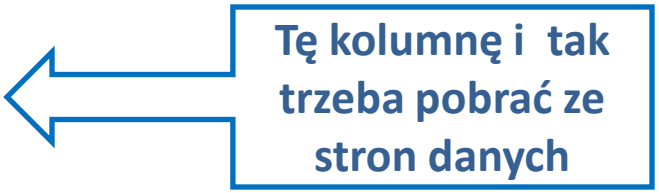
Indeksy pokrywające - przykład

- Zakładamy, że:
  - W bazie istnieje tabela Klienci z kolumnami:
    - ID, Nazwisko, Imie, Email, DataOstatniegoZamowienia
  - Istnieje indeks zgrupowany na kolumnie ID
  - Istnieje indeks niezgrupowany na kolumnie Nazwisko
- Wykonywane będzie zapytanie:

  SELECT Nazwisko, Email

  FROM klienci

  WHERE Nazwisko BETWEEN 'Iksiński' and 'Nowak'



- Pomimo istnienia indeksu niezgrupowanego i tak skanowany jest indeks zgrupowany. (clustered index scan)
- Wynika to z tego, że na wyjściu zapytania są kolumny, które nie są zawarte w żadnym indeksie
- Gdy na liście kolumn zostanie tylko nazwisko – zapytanie jest realizowane przez wyszukiwanie w indeksie niezgrupowanym (index seek)

Indeksy pokrywające - przykład c.d.

- Modyfikujemy zapytanie:

  SELECT Nazwisko

  FROM klienci

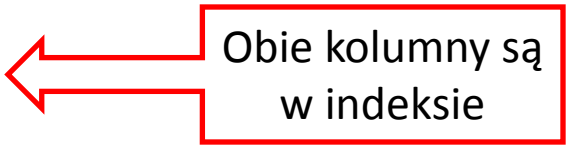
  WHERE Nazwisko BETWEEN 'Iksiński' and 'Nowak'

- Tym razem zapytanie realizowane jest przez wyszukiwanie w indeksie niezgrupowanym (index seek) znacznie niższym kosztem, bez dostępu do stron danych.
- Modyfikujemy indeks niezgrupowany (dodając kolumnę Email)
- Wykonujemy zapytanie:

  SELECT Nazwisko, Email

  FROM klienci

  WHERE Nazwisko BETWEEN 'Iksiński' and 'Nowak'



- Znów wykorzystana zostaje operacja wyszukiwania w indeksie niezgrupowanym, gdyż obie kolumny są w nim zawarte i nie trzeba pobierać ich wartości ze stron danych.
- Na tym polega istota indeksów pokrywających

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- **Plany wykonania zapytań**
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Wykonywanie zapytań

- Zapytanie zostało przekazane do wykonania
  …co dzieje się dalej?
- Całość procesu można opisać kilkoma etapami:
  - Parsowanie zapytania (błędy składniowe). Efektem jest drzewo zapytania.
  - Standaryzacja zapytania (drzewa). Usuwanie nadmiarowości, standaryzowanie podzapytań itp..
  - **Optymalizacja zapytania .**Wieloetapowy proces prowadzący do wyboru sposobu realizacji zapytania
  - Kompilacja wygenerowanego planu (zapisanie w cache)
  - Określenie metod fizycznego dostępu do danych
  - Wykonanie zapytania zgodnie ze stworzonym planem

Wykonywanie zapytań – optymalizacja zapytania

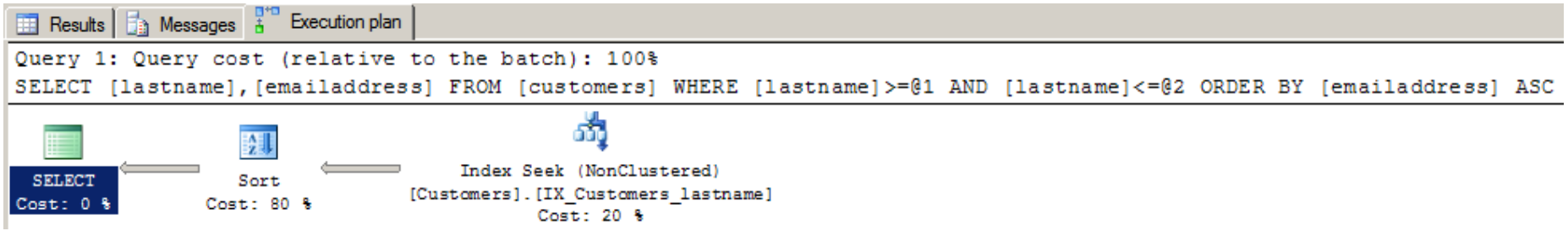
- Optymalizacja zapytania polega na:
  - Dokonaniu analizy zapytania (pod kątem kryteriów wyszukiwania oraz złączeń)
  - Dobraniu indeksów, które mogą okazać się pomocne przy realizacji zapytania (kryteria wyszukiwania, kolumny wyjściowe)
  - Określeniu strategii realizacji złączeń (selektywność, potrzebna pamięć)
- Generowanych jest kilka wariantów, dla każdego szacowany jest koszt wyrażony w operacjach wejścia/wyjścia (I/O) i czasie rocesora (CPU).
- Wybierany jest najtańszy wariant i przekazywany do kompilacji
- Plan wykonania można podejrzeć za pomocą włączenia jednej z opcji:
  - SET SHOWPLAN_TEXT ON, SET SHOWPLAN_XML ON , SET SHOWPLAN_ALL ON

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- **Plany wykonania zapytań**
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Graficzna reprezentacja planów wykonania zapytań

- SQL Server Management Studio pozwala obrazować plany wykonania w formie graficznej



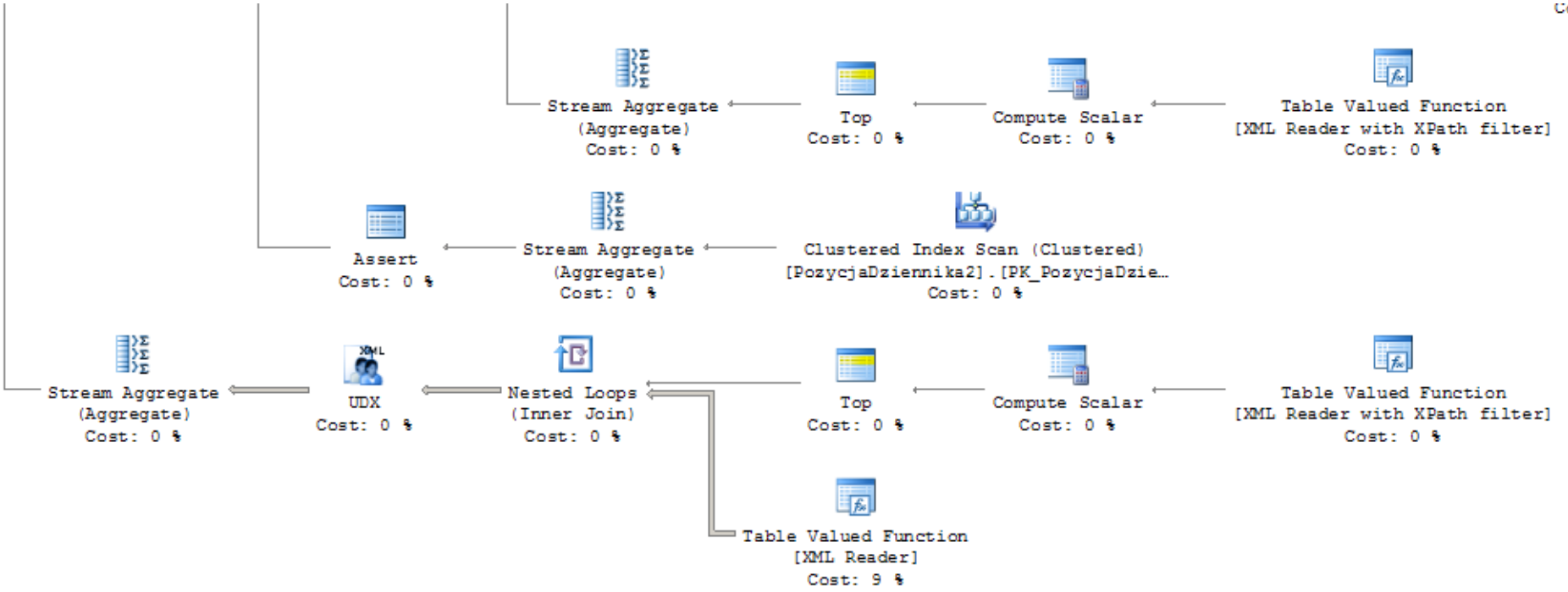
- Dwa tryby: estymowany i faktyczny plan wykonania:
  - Estymowany (estimated)
    - Nie trzeba wykonywać zapytania
    - Nie zawiera dokładnych informacji (jak sama nazwa wskazuje)
  - Faktyczny (actual)
    - Zapytanie musi zostać wykonane
    - Zawiera pełnie i dokladne informacje

Gdy zapytanie wykonuje się 3 godziny a nie można czekac…

Używać gdy tylko się da. Wiarygodny!

Graficzna reprezentacja planów wykonania zapytań

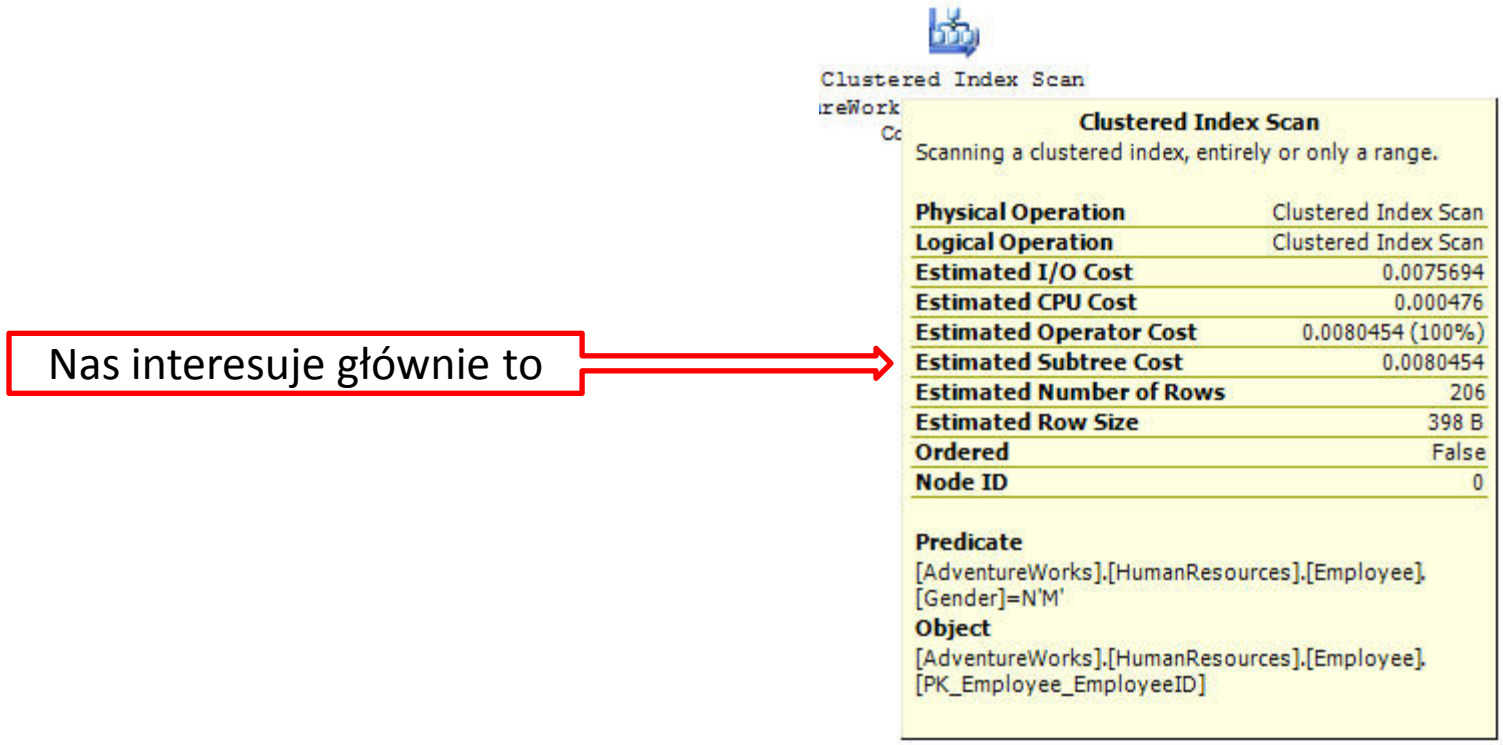
- Graficzna postać planów wykonania może zawierać kilkadziesiąt różnych symboli operatorów (logicznych i fizycznych)



- Narzędzie umożliwia wygodne poruszanie się po złożonych planach wykonania

Graficzna reprezentacja planów wykonania zapytań

- Najechanie kursorem na dowolny symbol powoduje wyświetlenie szczegółowych informacji na jego temat



- Początkowo sprawia to wrażenie czarnej magii... z czasem można się przyzwyczaić i zacząć rozumieć :-)

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- **Plany wykonania zapytań**
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Statystyki indeksów

- Co z tego, że mamy indeks na kolumnie Nazwisko, skoro 90% osób to Kowalscy?
- Do podjęcia decyzji o skorzystaniu z indeksu niezbędne są także dodatkowe informacje dotyczące zróżnicowania wartości kluczy (kolumn).
- Zwykle statystyki tworzone są i utrzymywane automatycznie
- Istnieje także wariant dla twardzieli – ręczne tworzenie i utrzymywanie statystyk
- Statystyka to w uproszczeniu histogram prezentujący zakresy wartości kolumn wraz z ilością konkretnych wartości mieszczących się w tych zakresach
- W ramach optymalizacji zapytania serwer może samoczynnie generować statystyki a nawet tworzyć indeksy na potrzeby realizacji jednego zapytania
- Nieaktualne statystyki mogą prowadzić do powstawania nieefektywnych planów wykonania (ważne przy „drastycznych” zmianach rozkładu danych w bazie – np.: usunięcie 40% rekordów)

Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- **Plany wykonania zapytań**
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

Wykorzystanie indeksów

- Zakładamy, że zapytania będą tworzone w oparciu o tabelę:

Klienci

	Column Name	Data Type	Allow Nulls
	ID	int	☐
	Imie	varchar(100)	☐
	Nazwisko	varchar(100)	☐
	Email	varchar(50)	☑
	Telefon	varchar(50)	☑
	smiec	char(1000)	☐
			☐

W celu zwiększenia rozmiaru wiersza i liczby stron:)

- Nie ma żadnych indeksów na tabeli Klienci
- Zapytanie, którym się zajmiemy jest proste:

```
SELECT
   Imie
  ,Nazwisko
FROM
   dbo.Klienci
WHERE
   nazwisko BETWEEN 'F' AND 'T'
```

Wykorzystanie indeksów

- Pierwsze wykonanie zapytania – plan wykonania

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Imie],[Nazwisko] FROM [dbo].[Klienci] WHERE [nazwisko]>=@1 AND [nazwisko]<=@2
```

Brak indeksów – skanowanie sterty

```
(12673 row(s) affected)
Table 'Klienci'. Scan count 1, logical reads 2854, physical reads 42, read-ahead reads 2854,
```

Pierwsze wykonanie: strony pobierane z dysku

Kolejne wykonania: strony znajdują się w cache

```
(12673 row(s) affected)
Table 'Klienci'. Scan count 1, logical reads 2854, physical reads 0, read-ahead reads 0,
```

- Koszt zapytania (estimated subtree cost) : **2,1385**

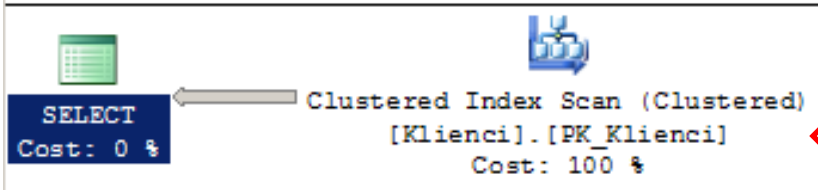
Wykorzystanie indeksów

- Stwórzmy najpierw indeks zgrupowany na kolumnie ID.
- Zrealizujemy to przez utworzenie klucza podstawowego na tej kolumnie (prowadzi to do utworzenia indeksu)

```
ALTER TABLE dbo.Klienci
ADD CONSTRAINT PK_Klienci PRIMARY KEY CLUSTERED (ID)
```

- Wykonanie naszego zapytania po utworzeniu indeksu przebiega według planu:

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Imie],[Nazwisko] FROM [dbo].[Klienci] WHERE [nazwisko]>=@1 AND [nazwisko]<=@2
```



Stworzyliśmy indeks zgrupowany, więc nie ma już sterty.

```
(12673 row(s) affected)
Table 'Klienci'. Scan count 1, logical reads 2862, physical reads 0, read-ahead reads 0,
```

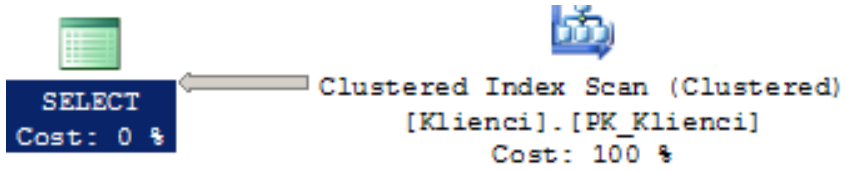
- Koszt zapytania pozostał bez zmian : **2,1385**

Wykorzystanie indeksów

- Spróbujmy teraz popracować nad wydajnością
- Stwórzmy indeks niezgrupowany na kolumnie, której używamy jako kryterium wyszukiwania

```
CREATE NONCLUSTERED INDEX IX_klienci_nazwisko ON klienci(Nazwisko)
```

- Skoro istnieje indeks na kolumnie Nazwisko, to powinien zostać użyty do wyszukiwania? Sprawdźmy…



Nic z tego! Nasz indeks nie został wykorzystany

- Dlaczego?
- Bo na wyjściu zapytania mamy jeszcze kolumnę Imie!
- Optymalizator stwierdził, iż nie warto korzystać z indeksu niezgrupowanego, skoro i tak trzeba pobrać strony danych, żeby uzyskać wartości z tej kolumny
- Koszt zapytania ciągle bez zmian : **2,1385**

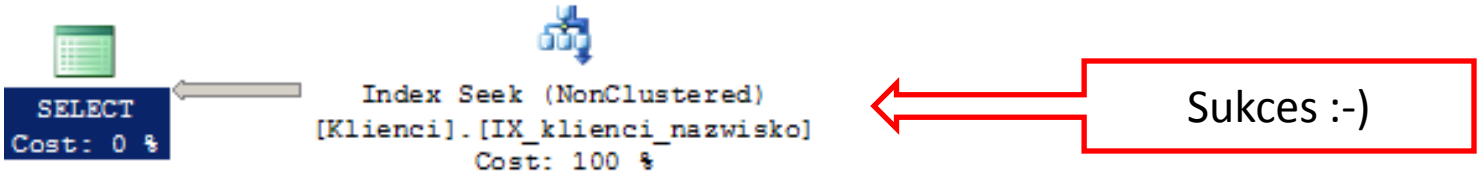
Wykorzystanie indeksów

- Zróbmy w końcu coś co przyniesie efekt!
- Wiemy dlaczego nasz indeks był nieprzydatny
- Uczyńmy go przydatnym! Dodajmy kolumnę Imie do indeksu

```
DROP INDEX Klienci.IX_klienci_nazwisko
CREATE NONCLUSTERED INDEX IX_klienci_nazwisko ON klienci(Nazwisko) INCLUDE (Imie)
```

- Wykonajmy kolejny raz nasze zapytanie

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Imie],[Nazwisko] FROM [dbo].[Klienci] WHERE [nazwisko]>=@1 AND [nazwisko]<=@2
```




```
(12673 row(s) affected)
Table 'Klienci'. Scan count 1, logical reads 41, physical reads 0, read-ahead reads 0,
```

Wcześniej było 2862 !

- Koszt wykonania: 0,0453
- Wcześniej było: 2,1385

Agenda

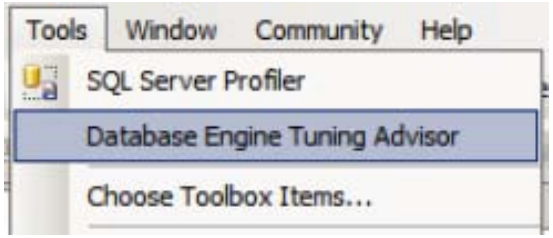
- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- **Plany wykonania zapytań**
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- Podsumowanie

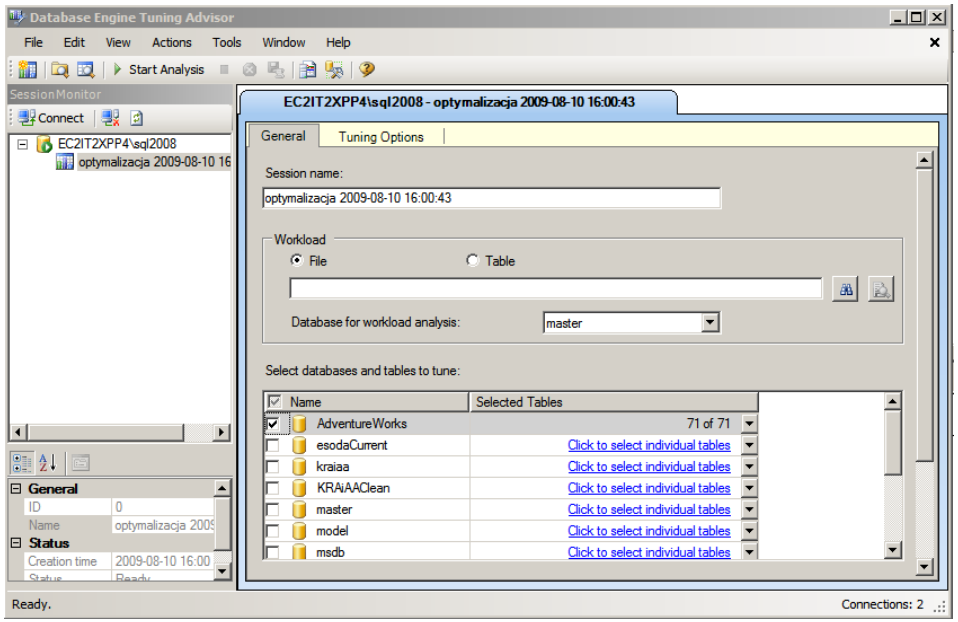
…ale masz jakieś wsparcie? – narzędzia pomocne przy optymalizacji zapytań

- Próbowaliśmy uzyskać poprawę wydajności dla jednego prostego zapytania
- Optymalizacja zapytań i planowanie indeksów w prawdziwych aplikacjach jest bardzo złożonym zagadnieniem
  - Wiele zapytań, różne kryteria, kolumny na wyjściu itp.
  - Dane są także modyfikowane – wpływ indeksów na wydajność
  - Poprawa wydajności przy jednym zapytaniu potrafi psuć ją przy innym
- Jak więc radzić sobie z tak złożonymi problemami?
  - Wsparcie ze strony narzędzi
  - Właściwe podejście do problemu!

Narzędzie Database Engine Tuning Advisor

- Sama struktura bazy danych to za mało, żeby zaplanować indeksy
- Trzeba jeszcze znać sposób korzystania z bazy – zapytania
- Jeśli zbierzemy informacje o zapytaniach wysyłanych do bazy, to z ich wykorzystaniem można użyć narzędzia Database Engine Tuning Advisor
- Jest ono w stanie na podstawie tych danych zasugerować szereg czynności prowadzących do wzrostu wydajności
- Potrafi zaplanować indeksy
- Umie skorzystać też z widoków indeksowanych
- Wynik pracy narzędzia może być podstawą do dalszych prac





Agenda

- Po co optymalizować?
- Fizyczna organizacja danych w SQL Server 2008
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań
  - Strategie wykonania zapytań
  - Graficzna reprezentacja planów wykonania zapytań
  - Statystyki indeksów
  - Wykorzystanie indeksów
  - …ale masz jakieś wsparcie?
- **Podsumowanie**

Podsumowanie

- Najlepsza nawet struktura bazy danych nie gwarantuje wysokiej wydajności zapytań
- Połączenie odpowiedniego projektu bazy z dobraniem formy zapytań oraz adekwatnym zaplanowaniem indeksów pozwala na zapewnienie pożądanej wydajności
- Żeby móc zająć się optymalizacją zapytań trzeba zrozumieć w jaki sposób SQL Server przechowuje dane i jakie są tego konsekwencje.
- Trzeba również zapoznać się ze sposobami realizacji zapytań, dostępnymi operacjami i specyfiką ich działania
- Można wspierać się narzędziami (SQL Server management Studio, Database engine Tuning Advisor), lecz nie rozwiążą one każdego problemu.
- Najlepszą drogą do nabycia umiejętności w zakresie optymalizacji jest praktyka!
- Gdy już nabierze się doświadczenia – zyskuje się rangę guru lub magika :-)

KONIEC

… czy są jakieś pytania?