Politechnika Wrocławska

Internetowe Bazy Danych

dr inż. Roman Ptak
Katedra Informatyki Technicznej
roman.ptak@pwr.edu.pl

Plan wykładu 5.

- Optymalizacja baz danych
- Struktura fizyczna systemu MS SQL Server
- Plan wykonania w MS SQL Server
- Optymalizacja zapytań
- Bazy NoSQL

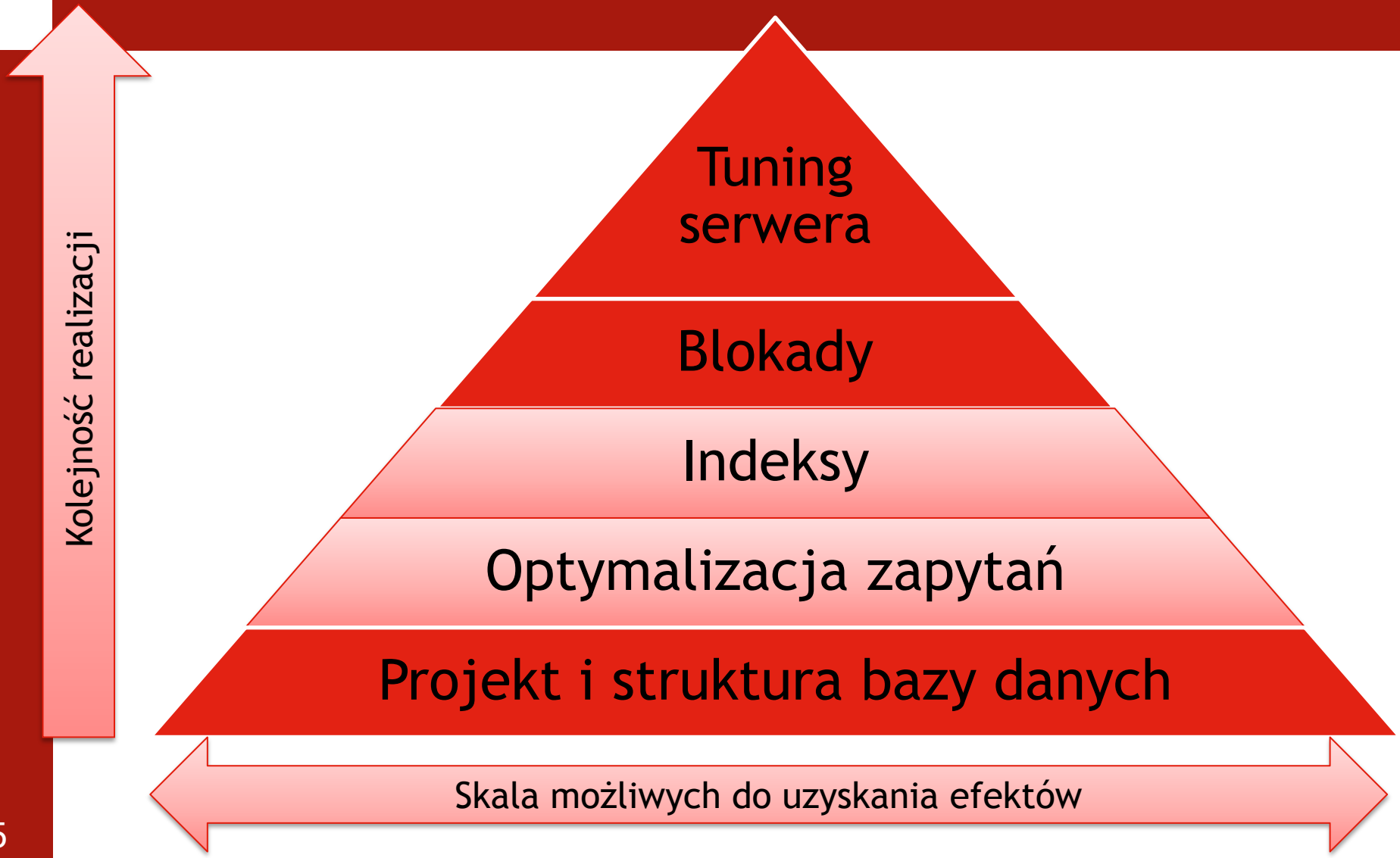
Na przykładzie MS SQL Server

OPTYMALIZACJA BAZ DANYCH

Optymalizacja IBD - wprowadzenie

- Problem C10k
- Optymalizacja serwerów: WWW, BD, …
- Zwiększenie przepustowości łącza internetowego
- Optymalizacji BD: normalizacja, denormalizacja, zastosowanie indeksów, procedury składowane i inne

Piramida optymalizacyjna



Model optymalizacji wydajności

- Projekt struktury bazy danych
  - Najważniejsze zadanie w procesie optymalizacji
  - Skutek złego projektu trudno poprawić
  - Warstwa abstrakcji danych
- **Optymalizacja zapytań**
- **Indeksy**
- Blokady
  - Baza danych działa prawidłowo z jednym użytkownikiem a przy kilku zaczyna „spowalniać”
- Tuning serwera
  - Dodawanie procesorów, zwiększanie pamięci RAM, zmiana dysków twardych itd.

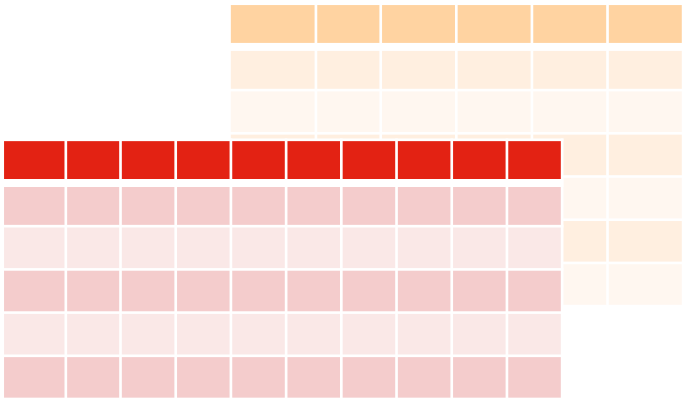
Kwestie związane z optymalizacją

- Po co optymalizować?
- Architektura fizyczna SQL Servera
- Indeksy
  - Zgrupowane
  - Niezgrupowane
  - Pokrywające
- Plany wykonania zapytań

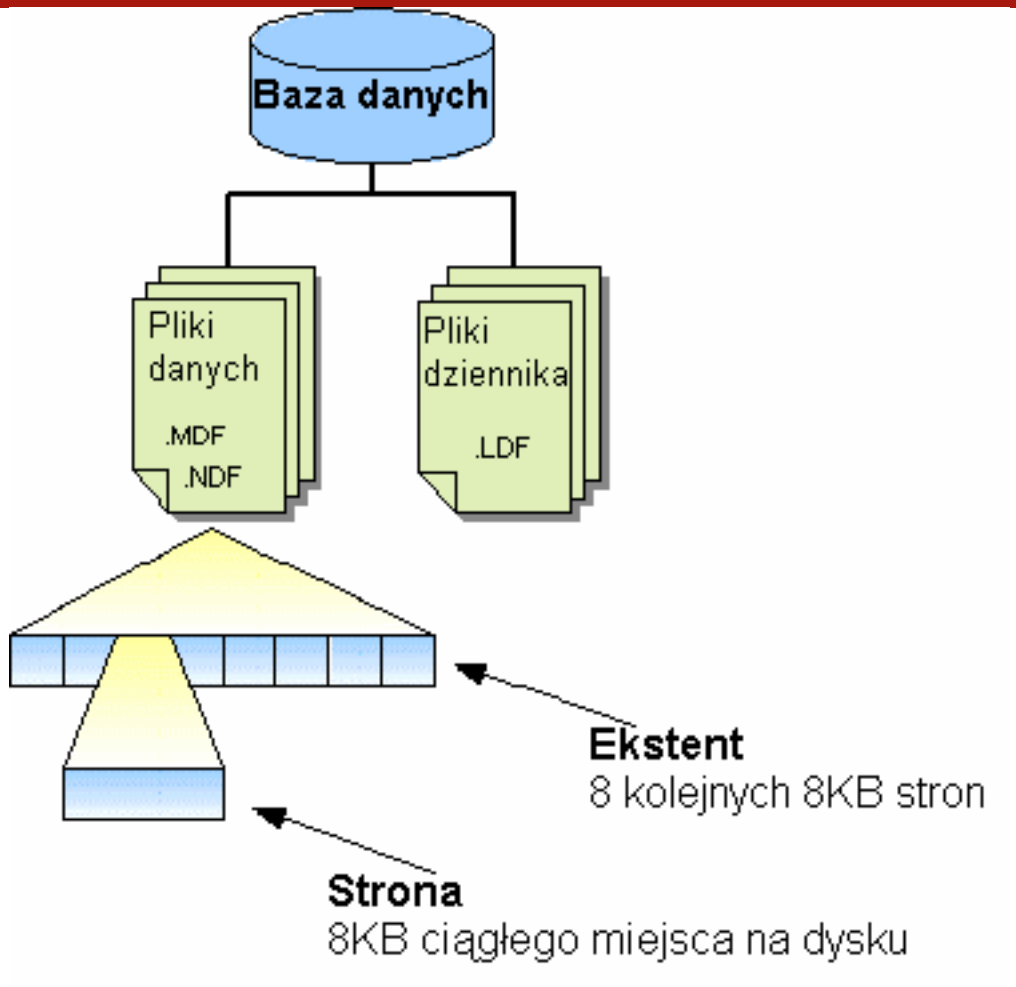
STRUKTURA FIZYCZNA SYSTEMU MS SQL SERVER

Logiczna architektura bazy danych

- Tabele
- Widoki
- Procedury składowane
- Wyzwalacze
- …



Podział bazy danych MS SQL Server na pliki, ekstenty i strony

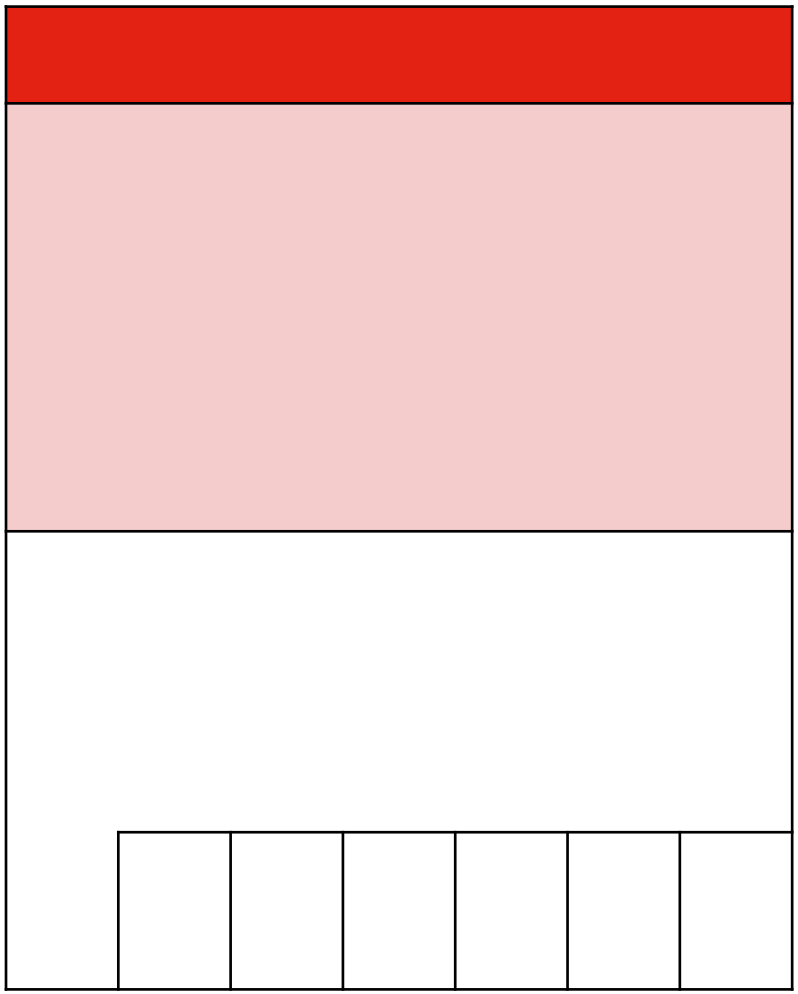


źródło: http://edu.pjwstk.edu.pl/wyklady/szb/scb/wyklad15/w15.htm

Rodzaje stron

- **data** – wszystkie dane z wyjątkiem atrybutów typów LOB (ang. *Large OBject*)
- **index** – wpisy indeksów
- **text/image** – typy LOB: text, ntext, image, varchar(max), nvarchar(max), varbinary(max), xml
- **GAM** (*Global Allocation Map*), **SGAM** (*Shered* GAM), **IAM** (*Index Allocation Map*) – bitowe mapy alokacji

Struktura strony danych



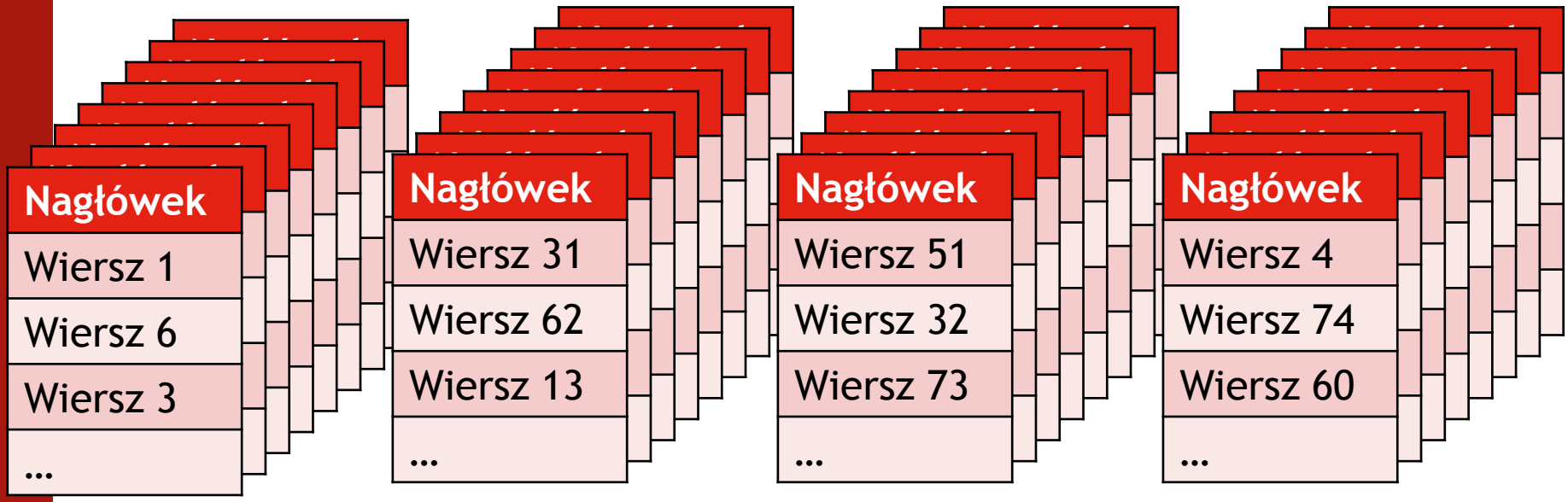
Nagłówek strony
96 bajtów

Wiersze danych

Tablica przesunięć
wierszy

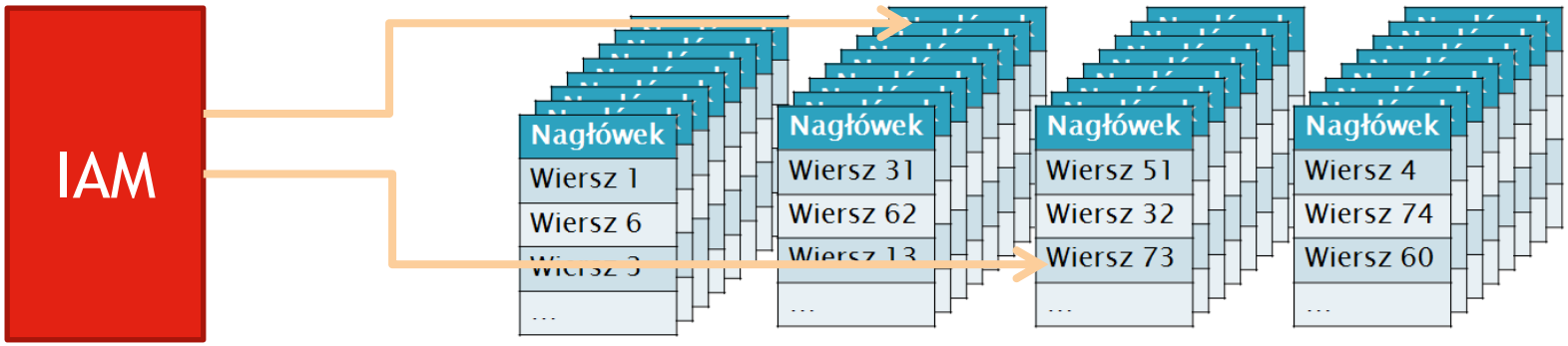
Sterta (ang. *heap*)

- Zbiór obszarów zawierających dane z jednej tabeli (lub partycji)
- Dane nie są ze sobą powiązane
- Wyszukiwanie wymaga przejrzenia wszystkich stron



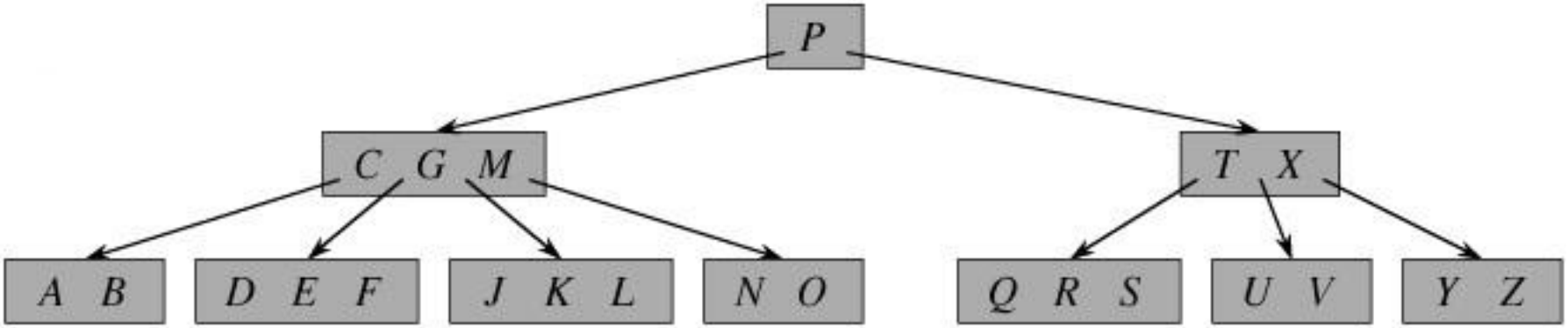
Fizyczna organizacja danych w SQL Server

- Strony GAM, SGAM informują, które obszary są wolne, zajęte
  - **GAM** – informacje o zajętych obszarach jednolitych (*uniform*)
  - **SGAM** – informacje o zajętych obszarach mieszanych (*mixed*)
- Strony **IAM** – informacje o przynależności obszarów do obiektów



B-drzewo

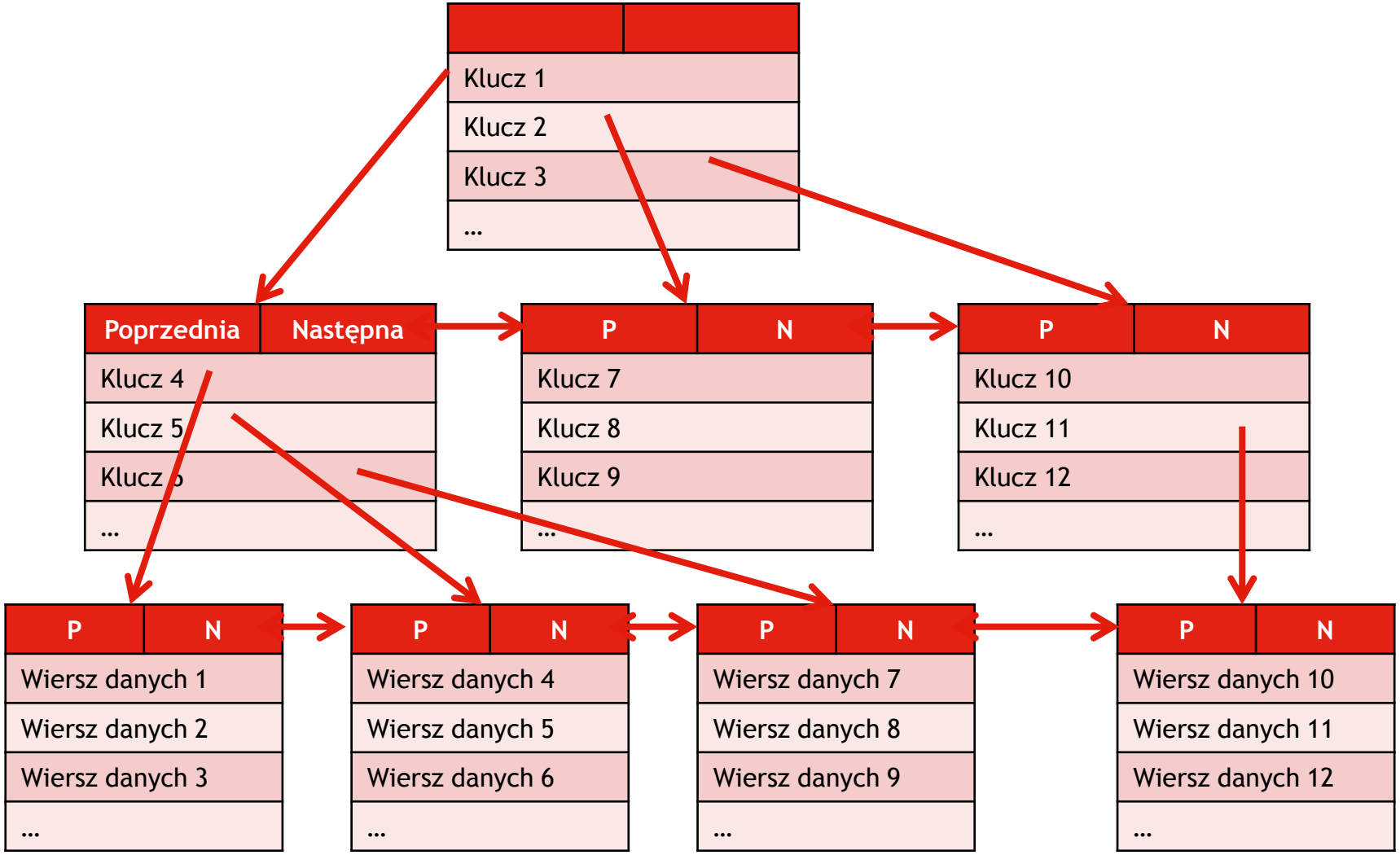
- Drzewo zbalansowane
- Węzły mogą mieć strukturę listy dwukierunkowej
- W praktyce do 5 poziomów



Indeks zgrupowany (z danymi)

- ang. *clustered index*
- Struktura drzewa (B-tree)
- Na poziomie korzeni i gałęzi – **strony indeksu**
- Na poziomie liści – **strony z danymi** z tabel
- Fizycznie porządkuje dane
- Dane fizyczne uporządkowane rosnące wg klucza indeksu
- Może istnieć tylko 1 indeks zgrupowany
- **Zakładany najczęściej na sztucznym kluczy podstawowym tzw. id**

Indeks zgrupowany



Na jakich kolumnach tworzyć indeks zgrupowany?

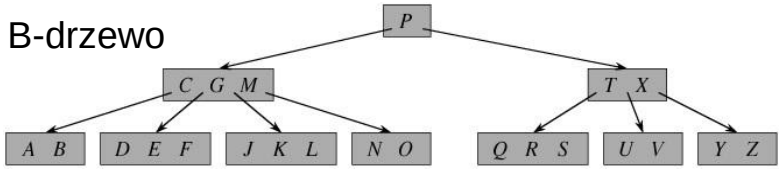
- Mała długość atrybutu (klucza)
- Wysoka selektywność (mało powtarzających się wartości klucza indeksu)
- Rzadko bądź wcale nie zmieniane wartości
- Wartości klucza dla kolejno dodawanych wierszy są rosnące

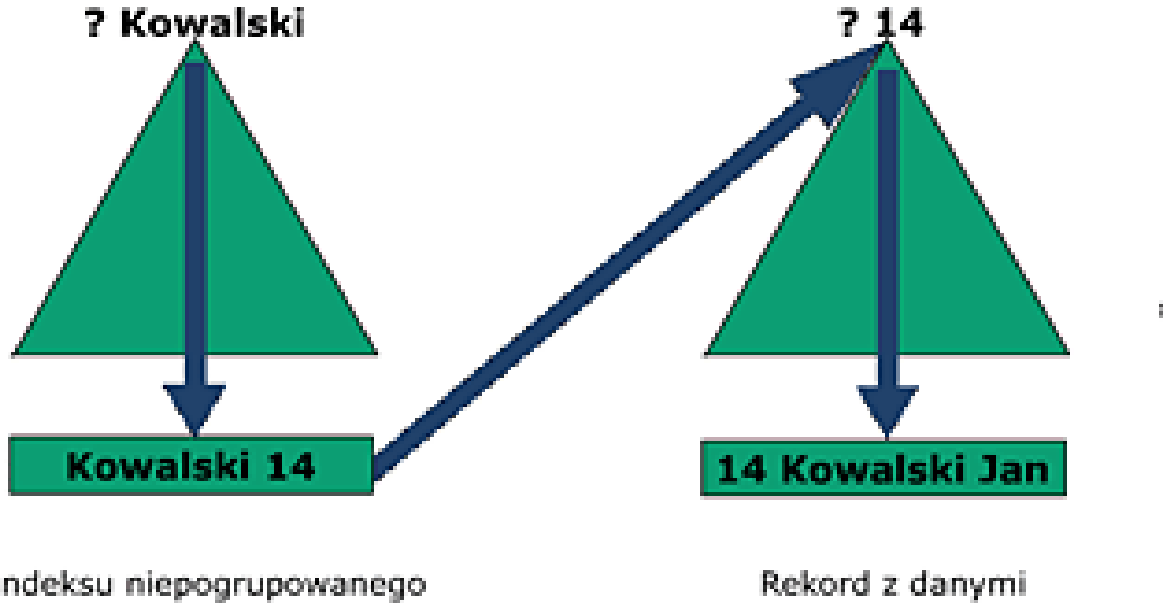
Indeks niezgrupowany

- Struktura drzewa (B-tree)
- Na **wszystkich** poziomach drzewa mamy **strony indeksu**
- Może być budowany na stercie lub na indeksie zgrupowanym
- Można stworzyć do 248 indeksów niezgrupowanych na tabeli
- Stosowane są, gdy dane wyszukiwane są według wielu kryteriów
- Maksymalnie 16 kolumn w kluczu

Indeksy niezgrupowany budowany na indeksie zgrupowanym

- Indeks niezgrupowany
- Indeks zgrupowany



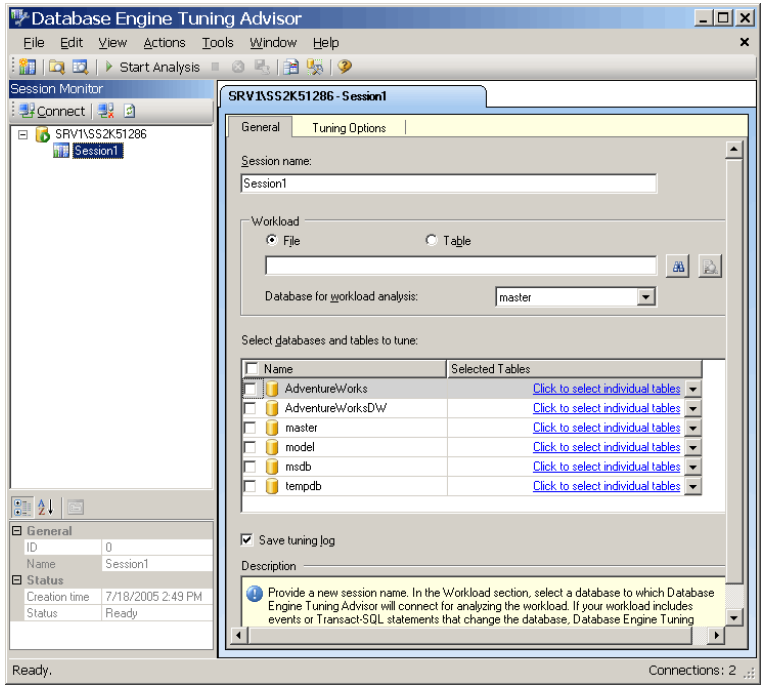


źródło: http://edu.pjwstk.edu.pl/wyklady/szb/scb/wyklad15/w15.htm

PLAN WYKONANIA W MS SQL SERVER

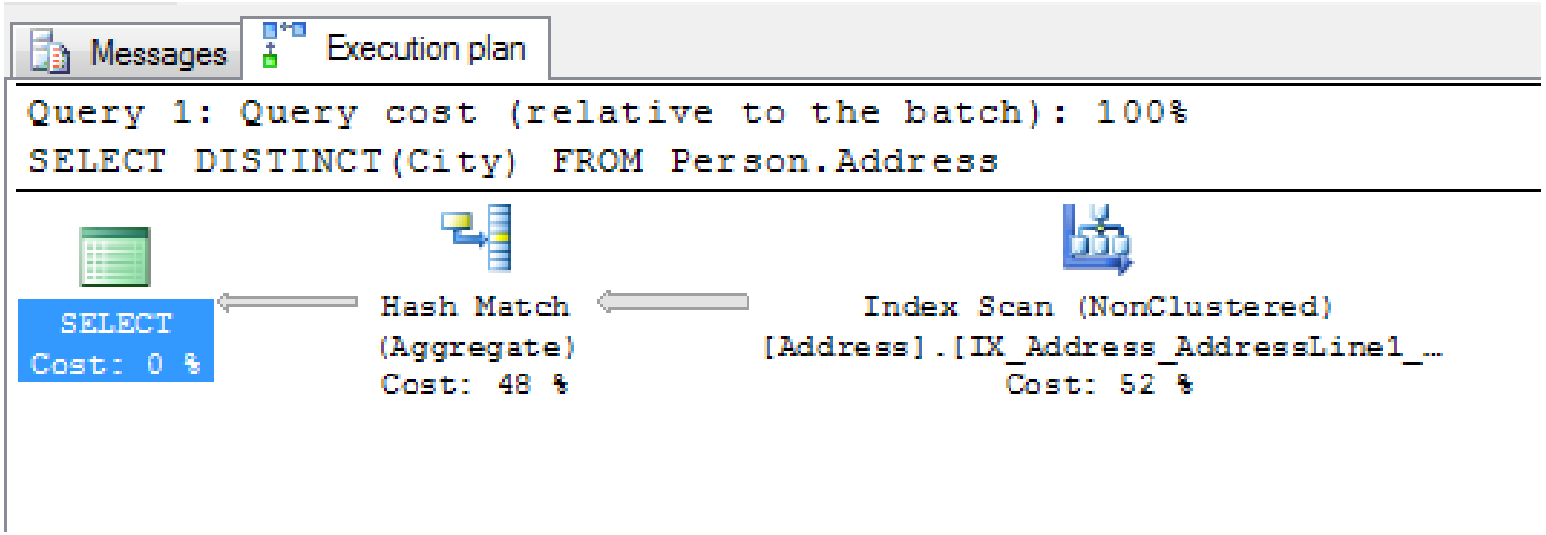
Narzędzia wspierające optymalizację

- MS SQL Server Management Studio
  - Plan wykonania
- Database Engine Tuning Advisor



Kwestie wydajnościowe

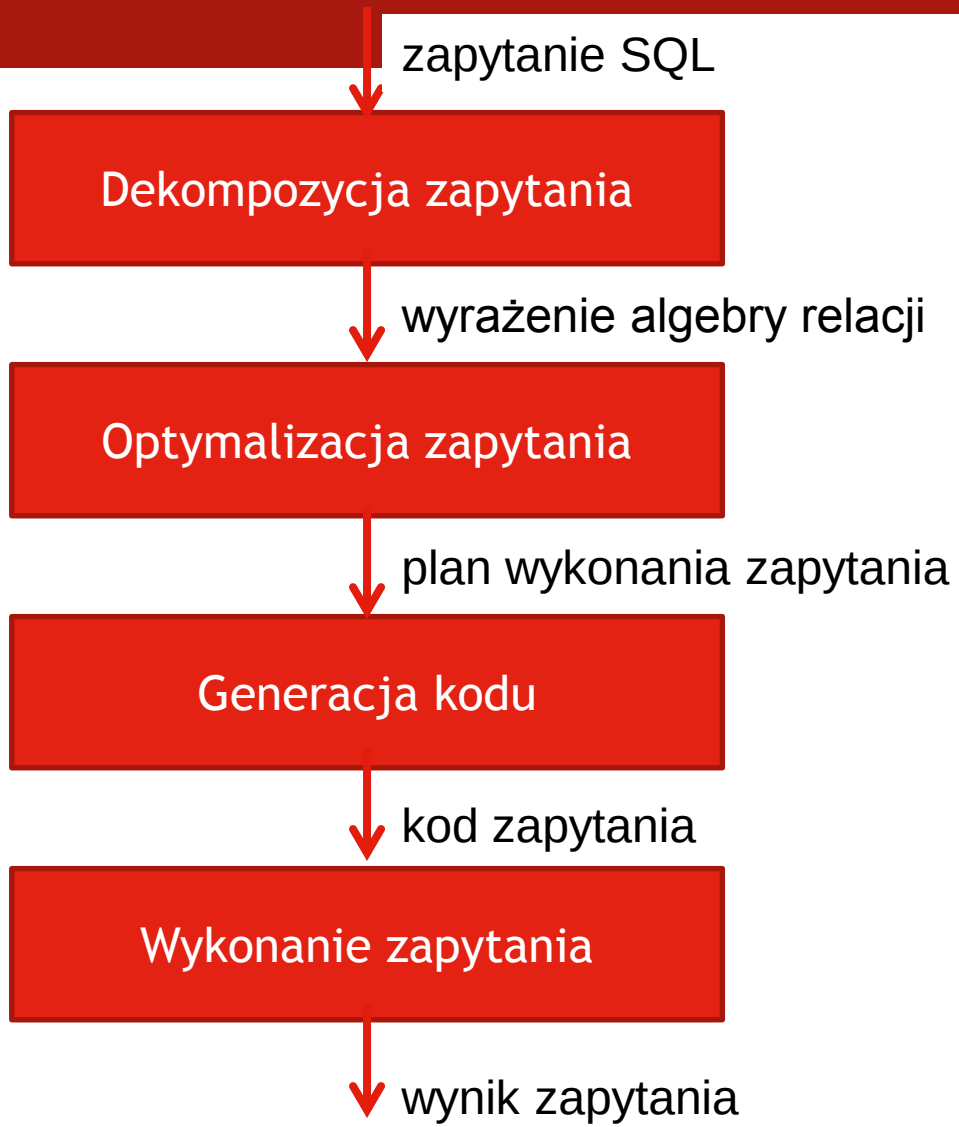
- Fizyczna organizacja danych wpływa na wydajność pracy BD
- Plan wykonania (ang. *query plan*)



Rola optymalizatora zapytań

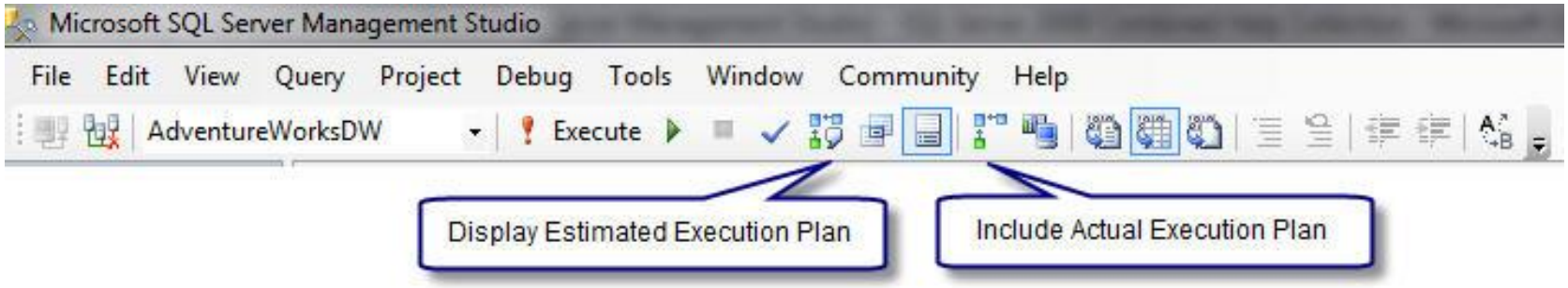
- T-SQL jest językiem deklaratywnym.
- Używając go, stwierdzamy **co** chcemy zrobić, a nie **jak** to zrobić.
- System zarządzania bazą danych posługując się **optymalizatorem zapytań** decyduje o sposobie wykonania zapytania.

Fazy przetwarzania zapytania



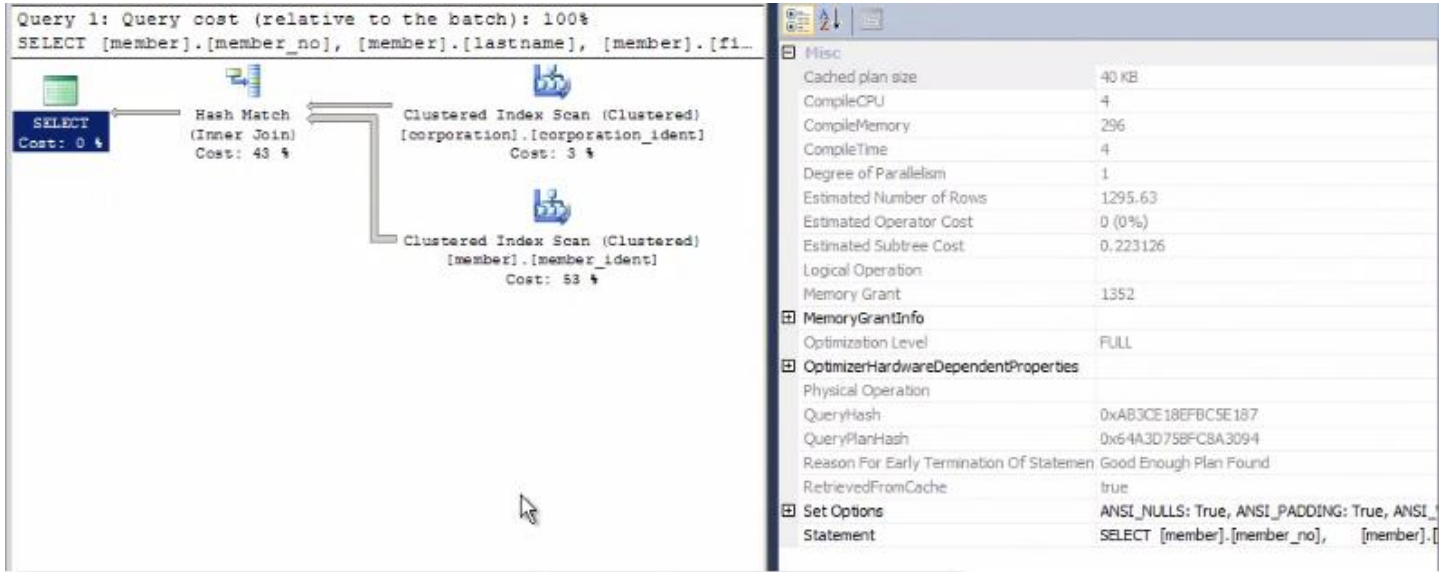
Plan wykonania MS SQL Server

- Estymowany plan wykonania (ang. *estimated execution plan*)
- Rzeczywisty plan wykonania (ang. *actual execution plan*)



Graficzny plan wykonania w SQL Server Management Studio

- Dostarcza możliwość reprezentacji graficznej planów zapytań w postaci drzewa operatorów
- Podpowiada kroki warte podjęcia celem optymalizacji zapytań (np. wykrywa brakujące indeksy)



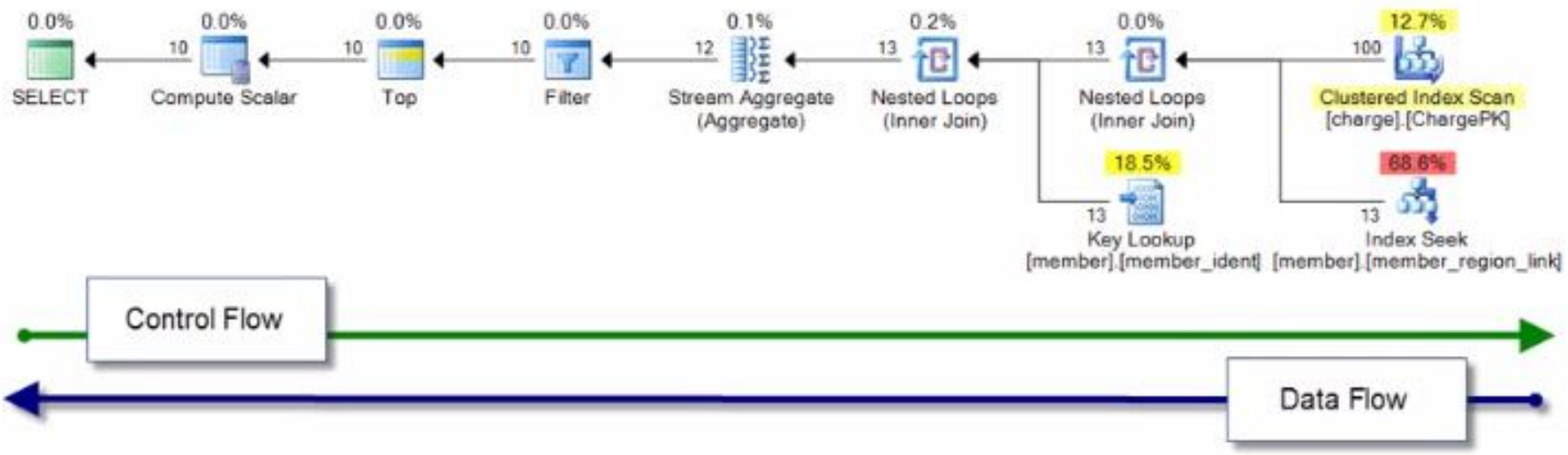
Ikony planu wykonania (wybór)

Ikona	Element języka/Operator	(ang.)
	Wynik	Result
	Skanowanie indeksu zgrupowanego	Clustered Index Scan
	**Szukanie indeksu zgrupowanego**	Clustered Index Seek
	Skanowanie indeksu zwykłego	Nonclustered Index Scan
	**Szukanie indeksu zwykłego**	Nonclustered Index Seek
	Pętla zagnieżdżona	Nested Loops
	Wyszukiwanie zakładki	Bookmark Lookup
	Sortowanie	Sort
	Dopasowanie wartości mieszającej	Hash Match

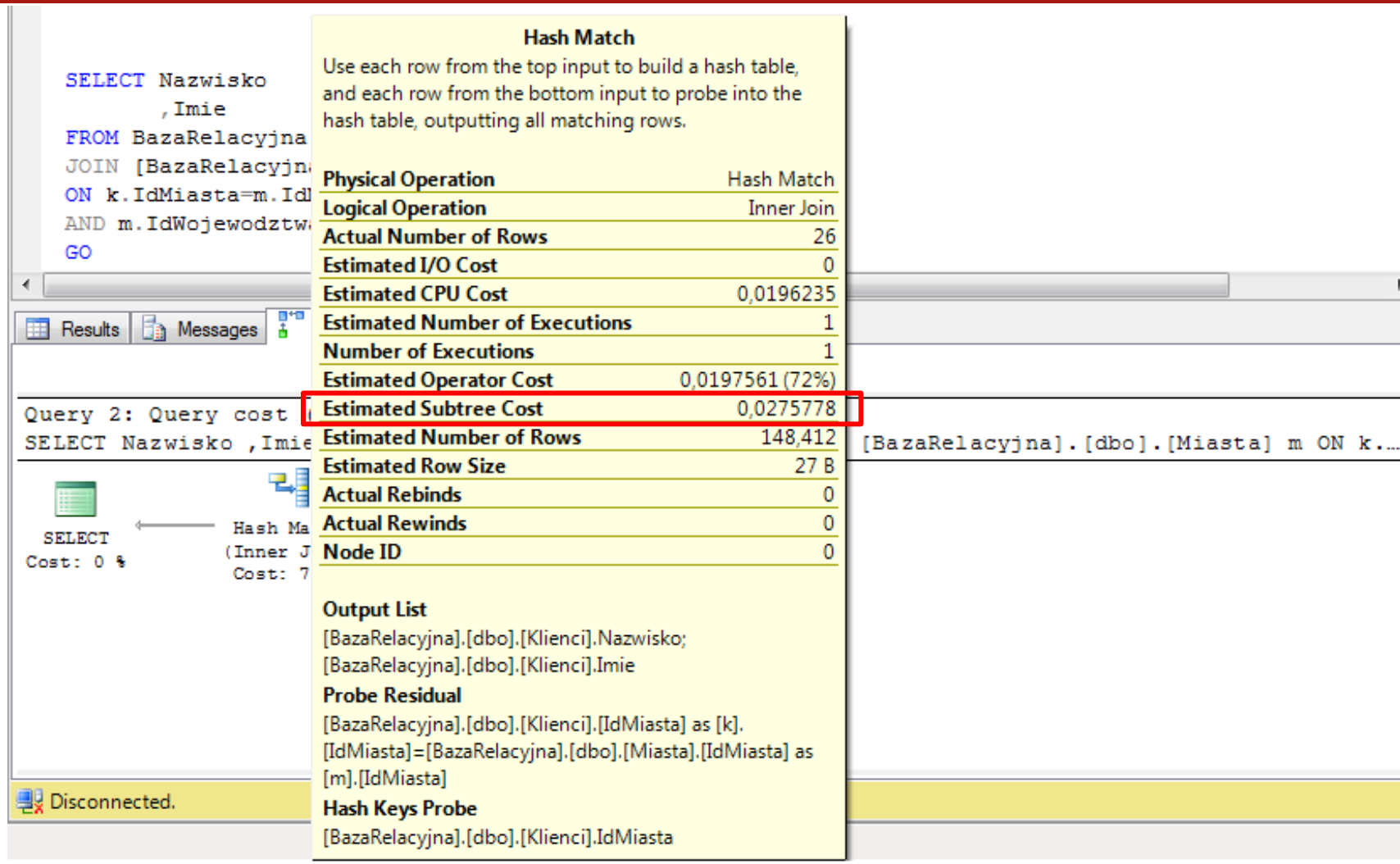
Ikony planu wykonania (cd.)

Ikona	Element języka/Operator	(ang.)
	Obliczanie wartość skalarna	Compute Scalar
	Top	Top
	Filtruj	Filtr
	Obliczanie wartość skalarnej	Compute Scalar
	Agregat strumienia	Stream Aggregate
	(...)	

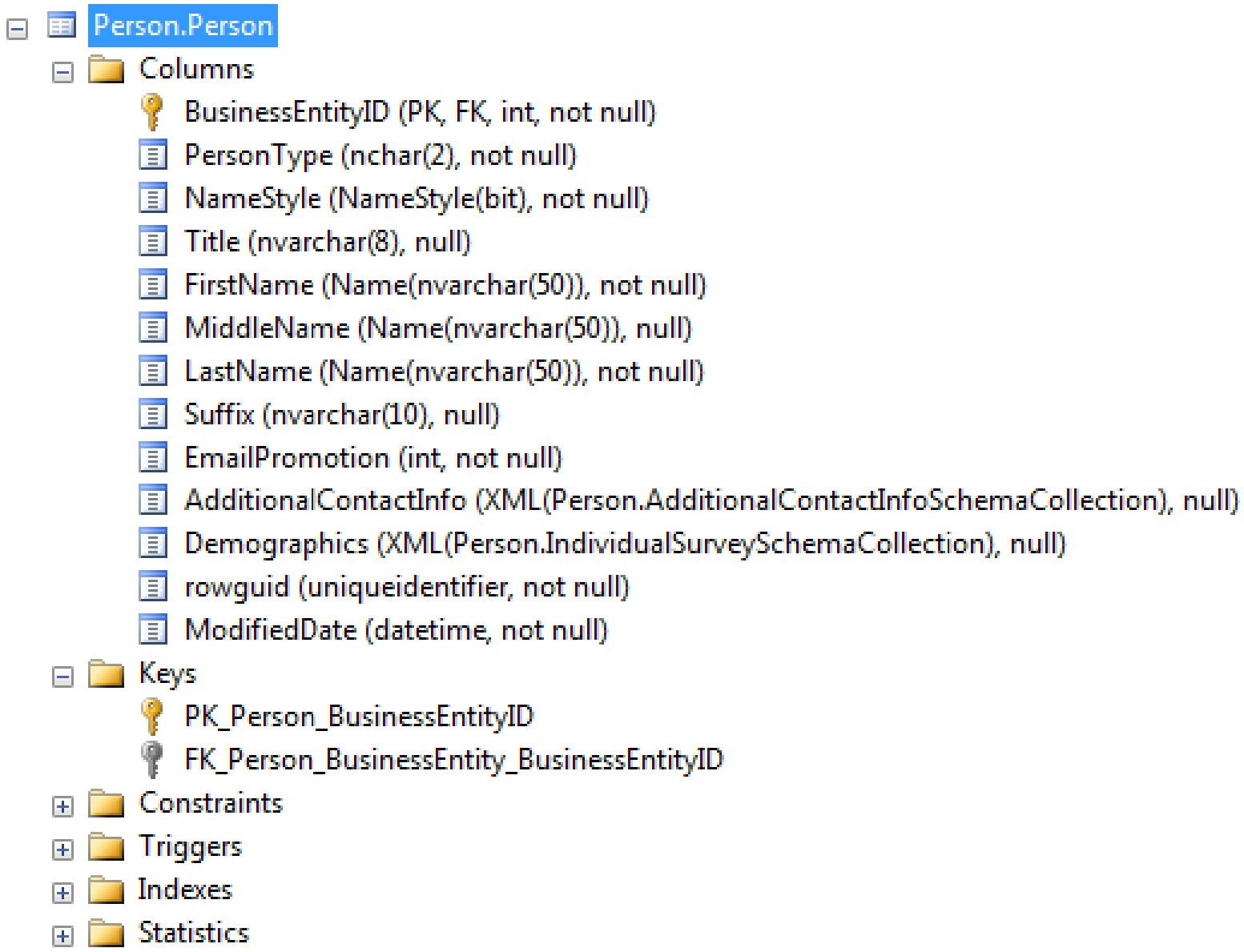
Interpretacja planu wykonania



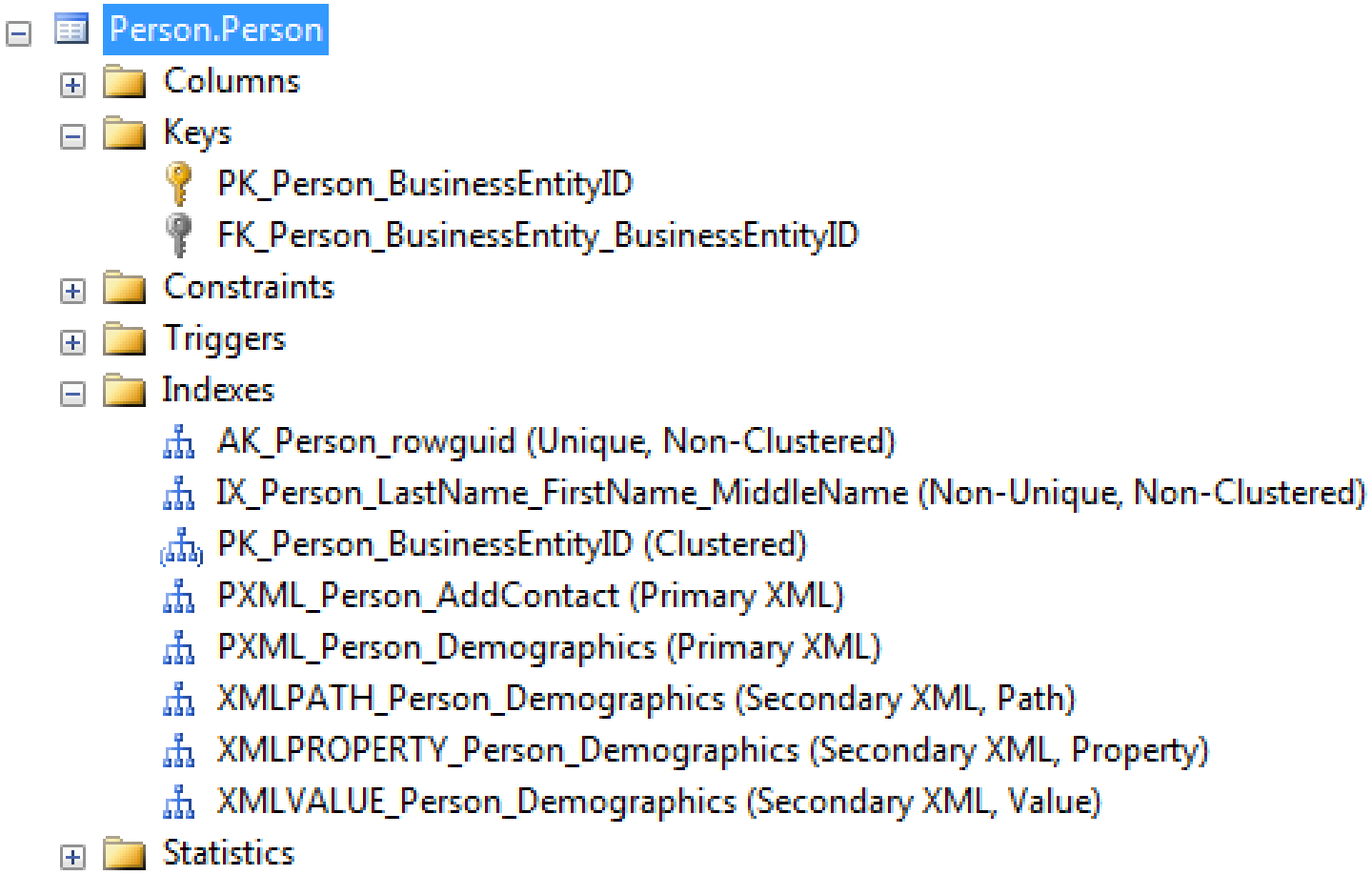
Okno podpowiedzi z informacjami o operatorze



Przykład z BD AdventureWorks2008



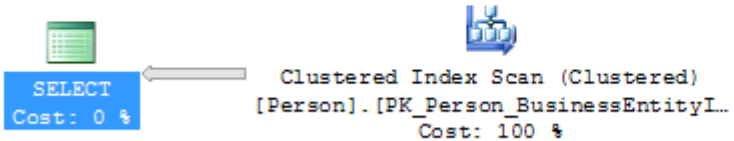
Indeksy w tabeli Person.Person



Przykład 1: Proste plany wykonania

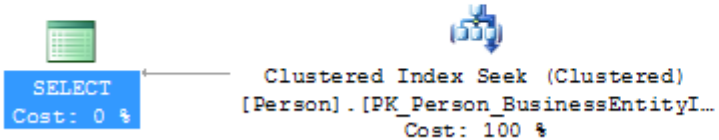
Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [AdventureWorks2008].[Person].[Person]
```



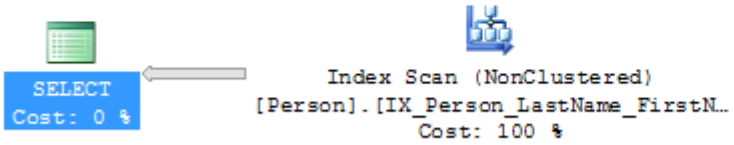
Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [AdventureWorks2008].[Person].[Person] WHERE [BusinessEntityID]=@1
```



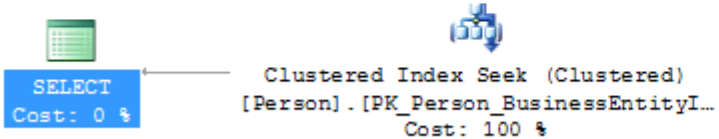
Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person]
```



Results | Messages | Execution plan

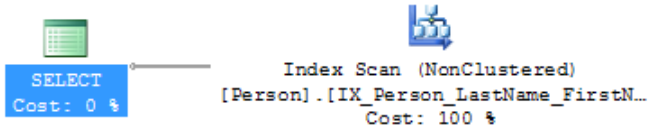
```
Query 1: Query cost (relative to the batch): 100%
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person] WHERE [BusinessEntityID]=@1
```



Przykład 1 cd.: Proste plany wykonania

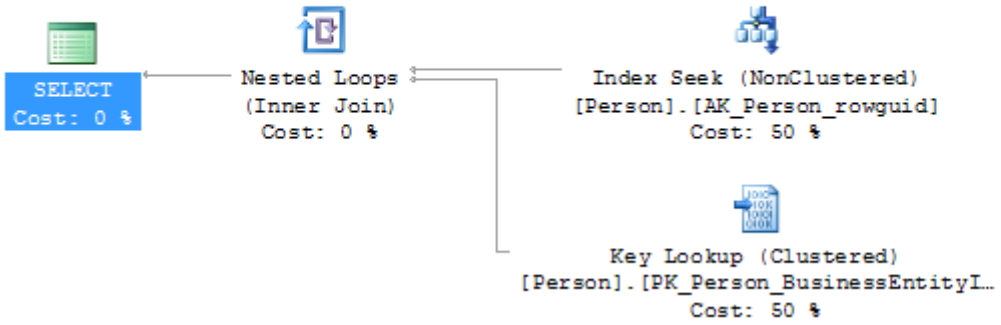
Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person] WHERE [FirstName]=@1
Missing Index (Impact 96.4982): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Person].[Person] ([FirstName])
```



Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person] WHERE [rowguid]=@1
```



Dwie wersje zapytania SQL

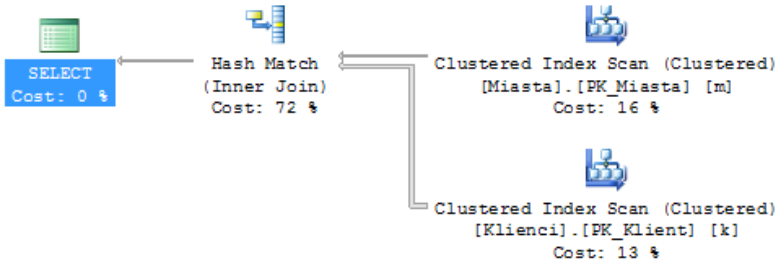
```
SELECT *
FROM BazaRelacyjna.dbo.Klienci k
WHERE k.IdMiasta IN
   (SELECT [IdMiasta]
    FROM [BazaRelacyjna].[dbo].[Miasta] m
    WHERE IdWojewodztwa=12)
GO

SELECT *
FROM BazaRelacyjna.dbo.Klienci k
JOIN [BazaRelacyjna].[dbo].[Miasta] m
ON k.IdMiasta=m.IdMiasta
AND m.IdWojewodztwa=12
GO
```

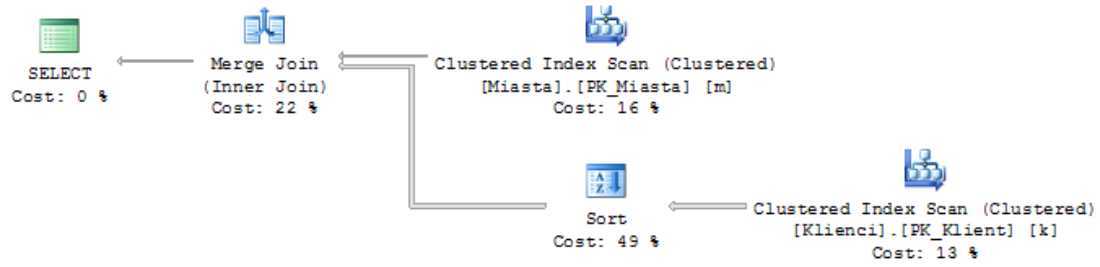
Uzyskane plany wykonania

Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 50%
SELECT * FROM BazaRelacyjna.dbo.Klienci k WHERE k.IdMiasta IN (SELECT [IdMiasta] FROM [BazaRelacyjna].[dbo].[Miasta] m WHERE IdWojewodztwa=12)
```




```
Query 2: Query cost (relative to the batch): 50%
SELECT * FROM BazaRelacyjna.dbo.Klienci k JOIN [BazaRelacyjna].[dbo].[Miasta] m ON k.IdMiasta=m.IdMiasta AND m.IdWojewodztwa=12
```



OPTYMALIZACJA ZAPYTAŃ SQL

Optymalizacja zapytań SQL i bazy danych - 11 przykazań

źródło:

http://www.bazy-danych.info/sql/optymalizacja-zapytan.htm

1. Nie używaj operatora NOT IN – być może składnia jest nieco mniej intuicyjna, aczkolwiek zdecydowanie wydajniejsza jest klauzula NOT EXISTS.
2. Nie używaj operatorów nierówności: ‘<>’, ‘=!’. O wiele korzystniej i wydajniej jest włączyć nawet dużą listę niż wykluczyć z zapytania pojedyncze rekordy.
3. Nie używaj klauzul IS (NOT) NULL. Wartości NULL często powodują problemy w obsłudze wektora wyników (np. w aplikacjach klienckich), wymagają użycia dodatkowych funkcji do obsługi itp. Najlepiej jest zamieniać puste wartości na konkretny znak lub liczbę i w tej postaci przechowywać je w bazie danych.
4. Unikaj zapytań z użyciem DISTINCT. Lepiej i wydajniej jest użyć GROUP BY.
5. Poprzedzaj nazwę tabeli właścicielem tabeli w części FROM zapytania. Unikniesz dzięki temu dwuznaczności i kłopotów.

6. W poleceniach INSERT zawsze jawnie wymieniaj listę kolumn. W przeciwnym wypadku każda zmiana struktury tabeli będzie generować błędy
7. Funkcja COUNT - w każdym wypadku lepiej będzie użyć COUNT(1) niż COUNT(*)
8. Przy łączeniu wielu tabel zwróć uwagę na kolejność występowania złączeń. Zawsze najkorzystniej będzie ograniczyć jak największą ilość wierszy w pierwszym złączeniu.
9. Zawsze przed wykonaniem SQL przyjrzyj się planowi wykonania zapytania (EXPLAIN PLAN).
10. Nie używaj funkcji SQL w części WHERE zapytania. Rozważ dodanie kolumny z wartością wynikową działania funkcji, a jeżeli użycie funkcji jest niezbędne, to korzystaj z indeksów bazujących na funkcjach (function-based indexes).
11. Staraj się też nie używać symboli wieloznacznych na początku szukanego słowa klauzuli WHERE w wyrażeniu LIKE. Czyli dużo wydajniejsze będzie wyszukiwanie nazwisko LIKE ‘K%owalski’ niż nazwisko LIKE ‘%owalski’, nawet jeżeli na tym polu jest założony jakiś indeks, to nie zostanie on wykorzystany poprawnie.

Optymalizacja - podsumowanie

- Nawet najlepsza struktura bazy danych nie gwarantuje wysokiej wydajności jej działania
- Należy odpowiednio zaprojektować bazę danych, dobrać postać zapytań i użyć adekwatnie zaprojektowanych indeksów aby osiągnąć wysoką wydajność
- Aby móc zając 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
- Rola doświadczenia w nabywaniu umiejętności w zakresie optymalizacji

Not Only SQL

BAZY NOSQL

Założenia technologii NoSQL

1. Odejście od założenia ACID bowiem uznano, że są one zbyt restrykcyjne.
2. Rezygnacja z wielu składników (elementów) baz relacyjnych, gdyż ścisły schemat bazy często bywa wadą.
3. Zmniejszenie znaczenia schematów danych, gdyż największą uwagę powinno się zwrócić na dane.

Cechy baz NoSQL

- Sposób przechowywania danych można dobrać w zależności od ich specyfiki.
- Przechowują bogatą strukturę bardzo blisko spokrewnionych danych, które zostały potraktowane jako jednostki (agregacja)
- Łatwo się skalują, gdyż mogą działać na rozproszonej architekturze (nie posiadają złączeń), którą daje się łatwo poszerzyć o kolejne jednostki (ang. node).
- Są przystosowane do pracy na klastrach (efektywna praca w klastrach w przeciwieństwie do baz relacyjnych), nie wszystkie jednak z nich korzystają. Istnieją również bazy grafowe, w którym model dystrybucji jest podobny do baz relacyjnych, lecz model danych jest inny, dzięki czemu można w nich przechowywać dane zawierające skomplikowane relacje.
- Ich budowa jest przystosowana do potrzeb aplikacji webowych powstałych po 2000 roku.

Rodzaje baz NoSQL

- klucz-wartość (ang. *key-value*)
- kolumnowe (ang. *column oriented stores*)
- dokumentowe
- bazy oparte na grafach (ang. *graph stores*)
- obiektowe (ang. *object databases*)
- inne bazy danych - zazwyczaj rozwiązania hybrydowe

PRZEGLĄD

Internetowe bazy danych (1)

- Przykłady serwisów internetowych
- Wybór technologii oraz architektury
- Języki programowania
- Serwery baz danych
- Technologie udostępniania informacji
  - HTML, ISAPI, CGI, ASP, PHP, JSP, XML
- Technologie dostępu do danych
  - ODBC, JDBC, OLE DB, DAO, ADO/.NET, ORM

Internetowe bazy danych (2)

- E-commerce
- Bezpieczeństwo i zagrożenia
- Ataki bazodanowe
- Umiejscowienie bazy danych w architekturze sieci firmowej
- Składowanie, archiwizacja i ochrona danych

Internetowe bazy danych (3)

- Uwierzytelnienie, autoryzacja
- Uwierzytelnienie i bezpieczeństwo w kontekście PHP i baz danych
- Szyfrowanie danych
- Bezpieczeństwo transakcji – podpis cyfrowy, SSL

Internetowe bazy danych (4)

- Języki skryptowe: PHP, Python, Ruby
- Frameworki
- Mapowanie obiektowo-relacyjne

Pytania?

DZIĘKUJĘ ZA UWAGĘ