



Politechnika Wroclawska

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



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

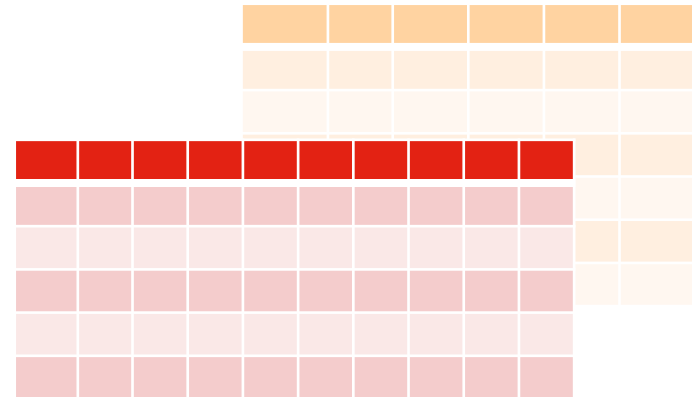
- Po co optymalizować?
- Architektura fizyczna SQL Servera
- Indeksy
 - Zgrupowane
 - Niezgrupowane
 - Pokrywające
- Plany wykonania zapytań



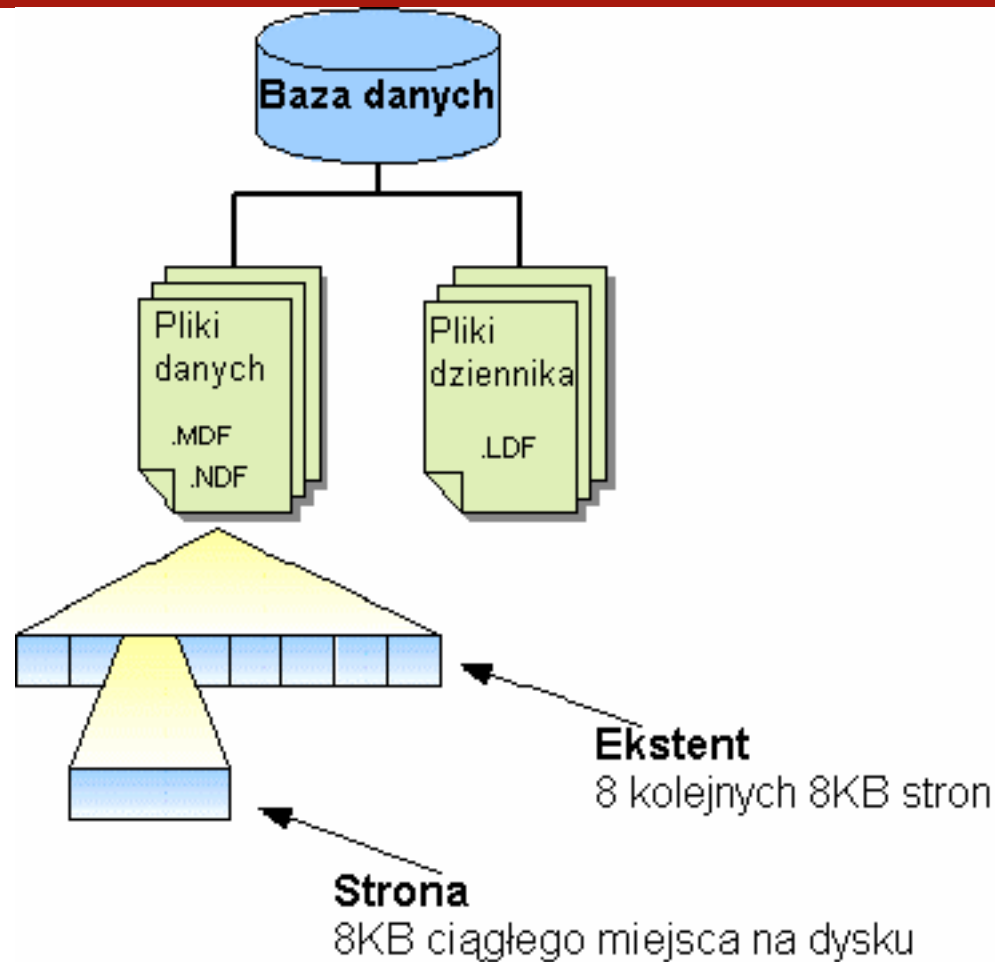
STRUKTURA FIZYCZNA SYSTEMU MS SQL SERVER

Logiczna architektura bazy danych

- Tabele
- Widoki
- Procedury składowane
- Wyzwalacze
- ...



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



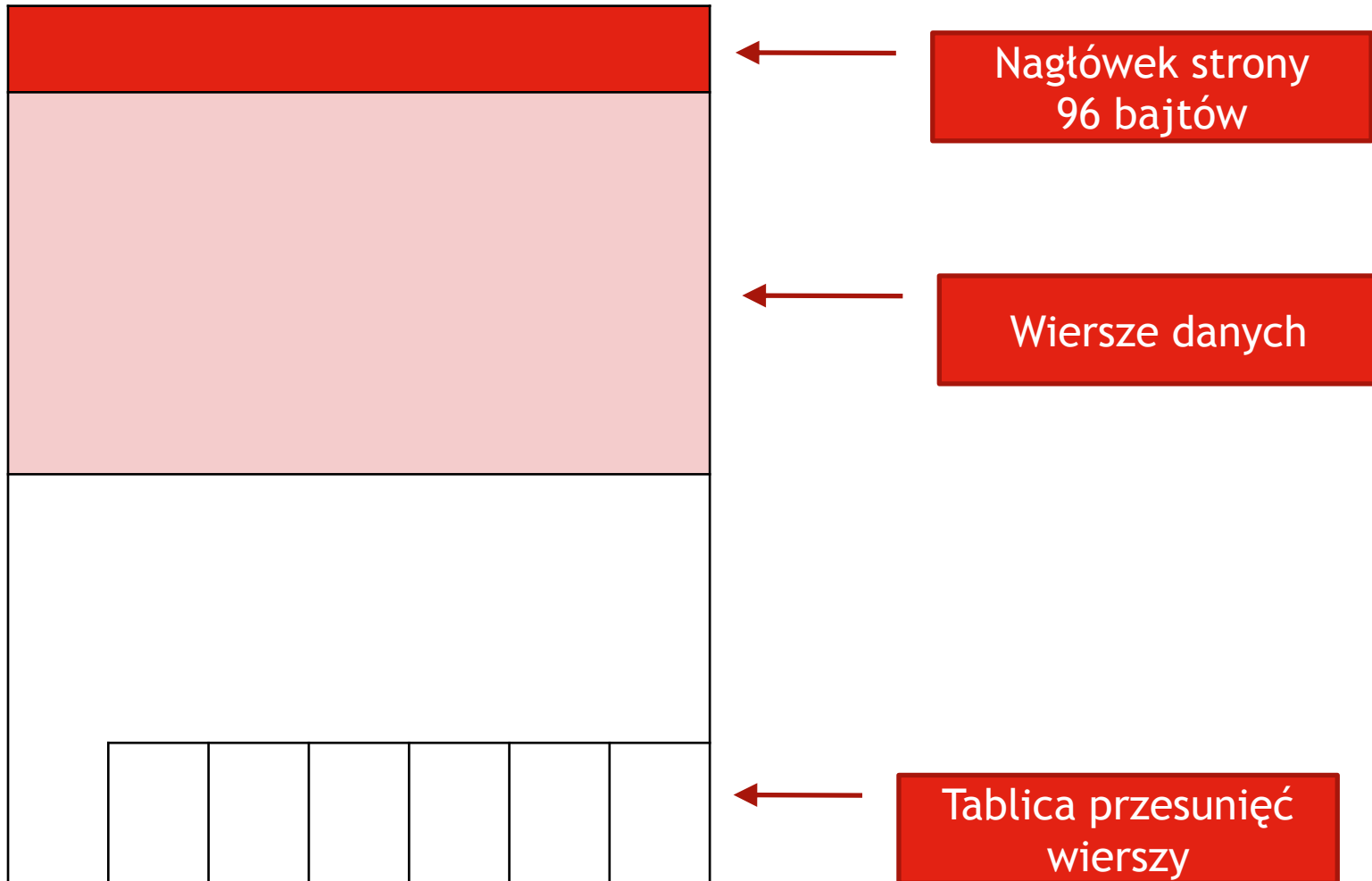


Rodzaje stron

- **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

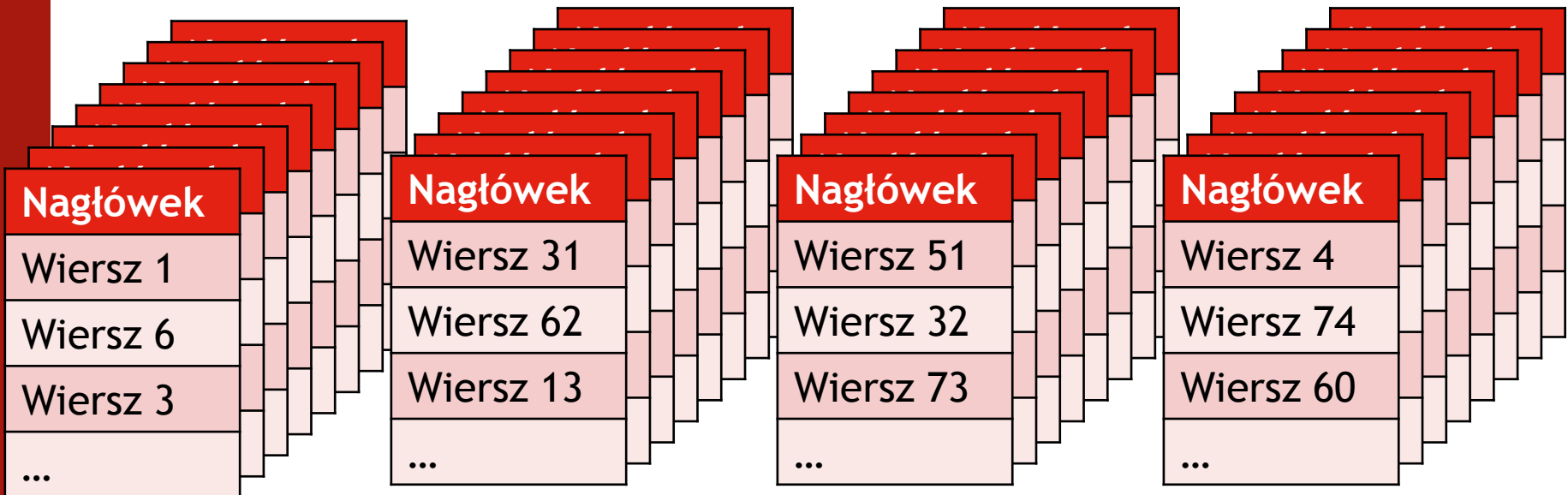


Struktura strony danych



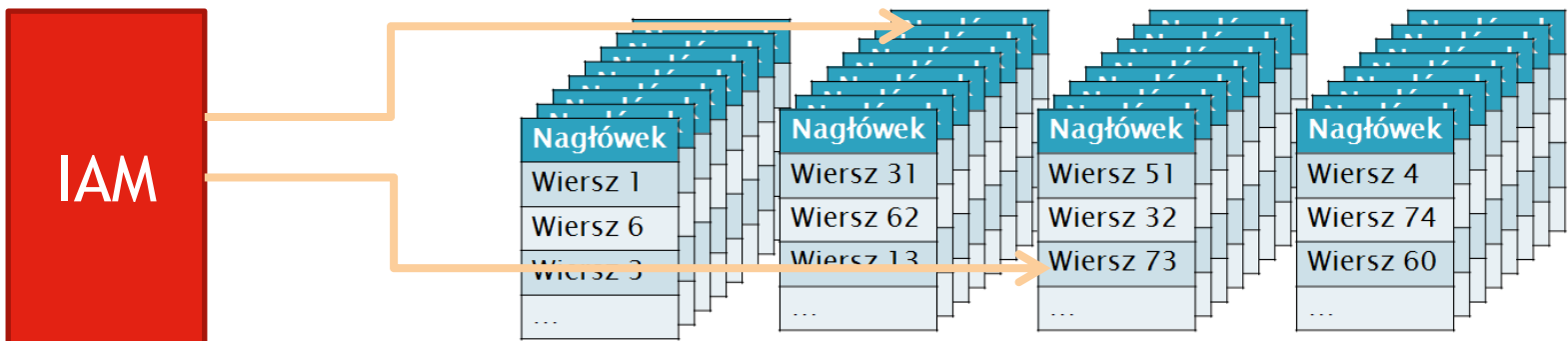
Szeregi (ang. *heap*)

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



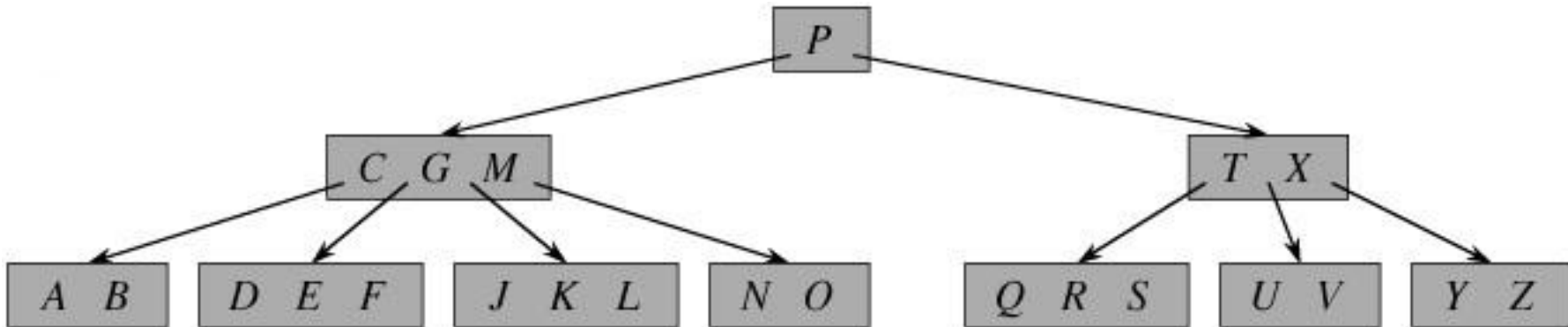
Fizyczna organizacja danych w SQL Server

- 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



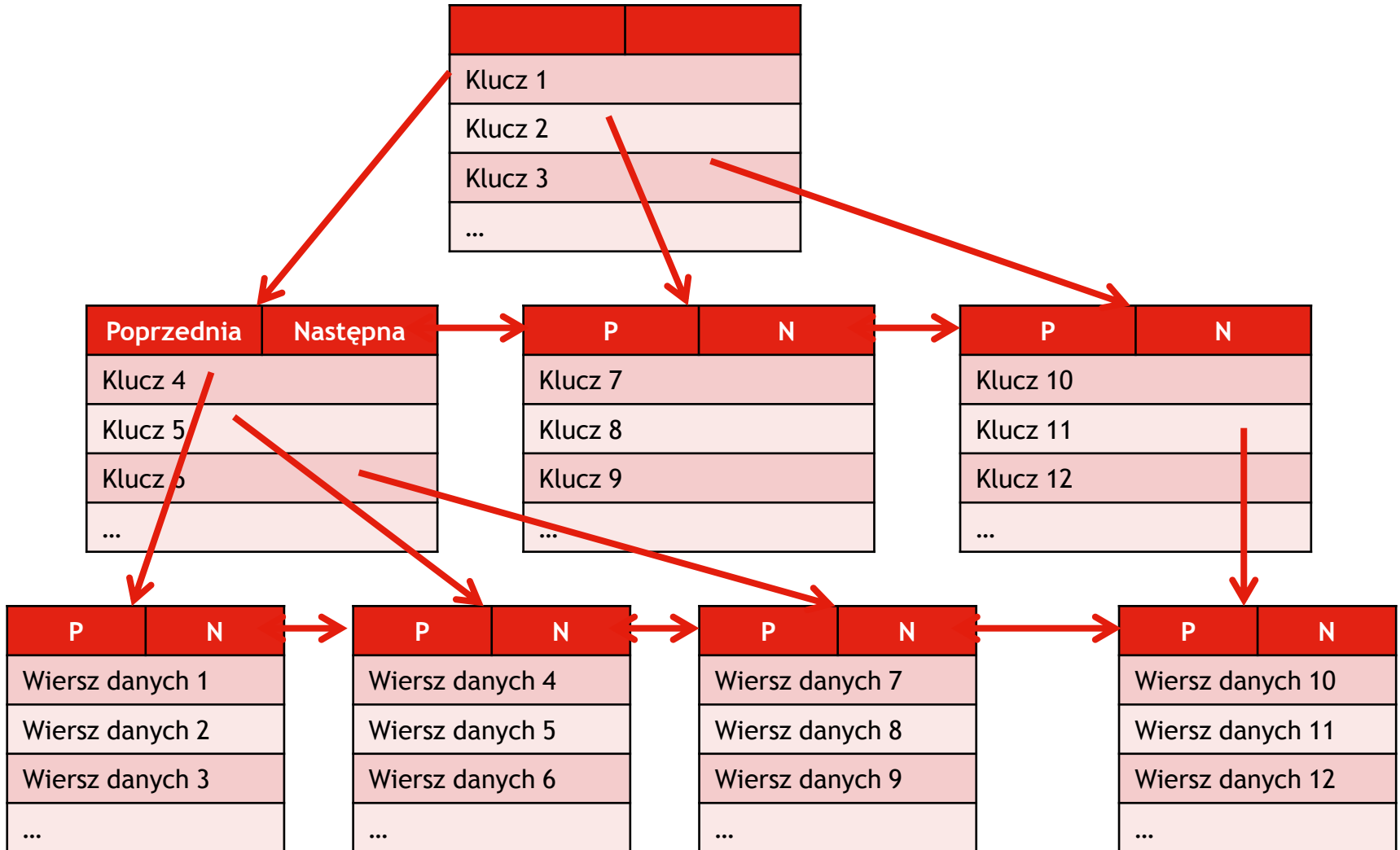


Indeks zgrupowany (z danymi)

- 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

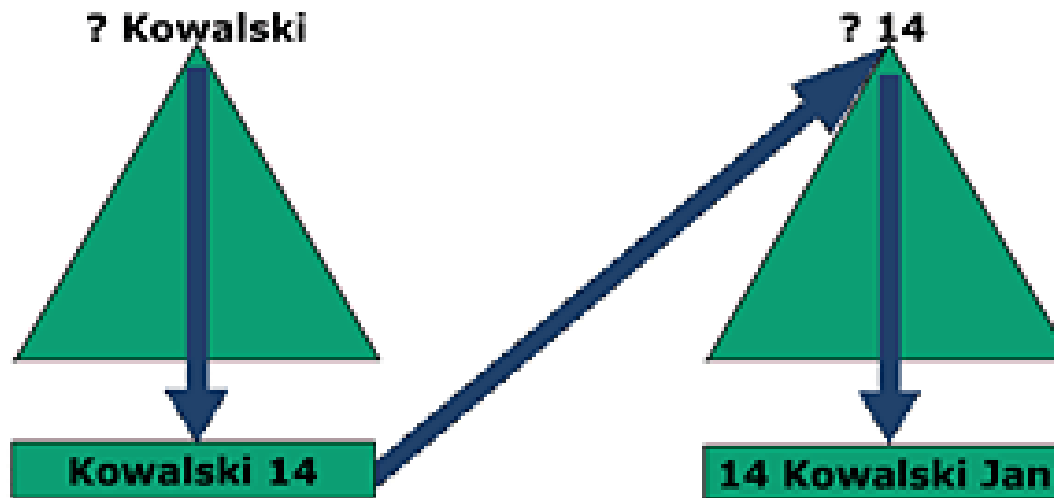
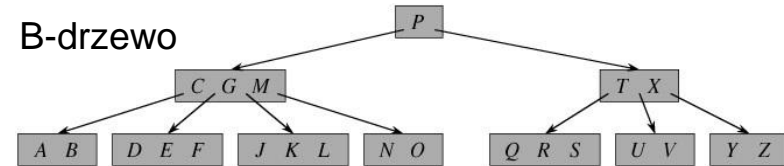


Indeks niezgrupowany

- 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



Liść indeksu niegrupowanego

Rekord z danymi

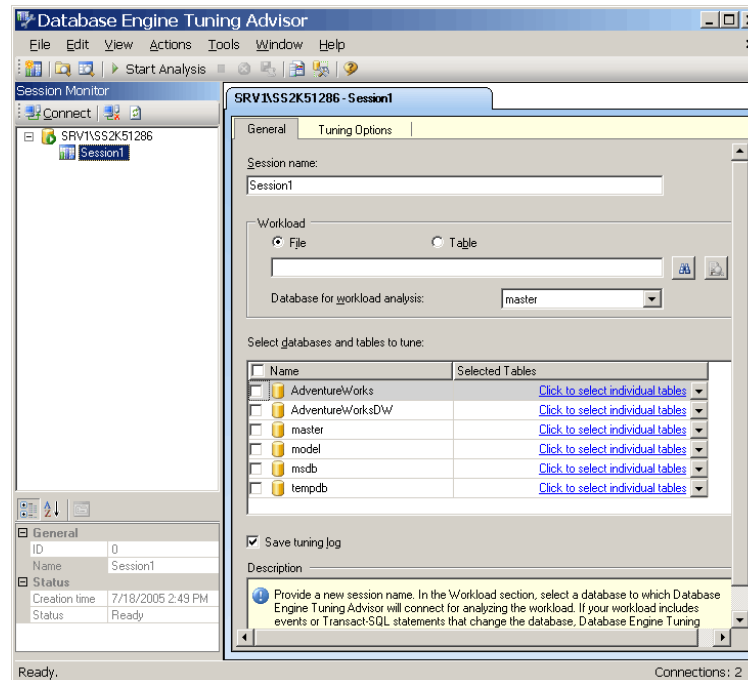


PLAN WYKONANIA W MS SQL SERVER



Narzędzia wspierające optymalizację

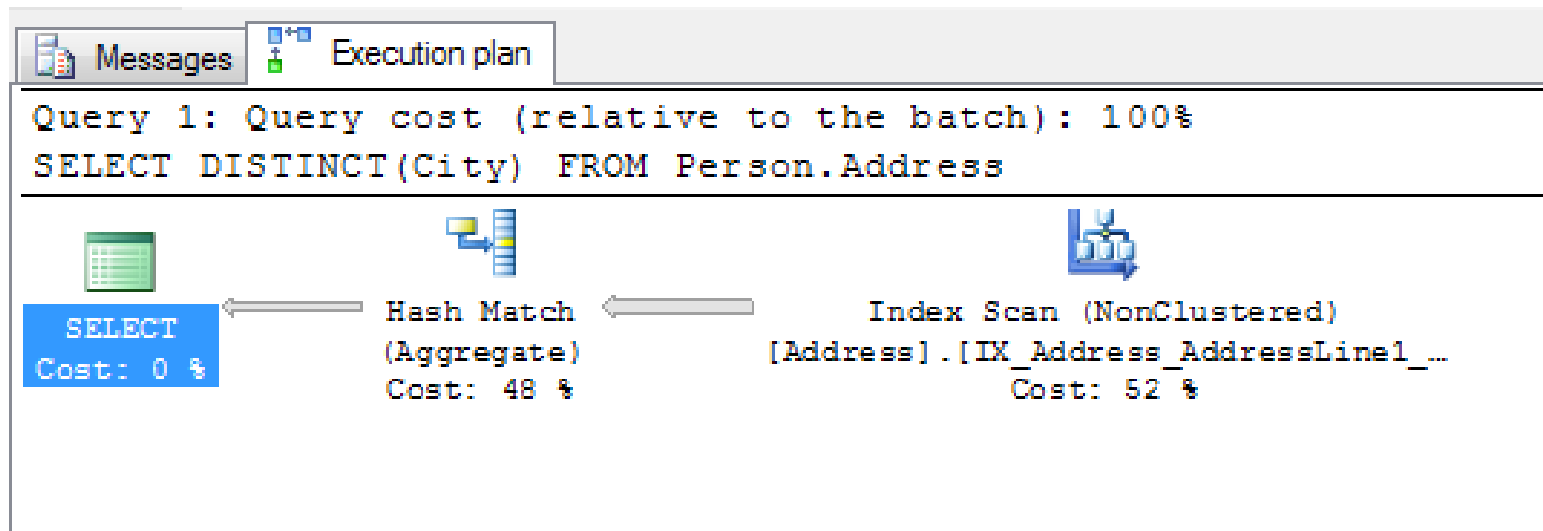
- MS SQL Server Management Studio
 - Plan wykonania
- Database Engine Tuning Advisor





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.

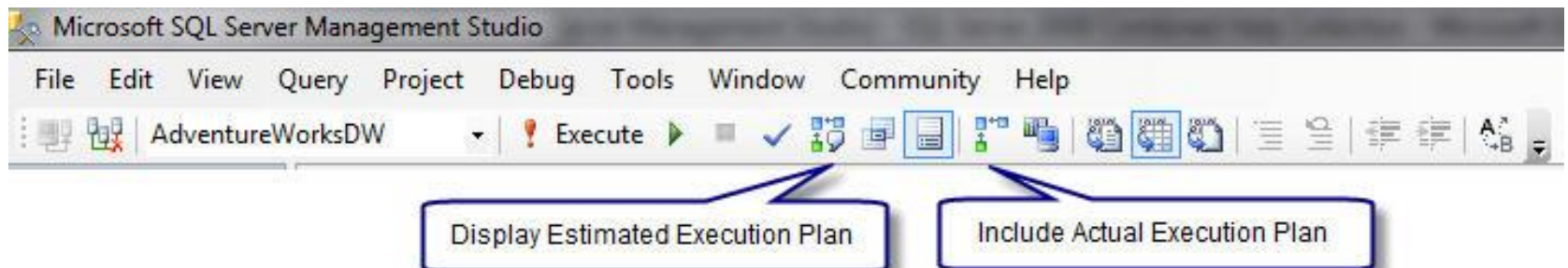


Fazy przetwarzania zapytania



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)

Query 1: Query cost (relative to the batch): 100%

```
SELECT [member].[member_no], [member].[lastname], [member].[fi...
```

The graphical plan shows the following operators and their costs:










- Hash Match (Inner Join): Cost: 43 %
- Clustered Index Scan (Clustered) [corporation].[corporation_ident]: Cost: 3 %
- Clustered Index Scan (Clustered) [member].[member_ident]: Cost: 53 %
- SELECT: Cost: 0 %

The right pane displays the following properties:






Property	Value
Cached plan size	40 KB
CompileCPU	4
CompileMemory	296
CompileTime	4
Degree of Parallelism	1
Estimated Number of Rows	1295.63
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.223126
Logical Operation	
Memory Grant	1352
Optimization Level	FULL
Physical Operation	
QueryHash	0xAB3CE18FBFC5E187
QueryPlanHash	0x64A3D75BFCBA3094
Reason For Early Termination Of Statement	Good Enough Plan Found
RetrievedFromCache	true
Statement	SELECT [member].[member_no], [member].[



Ikony planu wykonania (wybór)

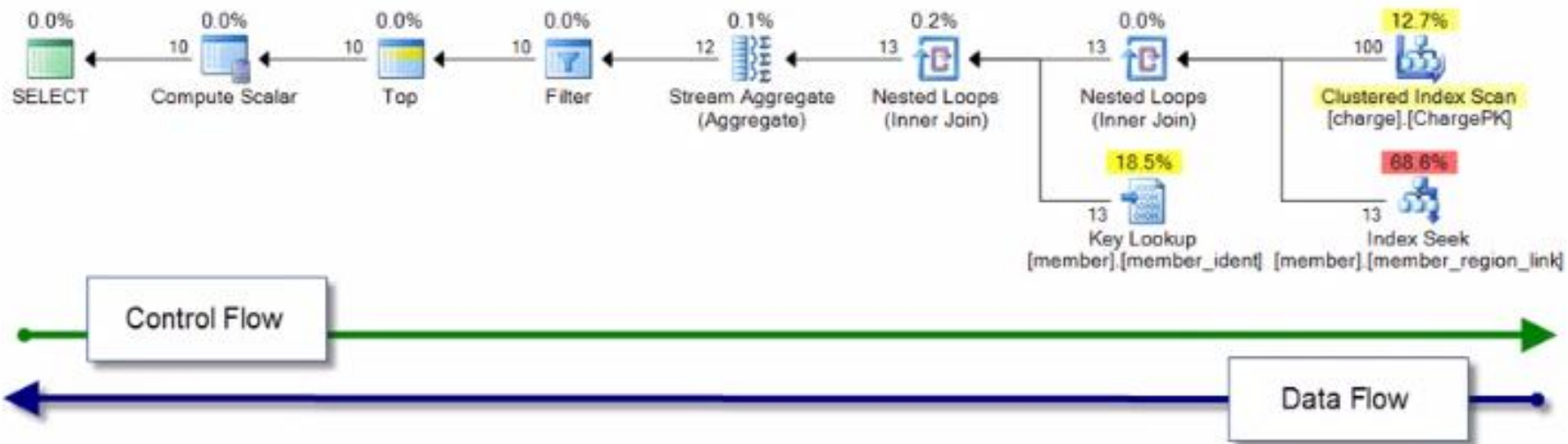
Ikona	Element języka/Operator	(ang.)
	Wynik	Result
	Skanowanie indeksu zgrupowanego	Clustered Index Scan
	Szukanie indeksu zgrupowanego	Clustered Index Seek
	Skanowanie indeksu zwykłego	Nonclustered Index Scan
	Szukanie indeksu zwykłego	Nonclustered Index Seek
	Pętla zagnieżdżona	Nested Loops
	Wyszukiwanie 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

```

SELECT Nazwisko
      , Imie
FROM BazaRelacyjna
JOIN [BazaRelacyjna].[dbo].[Klienci] k
ON k.IdMiasta=m.IdMiasta
AND m.IdWojewodztwa=k.IdWojewodztwa
GO
        
```

Hash Match

Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows.

Physical Operation	Hash Match
Logical Operation	Inner Join
Actual Number of Rows	26
Estimated I/O Cost	0
Estimated CPU Cost	0,0196235
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0,0197561 (72%)
Estimated Subtree Cost	0,0275778
Estimated Number of Rows	148,412
Estimated Row Size	27 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0

Output List
[BazaRelacyjna].[dbo].[Klienci].Nazwisko;
[BazaRelacyjna].[dbo].[Klienci].Imie

Probe Residual
[BazaRelacyjna].[dbo].[Klienci].[IdMiasta] as [k].
[IdMiasta]=[BazaRelacyjna].[dbo].[Miasta].[IdMiasta] as [m].[IdMiasta]

Hash Keys Probe
[BazaRelacyjna].[dbo].[Klienci].IdMiasta

Query 2: Query cost 0,0275778

```

SELECT Nazwisko , Imie
FROM [BazaRelacyjna].[dbo].[Klienci] k
JOIN [BazaRelacyjna].[dbo].[Miasta] m ON k.IdMiasta=m.IdMiasta
AND m.IdWojewodztwa=k.IdWojewodztwa
GO
        
```

Disconnected.



Przykład z BD AdventureWorks2008

The screenshot displays the schema for the `Person.Person` table in the AdventureWorks2008 database. The schema is organized into folders: Columns, Keys, Constraints, Triggers, Indexes, and Statistics.

- Columns:**
 - `BusinessEntityID` (PK, FK, int, not null)
 - `PersonType` (nchar(2), not null)
 - `NameStyle` (NameStyle(bit), not null)
 - `Title` (nvarchar(8), null)
 - `FirstName` (Name(nvarchar(50)), not null)
 - `MiddleName` (Name(nvarchar(50)), null)
 - `LastName` (Name(nvarchar(50)), not null)
 - `Suffix` (nvarchar(10), null)
 - `EmailPromotion` (int, not null)
 - `AdditionalContactInfo` (XML(Person.AdditionalContactInfoSchemaCollection), null)
 - `Demographics` (XML(Person.IndividualSurveySchemaCollection), null)
 - `rowguid` (uniqueidentifier, not null)
 - `ModifiedDate` (datetime, not null)
- Keys:**
 - `PK_Person_BusinessEntityID`
 - `FK_Person_BusinessEntity_BusinessEntityID`
- Constraints:** (Folder)
- Triggers:** (Folder)
- Indexes:** (Folder)
- Statistics:** (Folder)



Indeksy w tabeli Person.Person

- [-] [Table Icon] Person.Person
 - [+] [Folder Icon] Columns
 - [-] [Folder Icon] Keys
 - [Key Icon] PK_Person_BusinessEntityID
 - [Key Icon] FK_Person_BusinessEntity_BusinessEntityID
 - [+] [Folder Icon] Constraints
 - [+] [Folder Icon] Triggers
 - [-] [Folder Icon] Indexes
 - [Index Icon] AK_Person_rowguid (Unique, Non-Clustered)
 - [Index Icon] IX_Person_LastName_FirstName_MiddleName (Non-Unique, Non-Clustered)
 - [Index Icon] PK_Person_BusinessEntityID (Clustered)
 - [Index Icon] PXML_Person_AddContact (Primary XML)
 - [Index Icon] PXML_Person_Demographics (Primary XML)
 - [Index Icon] XMLPATH_Person_Demographics (Secondary XML, Path)
 - [Index Icon] XMLPROPERTY_Person_Demographics (Secondary XML, Property)
 - [Index Icon] XMLVALUE_Person_Demographics (Secondary XML, Value)
 - [+] [Folder Icon] Statistics



Przykład 1: Proste plany wykonania

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT * FROM [AdventureWorks2008].[Person].[Person]
```

Clustered Index Scan (Clustered)
[Person].[PK_Person_BusinessEntityI...]
Cost: 100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT * FROM [AdventureWorks2008].[Person].[Person] WHERE [BusinessEntityID]=@1
```

Clustered Index Seek (Clustered)
[Person].[PK_Person_BusinessEntityI...]
Cost: 100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person]
```

Index Scan (NonClustered)
[Person].[IX_Person_LastName_FirstN...]
Cost: 100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person] WHERE [BusinessEntityID]=@1
```

Clustered Index Seek (Clustered)
[Person].[PK_Person_BusinessEntityI...]
Cost: 100 %



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])

SELECT Cost: 0 %

Index Scan (NonClustered)
[Person].[IX_Person_LastName_FirstN...]
Cost: 100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT [FirstName] FROM [AdventureWorks2008].[Person].[Person] WHERE [rowguid]=@1
```

SELECT Cost: 0 %

Nested Loops (Inner Join)
Cost: 0 %

Index Seek (NonClustered)
[Person].[AK_Person_rowguid]
Cost: 50 %

Key Lookup (Clustered)
[Person].[PK_Person_BusinessEntityL...]
Cost: 50 %



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
```

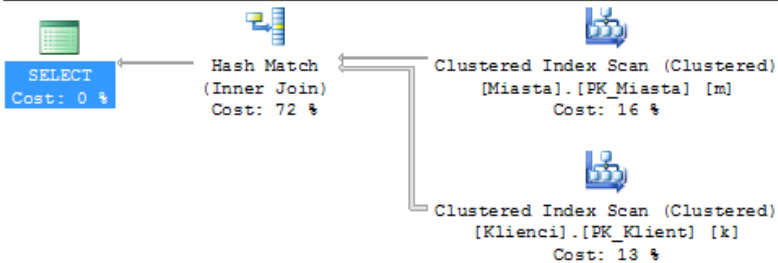
```
SELECT *  
FROM BazaRelacyjna.dbo.Klienci k  
JOIN [BazaRelacyjna].[dbo].[Miasta] m  
ON k.IdMiasta=m.IdMiasta  
AND m.IdWojewodztwa=12  
GO
```



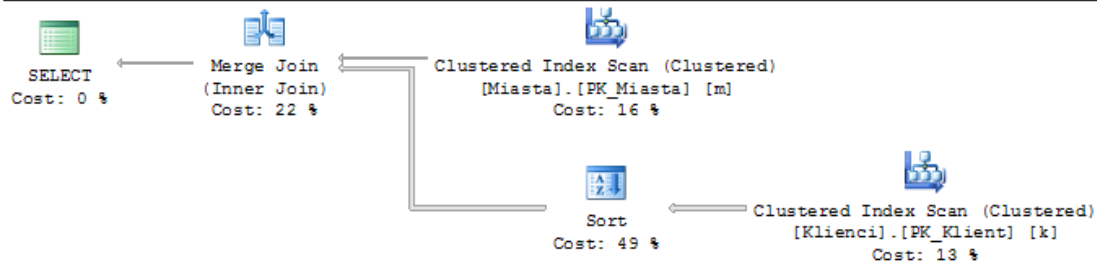
Uzyskane plany wykonania

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/optimalizacja-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 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
- Rola doświadczenia w nabywaniu umiejętności w zakresie optymalizacji



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