Sprawozdanie z obliczeń wykonanych na podstawie artykułu pt.: „Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych”

# Wprowadzenie

Baza danych to zorganizowany zbiór usystematyzowanych informacji, czyli danych, zwykle przechowywany w systemie komputerowym w formie elektronicznej. Bazą danych steruje zwykle system zarządzania bazami danych (DBMS). Dane i system DBMS oraz powiązane z nimi aplikacje razem tworzą system bazodanowy, czyli inaczej bazę danych.

Kluczowym jest, aby zapytania wykonywane na bazie danych miały krótki czas realizacji. W tym celu konieczne jest zastosowanie dobrego schematu bazy danych oraz optymalizacja pod kątem danych, jakie będą w bazie przechowywane. Wszystkie te przygotowania powinny zapewnić odporność na obciążenia generowane przez użytkowników baz danych.

Zapytania muszą opierać się na łączeniu wielu tabel, co można osiągnąć na dwa sposoby:

1. Poprzez złączenia (np. wewnętrze) tabel,
2. Poprzez zapytania zagnieżdżone skorelowane lub nieskorelowane

Najtrudniejsze w optymalizacji są zapytania zagnieżdżone skorelowane, nieskorelowane nieco mniej. Złączenia tabel natomiast są zazwyczaj efektywnie wykonywane przez DBMS.

## Normalizacja schematów tabel w bazach danych

Wyróżniamy 3 podstawowe postacie normalne:

1. 1NF – Pierwsza postać normalna. Jej jedynym warunkiem jest zapewnienie stanu atomowego każdej krotki (niemożność podziału na mniejsze wartości). Atomowość danych jest ściśle powiązana z ich typem (nazwanym i skończonym zbiorem wartości). Ważną cechą relacji utworzonych zgodnie z modelem relacyjnym jest to, że zawsze są znormalizowane – spełniają 1NF.
2. 2NF – Druga postać normalna zabrania, aby dla zdefiniowanego klucza istniał podzbiór atrybutów podstawowych, który identyfikuje atrybuty wtórne. Innymi słowy: aby każdy atrybut wtórny tej relacji był w pełni funkcyjnie zależny od wszystkich kluczy tej relacji.
3. 3NF – Relacja jest w trzeciej postaci normalnej tylko wtedy, gdy jest w drugiej postaci normalnej i każdy atrybut wtórny jest tylko bezpośrednio zależny od klucza głównego. Innymi słowy, wymaga usunięcia wszelkich pól niezwiązanych z kluczem głównym.

Istnieją również inne postacie, ale 3NF powszechnie uważana jest za wystarczającą dla większości projektów.

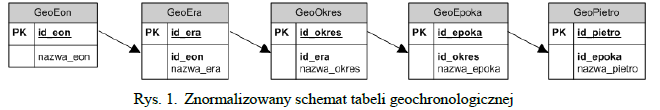
## Schematy hurtowni danych

Hurtownia danych jest centralnie zarządzaną i zintegrowaną bazą danych, gromadzącą dane z systemów i źródeł operacyjnych organizacji. Wyróżniamy następujące schematy:

1. Schemat gwiazdy
   1. Prosta struktura
   2. Duża efektywność zapytań z uwagi na małą liczbę powiązań
   3. Długi czas ładowania danych do tabel wymiarów z uwagi na denormalizację
2. Schemat płatka śniegu:
   1. Spadek wydajności zapytań w porównaniu z gwiazdą z uwagi na większą liczbę relacji
   2. Struktura łatwiejsza do modyfikacji
   3. Relatywnie krótszy czas ładowania danych do tabel ze względu na mniejszą redundancję dzięki normalizacji
   4. Wykorzystywany rzadziej niż schemat gwiazdy, gdyż efektywność zapytań jest ważniejsza niż efektywność ładowania danych
3. Schemat konstelacji faktów:
   1. Rozwiązanie pośrednie między gwiazdą a płatkiem śniegu
   2. Zwykle występuje więcej niż jedna tabela
   3. Część tabel wymiarów jest znormalizowana a część zdenormalizowana
   4. Tabele faktów są ze sobą powiązane w relacji 1:1 lub 1: *n*
   5. Tabele faktów mogą współdzielić te same tabele wymiarów

# Tabela geochronologiczna

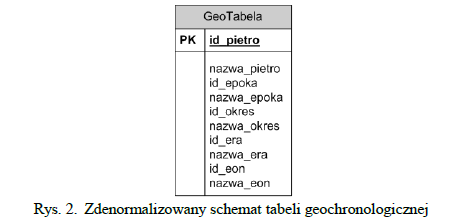
Tabela geochronologiczna obrazuje przebieg historii Ziemi na podstawie następstwa procesów i warstw skalnych. W różnych opracowaniach można spotkać różne podziały stratygraficzne z powodu ciągle trwających badań i typowania nowych profili wzorcowych. Wykorzystany schemat pochodzi ze strony Wikipedia.



# Konstrukcja wymiaru geochronologicznego

W tabeli geochronologicznej znalazły miejsce jednostki geochronologiczne mające wy-miar czasowy (eon, era, okres, epoka i wiek) oraz odpowiadające im jednostki stratygraficzne.

Podczas wykonywania testów skupiono się na konstrukcji tabeli geochronologicznej w dwóch przypadkach:

1. Schemacie znormalizowanym (płatka śniegu, rys. 1),
2. Schemacie zdenormalizowanym (schemat gwiazdy).

Formę zdenormalizowaną tabeli geochronologicznej osiągnięto tworząc jedną tabelę GeoTabela (rys. 2), zawierającą wszystkie dane z powyższych tabel. Dokonano tego za pomocą polecenia *JOIN … ON*, obejmującego wszystkie tabele tworzące hierarchię:

**SELECT p.id\_pietro, p.nazwa\_pietro, ep.id\_epoka, ep.nazwa\_epoka, o.id\_okres, o.nazwa\_okres, er.id\_era, er.nazwa\_era, eo.id\_eon, eo.nazwa\_eon**

**INTO tabela.GeoTabela**

**FROM tabela.GeoPietro p**

**JOIN tabela.GeoEpoka ep ON p.id\_epoka = ep.id\_epoka**

**JOIN tabela.GeoOkres o ON ep.id\_okres = o.id\_okres**

**JOIN tabela.GeoEra er ON o.id\_era = er.id\_era**

**JOIN tabela.GeoEon eo ON er.id\_eon = eo.id\_eon**

**ALTER TABLE tabela.Geotabela**

**ADD CONSTRAINT id\_pietro PRIMARY KEY (id\_pietro)**

*kod MSSQL*

Utworzenie tabeli GeoTabela umożliwia szybki dostęp do wszystkich danych tabeli geochronologicznej za pomocą jednego zapytania prostego, co nie jest możliwe w przypadku schematu znormalizowanego opisanego w punkcie pierwszym.

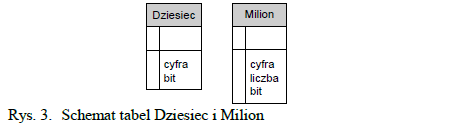
# Testy wydajności

W testach skupiono się na porównaniu wydajności złączeń oraz zapytań zagnieżdżonych, wykonywanych na tabelach o dużej liczbie danych. Testowane były następujące rozwiązania bazowe:

1. MSSQL – w DBMS Microsoft SQL Server Management Studio oraz Data Grip
2. SQLite – w DBMS Data Grip

W zapytaniach testowych łączono dane z tabeli geochronologicznej z syntetycznymi danymi o rozkładzie jednostajnym z tabeli *Milion*, wypełnionej kolejnymi liczbami naturalnymi od 0 do 999 999. Tabela *Milion* została utworzona na podstawie odpowiedniego autozłączenia tabeli *Dziesiec* wypełnionej liczbami od 0 do 9.  
**INSERT INTO tabela.Milion**   
**SELECT a1.cyfra + 10 \* a2.cyfra +100\*a3.cyfra + 1000\*a4.cyfra + 10000\*a5.cyfra + 100000\*a6.cyfra AS liczba,**  
**a1.cyfra AS cyfra, a1.bit AS bit**  
**FROM tabela.Dziesiec a1, tabela.Dziesiec a2, tabela.Dziesiec a3, tabela.Dziesiec a4, tabela.Dziesiec a5, tabela.Dziesiec a6;**

Schematy tabel przedstawiono na rysunku 3.



## Konfiguracja sprzętowa i programowa

1. CPU: Intel® Core™ i7-8750H, l. rdzeni: 6, l. wątków: 12, bazowa częstotliwość: 2.20 GHz, max częstotliwość: 4.10 GHz
2. GPU: Nvidia GeForce GTX 1050 Ti, 4GB GDDR5
3. RAM: 1x8GB DDR4 2666 MHZ Kingston, 1x4GB DDR4 2400 MHz Crucial
4. SSD 256GB Kingston
5. S.O.: Windows 10

Testy wykonywano wielokrotnie dla każdego systemu zarządzania bazą danych.

## Kryteria testów

W teście wykonano szereg zapytań sprawdzających wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji zdenormalizowanej i znormalizowanej. Procedurę testową przeprowadzono w dwóch etapach:

1. pierwszy etap obejmował zapytania bez nałożonych indeksów na kolumny danych (jedynymi indeksowanymi danymi były dane w kolumnach będących kluczami głównymi poszczególnych tabel),
2. w drugim etapie nałożono indeksy na wszystkie kolumny biorące udział w złączeniu.

Zaproponowano cztery zapytania:

1. Zapytanie 1 (1 ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym do warunku złączenia dodano operację modulo, dopasowującą zakresy wartości złączanych kolumn:  
     
   **SELECT COUNT(\*) FROM tabela.Milion**

**INNER JOIN tabela.GeoTabela ON (Milion.liczba % 77) = (GeoTabela.id\_pietro);**

1. Zapytanie 2 (2 ZL), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:  
   **SELECT COUNT(\*) FROM tabela.Milion**

**INNER JOIN tabela.GeoPietro p ON (Milion.liczba % 77 = p.id\_pietro)**

**JOIN tabela.GeoEpoka ep ON p.id\_epoka = ep.id\_epoka**

**JOIN tabela.GeoOkres o ON ep.id\_okres = o.id\_okres**

**JOIN tabela.GeoEra er ON o.id\_era = er.id\_era**

**JOIN tabela.GeoEon eo ON er.id\_eon = eo.id\_eon;**

1. Zapytanie 3 (3 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

**SELECT COUNT(\*) FROM tabela.Milion**

**WHERE Milion.liczba % 77 =**

**(SELECT id\_pietro FROM tabela.GeoTabela WHERE Milion.liczba % 77 = id\_pietro);**

1. Zapytanie 4 (4 ZG), którego celem jest złączenie syntetycznej tablicy miliona wyników z tabelą geochronologiczną w postaci znormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane, a zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych:

**SELECT COUNT(\*) FROM tabela.Milion**

**WHERE Milion.liczba % 77 IN**

**(SELECT p.id\_pietro FROM tabela.GeoPietro p**

**JOIN tabela.GeoEpoka ep ON p.id\_epoka = ep.id\_epoka**

**JOIN tabela.GeoOkres o ON ep.id\_okres = o.id\_okres**

**JOIN tabela.GeoEra er ON o.id\_era = er.id\_era**

**JOIN tabela.GeoEon eo ON er.id\_eon = eo.id\_eon);**

# Wyniki testów

Testy zostały wykonane ok. 10 razy dla każdego kryterium. Wyniki skrajne pominięto.  
Otrzymane wyniki zostały zaprezentowane w poniższej tabeli.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1 ZL** | | **2 ZL** | | **3 ZL** | | **4 ZL** | |
| **BEZ INDEKSÓW** | **MIN** | **ŚR** | **MIN** | **ŚR** | **MIN** | **ŚR** | **MIN** | **ŚR** |
| **MSSQL** (Server Management Studio) | 25 | 26.40 | 26 | 27.40 | 24 | 27.40 | 26 | 28.10 |
| **MSSQL** (Data Grip) | 56 | 68 | 61 | 72.80 | 60 | 68.90 | 54 | 63.30 |
| **SQLite** | 153 | 158.10 | 277 | 284.80 | 276 | 284.20 | 165 | 170.30 |
| **Z INDEKSAMI** |  |  |  |  |  |  |  |  |
| **MSSQL** (Server Management Studio) | 16 | 18.60 | 17 | 19.10 | 18 | 20.30 | 18 | 20.80 |
| **MSSQL** (Data Grip) | 50 | 60 | 48 | 62.10 | 46 | 55.50 | 46 | 54.90 |
| **SQLite** | 130 | 134.44 | 175 | 180 | 243 | 251.5 | 137 | 145.80 |

Dla lepszego zobrazowania różnic w czasach wykonania zapytań zostały wykonane wykresy – osobno dla testów wykonywanych bez indeksów (indeksami były tylko klucze główne) oraz z indeksami na kolumnach biorących udział w złączeniu.

# Wnioski

Przedstawienie wyników w tabeli oraz na wykresach pozwala wyciągnąć następujące wnioski:

1. Postać zdenormalizowana jest szybsza w przypadku zapytań, w których nie występuje zagnieżdżenie skorelowane, szczególnie w SQLite. W MSSQL różnice między 3 ZL a 4 ZL są jednak zazwyczaj rzędu 10-4 sekundy, więc nie jest to rozstrzygająca wartość.
2. W każdym DBMS wartość minimalna oraz średnia są zbliżone, co świadczy o stabilności przeprowadzanych prób.
3. Czas wykonywania zapytań w SQLite jest od 3 do 10 razy wolniejszy niż w przypadku MSSQL.
4. Zapytania wykonywane w dialekcie MSSQL w Server Management Studio miały krótszy czas wykonania niż te w Data Grip. Przyczyny takiego stanu rzeczy można się doszukiwać w dedykacji produktu Microsoftu do tego dialektu, natomiast produkt JetBrains zapewnia wsparcie wielu dialektom, co może spowalniać czas wykonania (średnio o 2 razy wolniej).
5. Nałożenie indeksów dało większe efekty w SQLite, czas został widocznie skrócony. W MSSQL również widoczne jest skrócenie czasu zapytań, jednak nie tak znaczące. Wynika to z i tak już krótkich czasów wykonywania.

Podsumowując, uzyskane wnioski są zbliżone do tych wyciągniętych we wzorcowym artykule. Normalizacja zazwyczaj prowadzi do spadku wydajności, choć spadek ten jest mniej widoczny niż wówczas. Warto jednak wspomnieć, że dane w postaci znormalizowanej umożliwiają łatwą konserwację, rozwój schematu oraz porządek, jaki owa postać wprowadza.

Autor:

Mateusz Zygmunt

Nr indeksu: 405685