WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH NA PODSTAWIE SYSTEMU ZARZĄDZANIA BAZAMI SQLServer I PostgreSQL

Poniższa praca stanowi rozważania nad wydajnością systemów zarządzania bazami danych na podstawie zastosowania znormalizowanych oraz nieznormalizowanych baz danych. Jako przykład zastosowano tabela geochronologiczną.

Szymon Górka

WGGiOŚ Geoinformatyka

1. Wprowadzenie

Bazy danych są obecnie bardzo popularną dziedziną informatyki, z której korzystają wszystkie gałęzie gospodarki. Z czasem popyt na dane staje się coraz większy, a co za tym idzie same bazy nabierają ogromnych rozmiarów, żeby móc sobie poradzić z tak ogromną dawką informacji powstało wiele programów do ich obróbki takich, jak: PostgreSQL, MySQL, Oracle SQL, MariaDB, Microsoft Access czy SQL server. Każda z nich ma różne wady i zalety, jedne potrafią szybciej pracować na bazach znormalizowanych o dużej objętości, inne zupełnie inaczej. Programy te za pomocą prostych złączeń i zapytań potrafią w szybki sposób znaleźć potrzebne informację. W celu usystematyzowania przechowywania baz oraz zredukowania potrzebnego miejsca na nośniku pamięci zaczęto rozbijać jednotablicowe bazy nieznormalizowane na mniejsze, które można później połączyć za pomocą wyżej wymienionych metod przy użyciu tak zwanych kluczy głównych oraz obcych, niestety wiąże się to z faktem, iż te operację potrzebują więcej czasu na połączenie tabel, dlatego czasami naumyślne zostawia się bazę zdemoralizowaną. Najważniejszym założeniem normalizacji baz jest zmniejszenie ilości danych oraz uproszczenie ich przy jednoczesnym zachowaniu wszystkich informacji. Początkowo zaproponowano 3 postacie normalne baz.

- 1NF – Pierwsza postać normalna. Jej jedynym warunkiem jest zachowanie atomowości w każdej krotce. By baza była znormalizowana, zawsze musi być w co najmniej pierwszej postaci normalnej.
- 2NF – Druga postać normalna. Możemy o niej mówić wtedy, gdy baza jest w pierwszej postaci oraz żadna kolumna nie jest funkcyjnie zależna od klucza głównego.
- 3NF – Trzecia postać normalna. Baza jest w 2 postaci normalnej oraz żaden atrybut niekluczowy nie jest funkcyjnie zależny od innego atrybutu niekluczowego.

Obecnie używa się także postaci normalnych BCNF, 4NF, 5NF, 6NF, lecz uważa, się że 3NF wystarczająca dla większości projektów. Pełne znormalizowanie jest zalecane by wykryć brak spójności.

2. Tabela geochronologiczna

Jest ona podstawowym narzędziem do określania wieku warstw skalnych oraz przebieg historii ziemi, jest ona podzielona na m.in. na: eony, ery, okresy, epoki i piętra jak przedstawiono w tabeli poniżej.

Tabela 1: Tabela geochronologiczna

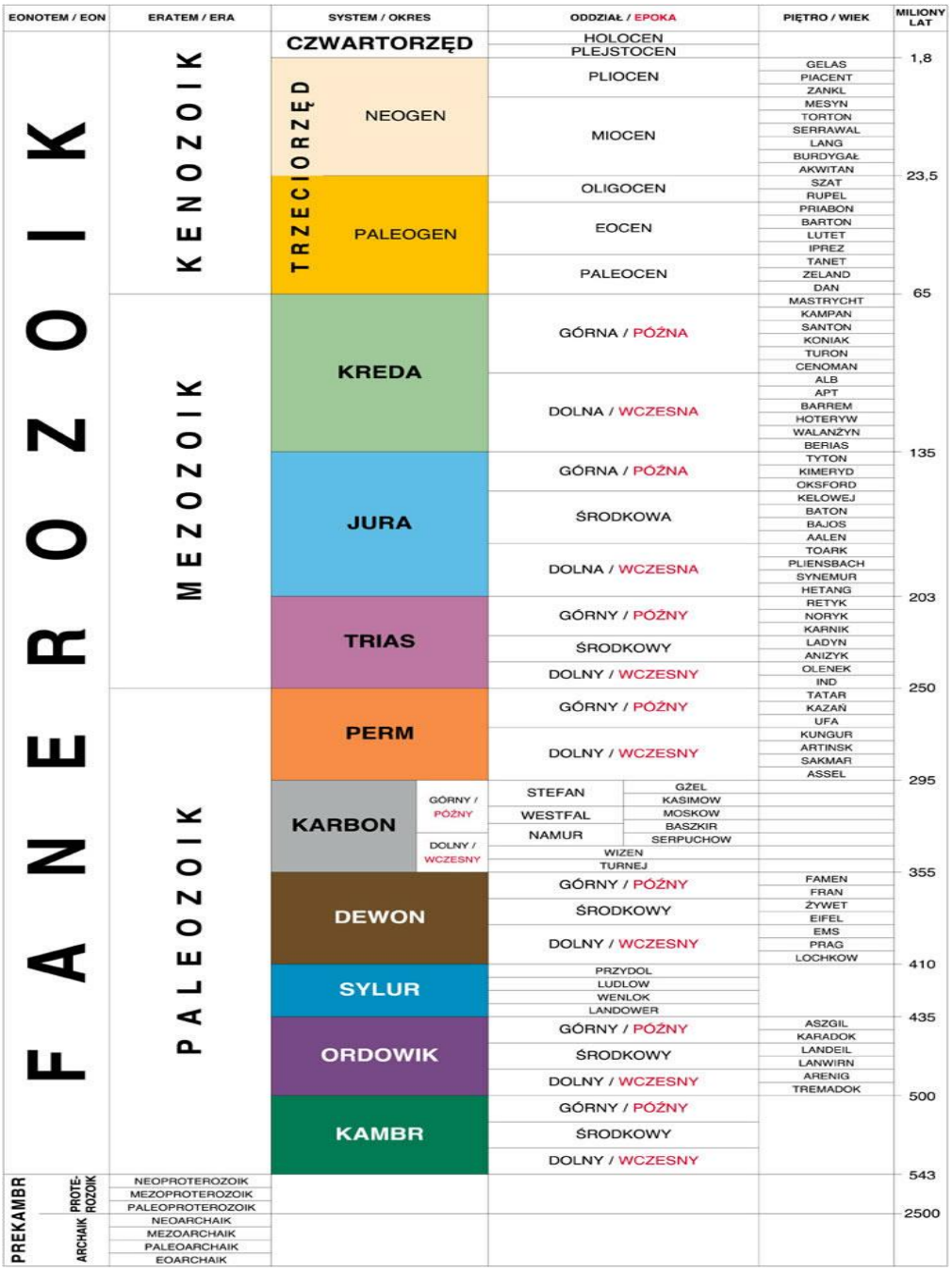
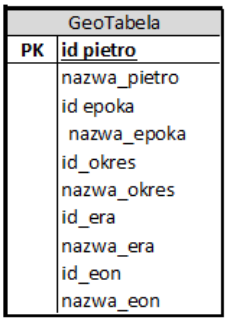


Tabela 1 jest zdenormalizowanym sposobem przedstawiania tabeli geochronologicznej, można ją także przedstawić za pomocą wersji znormalizowanej, jaka jest przedstawiona w schemacie poniżej:

Rys. 1. Znormalizowany schemat tabeli geochronologicznej

Przykładowy kod stworzenia jednej z tabel z rys. 1:

```
CREATE TABLE Tabela_strato.GeoEra (
id_era int  NOT NULL,
id_eon INT,
nazwa_era varchar(50),
PRIMARY KEY(id_era),
CONSTRAINT fk_GeoEon
FOREIGN KEY(id_eon)
REFERENCES Tabela_strato.GeoEon(id_eon)
);
```



Rys. 2. Zdenormalizowany schemat tabeli geochronologicznej

Powyższe rysunki przedstawiają model w schemacie płatka śniegu (rys. 1) oraz w schemacie gwiazdy (rys.2)

Po utworzeniu formy znormalizowanej możemy stworzyć za jej pomocą formę zdenormalizowaną za pomocą złączenia naturalnego:

```
CREATE TABLE GeoTabela AS
(SELECT * FROM Tabela_strato.GeoPietro
NATURAL JOIN Tabela_strato.GeoEpoka
NATURAL JOIN Tabela_strato.GeoOkres
NATURAL JOIN Tabela_strato.GeoEra
NATURAL JOIN Tabela_strato.GeoEon );
```

Za pomocą tabeli GeoTabela możliwy jest szybki dostęp do wszystkich danych tabeli za pomocą zapytania prostego, co nie jest możliwe w schemacie znormalizowanym.

3. Testy wydajności

W testach porównano wydajność złączeń i zapytań na dwóch różnych maszynach identycznych na tym samym komputerze. Przetestowane aplikacje to:

- PostgreSQL
- SQL Server

W celu przeprowadzenia badań stworzono dodatkowe tabele *Dziesięć* i *Milion* za pomocą następujących komend:

- *Dziesięć*

```
CREATE TABLE Dziesiec(
cyfra int,
bit int);
```

Tabele wypełniono cyframi od 0 do 9;

-*Milion*

```
CREATE TABLE Milion(
liczba int,
cyfra int,
bit int);

INSERT INTO Milion
SELECT a1.cyfra +10* a2.cyfra +100*a3.cyfra + 1000*a4.cyfra + 10000*a5.cyfra +10000*a6.cyfra AS
liczba ,
 a1.cyfra AS cyfra, a1.bit AS bit
FROM Dziesiec a1, Dziesiec a2, Dziesiec a3, Dziesiec a4, Dziesiec a5, Dziesiec a6 ;
```

Tabela ta zawiera liczby od 0 do 999 999.

3.1 Konfiguracja sprzętowa

Wszystkie testy zostały przeprowadzone na jednym komputerze o następujących parametrach:

- Procesor: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
- Zainstalowana pamięć RAM: 16,0 GB (dostępne: 15,9 GB)
- Wersja Windows: 11 Home
- Wersja:21H2

Oraz skorzystano z oprogramowania:

- PostgreSQL 6.4
- SQL Server Management Studio 15.0.18410.0

3.2 Kryteria testów

Test został podzielony na dwa etapy:

- W pierwszym za pomocą niżej opisanych zapytań została sprawdzona różnica w działaniach między schematem znormalizowanym a nieznormalizowanym (tabele zawierały indeksowane klucze główne),
- W kolejnym etapie zostały nałożone indeksy na wszystkie kolumny biorące w teście.

Zasadniczym celem testów była ocena wpływu normalizacji na zapytania złożone – złączenia i zagnieżdżenia. W tym celu zaproponowano cztery zapytania:

- Zap. 1 (1 ZL), którego celem jest złączenie tablicy *Milion* z tablicą geochronologiczną w postaci zdenormalizowanej, do złączenia dodano warunek modulo, który dopasowuje wartość złączanych kolumn:

```
SELECT COUNT(*) FROM Milion
```

```
INNER JOIN GeoTabela
ON (mod(Milion.liczba,68)=(GeoTabela.id_pietro));
```

- Zap. 2 (2 ZL), którego celem jest złączenie tablicy *Milion* z tabelą geochronologiczną w postaci znormalizowanej, reprezentowaną przez złączenia pięciu tabel:

```
SELECT COUNT(*) FROM Milion
INNER JOIN Tabela_strato.GeoPietro
ON (mod(Milion.liczba,68)=Tabela_strato.GeoPietro.id_pietro)
NATURAL JOIN Tabela_strato.GeoEpoka
NATURAL JOIN Tabela_strato.GeoOkres
NATURAL JOIN Tabela_strato.GeoEra
NATURAL JOIN Tabela_strato.GeoEon;
```

- Zap. 3 (3 ZG), którego celem jest złączenie tablicy *Milion* z tabelą geochronologiczną w postaci zdenormalizowanej, przy czym złączenie jest wykonywane poprzez zagnieżdżenie skorelowane:

```
SELECT COUNT(*) FROM Milion
WHERE mod(Milion.liczba,68)=
        (SELECT id_pietro FROM GeoTabela
        WHERE mod(Milion.liczba,68)=(id_pietro));
```

- Zap. 4 (4 ZG), którego celem jest złączenie tablicy *Milion* 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 Milion
WHERE mod(Milion.liczba,68) IN (SELECT id_pietro
        FROM Tabela_strato.GeoPietro
        NATURAL JOIN Tabela_strato.GeoEpoka
        NATURAL JOIN Tabela_strato.GeoOkres
        NATURAL JOIN Tabela_strato.GeoEra
        NATURAL JOIN Tabela_strato.GeoEon);
```

4. Wyniki testów

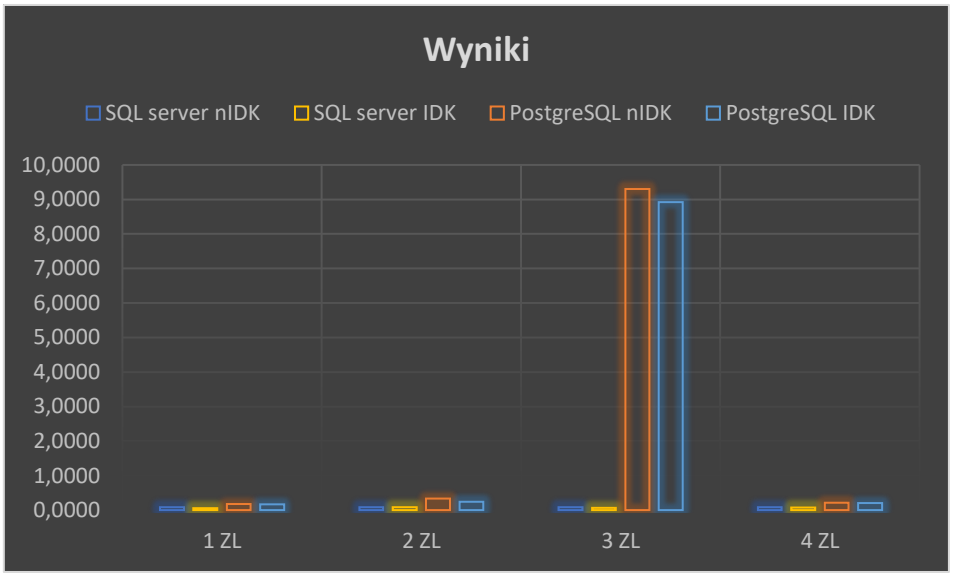
Każdy z testów został przeprowadzony kilkukrotnie, wartości skrajne zostały odrzucone. Wyniki testu zostały zamieszczone w tabeli 2.

Tabela 2: Czasy wykonania zapytań 1 ZL, 2 ZL, 3 ZG i 4 ZG [s]

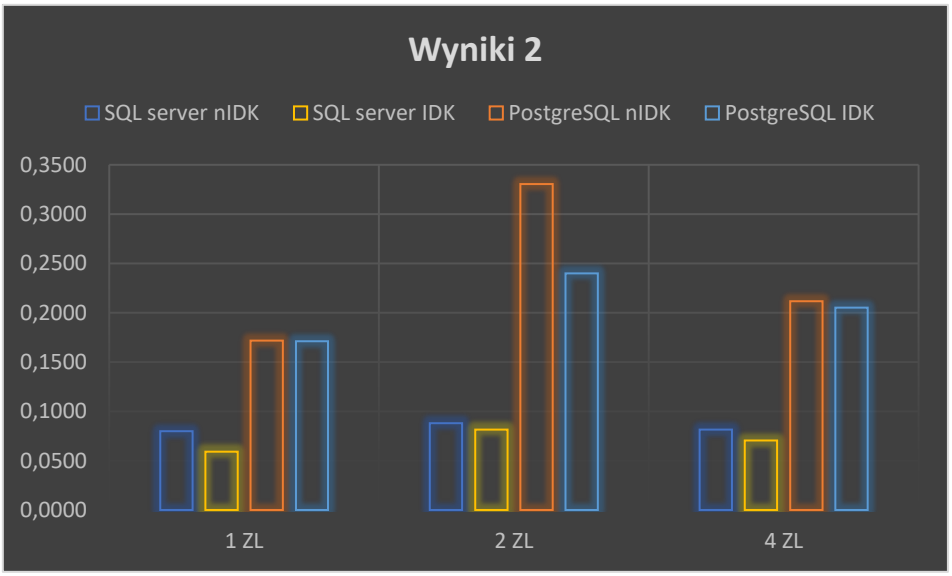
	1 ZL		2 ZL		3 ZL		4 ZL	
BEZ INDEKSÓW	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**
SQL server	0,0720	0,0800	0,0800	0,0880	0,0720	0,0816	0,0800	0,0816
PostgreSQL	0,1580	0,1718	0,3130	0,3304	9,0850	9,3034	0,1640	0,2116
Z INDEKSAMI								
SQL server	0,0560	0,0592	0,0640	0,0816	0,0640	0,0656	0,0640	0,0704
PostgreSQL	0,1530	0,1710	0,2320	0,2400	8,6890	8,9194	0,1850	0,2052

W celu ułatwienia analizy sporządzono dwa wykresy jeden dla wszystkich zapytań a drugi z pominięciem zapytania 3 który odstaje od pozostałych, aby ułatwić porównywanie niskich wartości.

Rys. 3: Wykres całościowy



Rys. 4: Wykres szczegółowy



5. Wnioski

Otrzymane wyniki pozwalają wyciągnąć następujące wnioski związane z tezą artykułu:

- Postać zdenormalizowana jest w większości przypadków wydajniejsza.
- W przypadku zagnieżdżenia skorelowanego dla programu PostgreSQL nie opłaca się używać zdenormalozowanej tablicy ponieważ długość zapytania jest kilkunastokrotnie większy.
- Jedynym przypadkiem, gdy w programie SQLServer mamy do czynienia z takim samym czasem zapytania dla tablicy znormalizowanej co zdenormalizowanej jest przypadek złączenia skorelowanego.

Testy pozwalają też przedstawić dodatkowe spostrzeżenia związane z rozważanym przypadkiem:

- Zagnieżdżenia skorelowane są dużo wolniejsze w wykonaniu niż zwykłe złączenia.
- Użycie indeksów we wszystkich rozważanych przypadkach przyśpieszyło wykonanie zapytań, lecz najlepiej widoczne jest to dla zapytań z tablicą geochronologiczną w postaci zdenormalizowanej.
- SQL Server poradził sobie znacznie lepiej w każdym badanym kryterium.
- PostgreSQL ma zbliżony czas działania dla tablic znormalizowanych niezależnie od występowania indeksów .
- PostgreSQL nie radzi sobie z złączeniem poprzez zagnieżdżenie skorelowane dla tabeli zdenormalizowanej.

Podsumowaniem rozważań jest wniosek, iż normalizacja w większości przypadków prowadzi do spadku wydajności, ale warto jest tu przypomnieć jej zalety, a mianowicie łatwą konserwację, rozwój schematu oraz porządek, jaki ona wprowadza.

Praca powstała na podstawie artykułu Łukasza JAJEŚNICA oraz Adama PIÓRKOWSKI o tytule „Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych”.