STUDIA INFORMATICA 2010
Volume 31 Number 2A (89)

Łukasz JAJEŚNICA, Adam PIÓRKOWSKI
Akademia Górniczo – Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH[1]

Streszczenie. Niniejszy artykuł stanowi rozważania na temat wydajnościowych aspektów normalizacji schematów baz danych. Przedstawiono ideę konstrukcji dużych baz danych i hurtowni w postaciach znormalizowanej (schemat płatka śniegu) i zdenormalizowanej (schemat gwiazdy). Jako przykład wybrano wymiar czasu geologicznego, nieodzownego w konstrukcji baz danych geologicznych. Wymiar ten tworzy hierarchię jednostek geochronologicznych. Przeprowadzono eksperymenty sprawdzające wydajność dla złączeń i zagnieżdżeń skorelowanych dla systemów zarządzania bazami danych MySQL i PostgreSQL.

Słowa kluczowe: postać normalna, normalizacja, schemat gwiazdy, schemat płatka śniegu, tabela geochronologiczna, złączenia, zagnieżdżenia skorelowane

PRODUCTIVITY AND NESTING JOIN THE SCHEMES AND STANDARDIZED AND DENORMALIZED

Summary. This article provides a reflection on the performance aspects of database normalization schemas. It resents the idea of the construction of large databases and warehouses on standardized forms (snowflake schema) and denormalized (star schema). Dimension was chosen as an example of geological time, indispensable in the construction of geological database. This dimension creates a hierarchy of chronostratigraphy units. Experiments were carried out to check the performance of the joints and nesting correlated to the database management system MySQL and PostgreSQL.

Keywords: normal form, normalization, star schema, snowflake schema, chronostratigraphy, join, correlated nesting Umieszczone tu pole Advance zapewnia poprawny odstêp

[1] Praca finansowana w ramach badań statutowych KGIS nr 11.11.140.561.

1. Wprowadzenie

Bazy danych są dziś powszechnie stosowane we wszystkich dziedzinach życia. Trudno sobie wyobrazić działanie bardziej złożonych aplikacji bez baz danych, których zadaniem jest przechowywanie i inteligentne zarządzanie gromadzonymi informacjami. W obecnych czasach bazy danych zawierają ogromne ilości informacji, a uzyskiwanie wyników zapytań kierowanych do bazy musi odbywać się w krótkim czasie, dlatego też nieodzownym elementem prócz dobrze przemyślanego projektu schematu bazy danych jest również jej optymalizacja pod kątem danych, jakie będą w niej przechowywane oraz pod kontem obciążenia generowanego przez użytkowników korzystających z usług bazy.

Celem uzyskania wymaganych wyników zapytania kierowane do bazy danych muszą opierać się na złączeniach wielu tabel przy odpowiednich warunkach ich kojarzenia. Wykonania tej operacji można dokonać na dwa sposoby:

- poprzez złączenia (np. wewnętrzne) tabel,
- poprzez zapytania zagnieżdżone nieskorelowane lub skorelowane.

Złączenia tabel są zazwyczaj efektywnie wykonywane przez systemy zarządzania bazami danych, natomiast zapytania zagnieżdżone, szczególnie skorelowane, są zazwyczaj bardzo trudne w optymalizacji, co jest związane z faktem, iż dla każdej krotki z tabeli w zapytaniu zewnętrznym musi być każdorazowo wykonane zapytanie wewnętrzne. Taka sytuacja nie występuje w przypadku zapytań zagnieżdżonych nieskorelowanych.

1.1. Normalizacja schematów tabel w bazach danych

Normalizacja bazy danych to proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych. Istnieją sposoby ustalenia, czy dany schemat bazy danych jest znormalizowany i jeżeli jest – to w jakim stopniu. Jednym ze sposobów jest transformowanie danej bazy do schematów zwanych postaciami normalnymi (ang. *normal forms*, NF) [1-4]. Normalizacja bazy danych do konkretnej postaci może wymagać rozbicia dużych tabel na mniejsze i przy każdym wykonywaniu zapytania do bazy danych ponownego ich łączenia. Zmniejsza to wydajność, dlatego w niektórych przypadkach świadoma denormalizacja (stan bez normalizacji) jest lepsza, jak np. w systemach OLAP [5].

Normalizacja nie usuwa danych, tylko zmienia schemat bazy danych, przeprowadza bazę danych z jednego stanu spójnego (przed normalizacją) w inny stan spójny (po normalizacji). Jedyna różnica polega na innym układzie danych i relacji pomiędzy nimi, ale bez utraty danych (ewentualnie dodawane są nowe klucze główne).

Edgar Frank Codd (prekursor normalizacji) początkowo zaproponował 3 postacie formalne [1,2]:

- 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.
- 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.
- 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.

Obecnie istnieją jeszcze inne postacie, ale 3NF jest powszechnie uznawana za wystarczającą dla większości projektów. Część tabel spełniając postać 3NF spełnia także BCNF (ang. *Boyce-Codd Normal Form*). 4NF i 5NF są następnymi rozszerzeniami, a 6NF jest używana do baz uwzględniających w modelu relacyjnym wymiar czasowy. Normalizacja według postaci powyżej 3NF może być skomplikowana ze względu na SQL, lecz brak normalizacji lub niepełna normalizacja narażają bazę danych na problemy uszkodzenia danych (anomalie). Pełne znormalizowanie, nawet gdy nie jest wspierane przez technologię, jest warte rozważenia, gdyż pomaga wykryć wszystkie potencjalne braki spójności.

1.2. Schematy hurtowni danych

Hurtownia danych jest centralnie zarządzaną i zintegrowaną bazą danych, gromadzącą dane z systemów i źródeł operacyjnych organizacji (takich jak systemy SAP, CRM, ERP).

Hurtownia danych może również zawierać ręcznie wprowadzane mapowania, parametry bądź kryteria w celu klasyfikacji danych. Baza, na której oparta jest hurtownia danych, zawiera dane będące źródłem analiz i do których użytkownicy mogą mieć bezpośredni dostęp. Hurtownia danych może być uaktualniana o dowolnym czasie, bez konieczności wyłączania systemów operacyjnych.

W artykule skupiono się na omówieniu dwóch najpopularniejszych schematów hurtowni baz danych: schemacie gwiazdy (ang. *star schema*) oraz schemacie płatka śniegu (ang. *snowflake schema*).

Schemat gwiazdy

Jest najprostszym modelem projektu bazy danych w hurtowni danych. Główną cechą schematu gwiazdy jest centralna tabela, nazywana tabelą faktów, z którą połączone są tabele wymiarów. Model taki umożliwia przeglądanie poszczególnych kategorii, operacje agregacji, czy też zaawansowane drążenie i filtrowanie danych.

W modelu gwiazdy tabela faktów jest w trzeciej postaci normalnej, podczas gdy tabele wymiarów są zdenormalizowane, czyli reprezentują drugą postać normalną. Tabela faktów jest zaprojektowana zupełnie inaczej niż typowa tabela relacyjnej bazy danych. Jej denormalizacja jest celowa i służy zwiększeniu wydajności i zmniejszeniu czasu wykonywania zapytań kierowanych do bazy. Tabela faktów zwykle zawiera rekordy gotowe do eksploracji, zwykle łatwo dostępne dla zapytań pisanych 'od ręki' (ad hoc). Rekordy w tabeli faktów mogą być postrzegane jako wydarzenia, co jest spowodowane naturą hurtowni danych. Klucz główny tabeli faktów jest zwykle złożony z wszystkich kolumn z wyjątkiem wartości numerycznych, reprezentujących miary.

Prawie wszystkie informacje w typowej tabeli faktów są reprezentowane również w jednej lub wielu tabelach wymiarów. Głównym celem utrzymywania tabeli wymiarów jest umożliwienie przeglądania kategorii prosto i szybko.

Klucz główny każdej z tabeli wymiarów jest związany z tabelą faktów i jest to składowa złożonego klucza głównego tabeli faktów. W schemacie gwiazdy występuje tylko jedna zdenormalizowana tabela dla każdego z wymiarów.

Schemat płatka śniegu

Jest bardziej złożoną wersją schematu gwiazdy. Główną różnicą między nimi stanowi fakt, że w schemacie płatka śniegu tabele wymiarów są znormalizowane, czyli są zaprojektowane zgodnie z modelem relacyjnej bazy danych. Schemat płatka śniegu jest używany przede wszystkim wtedy, gdy tabela wymiarów osiąga duże rozmiary i w schemacie gwiazdy jest trudno przedstawić kompleksowość takiej struktury danych.

2. Tabela geochronologiczna

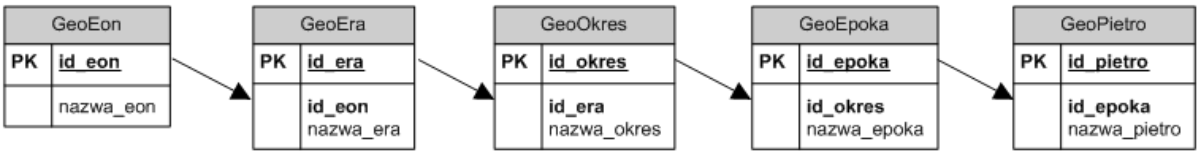
Tabela geochronologiczna obrazuje przebieg historii Ziemi na podstawie następstwa procesów i warstw skalnych [6,7]. Obecnie przyjęta tabela geochronologiczna została ustalona przez Międzynarodową Komisję Stratygrafii (ICS). W tabeli 1 przedstawiono taksonomię dla czterech jednostek geochronologicznych – eonu, ery, okresu i epoki. Wymiar ostatniej jednostki – piętra został pominięty ze względu na obszerność (68 elementów).

Tabela 1

Tabela geochronologiczna

Wiek (mln lat)	Eon	Era	Okres		Epoka
0,010	**FANEROZOIK**	**Kenzoik**	Czwar-torząd		Halocen
1,8					Plejstocen
22,5			Trzeciorząd	Neo-gen	Pliocen
					Miocen
65				Paleogen	Oligocen
					Eocen
					Paleocen
140		**Mezozoik**	Kreda		Górna
					Dolna
195			Jura		Górna
					Środkowa
					Dolna
230			Trias		Górna
					Środkowa
					Dolna
280		**Paleozoik**	Perm		Górny
					Dolny
345			Karbon		Górny
					Dolny
395			Dewon		Górny
					Środkowy
					Dolny

Tabela ta jest podstawowym narzędziem pracy geologów i paleontologów, jednoznacznie definiuje oficjalną terminologię okresów geologicznych w historii Ziemi, dzięki czemu unika się stosowania tych samych nazw w różnych znaczeniach w publikacjach naukowych. W różnych opracowaniach można spotkać różne podziały stratygraficzne z powodu ciągle trwających badań i typowania nowych profili wzorcowych (stratotypów).

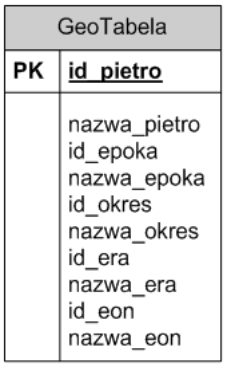


Rys. 1. Znormalizowany schemat tabeli geochronologicznej
Fig. 1. Standardized schema of the geochronologic table

3. Konstrukcja wymiaru geochronologicznego

W tabeli geochronologicznej znalazły miejsce jednostki geochronologiczne mające wymiar czasowy (eon, era, okres, epoka i wiek) oraz odpowiadające im jednostki stratygraficzne. Dane te tworzą wymiar, który występuje w wielu bazach danych geologicznych [8, 9, 10, 11, 12 i 13]. W niniejszym opracowaniu skupiono się na konstrukcji tabeli geochronologicznej w dwóch przypadkach:

- schemacie znormalizowanym (płatka śniegu, rys. 1),
- schemacie zdenormalizowanym (schemat gwiazdy).



Rys. 2. Zdenormalizowany schemat tabeli geochronologicznej
Fig. 2. Denormalized schema of the geochronologic table

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

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

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.

4. Testy wydajności

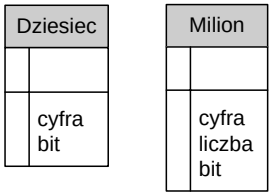
W testach skupiono się na porównaniu wydajności złączeń oraz zapytań zagnieżdżonych, wykonywanych na tabelach o dużej liczbie danych. Testy wykonano na dwóch różnych maszynach identycznych pod względem konfiguracji sprzętowej oraz oprogramowania. Przetestowano najpopularniejsze darmowe rozwiązania bazodanowe:

- MySQL,
- PostgreSQL.

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 [14]:

```
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 ;
```

Schematy tabel przedstawiono na rysunku 3.



Rys. 3. Schemat tabel Dziesiec i Milion
Fig. 3. Schema of the tables Dziesiec (Ten) and Milion (Milion)

4.1. Konfiguracja sprzętowa i programowa

Wszystkie testy omówione w niniejszym artykule wykonano na dwóch symetrycznych komputerach o następujących parametrach:

- CPU: Intel Pentium 4 Hyper-Threading Technology 2,8 GHz,
- RAM: Pamięć DDR1 1024 MB (533 MHZ),
- HDD: Seagate 7200 rpm, 1 MB cache,
- S.O.: Windows 2000.

Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne:

- MySQL, wersja Community Server 5.1.42,
- PostgreSQL, wersja 8.4.2-1.

Testy wykonywano wielokrotnie na każdym komputerze dla każdego systemu zarządzania bazą danych, przy czym w trakcie testów na danym komputerze zainstalowany był tylko jeden z nich, kolejność instalacji była krzyżowa.

4.2. 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:

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

Zasadniczym celem testów była ocena wpływu normalizacji na zapytania złożone – złączenia i zagnieżdżenia (skorelowane) [14]. W tym celu zaproponowano cztery zapytania:

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

- 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 Milion INNER JOIN  GeoPietro  ON
(mod(Milion.liczba,68)=GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

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

- 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 Milion WHERE mod(Milion.liczba,68)=
(SELECT GeoPietro.id_pietro FROM GeoPietro NATURAL JOIN GeoEpoka NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

5. Wyniki testów

Każdy test przeprowadzono wielokrotnie, wyniki skrajne pominięto. W przypadku systemu MySQL miała miejsce dość duża zgodność kolejnych prób, także w przypadku obydwu komputerów testowych. Wynika to z determinizmu, jakim chwalą się konstruktorzy bazy [15]; niestety, jest to też związane z brakiem dynamicznych algorytmów optymalizacji. Wyniki testów zamieszczono w tabeli 2.

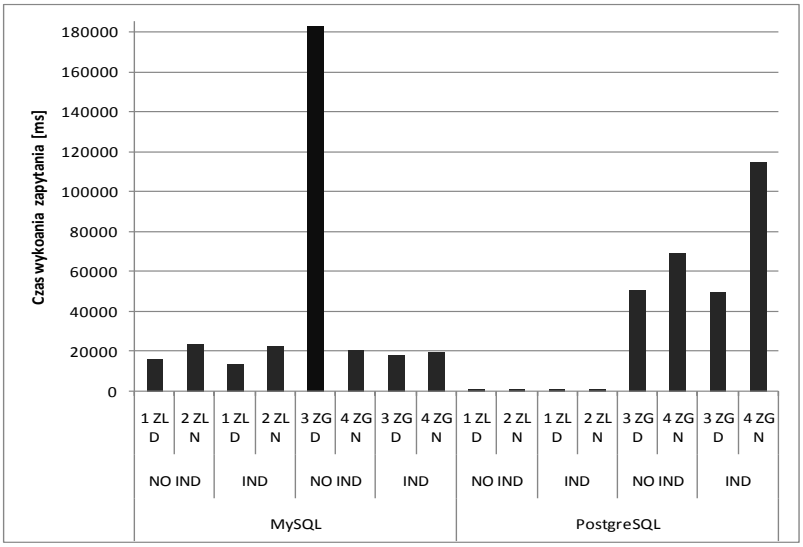
Tabela 2

Czasy wykonania zapytań 1 ZL, 2 ZL, 3 ZG i 4 ZG [ms]

	1 ZL		2 ZL		3 ZG		4 ZG	
BEZ INDEKSÓW	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**	**MIN**	**ŚR**
MySQL	15920	16041	23800	23863	182560	182982	20480	20493
PostgreSQL	1312	1333	1312	1330	50234	50534	68828	69055
Z INDEKSAMI								
MySQL	13310	13398	22410	22565	18220	18262	19250	19340
PostgreSQL	1328	1336	1313	1333	49922	50284	114672	114821

Analizę wyników ułatwiają wykresy (rys. 4 i 5) – ze względu na dość duże wartości pojedynczych przypadków rozważono dwie wersje – pełna skala liniowa i skala liniowa częścio-

wa (aby ułatwić porównanie niskich wartości). Wyniki zestawiono wysuwając na pierwszy plan związki z tezą artykułu – czy wersja znormalizowana (N) jest wolniejsza czy szybsza od wersji zdenormalizowanej (D).



Rys. 4. Wyniki testów w ujęciu celu normalizacji
Fig. 4. View of the impact of normalization

6. Wnioski

Otrzymane wyniki pozwalają wyciągnąć następujące wnioski związane z tezą artykułu:

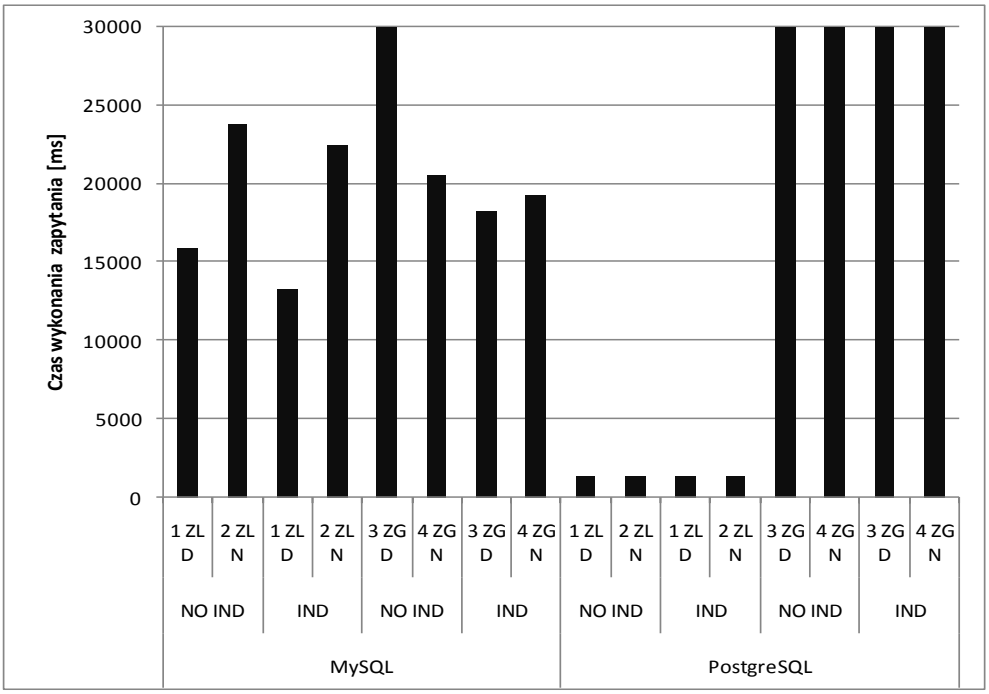
- Postać zdenormalizowana jest w większości przypadków wydajniejsza.
- Jedyny przypadek, kiedy postać znormalizowana jest szybsza, to zagnieżdżenie skorelowane – w podzapytaniu wewnętrznym w wersji zdenormalizowanej dokonywany jest odczyt dużej tabeli danych niezaindeksowanych; wydłużenie czasu wykonania ma miejsce w systemie MySQL, PostgreSQL, bowiem dokonuje skutecznej optymalizacji takiego zapytania.

Testy pozwalają też przedstawić dodatkowe spostrzeżenia związane z rozważanym przypadkiem:

- Zagnieżdżenia skorelowane są dużo wolniejsze w wykonaniu niż złączenia.
- Użycie indeksów w systemie MySQL we wszystkich rozważanych przypadkach przyśpiesza wykonanie zapytań, zarówno złączeń, jak i zagnieżdżeń skorelowanych.
- System PostgreSQL dokonuje analizy tabeli i indeksacja nie przyspieszyła (ani nie spowolniła) wykonania przedstawionych złączeń i zagnieżdżenia 3 ZG, jedynie dla zapytania 4 ZG użycie indeksów wydłużyło czasy zapytań.
- Złączenia w PostgreSQL są tak optymalizowane, iż zapytanie składające się z samych złączeń wykonuje się równie szybko dla postaci znormalizowanej, jak i zdenormali-

zowanej, różnica (i to istotna) pozostaje w przypadku wystąpienia złączeń jako wewnętrznego podzapytania zapytania skorelowanego, które jest często wykonywane.

- Systemów zarządzania bazami danych nie da się porównać jednoznacznie – o ile złączenia najszybciej były wykonane przez system PostgreSQL, to zagnieżdżenia w większości przypadków były szybciej przetworzone przez MySQL.



Rys. 5. Wyniki testów w ujęciu celu normalizacji dla niższych wartości
Fig. 5. View of the impact of normalization for the lower values

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.

BIBLIOGRAFIA

1. Codd, E. F.: Further Normalization of the Data Base Relational Model. IBM Research Report, San Jose, California RJ909, 1971.
2. Codd, E. F.: Recent Investigations into Relational Data Base Systems. IBM Research Report RJ1385, 1974.
3. Fagin R.: Multivalued Dependencies and a New Normal Form for Relational Databases. ACM Transactions on Database Systems Vol. 2(1), 1977, s. 267.
4. Fagin R.: A Normal Form for Relational Databases That Is Based on Domains and Keys. Communications of the ACM Vol. 6, 1981, s. 387÷415

5. Gorawski M.: Ocena efektywności architektur OLAP. Problemy i metody inżynierii oprogramowania. Red. Z. Huzar, Z. Mazur. Wydawnictwo Naukowo-Techniczne, Warszawa 2003, s. 233÷250.
6. Waksmundzki B.: Historia Ziemi. Wydawnictwo Naukowe PWN, Warszawa, 2004.
7. Stanley S. M.: Historia Ziemi. Wydawnictwo PWN. Warszawa 2008.
8. Kotlarczyk J., Krawczyk A., Leśniak T., Słomka T.: Geologiczna baza danych GeoKarpaty dla polskich Karpat fliszowych. Wydawnictwo własne WGGiOŚ AGH, Kraków 1997.
9. Piórkowski A., Gajda G.: Konstrukcja wielowymiarowej bazy danych geologicznych. ZN Pol. Śl. Studia Informatica Vol. 30, No. 2B, Gliwice 2009, s. 179÷190.
10. Onderka Z., Piórkowski A. : Projekt i implementacja geologicznej bazy danych w sieci Internet. Nowe technologie sieci komputerowych – praca zbiorowa. Wydawnictwa Komunikacji i Łączności, Warszawa 2006.
11. Valenta M., Siwik L.: Geo-zagrożenia – komputerowy system ewidencji zagrożeń geodynamicznych w Polsce. Bazy danych – struktury, algorytmy, metody – wybrane technologie i zastosowania. Praca zbiorowa pod red. Stanisława Kozielskiego [et al.]. Wydawnictwa Komunikacji i Łączności, Warszawa 2006.
12. Gajda G., Piórkowski A.: Możliwości konstrukcji hurtowni danych geologicznych. Bazy danych – rozwój metod i technologii – bezpieczeństwo, wybrane technologie i zastosowania. Praca zbiorowa pod red. Stanisława Kozielskiego [et al.]. Wydawnictwa Komunikacji i Łączności, Warszawa 2008.
13. Malec O., Kyc M., Piórkowski A.: Internetowa baza danych odsłonięć GeoKarpaty II. Materiały kongresowe Pierwszego Polskiego Kongresu Geologicznego, Kraków 26–28 czerwca 2008. Polskie Towarzystwo Geologiczne, 2008.
14. Piórkowski A.: Bazy Danych II – materiały do wykładu.
15. MySQL 5.4 Reference Manual, www.mysql.com

Recenzent: Dr inż. Marcin Gorawski,
Dr inż. Aleksandra Werner

Wpłynęło do Redakcji 30 stycznia 2010 r.

Abstract

This article provides a reflection on the performance aspects of database normalization schemas. It presents the idea of the construction of large databases and warehouses on stan-

dardized forms called also snowflake schema (fig.1) and denormalized forms called star schema (fig. 2). Denormalized form of table was achieved by creating single table GeoTabela (fig. 2), containing all the data from tables. This was done by using the natural join including all tables creating a hierarchy. Creating a table GeoTabela allows quick access to all data with one simple question, which is not possible in the case of a standardized schema described in section first.

Experiments were carried out to check the performance of the joins and nesting correlated to the free database management system MySQL and PostgreSQL. Test procedure was carried out in two stages:

- the first stage involved imposed queries without indexes on the columns of data (data were only indexed in the columns by the main keys of individual tables),
- the second stage was imposed on the indices of all columns involved in the query.

The primary objective of the tests was to assess the impact of normalization on the complex question – join and nesting (correlated).

The obtained results allowed us to formulate the following conclusions:

- denormalised form is in most cases more efficient,
- the only case when the normalized form is faster is the implantation correlated (fig.4, 5).

In most cases, the normalization gives rise to a decrease in database performance, but allows for easy development and maintenance of the system, introduces a policy that can prevent data loss.

Adresy

Łukasz JAJEŚNICA: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska, lukasz.jajesnica@agh.edu.pl .
Adam PIÓRKOWSKI: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska, pioro@agh.edu.pl .