STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Radosław ZATOKA
Uniwersytet Ekonomiczny we Wrocławiu, Katedra Systemów Sztucznej Inteligencji

WYKORZYSTANIE DOKUMENTOWYCH BAZ DANYCH W APLIKACJACH INTERNETOWYCH

Streszczenie. W opracowaniu scharakteryzowano dokumentowe bazy danych w odniesieniu do popularnych relacyjnych i nierelacyjnych baz danych używanych w zastosowaniach internetowych. Przedstawiono propozycję wykorzystania dokumentowej bazy (MongoDB) do buforowania danych w architekturze MVC aplikacji webowej celem zwiększenia jej wydajności.

Słowa kluczowe: dokumentowe bazy danych, aplikacje internetowe, MongoDB

THE USE OF DOCUMENT-ORIENTED DATABASES IN WEB APPLICATIONS

Summary. This paper characterizes document-oriented databases with reference to relational and non-relational databases used in the development of Web applications. It presents the method of using document database (MongoDB) in the data tier of MVC architecture. Applying additional data cache can increase performance of Web systems.

Keywords: document databases, web applications, MongoDB

1. Ruch opensource'owych nierelacyjnych baz danych

Olbrzymie ilości danych, pod których obciążeniem pracują dzisiejsze aplikacje internetowe, sprowokowały w połowie pierwszej dekady XXI wieku rozpoczęcie dyskusji, czy relacyjne SZBD, oparte na koncepcjach pochodzących z czasów, kiedy nikt nie wyobrażał sobie takich wielkości, będą w najbliższej przyszłości w stanie podołać wydajnemu przetwarzaniu baz o rozmiarach wyrażanych w petabajtach.

Pojawienie się tego problemu i inspiracja pochodząca z projektów takich potentatów, jak Google (BigTable [2]) czy Amazon (Dynamo [3]) zrodziły ruch, którego celem jest przedstawienie i dostarczenie nowoczesnej opensource'owej technologii mogącej być alternatywą dla powszechnie używanych w typowych rozwiązaniach webowych MySQL i PostgreSQL lub komercyjnego MS SQL Server. W przygotowaniach do spotkania przedstawicieli społeczności skupiającej się wokół nierelacyjnych baz danych, które odbyło się w czerwcu 2009 roku na warsztatach poświęconych opensource'owym rozproszonym bazom danych [13], spopularyzowano nazwę *NoSQL* („*Not Only SQL*"). Pierwsze projekty kojarzone dziś z tym nurtem są związane z wielkimi spółkami internetowymi, takimi jak Facebook (Cassandra), Apache (HBase) i LinkedIn (Voldemort) [5, 6]. Aktualnie lista większych nierelacyjnych baz danych zawiera kilkadziesiąt pozycji [12].

2. Rodzaje nierelacyjnych baz danych

Poszczególne nierelacyjne bazy danych mogą znacząco różnić się od siebie. Na podstawie modeli danych wyróżnia się wśród nich 4 podstawowe kategorie (por. [11, 4]):

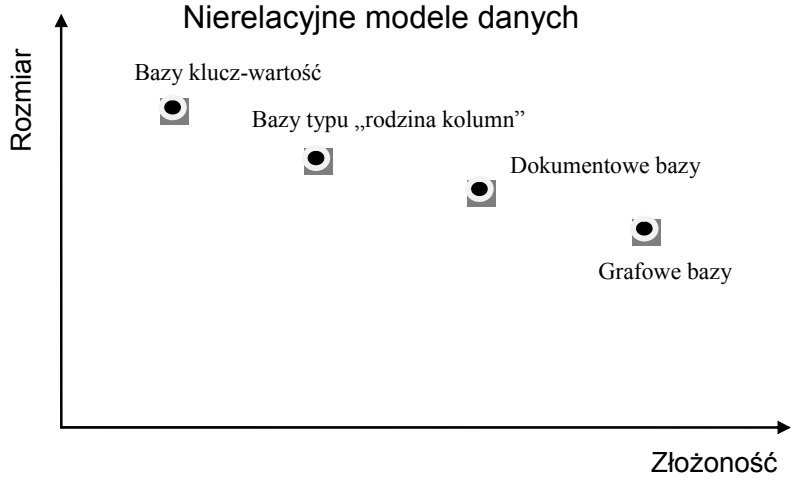
- Bazy typu klucz-wartość (ang. *Key-Value Stores*), ich model danych jest wykonany na podstawie podejścia opracowanego przez Amazon w implementacji Dynamo – przypominającej tablicę haszującą globalnej kolekcji par klucz-wartość (przykłady: Voldemort, Tokyo Cabinet, Dynomite).
- Bazy implementujące „rodzinę kolumn" (ang. „*Column Family*", *Wide Column Store*, *BigTable Clones*), opierają się na koncepcji Google'a „rodziny kolumn" – tabelarycznym modelu, w którym każdy wiersz może posiadać inną konfigurację kolumn (przykłady: Cassandra, Hypertable, HBase).
- Dokumentowe (dokumentowo zorientowane) bazy danych (ang. *Document-Oriented Databases*) realizują koncepcję znaną z Lotus Notes – baza jest zbiorem dokumentów składających się z par klucz-wartość (przykłady: MongoDB, CouchDB, Riak).
- Grafowe bazy danych (ang. *Graph Databases*), wywodzące się z teorii grafów modele, w których zarówno wierzchołki, jak i krawędzie mogą przechowywać parę klucz-wartość (przykłady: AllegroGraph, InfoGrid, Neo4j).

Główną przesłanką powstania *NoSQL* są problemy skalowalności relacyjnych baz danych, które wynikają z faktu, że nie były one projektowane do wydajnego przetwarzania ogromnych ilości danych w czasie rzeczywistym.

3. Skalowalność nierelacyjnych baz danych

Zwiększeniu wydajności *NoSQL* w porównaniu do relacyjnych baz danych ma służyć zapewnienie szybkich procesów replikacji, rozproszonego partycjonowania (uproszczonego m.in. brakiem konieczności złączeń tabel) oraz przetwarzania (wiele z nierelacyjnych baz natywnie realizuje *load balancing* na bazie modelu *MapReduce*).

Omawiane bazy danych nie są rozwiązaniami uniwersalnymi. Konsekwencją sposobów składowania danych jest uzyskanie określonych typów skalowalności: do rozmiarów zbioru danych albo do poziomu ich skomplikowania. Najmniejszy narzut modelu danych, a co za tym idzie, największą wydajność przetwarzania uzyskać można przy zastosowaniu baz danych klucz-wartość. Wtedy jednak cały kod obsługujący zależności występujące w danych musi zostać przeniesiony do logiki biznesowej aplikacji, co w przypadku skomplikowanych związków może okazać się nieopłacalne (por. rys. 1).



Rys. 1. Dwuwymiarowe porównanie skalowalności baz nierelacyjnych wykorzystujących różne modele danych. *Źródło: opr. na podst. [11]*
Fig. 1. Two-dimensional comparison of scalability of non-relational bases which use data models

Jak sama nazwa wskazuje, ruch „*Not Only SQL*” nie ma na celu zastąpienie technologii relacyjnej, ale jej uzupełnienie. W rozwiązaniach wymagających przeprowadzania częstych transakcji rezygnowanie z baz relacyjnych nie byłoby uzasadnione.

4. Charakterystyka dokumentowych bazy danych

Szczególnie interesującym pod względem możliwości zastosowania w typowych aplikacjach webowych rozwiązaniem z wyżej omawianych są dokumentowe bazy danych, których filozofię przechowywania danych można dopasować do sposobu prezentowania użytkownikowi informacji końcowej w postaci witryny internetowej – dokumentu HTML.

Koncepcja projektowania modelu opiera się na przechowywaniu połączonych danych w jednym dokumencie, w jak najbardziej zdenormalizowanej postaci. Idea polega na umieszczeniu razem wszystkich danych w jednym miejscu, tak jak znajdują się na kartkach papieru, w konwencjonalnym, biurowym raporcie.

Na poniższym listingu zaprezentowano przykład dokumentu reprezentującego artykuł z katalogu produktów części elektronicznych w MongoDB.

```
{
  art_partno: "BKN179C88",
  art_stock: 56000,
  art_price: 0.047,
  art_manufacturer:  "TVL",
  art_distributor: [
    { disti_name: "FVBX", disti_loc: "DE" },
    { disti_name: "RYG", disti_loc: "CZ" },
  ],
  art_quantity_columns: { art_qty1: 2500, art_qty2: 10000, art_qty3: 17500 } ,
  art_price_columns: { art_price1: 0.045, art_price1: 0.042, art_price3: 0.04 }
}
```

W dokumentowo zorientowanych bazach dane stają się kolekcjami dokumentów. Pojedynczy dokument przedstawia hierarchię danych dotyczącą określonego pojęcia z modelowanej rzeczywistości. Dokument nie posiada z góry zdefiniowanej struktury danych, jak występuje to w technologii relacyjnej. Może składać się z dowolnej ilości elementów, przechowywanych w formacie klucz-wartość, o całkowicie rożnej strukturze. W każdej chwili istnieje możliwość dodania do rekordu nowego pola o dowolnej długości. Kolekcje mogą być zagnieżdżane podobnie jak w paradygmacie obiektowym, tj. pole może zawierać nie tylko pojedynczą wartość, ale także cały zbiór – inną kolekcję. Dokument pobiera się na podstawie klucza głównego lub poprzez sformułowanie zapytania odnoszącego się do jego pól. Na polach dokumentu mogą być zakładane indeksy.

Różnice między relacyjnym a dokumentowym modelem danych zebrano w tabeli 1.

Tabela 1

Porównanie relacyjnej i dokumentowej bazy danych

Relacyjne bazy danych	**Dokumentowe bazy danych**
Predefiniowana struktura danych	Dynamiczna struktura danych
Jednakowe tabele	Kolekcje dokumentów o tej samej lub różnej strukturze
Dane znormalizowane Ograniczona nadmiarowość	Dane zdenormalizowane Występuje redundancja
Wymagana znajomość schematu dla operacji *read / write*	Wymagana nazwa dokumentu
Dynamiczne zapytania dla statycznej struktury	Statyczne zapytania dla dynamicznej struktury

Dwa najpopularniejsze obecnie dokumentowe rozwiązania to MongoDB i CouchDB. Na potrzeby artykułu zdecydowano się wykorzystać to pierwsze. Zdaniem autora, jest przyjaźniejsze dla użytkownika i łatwiejsze do wdrożenia w działających już aplikacjach (do komunikacji CouchDB dostarcza interfejs REST – API działające bezpośrednio na protokole

HTTP, podczas gdy MongoDB wykorzystuje natywne sterowniki dla określonego języka programowania).

5. MongoDB w aplikacjach internetowych

Od pojawienia się MongoDB w szybkim tempie rośnie liczba serwisów decydujących się na jej użycie w realizowaniu części swoich funkcjonalności. Ze znanych firm i projektów należy wyróżnić SourceForge.net (serwis hostujący ponad 2,7 mln opensource'owych projektów), Disqus Comments (usługę dostarczającą system komentarzy dla docelowej strony internetowej), producenta oprogramowania rozrywkowego Electronic Arts czy nawet portal *The New York Timesa*. Powodem zdobywania popularności jest dostosowanie produktu do wspomagania konkretnych problemów. Intencją twórców MongoDB jest stworzenie bazy, która może [10]:

- Pełnić funkcję operacyjnej składnicy danych na potrzeby stron internetowych. Zaletami MongoDB są bardzo szybko realizowane operacje zapisu, aktualizacji i pobierania rekordów.
- Pełnić rolę warstwy buforującej dane na potrzeby witryn internetowych. Szerszemu przedstawieniu tego przypadku poświęcono kolejny punkt artykułu.
- Być wykorzystana do przechowywania danych o dużej objętości, takich jak pliki video, pliki graficzne, itp. Eliminuje to potrzebę pisania dodatkowego kodu operującego na systemie plików.
- Sprawdzić się w warunkach silnego obciążenia. Duża skalowalność (wg twórców 64-bitowa wersja MongoDB jest w stanie obsługiwać dowolną ilość danych) i zapewnienie rozproszonej replikacji są cechami mającymi pozwolić jej sprostać temu zadaniu.
- Bezpośrednio składować dane wymieniane pomiędzy serwerami. Format danych MongoDB jest kompatybilny z notacją obiektową JSON wykorzystywaną przez Web Services. Otwiera to możliwości zastosowania jej w aplikacjach typu mash-up.

Pozbawione struktury danych bazy doskonale komponują się z dynamicznymi językami, takimi jak np. PHP czy Ruby, używanymi w programowaniu aplikacji internetowych. Języki te dostarczają bibliotek do komunikacji z bazą i wykonywania podstawowych operacji (np. klasa MongoDB w PHP i MongoMapper w Rubym). Ponadto MongoDB dostarcza powłokę wykorzystującą język JavaScript, za pomocą której możliwe jest wywoływanie na bazie poleceń podobnych do zapytań SQL. Przykłady poleceń pobierających dane przedstawiono w tabeli 2.

Tabela 2

Zapytania w języku SQL i ich odpowiedniki napisane w powłoce MongoDB za pomocą języka JavaScript

SQL	Powłoka MongoDB
`SELECT * FROM articles WHERE art_price > 0 LIMIT 20`	`db.articles.find({art_price:{$gt: 0}}).limit(20)`
`SELECT DISTINCT m.name FROM articles a JOIN manufacturers m ON a.art_manufacturer = m.id`	`db.articles.distinct({"art_manufacturer"})`
`SELECT COUNT(*) FROM articles a JOIN distributors d ON a.distributor = d.id WHERE d.name LIKE 'RYG'`	`db.articles.find({art_distributor: /RYG/}).count()`

Samodzielnym rzeczywistym zastosowaniem MongoDB mogą stać się aplikacje przetwarzające duże ilości szybko napływających danych (np. analiza logów błędów przesyłanych z wielu serwerów, statystyki witryny), aplikacje składujące dane o dużych rozmiarach (galerie obrazków, repozytoria plików) i wszelkie aplikacje, których model warstwy danych pasuje do stylu dokumentowego (systemy komentarzy, dashboardy itp.).

Ponadto MongoDB może być wykorzystywane jako uzupełnienie technologii relacyjnej w architekturach aplikacji webowych.

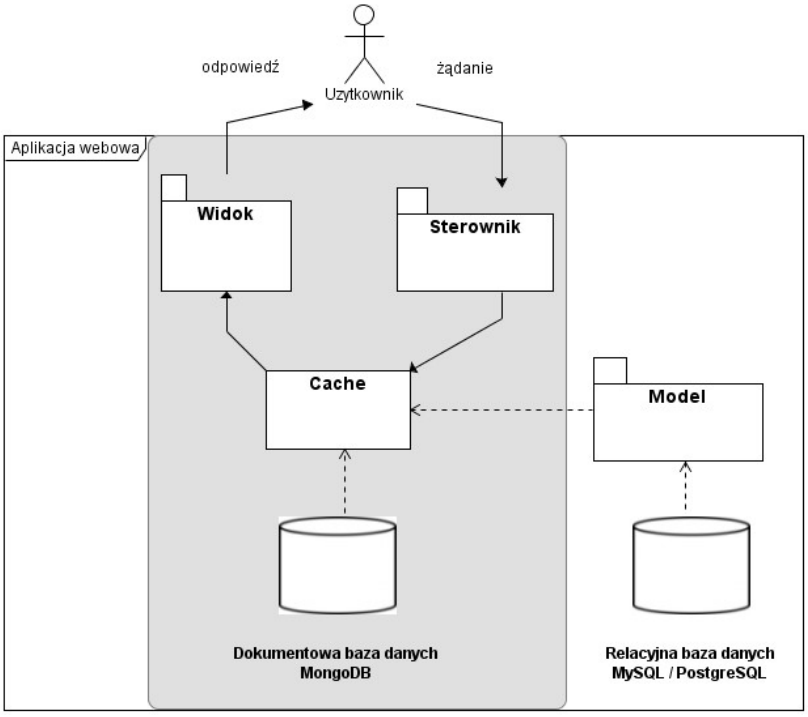
6. Studium przypadku: MongoDB w implementacji warstwy buforującej dane w architekturze aplikacji webowej

Nowoczesne serwisy WEB 2.0 często prezentują użytkownikowi mnóstwo informacji na jednej witrynie. Nawet jeżeli charakter pokazywanych danych jest naturalnie relacyjny, nie ulega wątpliwości, że wyświetlenie strony może wymagać wielu zapytań, czasami wymagających 5 lub więcej złączeń tabel. Nawet w przypadku fragmentu podręcznikowego przykładu systemu typu e-commerce – katalogu produktów, niektóre z tych tabel mogą zawierać od kilkunastu tysięcy do kilku milionów rekordów. Ruch generowany przez setki użytkowników znacząco obciąża serwery baz danych. Powszechnie stosowanym, niezbędnym w dużych aplikacjach webowych, rozwiązaniem pozwalającym na uzyskanie wydajności jest buforowanie stron (i/lub ich fragmentów) generowanych podczas obsługi żądań użytkowników.

Stosowanie cache'a warstwy prezentacji ma jednak swoje wady. Operacje wykonywane na systemie plików, konieczne przy zapisywaniu, jak i pobieraniu części zbuforowanego kodu HTML, nie należą do najszybszych. Mogą prowadzić również do sytuacji, gdy w przypadku konieczności zmiany wyglądu strony zajdzie potrzeba ponownego wygenerowania kodu kilkuset tysięcy podstron.

Przedstawione poniżej podejście zakłada przeniesienie buforowania na inny poziom aplikacji: cache widoku zostaje zastąpiony cachem danych. Buforowanie żądań użytkowników

odbywa się nie na poziomie utrwalonych fragmentów kodu stron, lecz na poziomie danych przekazywanych do warstwy prezentacji. Podobne do zaprezentowanego na rys. 2 rozwiązanie zostało sprawdzone w serwisie SongKick (www.songkick.com) zawierającym przeszło 1,3 mln sprawozdań z koncertów różnych artystów i pozytywnie ocenione podczas prezentacji na Ruby Manor 2, konferencji poświęconej językowi Ruby [9].



Rys. 2. MongoDB jako *backend* modułu buforującego dane rozszerzającego architekturę MVC
Fig. 2. MongoDB as a backend of data buffering module that widen MVC architecture

Dane potrzebne do wygenerowania całej strony są po raz pierwszy pobierane z relacyjnej bazy danych poprzez wykonanie ciągu zapytań, następnie utrwalane w dokumentowo zorientowanej bazie danych jako całość. Kolejne żądania uzyskania określonego zasobu spowodują już pobranie danego dokumentu bez narzutu związanego z łączeniem danych i odpowiednie go sformatowanie. Kolejna komunikacja z bazą relacyjną będzie konieczna dopiero w przypadku uaktualnienia danych, kiedy bufor musi zostać odtworzony.

Rozwiązanie to elastycznie wpasowuje się w stosowaną w popularnych frameworkach dla webowych języków programowania (Symfony, Rails, Django) architekturę Model-View-Controller i może być łatwo dodane do aplikacji zbudowanej w tej konwencji.

7. Podsumowanie

Dokumentowo zorientowane bazy danych są interesującym narzędziem dającym architektom aplikacji webowych alternatywę w wyborze rozwiązań, które mogą być zastosowane w projektowaniu warstwy danych. O ich przydatności świadczy fakt, że nie

w projektowaniu warstwy danych. O ich przydatności świadczy fakt, że nie tylko są wykorzystywane w przypadku serwisów WWW działających w warunkach dużego obciążenia, ale także mogą stanowić doskonałe uzupełnienie technologii relacyjnej w typowych rozwiązaniach internetowych, przejmując odpowiedzialność za niektóre funkcjonalności webowych systemów.

Dalsze badania powinny skoncentrować się na opracowaniu wzorców projektowych przedstawiających możliwości zastosowania dokumentowych baz danych w aplikacjach internetowych wskazujących zalety i wady rozwiązań konkretnych problemów.

BIBLIOGRAFIA

1. Anderson J. C., Lehnardt J., Slater N.: CouchDB: The Definitive Guide. O'Reilly, 2009.
2. Chang F., Dean J., Ghemawat S., Hsieh W. C., Wallach D. A., Burrows M., Chandra T., Fikes A., Gruber R. E.: Bigtable: A Distributed Storage System for Structured Data. OSDI, 2006.
3. DeCandia G., Hastorun D., Jampani M., Kakulapati G., Lakshman A., Pilchin A., Sivasubramanian S., Vosshall P. and Vogels W.: Dynamo: Amazon's Highly Available Key-value Store. SOSP, 2007.
4. http://s3.amazonaws.com/AllThingsDistributed/sosp/amazon-dynamo-sosp2007.pdf .
5. Eifrem E.: A NoSQL Overview And The Benefits Of Graph Databases. http://www.slideshare.net/emileifrem/nosql-east-a-nosql-overview-and-the-benefits-of-graph-databases, NoSQL East, 2009, (2009-09).
6. Ellis J.: NoSQL Ecosystem, http://www.rackspacecloud.com/blog/2009/11/09/nosql-ecosystem/, (2009-11-09).
7. Gupta V.: NoSql Databases – Part 1 – Landscape. http://www.vineetgupta.com/2010/01/nosql-databases-part-1-landscape.html, (2010-01-05).
8. Jones R.: Anti-RDBMS: A list of distributed key-value stores. http://www.metabrew.com/article/anti-rdbms-a-list-of-distributed-key-value-stores/, (2009-01-19).
9. Katz D.: The CouchDB Project: a document oriented database.
10. http://damienkatz.net/files/What%20is%20CouchDB.pdf, (2008-01-24).
11. Lucraft D.: Denormalizing Your Rails Application, Ruby Manor 2.
12. http://github.com/danlucraft/presentations/raw/master/denormalizing.pdf, (2009-12-12).
13. MongoDB. http://www.mongodb.org/, (2009-12-03).
14. Neo E.: NOSQL: scaling to size and scaling to complexity.
15. http://blogs.neotechnology.com/emil/2009/11/nosql-scaling-to-size-and-scaling-to-complexity.html, (2009-11-15).

16. NoSQL Databases – List of NoSQL Databases. http://nosql-databases.org/, (2010-01-17).
17. Oskarsson J.: NoSQL debrief. http://blog.oskarsson.nu/2009/06/nosql-debrief.html, (2009-06-13).
18. Popescu A.: Quick Reference to Alternative data storages. http://themindstorms. blogspot.com/2009/05/quick-reference-to-alternative-data.html (2009-05-28).
19. Smith R.: Non-Relational Databases. php|architect, Vol. 8, August 2009.

Recenzenci: Dr inż. Paweł Kasprowski
Dr inż. Adam Świtoński

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

Document-oriented databases are one of the branches of "Not Only SQL" (NoSQL), a young community aiming to provide an alternative technology to traditional relational databases, better suited for scaling out to large and heterogeneous datasets.

Various types of non-relational databases differ in their data models. As a consequence, they provide different types and levels of scalability. Data models of non-relational databases can better scale to size or better scale to complexity of datasets (Fig.1).

Document databases have several features that find them to be a good fit for building web infrastructure. They offer flexible data schema and store related data together in denormalized form which can simplify the process of modelling some business domains and facilitate the process of making changes later. Furthermore, they can easily store high volume data, so programmers don't have to use the file system in addition to DB. As the structure of the data model often corresponds with the way in which information are presented to the user, document databases have a great potential for caching.

The paper presents the method of using document database (MongoDB) in the architecture of a web application to increase its performance. The original Model-View-Controller pattern was extended to work with cache layer which links the data tier with business logic (Fig. 2). MongoDB was placed in the data tier simultaneously with the relation database. Data collected from the relational DB are immediately saved in MongoDB. Thus, for the next request they can be reached without querying across many various tables.

The further research should concentrate on trying to work out design patterns for web applications concerning the use of document databases.

Adres

Radosław ZATOKA: Uniwersytet Ekonomiczny we Wrocławiu, Katedra Systemów Sztucznej Inteligencji, ul. Komandorska 118/120, 51-250 Wrocław, Polska, radoslaw.zatoka@ue.wroc.pl .