STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Robert BRZESKI
Politechnika Śląska, Instytut Informatyki

ROZWIĄZANIA PODSTAWOWYCH IDEI BAZ DANYCH TYPU NOSQL W KONTEKŚCIE BEZPIECZEŃSTWA DANYCH

Streszczenie. W artykule przedstawiono powody powstania baz danych typu NoSQL oraz podstawowe metody realizacji tego typu systemów. Następnie odniesiono się do bezpieczeństwa składowania i operacji na danych. Na końcu przedstawiono przykładowe rozwiązania na podstawie dokumentowej bazy danych MongoDB, będącej przedstawicielem systemu Not Only SQL (nie tylko SQL).

Słowa kluczowe: NoSQL, Not Only SQL, MongoDB, transakcje, spójność, integralność, baza dokumentowa

THE SOLUTION OF BASIC IDEAS OF THE NOSQL'S DATABASES IN CONTEXT OF THE DATA'S SAFETY

Summary. The paper presents main reasons of the NoSQL's database arises and the basic methods of the realization of this type of systems. It was referred to the safety of storing and operation on the data. Example solutions were introduced at the end of the paper – MongoDB, document-oriented databases, the representative of the Not Only SQL system.

Keywords: NoSQL, Not Only SQL, MongoDB, transaction, data integrity, consistency, document-oriented

1. Geneza ruchu NoSQL

Na początku warto zadać sobie pytanie o przyczyny powstania tzw. ruchu NoSQL oraz wyjaśnić, czym on tak naprawdę jest, a czym nie jest? Rosnące zapotrzebowanie wielu użytkowników systemów informatycznych na dostęp do informacji powoduje, że wiele instytucji ma w swoich bazach stosunkowo ogromne ilości danych (GB, TB czy nawet PB (petabajty)). Duży rozmiar tych danych powoduje, iż zagadnienie ich przetwarzania staje się samo w sobie

problemem niebanalnym. Obecnie najczęściej stosowane relacyjne bazy danych, wprawdzie „potrafią poradzić" sobie z tak dużymi zbiorami, jest to jednak okupione zarówno wysokimi kosztami, związanym z potrzebną infrastrukturą bazodanową, jak i odpowiednio długimi czasami, potrzebnymi do przeprowadzenia realizowanych operacji, m.in. operacji wyszukiwania i udostępniania wymaganych danych. Biorąc pod uwagę, że w wielu zastosowaniach czas dostępu do danych ma znaczenie kluczowe, zrodziła się potrzeba przyspieszenia wykonywania tego typu operacji.

Pojawienie się takiej odpowiednio silnej potrzeby spowodowało rozpoczęcie stosownych prac i powstawanie adekwatnych rozwiązań. Szczególne nasilenie tego typu prac można zaobserwować od 2009 r. Rezultatem są bazy danych, które mogą być wykorzystane w wielu specyficznych sytuacjach. W jednym zdaniu sytuacje te można określić poprzez warunki pracy, w których kluczowym czynnikiem jest szybkość przetwarzania danych – szczególnie ich wyszukiwania, przy jednoczesnym braku potrzeby korzystania z wszystkich możliwości, jakie dają relacyjne bazy danych. Twórcy systemów NoSQL zakładają, że relacyjne bazy danych są zbyt wolne, a finansowe koszty są często zbyt duże. Uważają, że jest to spowodowane przerostem ich możliwości nad rzeczywistymi potrzebami (przynajmniej w pewnych zastosowaniach). Dlatego proponują rozwiązania, w których użytkownik otrzyma tylko to, co jest mu potrzebne.

Tak więc głównym motywem narodzin idei NoSQL-a jest odpowiednio wydajne przetwarzanie ogromnych ilości danych w czasie rzeczywistym. Dodatkowym argumentem na rzecz powstania NoSQL jest idea utworzenia nowoczesnego systemu typu open source, który mógłby zastąpić obecnie stosowane komercyjne bazy danych: MS SQL Server, Oracle czy też IBM DB2.

W ten sposób powstał ruch pod ogólną nazwą NoSQL. W tym momencie warto jednak podkreślić czy też wyjaśnić, że mamy tutaj do czynienia z rozwiązaniami będącymi uzupełnieniem dla relacyjnych baz danych, nie natomiast z ich zastąpieniem (a w chwili obecnej czasami można się spotkać z takim błędnym rozumieniem idei NoSQL). Dlatego, aby skrót NoSQL był lepiej rozumiany w szerszym kręgu odbiorców, coraz częściej zastępuje się go nazwą „Not Only SQL" – czyli „nie tylko SQL". Przy czym oba skróty są obecnie szeroko stosowane jako swoje zamienniki.

2. Realizacja idei „Not Only SQL" poprzez „uwstecznienie" rozwiązań znanych z relacyjnych baz danych

Kolejne pytanie, jakie warto rozważyć to, w jaki sposób zrealizowano główne zadanie idei NoSQL, czyli przyspieszenie działania bazy danych, w szczególności operacji przeszu-

kiwania zgromadzonych danych? Zadanie to trzeba realizować i rozważać w kontekście wymogów, jakie stawia docelowe zastosowanie. I tak, zauważono, że wiele z mechanizmów dostępnych w relacyjnych bazach danych, mechanizmów będących wynikiem wieloletniego ich rozwoju i stanowiących ogromny dorobek wielu środowisk i instytucji (w wielu obecnych zastosowaniach) jest wykorzystywanych w stopniu niewielkim lub nawet występuje całkowita z nich rezygnacja. Jednocześnie mechanizmy te, będąc częścią wykorzystywanego systemu, wprowadzają wydłużenie czasu realizacji operacji, także tych najprostszych. Czasami wydłużenie to jest bardzo duże. Zdecydowano się więc na rezygnację z pewnej części dorobku, używanego w relacyjnych bazach danych, stanowiących potężną siłę w wielu zastosowaniach, ale jednocześnie będących ogromnym obciążeniem czasowym w innych. Takie podstawowe mechanizmy to:

1. zarządzanie transakcjami,
2. złączenia tabel używane w języku SQL.

Pierwszy mechanizm zostanie szerzej omówiony w kolejnym punkcie [Konsekwencje w zakresie bezpieczeństwa danych dla rozwiązań typu NoSQL], natomiast mechanizm drugi: Język SQL pozwala szukać powiązań pomiędzy danymi poprzez m.in. złączenia „join". Umożliwia to realizację zapytań na wysokim stopniu abstrakcji, ale realizacja tych zapytań jest czasochłonna. Tak więc, do prostych zapytań te mechanizmy są „ociężałe". Możemy więc uzyskać duże przyspieszenie, kosztem rezygnacji z możliwości formułowania bardzo złożonych zapytań (z użyciem złączeń).

Oprócz rezygnacji z mechanizmów operujących na danych, przyspieszenie przetwarzania uzyskuje się poprzez uproszczenie samej struktury przechowywanych danych. I tak, zamiast ściśle zdefiniowanej struktury tabel powiązanych relacjami, w bazach danych NoSQL struktura jest oparta na parze „klucz – wartość", przy czym struktura ta jest odpowiednio rozszerzona, w zależności od typu bazy, można tutaj wyróżnić m.in.:

1. Bazy typu „klucz-wartość" (ang. *Key-Value*) – struktura danych jest oparta na tablicy haszującej dla poszczególnych danych, stanowiących parę klucz-wartość. Struktura taka została zrealizowana przez firmę Amazon w systemie Dynamo, w bazie Voldemort firmy LinkedIn oraz systemie Tokyo Cabinet.
2. Dokumentowe bazy danych (ang. *Document-Oriented Databases*) – struktura danych jest zbiorem dokumentów składających się z par klucz-wartość. Przykłady rozwiązań to: MongoDB, CouchDB.
3. Grafowe bazy danych (ang. *Graph Databases*) – w strukturze danych, zapożyczonej z teorii grafów, wierzchołki oraz krawędzie mogą przechowywać parę klucz-wartość. Przykłady rozwiązań to: AllegroGraph, FlockDB, InfoGrid, Neo4j.
4. Bazy oparte na „kolumnach" (ang. *Tabular, Column-oriented, Wide Column*) – mamy tutaj strukturę tabelaryczną, w której każdy wiersz może składać się z innego zestawu ko-

lumn. Koncepcja pochodzi od firmy Google i bazy BigTable, a przykłady rozwiązań to: Hypertable czy HBase firmy Apache.

W zakresie stosowanych struktur danych mamy więc do czynienia z pewnym „uwstecznieniem" w stosunku do rozwiązań używanych w relacyjnych baz danych. W relacyjnych bazach, struktura danych musi odpowiadać stosunkowo sztywnej organizacji tabelarycznej. Natomiast w bazach danych NoSQL struktura danych jest w znacznie większym stopniu dopasowana do postaci informacji, jaka istnieje w popularnych systemach informatycznych, „postaci naturalnej" (dokumenty, strony WWW). Rozwiązania NoSQL wydają się być bardzo przydatne dla danych nieustandaryzowanych, w sytuacjach, w których nie posiadamy pełnej informacji o danych napływających do bazy.

Minusem stosowania tego typu struktur danych jest występowanie nadmiarowości, powtarzania się tych samych informacji w bazie danych. Możliwość występowania redundancji stanowi jednak pewną korzyść, dającą możliwość zapisu do bazy porcji danych bez konieczności sprawdzania, czy zapisywane wartości nie znajdują się już w bazie. Jest to kolejny element przyspieszający działanie, tym razem operacji zapisu.

Przy czym, aby pełniej zrozumieć i wykorzystać możliwości systemów NoSQL, warto w pewnym stopniu odsunąć na bok doświadczenie zdobyte w trakcie projektowania i użytkowania relacyjnych bazach danych. Warto spojrzeć na nowy system w nieco inny sposób. Na przykład używając SQL-a istniała konieczność optymalizacji realizowanych zapytań. W NoSQL tego typu zagadnienia co najmniej znacznie się upraszczają.

3. Konsekwencje w zakresie bezpieczeństwa danych dla rozwiązań typu NoSQL

Ostatnie z pytań stawianych w tym artykule to, jakie są konsekwencje w zakresie bezpieczeństwa, spowodowane rezygnacją z niektórych mechanizmów stosowanych w relacyjnych bazach danych? „Zagrożenia wobec baz danych mogą spowodować utratę lub uszkodzenie niektórych lub wszystkich z następujących obszarów bezpieczeństwa: integralność, dostępność i poufność." [1]. Wpływ wcześniej przedstawionych mechanizmów na dostępność i poufność danych jest minimalny. Natomiast zdecydowany wpływ zmian wprowadzonych w NoSQL istnieje w zakresie integralności, gdzie „Integralność bazy danych odnosi się do wymagania, aby dane były zabezpieczone przed niewłaściwymi modyfikacjami." [1]. Takim elementem są np. mechanizmy zarządzania transakcjami. Mechanizmy te wprawdzie dają duży narzut czasowy, jednocześnie jednak, w klasycznych systemach zarządzania relacyjnymi bazami danych, należą do podstawowych mechanizmów zabezpieczających integralność (a więc spójność) danych.

Dwa główne rodzaje zagrożeń spójności danych to:

- Upadek systemu w trakcie wykonywania operacji, np. zapisu zmodyfikowanych danych i wtedy do naprawy wystarczający jest dziennik. Jeżeli nie ma dziennika, to nie ma jak odtworzyć spójnego stanu bazy sprzed rozpoczęcia transakcji.
- Bezpieczeństwo w zakresie operacji współbieżnych – tutaj kluczowe znaczenie ma zagadnienie blokowania (zwykle 4 poziomy izolacji transakcji – od całkowitej blokady – system jest wtedy mało wydajny, poprzez kolejne poziomy izolacji – zwiększamy poziom współbieżności, ale kosztem może być brak integralności danych).

Jako że w bazach NoSQL zrezygnowano z mechanizmów zarządzania transakcjami, a przez to blokowania danych, to nie mamy mechanizmów regulujących dostęp współbieżny. Nie ma to znaczenia, gdy jedynymi operacjami są odczyty. Problemy pojawiają się natomiast przy modyfikacji danych. Przykładem niech będzie sytuacja, gdy jeden użytkownik próbuje zmodyfikować dane w dokumencie nr1 na podstawie danych z dokumentu nr2, a w tym samym czasie użytkownik drugi próbuje zrealizować zadanie odwrotne. Może to prowadzić do powstania niespójności danych. W związku z tym, zastosowanie tego typu baz będzie ograniczone np. do systemów, gdzie podstawową operacją będzie odczyt danych.

W relacyjnych bazach danych jednym z mechanizmów zarządzania transakcjami jest prowadzenie dziennika. Zazwyczaj zapisuje się w nim informacje o wszystkich transakcjach, realizowanych od czasu utworzenia ostatniej kopii bezpieczeństwa. Przy czym zwykle odpowiedni opis transakcji najpierw zapisywany jest do dziennika, a następnie właściwe zmiany wpisywane są do bazy danych. W przypadku awarii, dziennik służy do przywrócenia spójnego stanu bazy danych.

Kolejny element wpływający na integralność danych, to konsekwencje uproszczenia struktury przechowywanych danych. Jako że dane w bazach typu NoSQL mogą się powtarzać (np. kolejne, napływające dokumenty mogą zawierać w pewnym zakresie takie same dane, jak te już przechowywane w zapisanych wcześniej innych dokumentach), to można się zastanowić, jak to wpływa na spójność udostępnianych danych – na ich wiarygodność? Jaki będzie rezultat działania systemu w przypadku, gdy z powtarzających się tych samych danych zmodyfikowana zostanie tylko część? W relacyjnych bazach danych, nawet jeżeli istnieje redundancja danych, to jest ona wprowadzona świadomie, wraz z procedurami realizującymi modyfikację danych w taki sposób, aby baza pozostała w stanie spójnym.

Jeszcze jeden aspekt bezpieczeństwa związany z bazami danych NoSQL to „młody wiek" tych systemów. Może mieć to niekorzystny wpływ na zabezpieczenie systemu przed niepowołanym dostępem. Także fakt, że wiele z nich jest realizowanych przez opensource'ową społeczność powoduje, że dostępna obsługa techniczna, przynajmniej w początkowej fazie rozwoju takich systemów, może być na gorszym poziomie niż dla komercyjnych baz danych.

Oczywiście bezpieczeństwo baz danych jest tematem wielowątkowym i w artykule tym, wybrano zagadnienia głównie dotyczące danych, szczególnie ich integralności, w kontekście porównania NoSQL do baz relacyjnych.

4. Przykład realizacji dokumentowej bazy danych typu NoSQL

Wytyczne opisane w poprzednich rozdziałach dotyczą ogólnej idei systemów typu NoSQL. W konkretnych rozwiązaniach mogą one się nieco różnić i nie obejmować wszystkich przedstawionych „uproszczeń". Dla sprawdzenia jak to wygląda w rzeczywistej implementacji, pokrótce opisana zostanie jedna z dokumentowych baz danych. Najpopularniejsze obecnie dokumentowe bazy danych to MongoDB oraz CouchDB. Do zaprezentowania wybrana została pierwsza z nich.

W bazie tej dane przechowywane są w dokumentach, w których podstawową jednostką struktury danych jest para „atrybut – wartość". Dokumenty są zebrane w zbiory zwane kolekcjami. Przy czym nawet w danej kolekcji, struktury dokumentów mogą się od siebie różnić. Dokument nie ma z góry ściśle zdefiniowanej struktury danych. W jednym dokumencie może być pewien zestaw atrybutów i wartości, a drugi dokument może zawierać inne atrybuty i wartości.

Przykład takiego pojedynczego dokumentu może być następujący:

```
{
imię: "Robert",
wiek: 15,
adresy: [{ulica: "Wolności", nr: 30}, {ulica: "Akademicka", nr: 45} ]
}
```

Jak widać na podstawie atrybutu „adresy", struktura takiego dokumentu może być zagnieżdżona. Dane pole może zawierać nie tylko pojedynczą wartość, ale także cały zestaw danych. Jest to tzw. dokument zagnieżdżony. Struktura takiego dokumentu implikuje oczywiście problemy, związane z możliwością wystąpienia powtarzających się danych, co zostało opisane w poprzednich punktach.

W bazie tej nie ma transakcji oraz możliwości blokowania danych, co w znacznym stopniu przyspiesza wykonywanie operacji. Dla twórców tej bazy kluczowym priorytetem była szybkość całego systemu, umożliwiająca realizację napływających zadań w czasie rzeczywistym. Jednak w takiej sytuacji aktualne pozostają występujące problemy czy też ograniczenia, związane z brakiem transakcji.

Dlatego głównym przeznaczeniem tej bazy może być np. przechowywanie, buforowanie stron internetowych. Podczas realizacji MongoDB twórcy dbali o odpowiednią szybkość zapisu, aktualizacji oraz odczytu danych. Dlatego, przynajmniej w założeniu, baza ta nie posia-

da ograniczenia w zakresie skalowalności i dzięki rozproszonej replikacji może obsłużyć dowolnie duży zasób danych. W bazie tej można także przechowywać duże obiekty takie, jak: pliki video, pliki graficzne czy repozytoria plików.

5. Podsumowanie

Celem nadrzędnym powstania baz danych NoSQL jest zwiększenie szybkości ich działania w porównaniu do baz relacyjnych.

Cel ten jest realizowany głównie przez:

1. Uproszczenie struktury, w której przechowywane są dane.
2. Rezygnację z niektórych mechanizmów stosowanych w bazach relacyjnych.

Sposób realizacji tego celu ma jednak znaczący wpływ na niektóre aspekty bezpieczeństwa danych, przechowywanych i przetwarzanych przez taki system.

Nikt z grona twórców rozwiązań typu NoSQL prawdopodobnie nie myśli o zastosowaniu ich w systemach, gdzie zagadnienie spójności danych oraz możliwości ich utraty mają kluczowe znaczenie (np. systemy bankowe, które same w sobie często określa się systemami transakcyjnymi). Warto chyba to jednak zapisać jawnym tekstem, aby szersze grono odbiorców miało tego świadomość. Typowym zastosowaniem mogą być zbiory danych, gdzie zagadnienie integralności nie ma kluczowego znaczenia, gdzie dostęp współbieżny do danych nie występuje lub konsekwencje są bez znaczenia, czy też stanowią problem pomijalny. Takim zastosowaniem mogą być np. blogi czy też wiele stron z witryn internetowych, gdzie dominującą operacją jest odczyt danych.

Zastosowanie baz NoSQL jest w wielu sytuacjach jak najbardziej możliwe i korzystne. Pewnym rozwiązaniem może także stać się system hybrydowy, powstały z połączenia relacyjnych baz oraz tych typu NoSQL.

BIBLIOGRAFIA

1. Elmasri R., Navathe S.B.: Wprowadzenie do systemów baz danych, Helion 2005.
2. http://nosql-databases.org/
3. http://www.mongodb.org/
4. http://couchdb.apache.org/
5. http://www.mongodb.org/display/DOCS/Comparing+Mongo+DB+and+Couch+DB
6. http://www.mongodb.org/display/DOCS/MongoDB%2C+CouchDB%2C+MySQL+Compare+Grid

7. http://fallabs.com/tokyocabinet/
8. http://nosql.blip.pl/
9. http://www.mongodb.org/display/DOCS/Atomic+Operations
10. http://webhosting.pl/Ruch.przeciwnikow.SQL_owych.baz.danych.zyskuje.na.popularnosci
11. http://s3.amazonaws.com/AllThingsDistributed/sosp/amazon-dynamo-sosp2007.pdf.
12. http://project-voldemort.com/
13. http://cassandra.apache.org/
14. http://hypertable.org/
15. http://hbase.apache.org/
16. http://www.basho.com/Riak.html
17. http://www.franz.com/agraph/allegrograph/
18. http://infogrid.org/
19. http://neo4j.org/
20. https://github.com/twitter/flockdb

Recenzent: Prof. dr hab. inż. Stanisław Wrycza

Wpłynęło do Redakcji 30 stycznia 2011 r.

Abstract

Primary aim of the NoSQL's database arises is the speed of their working, processing of data. This aim is realized mainly by:

1. The simplification of the structure in which the data is stored.
2. Resignation from some mechanisms applied in relational databases.

The way of the realization of this aim, has however the significant influence on some aspects of safety of data storage and processed, by such system.

From team of the creators of the NoSQL's solutions, probably nobody thinks about their use in systems, where issue of the consistency of data and the possibility of their loss, has the crucial meaning (e.g. bank's systems which often are defines himself as transaction systems). This is probably worth to write down, to give to the bigger team of addressees, the consciousness of this issue. So, the typical use of NoSQL, can be found in system, where the problem of the data integrity has no crucial meaning, where the concurrent access to the data is not present, or consequences are without the meaning. Such typical use can be internet pages, where the predominant operation is the reading of the data.

The use of NoSQL databases is in many situations possible and profitable. Certain solution can be also the hybrid system, as connection NoSQL database and the relational one.

Adres

Robert BRZESKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, robert.brzeski@polsl.pl.