Rozproszone systemy operacyjne

Projekt – RSO NoSQL

Autorzy:

* Tomasz Adamiec
* Piotr Cebulski
* Marek Kowalski
* Mateusz Rosiewicz
* Paweł Sokołowski
* Marcin Wnuk

Warszawa, 2013

[1. Wstęp 4](#_Toc353057015)

[1.1 Co to jest NoSQL? 4](#_Toc353057016)

[1.2 Potrzeba baz rozproszonych 4](#_Toc353057017)

[1.3 Czemu NoSQL? 4](#_Toc353057018)

[1.4 NoSQL movement 4](#_Toc353057019)

[1.5 Pierwsze projekty 4](#_Toc353057020)

[2. Ogólna koncepcja 4](#_Toc353057021)

[2.1 Spójność 4](#_Toc353057022)

[2.2 Partycjonowanie 4](#_Toc353057023)

[2.3 Struktura danych 4](#_Toc353057024)

[2.4 Model zapytań 4](#_Toc353057025)

[3. Model relacyjny vs. model zorientowany na dokumenty 4](#_Toc353057026)

[4. Wnioski 5](#_Toc353057027)

1. Wstęp

* 1. Co to jest NoSQL?
  2. Potrzeba baz rozproszonych
  3. Czemu NoSQL?
  4. NoSQL movement
  5. Pierwsze projekty

1. Model relacyjny vs. model zorientowany na dokumenty

Stosowanie baz NoSQL niesie ze sobą wiele korzyści, jednak deweloperzy przyzwyczajeni do baz relacyjnych będą zmuszeni do zmiany dotychczasowego podejścia. Przestawienie się na model zorientowany dokumentowo może z początku nastręczać problemów, gdyż wiąże się to z zerwaniem z praktykami, które dla bazodanowców są naturalne i weszły w nawyk. Główną motywacją do podjęcia takiego trudu jest potrzeba elastyczności w skalowaniu systemu i w modelu danych.

Technologia baz relacyjnych to technologia skalująca się w górę (*scale up*) – jeśli zachodzi potrzeba zwiększenia przepustowości lub pojemności, należy kupić większy serwer. Dzisiaj preferuje się skalowalność horyzontalną (*scale out*) – zamiast większego serwera, kupuje się więcej serwerów. Zmniejsza to koszty, gdyż umożliwia użycie sprzętu łatwo dostępnego (*commodity servers*) oraz korzystanie z usług serwerowych oferowanych w różnych innych formach. Skalowanie horyzontalne, które jest standardem na poziomie logiki biznesowej, wchodzi dopiero w dziedzinę baz danych.

Równie ważne co skalowalność horyzontalna, jest podejście do zarządzania danymi. Bazy relacyjne opierają się na schemacie. Aby dodać rekord do takiej bazy musi on być zgodny z tym schematem, narzucona jest z góry np. szerokość kolumn czy typ danych. Zmiana schematu w istniejącej bazie jest trudna, w szczególności gdy baza jest podzielona i znajduje się na wielu serwerach. Bazy NoSQL nie wymagają żadnego schematu, ani przy dodawaniu rekordów, ani przy dalszym zarządzaniu danymi, co daje im istotną przewagę w pewnych zastosowaniach.

* 1. Model relacyjny

Każdy rekord w bazie relacyjnej musi być zgodny ze schematem – musi mieć stałą liczbę kolumn, każda o określonym znaczeniu i typie danych. Każdy rekord jest taki sam. Jeśli chce się umieścić w bazie rekord innego typu, należy zmienić schemat. Dodatkowo, model relacyjny podlega normalizacji, co oznacza, że duże tabele są dzielone na mniejsze będące ze sobą relacji.

Efekt normalizacji jest taki, że kawałki rekordów są porozrzucane pomiędzy tabelami i rekordy z jednej tabeli mogą być wspólne dla wielu rekordów z innej tabeli. Zaletą tego jest oczywiście brak duplikacji danych. Wadą natomiast, jest blokowanie wielu tabel przy zmianie rekordu w jednej tabeli. Poprawne przetwarzanie transakcji (ACID) w bazach relacyjnych jest więc skomplikowane i jest przyczyną ograniczonej wydajności rozproszonych baz tego typu.

Teraz, gdy cena przestrzeni dyskowej jest niewielka, coraz rzadziej pojawia się realna potrzeba ograniczenia wielkości bazy. Zużycie większej ilości przestrzeni w zamian za lepszą wydajność i możliwość rozłożenia obciążenia pomiędzy maszyny jest lepszym rozwiązaniem przy tworzeniu wielu współczesnych aplikacji.

* 1. Model zorientowany na dokumenty

Dokument w bazach NoSQL oznacza rekord, który sam dostarcza informacji o sobie – o tym jakie pola się w nim znajdują i jakie są ich wartości. Przeważnie są to dokumenty w formacie XML, HTML lub JSON. Dane w tych dokumentach są zdenormalizowane. Dokument zawiera więc wszystkie właściwe dla niego informacje, nie odwołuje się do innych dokumentów. Nie ma problemów z zachowaniem spójności danych w obrębie pojedynczego rekordu, a więc także zachowanie poprawności transakcji jest o wiele prostsze. Wydajność zarówno modyfikacji, jak i odczytu rekordów jest w tym przypadku dużo wyższa niż w bazach relacyjnych.

Dokumenty w bazie NoSQL posiadają identyfikatory, które stanowią odpowiednik klucza głównego w bazie relacyjnej. Zazwyczaj identyfikator występuje w bazie tylko raz. Dane mogą być sortowane po identyfikatorze w taki sposób, aby dokumenty z podobnymi identyfikatorami były bliżej siebie. W ten sposób dane, dla których jest największe prawdopodobieństwo zażądania, będą dostępne w najkrótszym czasie.

Przykładem wykorzystania takiego podejścia, może być tworzenie oddzielnych dokumentów na komentarze pod artykułem. Komentarze nie powinny być umieszczone w tym samym dokumencie co artykuł, ponieważ jednoczesna edycja przez wielu komentatorów, byłaby utrudniona – blokowali by siebie na wzajem. W zamian, należy dla każdego komentarza utworzyć nowy dokument o identyfikatorze bliskim ID artykułu. Przykład ten pokazuje przeciwieństwo baz relacyjnych i NoSQL – w pierwszych wszystko podlega normalizacji, a w drugich odwrotnie.

1. Ogólna koncepcja
   1. Spójność
   2. Partycjonowanie
   3. Struktura danych
   4. Model zapytań
2. Wnioski