Rozproszone systemy operacyjne

Nazwa projektu „nie wiem jaka jest dokładna nazwa ☺”

Autorzy:

Imię nazwisko

Imię nazwisko

Imię nazwisko

Imię nazwisko

Imię nazwisko

Warszawa, 2013

[1 Wstęp 3](#_Toc352782441)

[2 Cechy MongoDB 3](#_Toc352782442)

[2.1 API 4](#_Toc352782443)

[find 4](#_Toc352782444)

[findAndModify 4](#_Toc352782445)

[findOne 4](#_Toc352782446)

[Insert 5](#_Toc352782447)

[Save 5](#_Toc352782448)

[Update 5](#_Toc352782449)

[2.2 Replikacja danych 5](#_Toc352782450)

[2.3 Bezpieczeństwo 5](#_Toc352782451)

1. Wstęp

MongoDB jest to otwarty i nierelacyjny system zarządzania bazą danych napisany w języku C++. Prace nad systemem rozpoczęła firma 10gen w 2007 roku. Pierwsze wydanie stabilnej wersji nastąpiło w 2009 roku. MongoDB charakteryzuje się dużą skalowalnością oraz wydajnością. Nie posiada ściśle określonej struktury obsługiwanych baz danych. Dane składowane są jako dokumenty w stylu JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchio raz indeksowania.

MongoDB charakteryzuje się następującymi właściwościami:

* jednorodne wsparcie dla standardu unicode ,
* obsługa danych w innych kodowaniach w formacie binarnym,
* duża liczba obsługiwanych typów danych,
* obsługa kursorów,
* zapytania ad-hoc,
* zapytania do zagnieżdżonych pól dokumentów,
* indeksowanie,
* wsparcie dla agregacji danych,
* możliwość składowania plików w bazie,
* architektura zaprojektowana z myślą o łatwej replikacji.

Wewnętrznym językiem do definiowania zapytań oraz funkcji agregujących jest [JavaScript](http://pl.wikipedia.org/wiki/JavaScript) wykonywany bezpośrednio przez serwer MongoDB. Interfejsy programistyczne pozwalające obsługiwać bazy MongoDB powstały dla wszystkich wiodących języków programowania, w tym dla C, C++, C#, Javy, PHP, Perla, Pythona, Rubiego.

MongoDB posiada ograniczone wsparcie dla transakcji - ich zasięg jest ograniczony do zmian w pojedynczym dokumencie, aczkolwiek zmiany te mogą być bardzo skomplikowane. Z tego powodu część użytkowników ogranicza zastosowanie MongoDB do niekrytycznych danych informacyjnych, pozostawiając obsługę krytycznych operacji (np. obsługa zamówień w sklepie) relacyjnym bazom danych, gdzie takie ograniczenia nie występują.

1. Cechy MongoDB

W celu utworzenia bazy danych wystarczy polecenie use <nazwa bazy> , baza jest gotowa do użycia. Domyślnie przy użyciu wiersza poleceń ( mongo ) używana jest baza test.

Baza jest w stu procentach dynamiczna. Ma to swoje plusy i minusy. Bazy danych i kolekcje (odpowiedniki tabel) tworzone są dynamicznie – oznacza to, że baza nie krzyczy na nas za literówki. Jeżeli raz napiszemy

db.users.insert({"name": "Student" });

(komenda wstawiająca dokument – odpowiednik rekordu – do kolekcji *user* z polem – kolumną - *name* która ma wartość *“Student”*) to Mongo sprawdzi czy w używanej przez nas bazie jest już kolekcja *users*, jeżeli nie to ją utworzy. Gdy za drugim razem wpiszemy

db.usrs.find();

(komenda listująca wszystkie dokumenty z kolekcji *usrs* ) to Mongo nie nakrzyczy na nas, tylko wyświetli pusty zbiór wyników. Gdy wpiszemy

db.usrs.insert( {"name": "Student" } );

to Mongo stworzy kolekcję *usrs*, wstawi do niej dokument z polem *name* o wartości *Student* i będzie się cieszyć z dobrze wykonanego zadania.

Co ciekawe każdy dokument przy insercie dostaje generowany przez bazę identyfikator. Dotyczy to też sterowników do PHP – po wywołaniu inserta obiekt który dodawaliśmy dostaje automatycznie pole \_id .

* 1. API

find

db.collection.find();

Metoda find jest to odpowiednik selecta w relacyjnych bazach danych. Składnia tej metody została przedstawiona poniżej.

db.products.find( { qty: { $gt: 25 } } );

Powyższy zapis należy rozumień jako: znajdź wszystkie dokumenty o nazwie products w których wartość dla pola gty jest większa niż 25.

findAndModify

db.collection.findAndModify( {

query: <document>,

sort: <document>,

remove: <boolean>,

update: <document>,

new: <boolean>,

fields: <document>,

upsert: <boolean>

} );

Przykład wykorzystania został zamieszczony poniżej.

db.people.findAndModify( {

query: { name: "Tom", state: "active", rating: { $gt: 10 } },

sort: { rating: 1 },

update: { $inc: { score: 1 } }

} )

Powyższy zapis należy rozumień jako: znajdź dokumenty z kolekcji people, gdzie wartość dla pola name jest równa Tom, dla pola state jest równa active, dla pola rating jest większa niż 10. Następnie posortuj rosnąco dokumenty z pierwszego zapytania jeśli takie istnieją i wybierz pierwszy dokument. Następnie zwiększ wartość dla pola score o 1.

findOne

db.collection.findOne();

Powyższe polecenie zwraca tylko jeden dokument spełniający warunek zadany w kwerendzie. Jeśli wiele dokumentów spełnia warunek, wyświetlony zostanie tylko pierwszy dokument zgodnie z porządkiem jaki jest on umieszczony w bazie.

Insert

db.collection.insert(document);

Przykład zastosowania:

db.products.insert( { item: "card", qty: 15 } );

Operacja ta wstawi nowy dokument do kolekcji protucts z polem item z wartością card oraz polem qty z wartością 15, oraz z unikalnym kluczem ObjectID.

Save

Save();

Metoda save aktualizuje istniejący dokument, lub wstawia dokument w zależności od parametru.

Przykład zastosowania:

db.products.save( { item: "book", qty: 40 } );

Jeśli wartość w dokumencie dla pola \_id będzie podana oraz wartość ta będzie już istniała wówczas dokument zostanie zaktualizowany z wyszczególnionymi wartościami dla danych pól. Jeśli w dokumencie nie będzie istniała wartość dla pola \_id wówczas do dokumentu zostaną dodane wyszczególnione pola z ich wartościami.

Update

db.collection.update(query, update[, options]);

Metoda update aktualizuje istniejący dokument, lub dokumenty w kolekcji. Domyślnie update aktualizuje jeden dokument.

Przykład zastosowania:

db.products.update( { item: "book", qty: { $gt: 5 } }, { $set: { x: 6 }, $inc: { y: 5} } );

Metoda ta aktualizuje dokument w kolekcji products, która odpowiada kryterium zapytania i ustawia wartość pola x na 6 oraz zwiększa wartość dla pola y o 5. Wszystkie pozostałe pola w dokumencie pozostają niezmienione.

* 1. Replikacja danych

MongoDB oferuje dwa rodzaje replikacji danych. Są nimi:

* *Replica set replication*
* *Master-**slave replicatio*

*Master-slave replication* jest klasycznym sposobem replikacji danych z jednym nadrzędnym węzłem i wieloma podrzędnymi. Podobne podejście oferowane jest przy *replica set replication,* jednak zawiera ono mechanizmy do automatycznego przełączania awaryjnych węzłów. Pierwszy z wymienionych sposobów uważany jest już za przestarzały i poza sytuacjami w których istnieją racjonalne wskazania na używanie go zaleca się używanie drugiego z wymienionych sposobów.

*Replica set* jest klastrem złożonym z co najmniej dwóch instancji *mongod[[1]](#footnote-1)*. Co najwyżej jedna z nich uważana jest za podstawową, a pozostałe ze drugorzędne. Wszystkie zapisy danych przeprowadzane są poprzez podstawowy węzeł, dane w węzłach drugorzędnych są replikowane w sposób asynchroniczny.

Ten sposób replikacji danych w MongoDB zwiększa złożoność systemu, ale zarazem zwiększa również jego dostępność, prędkość odczytów oraz ułatwia przeprowadzenie niektórych z zadań administracyjnych takich jak kopie zapasowe danych. Jedną z najważniejszych cech *replica set* jest automatyczne przełączanie głównego węzła w przypadku jego awarii. Pozostałe węzły przeprowadzają przezroczysty dla użytkownika wybór nowego węzła nadrzędnego. Mechanizm ten wprowadza jednak pewne ograniczenia. W jednym *replica set* może być maksymalnie 12 węzłów, w tym maksymalnie siedem mających prawo głosu.

Właściwości opisujące cechy węzłów

Każdy z członków *replica set* jest opisany następującymi właściwościami:

* **Secondary-Only –** taki węzeł nie może zostać nigdy węzłem głównym
* **Hidden** – węzeł nie jest widoczny przez aplikacje klienckie
* **Delayed –** operacje z głównego węzła replikowane są z określonym opóźnieniem
* **Arbiters –** węzeł nie zawierający danych, występuje jedynie w celu brania udziału w wyborach
* **Non-Voting –** węzeł nie ma prawa głosu

Przełączanie węzła głównego i głosowanie

Aby doszło do automatycznego przełączenia węzła głównego muszą być spełnione co najmniej dwa warunki:

* Nadrzędny członek klastra zaczyna nie odpowiadać
* Większość z drugorzędnych członków klastra jest połączona każdy z każdym

Jeżeli powyższe warunki są spełnione węzły drugorzędne wybierają spośród siebie nowy węzeł główny. Dzieje się to poprzez procedurę głosowania. Członkowie klastra poza członkami niegłosującymi mają jeden głos, włączając w to członków **Secondary-Only**, **Hidden** i **Arbiters** i wybierany jest pierwszy z członków który zdobędzie większość głosów (może się zdarzyć, że któryś z członków ma ustawiony wyższy priorytet, wtedy ma on większe szanse na zostanie wybranym na nowy węzeł główny. Nie jest to jednak zalecana rozwiązanie). W wyborach uczestniczą także członkowie niegłosujący.

Spójność danych

W przypadku węzła głównego wszystkie dane odczytywane są spójne z ostatnią operacją zapisu. Gdy preferencje odczytu zostaną tak ustawione, że pozwalają na odczyt z członków drugorzędnych możliwe jest odczytanie danych, które nie są całkowicie zgodne z ostatnim zapisem do głównego serwera. W normalnej, domyślnej sytuacji nie można zapewnić takiej spójności. Można jednak tak skonfigurować driver MongoDB, aby operacja zapisu była uznana za zakończoną z sukcesem dopiero w momencie gdy zapis zostanie przeprowadzony we wszystkich węzłach.

W pewnych przypadkach awarii węzła głównego i zmiany jego na jeden z węzłów drugorzędnych możliwe jest zaistnienie danych jeszcze niezreplikowanych do pozostałych członków klastra. W takiej sytuacji członek, które poprzednio był głównym członkiem i ponownie dołącza do klastra dokonuje operacji odwrócenia[[2]](#footnote-2) danych w celu zachowania ich spójności. Dane takie zapisywane są do pliku *BSON* w katalogu *dbpath.* Dane te mogą być odczytane za pomocą polecania *bsondump[[3]](#footnote-3).* Zaktualizowanie nowego węzła głównego o te dane musi zostać wykonane ręcznie.

* 1. Auto-Sharding

1. **mongod** – podstawowy proces systemu MongoDB. Zarządza on formatem danych, obsługą danych oraz wykonuje operacje w tle. [↑](#footnote-ref-1)
2. rollback [↑](#footnote-ref-2)
3. **Bsondump –** konwertujedane w postaci BSON w formę czytelną dla człowiek, np.: JSON [↑](#footnote-ref-3)