Szczegółowa koncepcja proponowanego rozwiązania i dokumentacja techniczna

Projekt – RSO MongoDB

Autorzy:

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

Warszawa, 2013

[CRUD 3](#_Toc358654210)

[FileOperations - Operacja na plikach 3](#_Toc358654211)

[Selector -Wybór plików z zapytania 3](#_Toc358654212)

[Insert - Zapisywanie plików na dysku 3](#_Toc358654213)

[Delete - Usuwanie plików z dysku 3](#_Toc358654214)

[Update - Modyfikacja plików na dysku 3](#_Toc358654215)

[Komunikacja i przetwarzanie wiadomości 3](#_Toc358654216)

[Ogólny zarys 3](#_Toc358654217)

[Opis rozwiązania 4](#_Toc358654218)

[Struktura 4](#_Toc358654219)

[Parsowanie wiadomości 4](#_Toc358654220)

[Wysyłanie komunikatów 5](#_Toc358654221)

[BSON 5](#_Toc358654222)

[Czym jest BSON 5](#_Toc358654223)

[Zastosowanie BSON 5](#_Toc358654224)

[Ogólny zarys 5](#_Toc358654225)

[Parsowanie BSON 5](#_Toc358654226)

[Tworzenie BSON 6](#_Toc358654227)

CRUD

FileOperations - Operacja na plikach

Klasa FileOperations odpowiada za czynności wykonywane na plikach. Jest to klasa zawierająca najczęściej powtarzane czynności, zapewniająca tworzenie i modyfikację struktury danych. Struktura bazy danych tworzona jest w folderze domowym użytkownika w folderze „exampleDB”

Zaimplementowane funkcje:

* Wczytywanie bajtów z pliku reprezentujących BSON dokument.
* Wyszukiwanie pliku zawierającego daną wartość ObjectID.
* Wyszukiwanie pliku po nazwie pliku i nazwie kolekcji.
* Tworzenie pliku na dysku.
* Zwrócenie listy plików z danego folderu.

Selector -Wybór plików z zapytania

Klasa Selector posiada funkcje zwracające listę ID plików wymienionych w zapytaniu. Porównywane są dokumenty zapisane w plikach jak i dokument stanowiący pole selektora.

Insert - Zapisywanie plików na dysku

Funkcja odpowiedzialna za dodanie danych wprowadzonych przez użytkownika do bazy przyjmuje, jako argument listę dokumentów stanowiących pole obiektu InsertMessage. Dla każdego elementu zostaje wywołana metoda zapisująca dokument na dysk w postaci bajtów. W zależności czy dana kolekcja wymieniona w InsertMessage istnieje, tworzona jest nowa, bądź aktualizowana istniejąca. Pozostałe operacje opierają się o wyżej opisaną klasę FileOperations.

Delete - Usuwanie plików z dysku

Delete dzieli się na dwa etapy. Pobranie przez Selector listy plików wybranych przez użytkownika do usunięcia oraz wyszukanie wybranych plików i ich usunięcie.

Update - Modyfikacja plików na dysku

Realizacja update polega na pobraniu nazwy pliku do zmodyfikowania z updateMessage i wyszukaniu go na dysku. Następnie dane są porównywane. Modyfikacja to stworzenie nowego dokumentu zawierającego niekonfliktowe pola oraz dodanie modyfikacji z updateMessage. Na koniec kasowany jest stary plik i tworzony nowy zawierający zmodyfikowany dokument.

Komunikacja i przetwarzanie wiadomości

Ogólny zarys

Ważną cechą naszego projektu jest możliwość integracji z innymi instancjami mongoDB. Integracja ta opiera się na dostosowaniu się do protokołu sieciowego Mongo. Specyfikacja tego protokołu znajduje się http://docs.mongodb.org/meta-driver/latest/legacy/mongodb-wire-protocol/.

Aplikacja nasłuchuje połączeń na ustalonym porcie, odbiera i rozpoznaje kod wiadomości, a następnie reaguje na nie wykonując odpowiednie czynności CRUD. Treść wiadomości jest zróżnicowana i opisana w linku podanym powyżej. Jedyną cechą wspólną wszystkich wiadomości jest nagłówek, który przechowuje informację o długości wiadomości, adresacie i nadawcy wiadomości oraz o typie wiadomości. Wiadomości o następujących typach są obsługiwane w naszej bazie danych:

* Wiadomość o kodzie OP\_INSERT jest komendą, odpowiada za operację Create (tworzenia) nowych plików z danymi. W nim jest zawarta cała tablica z dokumentami przeznaczonymi do przechowania w bazie.
* Wiadomość o kodzie OP\_QUERY powoduje utworzenie kursora, który zwraca do klienta pewną ustaloną liczbę dokumentów, zgodnie z wzorcem zawartym w tej wiadomości.
* Wiadomość OP\_GET\_MORE odnosi się do konkretnego kursora, otrzymanie tej wiadomości powoduje odczytanie z kursora o pewnej ilości dokumentów i zwraca ją klientowi.
* Zarówno OP\_QUERY jak i OP\_GET\_MORE odpowiadają operacji Read.
* Wiadomość OP\_KILL\_CURSOR niszczy kursor.
* Wiadomość o kodzie OP\_UPDATE powoduje wykonanie operacji Update.
* Wiadomość o kodzie OP\_DELETE powoduje wykonanie operacji Delete.

Opis rozwiązania

Struktura

Rozwiązanie modułu messages składa się z następujących klas:

* MessageParser: odpowiada za rozpoznanie typu wiadomości korzystając z surowej tablicy bajtowej, dostarcza również funkcje parsujące tablice bajtowe na wiadomości o konkretnym typie.
* MessageHeader: przechowuje informacje o nagłówku wiadomości.
* KillCursorsMessage, DeleteMessage, GenericMessage, GetMoreMessage, InsertMessage, QueryMessage, ReplyMessage, UpdateMessage: przechowują informacje o danych zawartych w wiadomościach.

Parsowanie wiadomości

Parsowanie wiadomości polega na odczytywaniu kolejnych zmiennych z tablicy bajtowej. Nie jest to trudne, gdyż w specyfikacji jest ściśle ustawione gdzie znajdują się liczby, gdzie napisy, i gdzie znajduje się BSON. Istnieje kilka wyjątków, gdy nie da się z góry przewidzieć treści wiadomości – występuje on w wiadomości typu OP\_INSERT, OP\_QUERY i OP\_REPLY. Jedyne co wiadomo, to fakt że na końcu wiadomości może wystąpić pewna ilość dokumentów BSON ( w przypadku OP\_QUERY, maksymalnie jedna, w pozostałych przypadkach może ich być więcej ). Porównuję wówczas liczbę wczytanych bajtów z długością wiadomości. Jeżeli cała wiadomość nie została jeszcze wczytana, jest to znak że można wczytać jeszcze jeden dokument.

Dla prostoty implementacji przyjęliśmy założenie, że wszystkie wiadomości są poprawne, gdyż obsługa błędów syntaktycznych jest niekreatywna i nieciekawa, i nie pokrywa się z właściwym sensem projektu.

Wysyłanie komunikatów

Jedynym komunikatem wysyłanym przez nasz projekt jest komunikat typu OP\_REPLY. Komunikat OP\_REPLY jest tworzony w trakcie pracy kursora i zawiera pewną liczbę dokumentów pasujących do wzorca zapytania a także informację pomocnicze. Następnie jest on translowany do postaci binarnej za pomocą klasy ByteBuffer i wysyłany do klienta.

BSON

Czym jest BSON

BSON (skrót od Binary JSON) jest binarnym formatem stosowanym do przechowywania dokumentów. Każdy dokument składa się z zestawu elementów typu nazwa-wartość. Gdzie wartość może należeć do wielu różnych typów (takich jak int, double, string) w tym typów złożonych takich jak dokument w dokumencie czy tablica elementów.

Zastosowanie BSON

W naszym projekcie BSON stosowany jest do przesyłania, zapisywania i przeszukiwania zawartości bazy danych. Pole "\_id", które każdy BSON zawiera, stosowane jest, jako klucz główny ze względu na swoją unikalność.

Ogólny zarys

Rozwiązanie modułu BSON składa się z pięciu klas:

* BSON - klasa abstrakcyjna zawierająca statyczne metody służące do parsowania tablicy bytów zawierającej BSON na BSONDocument
* BSONDocument - sparsowany dokument BSON, zawierający obiekty BSONElement różnych typów
* BSONElement - pojedyncza para nazwa - wartość
* BSONtype - wyliczenie możliwych typów wartości zawartych w BSON
* ObjectID - pole ObjectID dokumentu BSON, zawiera 4 numeryczne pola: czas utworzenia, id maszyny, id procesu, licznik

Najważniejszymi elementem rozwiązania jest klasa BSON. Zawarte w niej metody parseBSON i getBSON stanowią podstawę działania całego pakietu.

Parsowanie BSON

Parsowanie BSON (czyli przejście z tablicy bajtów na obiekt klasy BSONDocument) wykonywane jest przez metodę parseBSON. Najpierw odczytywane jest pierwsze pole dokumentu BSON, czyli jego długość.

Następnie dla każdego elementu (pary nazwa-wartość) zawartego w dokumencie proces wygląda tak samo. W pierwszej kolejności odczytywany jest jego typ i nazwa. Parsowanie wartości przechowywanej w elemencie jest zależne od typu i jest wykonywane przez metodę parse klasy BSON.

Tworzenie BSON

Tworzenie BSON (czyli tablicy bajtów z obiektu klasy BSONDocument) jest procesem odwrotnym do jego parsowania. W pierwszej kolejności tworzona jest lista, w której będą przechowywane tablice bajtów odpowiadające poszczególnym elementom zawartym w dokumencie.

Kiedy wszystkie elementy są już przetworzone na tablice bajtów (zajmuje się tym metoda getByte) obliczana jest łączna długość wszystkich tablic. W ostatnim kroku wszystkie tablice łączone są w jedną całość, do której dodawana jest na początku łączna długość i na końcu bajt zerowy.