Bazy noSQL’owe ze szczególnym uwzględnieniem MongoDB

Co to jest MongoDB?

MongoDB *jest bazą danych typu noSQL*

Ale czym jest noSQL?

Jak tłumaczyć noSQL?

noSQL można przetłumaczyć jako:

NO to SQL – czyli „**NIE dla SQL**”

ale lepsze tłumaczenie to:

Not Only SQL – czyli „**Nie Tylko SQL**”

Czym jest noSQL?

Bazy noSQL to systemy nieoparte na tradycyjnym modelu relacyjnym.

Nie jest to jedna spójna technologia, ale raczej **cała klasa systemów bazodanowych stojących w opozycji do modelu relacyjnego**.

Cechy wspólne wielu baz noSQL

Można przechowywać w nich dane, w bardzo elastyczny sposób.

Często dają się **łatwo skalować wszerz**.

Zazwyczaj są **open source**.

Często **unikają JOIN'ów**.

Mogą **nie gwarantować pełnej transakcyjności**.

Wiele baz noSQL posiada powyższe cechy, lecz nie należy traktować tego wyliczenia jako reguły lub wyznacznika...

Klasyfikacja baz noSQL

1. Klucz–wartość

Jest szczególnie prosty i zbliżony do słownika lub mapy.

Czasem przechowywanymi wartościami są nie tylko pojedyncze wartości, ale także listy lub zbiory wartości.

Amazon Dynamo (Amazon)

Memcached (Wikipedia, Twitter, YouTube)

Redis (Twitter, GitHub, StackOverflow)

Klasyfikacja baz noSQL

2. Kolumnowe

Podobne do klucz-wartość z tym, że klucz jest tablicą, a wartość dowolnym ciągiem bajtów.

Klucz zazwyczaj składa się z tzw. klucza wiersza, klucza kolumny i sygnatury czasowej.

Google BigTable (Google)

Apache Cassandra (Apache)

Klasyfikacja baz noSQL

3. Dokumentowe

Podobne do klucz-wartość z tym, że wartość jest tzw. dokumentem, czyli zależnie od wybranej implementacji – najczęściej pewnym tekstowym zapisem danych, np. w postaci XML lub JSON.

Dane wewnątrz dokumentu mają strukturę hierarchiczną, a każdy szczebel składa się z dowolnej liczby par klucz-wartość.

MonogDB (Adobe, Facebook, Ebay, Nokia, Google)

CouchDB (Ubuntu one)

Klasyfikacja baz noSQL

4. Grafowe

Wspierają połączenia między danymi, ale nie są zdefiniowane za pomocą kluczy głównych i obcych, lecz przy pomocy węzłów grafu, które są z założenia równorzędne i mogą być w dowolny sposób i w dowolnej chwili wiązane z innymi węzłami.

Neo4j (Cisco, Gamesys, Ebay, Linkedin)

Klasyfikacja baz noSQL

5. XML

Bazy traktujące plik w formacie xml jako źródło danych. W momencie gdy niektóre bazy relacyjne wprowadziły typ kolumny XML i umożliwiły obsługę xPath'ów, rozwiązanie to stało się mało popularne.

dbXML

Klasyfikacja baz noSQL

6. Obiektowe

Bazy bazujące na modelu obiektowym. W praktyce jednak okazały się mało wydajne.

ObjectDB

Dlaczego skupiamy się na MongoDB?

Ponieważ jest aktualnie najbardziej popularny spośród baz noSQL.

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

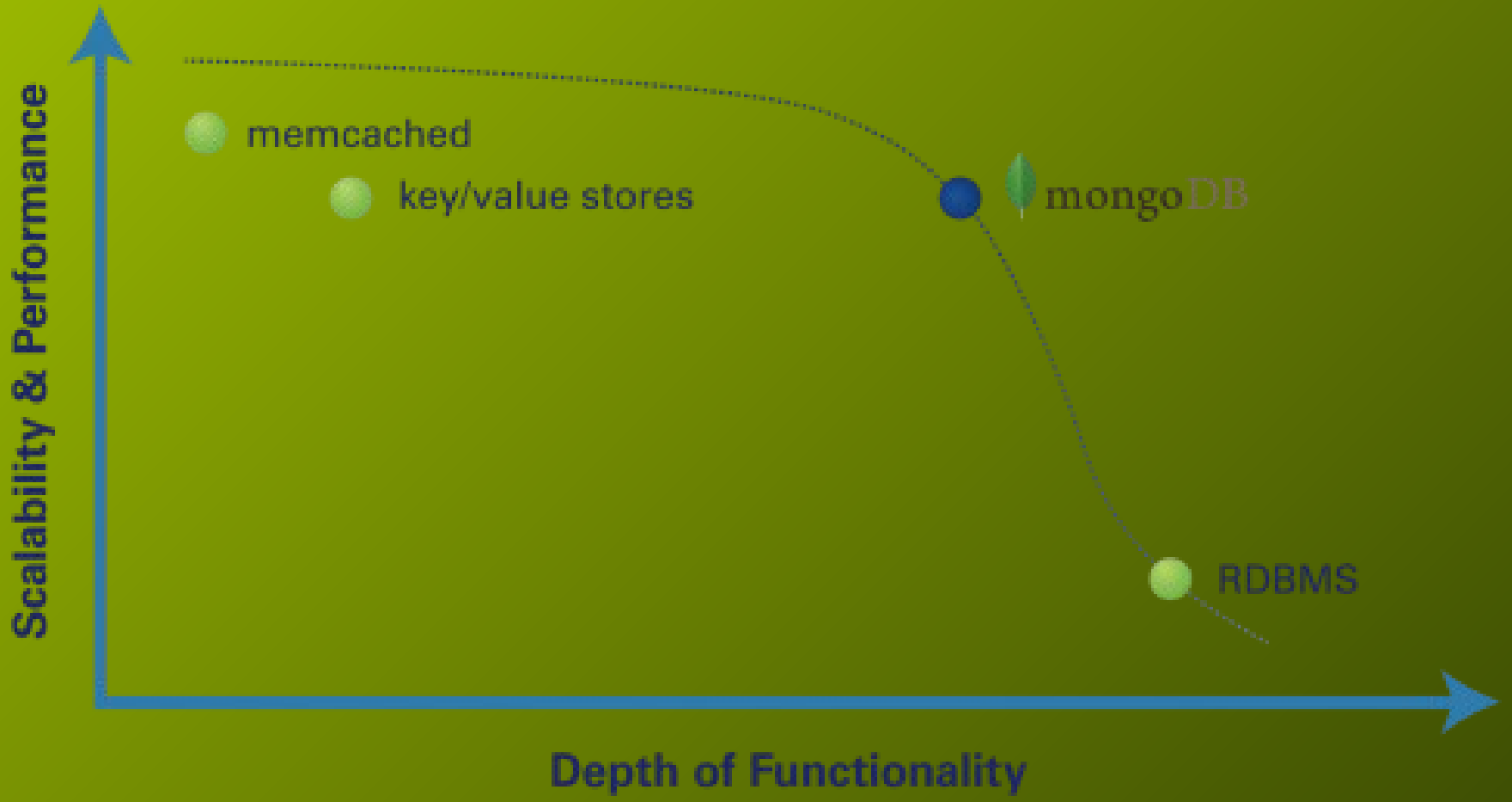
trend chart

299 systems in ranking, March 2016

Rank Mar 2016	Rank Feb 2016	Rank Mar 2015	DBMS	Database Model	Score Mar 2016	Score Feb 2016	Score Mar 2015
1.	1.	1.	Oracle	Relational DBMS	1472.01	-4.13	+2.93
2.	2.	2.	MySQL	Relational DBMS	1347.71	+26.59	+86.62
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1136.49	-13.73	-28.31
4.	4.	4.	MongoDB	Document store	305.33	-0.27	+30.32
5.	5.	5.	PostgreSQL	Relational DBMS	299.62	+10.97	+35.19
6.	6.	6.	DB2	Relational DBMS	187.94	-6.55	-10.91
7.	7.	7.	Microsoft Access	Relational DBMS	135.03	+1.95	-6.66
8.	8.	8.	Cassandra	Wide column store	130.33	-1.43	+23.02

http://db-engines.com/en/ranking

Relacja MongoDB do innych BD



MongoDB z bliska

MongoDB jest napisana w C++ i jest dostępne pod Windows, Linux, OSX i Solaris na licencji GNU AGPL v3.0. oraz komercyjnej.

Posiada sterowniki dla popularnych platform (PHP, Java, C#, Phyton, Ruby,...).

Powiązanie terminologii RDBMS z MongoDB

baza danych – baza danych
tabela – kolekcja
krotka/wiersz – dokument
kolumna – pole
złączenia tabel – osadzone dokumenty
klucz główny – klucz główny (domyślne pole _id)
index – index
relacja 1:n – zagnieżdżenie
relacja n:n – tablica referencji

MongoDB, a JavaScript

Do komunikowania się z bazą służą polecenia oparte o **JavaScript** oraz obiekty **JSON**.

Shell MongoDB interpretuje JavaScript, można więc np. używać pętli, deklarować zmienne, przypisywać do nich pobrane dane, zmodyfikować je i z powrotem wprowadzić do BD...

JSON / BSON

MongoDB wewnątrz przechowuje dane (dokumenty) w kolekcjach, które są obiektami **JSON**, a ściślej mówiąc **BSON**, czyli obiektami JSON, przechowywanymi w postaci binarnej i rozszerzonymi o definicję typu każdej wartości.

Typy danych wspierane przez BSON: **string**, **integer** (32- i 64-bit), **double** (64-bit), **date** (Unix Timestamp), **byte array** (binary data), **boolean** (true i false), **null** i inne...

Zorientowany dokumentowo

Nie musimy definiować struktury bazy danych, mongo samo ją utworzy na podstawie danych, które przekazujemy do bazy ;-)

Dokumenty to JSON'y – można zapomnieć o migracjach ;-)

Każdy dokument można dowolnie modyfikować, tzn. nie trzeba się troszczyć o schemat kolekcji – można zapomnieć o ALTER'ach ;-)

Możliwość osadzania dowolnie zagłębionych JSON'ów – można zapomnieć o JOIN'ach ;-)

Wstawianie danych

```
var document = {"name" : "Jan Kowalski"};
db.kolekcja.insert(document);


db.students.insert({
    "name" : "Cezary Cezary",
    "scores": [
        { type: "1st_exam", "score": 100},
        { type: "2nd_exam", "score": 80}
    ]
});
```

Odczyt danych

```
db.students.find();

db.students.find().count();

db.students.find({"name" : "Cezary Cezary"});

db.students.find({}, {name: 1}).sort({name: 1}).limit(6);
```

Aktualizacja danych

```
db.students.update(
    { "name": "Cezary Cezary"},
    { $push:
        { "scores":
            {
                type: "2nd_exam",
                "score": 90
            }
        }
    }
);
```

Usuwanie danych

```
db.students.remove();

db.students.remove({ "name": "Cezary Cezary"});

db.students.remove({ "name": "Cezary Cezary"}, 2);
```

Usuwanie danych

```
db.students.remove();

db.students.remove({ "name": "Cezary Cezary"});

db.students.remove({ "name": "Cezary Cezary"}, 2);
```

Kolejne przykłady

Ilość osób o imieniu rozpoczynającym się do G:

```
db.students.count({ name :/G/ });
```

Tablica różnych imion studentów:

```
db.students.distinct("name");
```

Agregacja – struktura dokumentu


```
{
  "_id" : 19,
  "name" : "Gisela Levin",
  "scores" : [
      {
        "type" : "exam",
        "score" : 44.51211101958831
      },
      {
        "type" : "quiz",
        "score" : 0.6578497966368002
      },
      {
        "type" : "homework",
        "score" : 93.36341655949683
      },
      {
        "type" : "homework",
        "score" : 49.43132782777443
      }
  ]
}
```

Lista niektórych przełączników

$project - wybranie tylko określonych pól z dokumentów

$match - filtruje dokumenty wedle okreslonego kryterium

$group- dokonuje grupowania elementów, jest to faktyczne miejsce agregowania wyników

$sort - sortowanie dokumentów według zdefiniowanych pól

$skip - pomija n dokumentów z listy

$limit - określa maksymalną liczbę dokumentów, które mają zostać przetworzone

$unwind - używane gdy chcemy zastąpić dokument zawierający tablicę wartości, zbiorem dokumentów, które w polu reprezentującym rozwijaną tablicę będą posiadały kolejne elementy wcześniejszej tablicy

Przełączniki – przykład 1

Imiona i nazwiska studentów w kolejności alfabetycznej:

```
db.students.aggregate({
    $project: {
        name: 1,
        _id: 0
    }
}, {
    $sort: {
        name: 1
    }
});
```

Przełączniki – przykład 2

Dziesięciu studentów o najwyższej sumie z egzaminów:

```
db.students.aggregate({
    $unwind: "$scores"
}, {
    $group: {
        _id: "$name",
        scores_sum: {
            $sum: "$scores.score"
        }
    }
}, {
    $sort: {
        scores_sum: -1
    }
}, {
    $limit: 10
});
```

Jak użyć MongoDB w PHP?

```
<?php

// connect
$mongo = new MongoClient();

// select a database
$db = $mmongo->comedy;

// select a collection (analogous to a relational database's table)
$collection = $db->cartoons;

// add a record
$document = array( "title" => "Calvin and Hobbes", "author" => "Bill Watterson" );
$collection->insert($document);

// add another record, with a different "shape"
$document = array( "title" => "XKCD", "online" => true );
$collection->insert($document);

// find everything in the collection
$cursor = $collection->find();

// iterate through the results
foreach ($cursor as $document) {
    echo $document["title"] . "\n";
}

?>
```

Zalety

Wspiera **łatwe skalowanie wszerz** i **replikację**

Kolekcje przechowują dane w postaci **dokumentów – bardzo elastyczna** struktura danych

Operacje **CRUD** na danych, **to operacje na JSON'ach**

Shell **obsługuje JavaScript**

Posiada mechanizmy do **walidacji** poprawności dokumentów

Ma wsparcie dla zapytań do **zagnieżdżonych** pól dokumentów

Wspiera standardową **agregację** danych oraz tworzenie **własnych agregacji** dzięki **Map-Reduce**

Zalety

Wspiera **indeksy** dla dowolnie zagnieżdżonych kluczy, dodatkowo automatycznie nadaje **klucz główny** każdemu dokumentowi

Ma wsparcie dla **sortowania** wyników

Ma wsparcie dla **Unicode**

Obsługuje dużą liczbę **typów danych**

Optymalizuje pracę **przechowując wyniki zapytań w RAM'ie**

Wspiera **administrację** bazą danych i **bezpieczny dostęp**

Wady

Ograniczone możliwości transakcji – transakcyjne są jedynie atomowe operacje CRUD

Maksymalna **pojemność dokumentu to 16MB** – można to łatwo „obejść” stosując w MongoDB **GridFS**

Przy **sortowaniu brak wsparcia dla UTF-8**

Zazwyczaj **wolniejsze od RDBMS** gdy w logice aplikacji przeważają operacje **wstawiania i modyfikowania danych**

Pożeracz przestrzeni dyskowej – dane zazwyczaj nie są znormalizowane

Na początku nieintuicyjna składnia przy bardziej zaawansowanych operacjach – kwestia nauki MongoDB

Gdzie warto użyć?

Serwisy internetowe

Np. to portale społecznościowe, portale informacyjne, serwisy o globalnym zasięgu, cechujące się dużą ilością odczytów z bazy w stosunku do operacji zapisów.

MongoDB pozwala na wydajne skalowanie np. na podstawie położenia geograficznego.

Gdzie warto użyć?

BigData (klikmapy, logowanie)

Analiza bardzo dużej ilość informacji o różnej strukturze.

MongoDB świetnie sprawdza się w tej roli dzięki elastycznym schematom danych oraz możliwości zapisu danych bez potwierdzania.

Gdzie warto użyć?

W sytuacjach gdy zapisujemy stan obiektu, który zazwyczaj przyjmuje tylko kilka spośród wielu rozmaitych cech.

Np. kartoteka pacjenta, cechy towaru...

Gdzie warto użyć?

Gdy przewidujemy, że w przyszłości nasza baza danych będzie musiała się skalować wszerz.

Gdzie warto użyć?

Jako Cache lub jako storage

W sytuacjach, gdy nie musimy mieć zapewnionej pełnej transakcyjności.

Kiedy unikać?

Systemy wymagające złożonych transakcji

Np. systemy bankowe

Kiedy unikać?

Aplikacje w których RDBMS jest naturalnym kandydatem

Wiele problemów jest łatwo przedstawić jako schemat relacyjnej bazy danych.

Jeśli dodatkowo ilość zapisów do bazy danych jest duża to MongoDB traci część swoich zalet.

Linki

Manual MongoDB
https://docs.mongodb.org/manual/

MongoDB University
https://university.mongodb.com/

Dziękuję za uwagę