Bazy danych - NoSQL

Michał Szymański

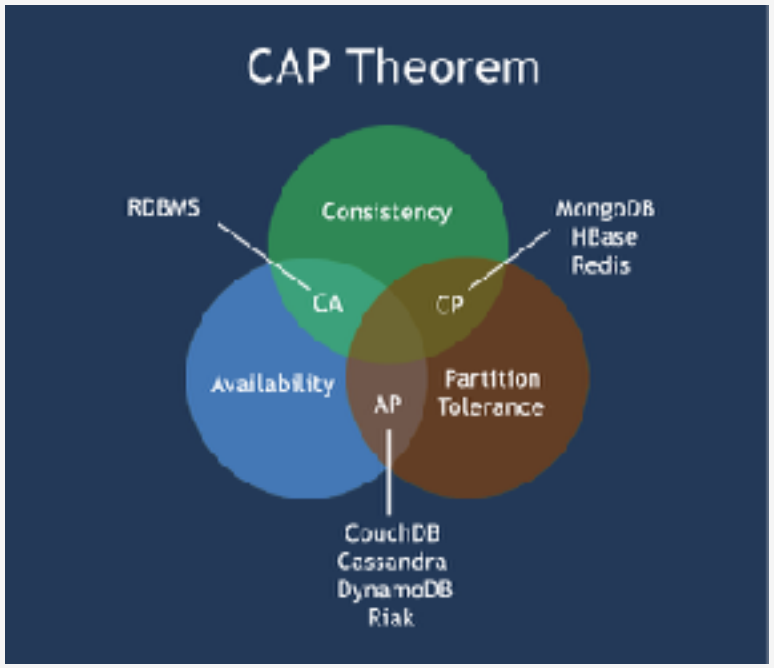
NoSQL dla kogo

- Skalowalność
- Elastyczność w przechowywaniu danych
- Łatwiejszy w użytkowaniu
- Wyższa wydajność

..ale

- Brak standardu języka dla NoSQL
- Zwykle nie są to bazy transakcyjne

Twierdzenie CAP



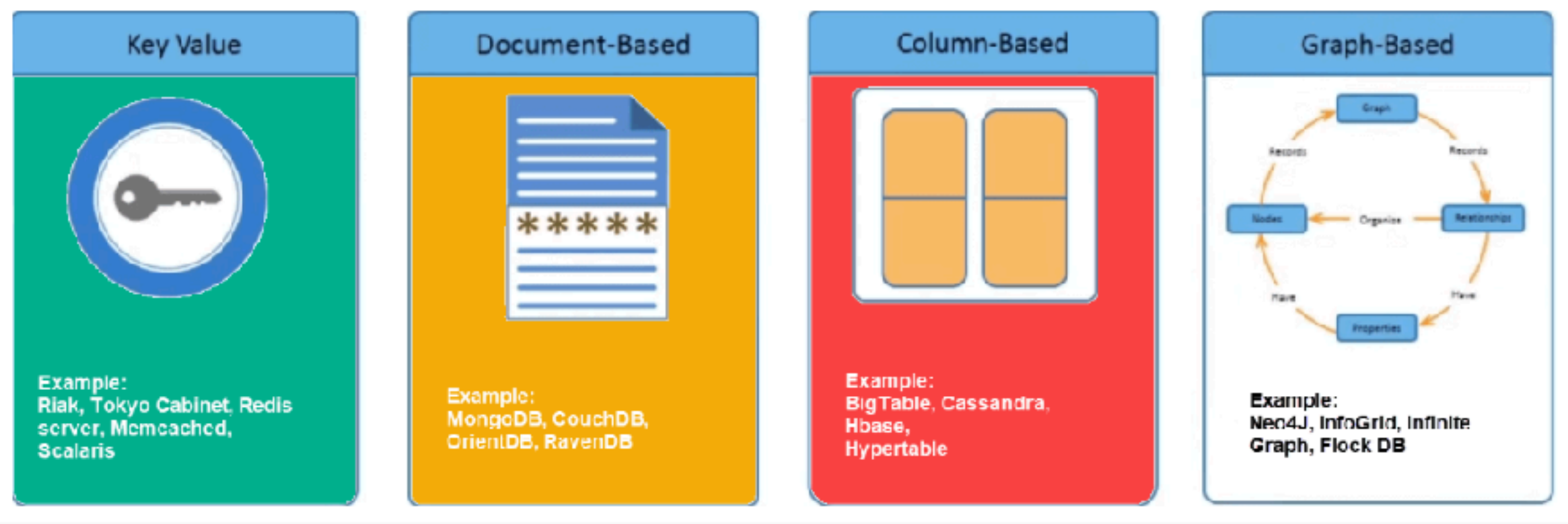
Twierdzenie mówiące o tym, że w systemach gdzie dane są przechowywane w sposób rozproszony da się osiągnąć jedynie 2 z 3 „właściwości":

Spójność: Każdy klient w każdej chwili czasowej ‚widzi' takie same dane.

Dostępność: Każde klient może czytać i zapisywać, nie może być przerw w dostępie.

Odporność na partycjonowane rozbicia: System będzie działał właściwie niezależnie od utraty komunikacji z innymi partycjami.

NoSQL – typy baz danych



Źródło: https://www.guru99.com/nosql-tutorial.html

NoSQL – Key-Value

Klucz	Wartość
GD1234	Jan Kowalski
GD3234	Piotr Nowak
GD2344	Julian Tuwim
…	…

NoSQL – Key-Value

```
package org.infinispan.tutorial.simple.map;

import org.infinispan.Cache;
import org.infinispan.configuration.cache.ConfigurationBuilder;
import org.infinispan.manager.DefaultCacheManager;

public class InfinispanMap {

   public static void main(String[] args) {
      // Construct a simple local cache manager with default configuration
      DefaultCacheManager cacheManager = new DefaultCacheManager();
      // Define local cache configuration
      cacheManager.defineConfiguration("local", new ConfigurationBuilder().build());
      // Obtain the local cache
      Cache<String, String> cache = cacheManager.getCache("local");
      // Store a value
      cache.put("key", "value");
      // Retrieve the value and print it out
      System.out.printf("key = %s\n", cache.get("key"));
      // Stop the cache manager and release all resources
      cacheManager.stop();
   }

}
```

Document Store - MongoDB, główne cechy

- Zorientowane na przechowywanie dokumentów JSON
- Możliwość zadawania pytań
- Indeksowanie pól
- Replikacja danych
- Load balancing
- Funkcje agregujące
- Rozwiązanie skalowalne
- OpenSource

JSON - wstęp

JavaScript Object Notation – lekki tekstowy format wymiany danych. Pomimo nazwy, JSON jest formatem niezależnym od konkretnego języka. Wiele języków programowania obsługuje ten format danych przez dodatkowe pakiety bądź biblioteki.

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

JSON – typy danych

- Ciąg tekstowy
- Liczba
- Obiekt
- Tablica
- Boolean - „true" / „false"
- null

JSON - podstawy

- Dane są przechowywane w postaci pary klucz/wartość

```
"id" : "01",
"tytul": "Powstanie 1944",
"autor_imie": "Władysław",
"cena": "20.50"
```

- Obiekty przechowywane są w nawiasach klamrowych

```
{
      "id": "01",
      "tytul": "Powstanie 1944"
}
```

JSON - obiekty

Obiekt może być zagnieżdżony w innym obiekcie

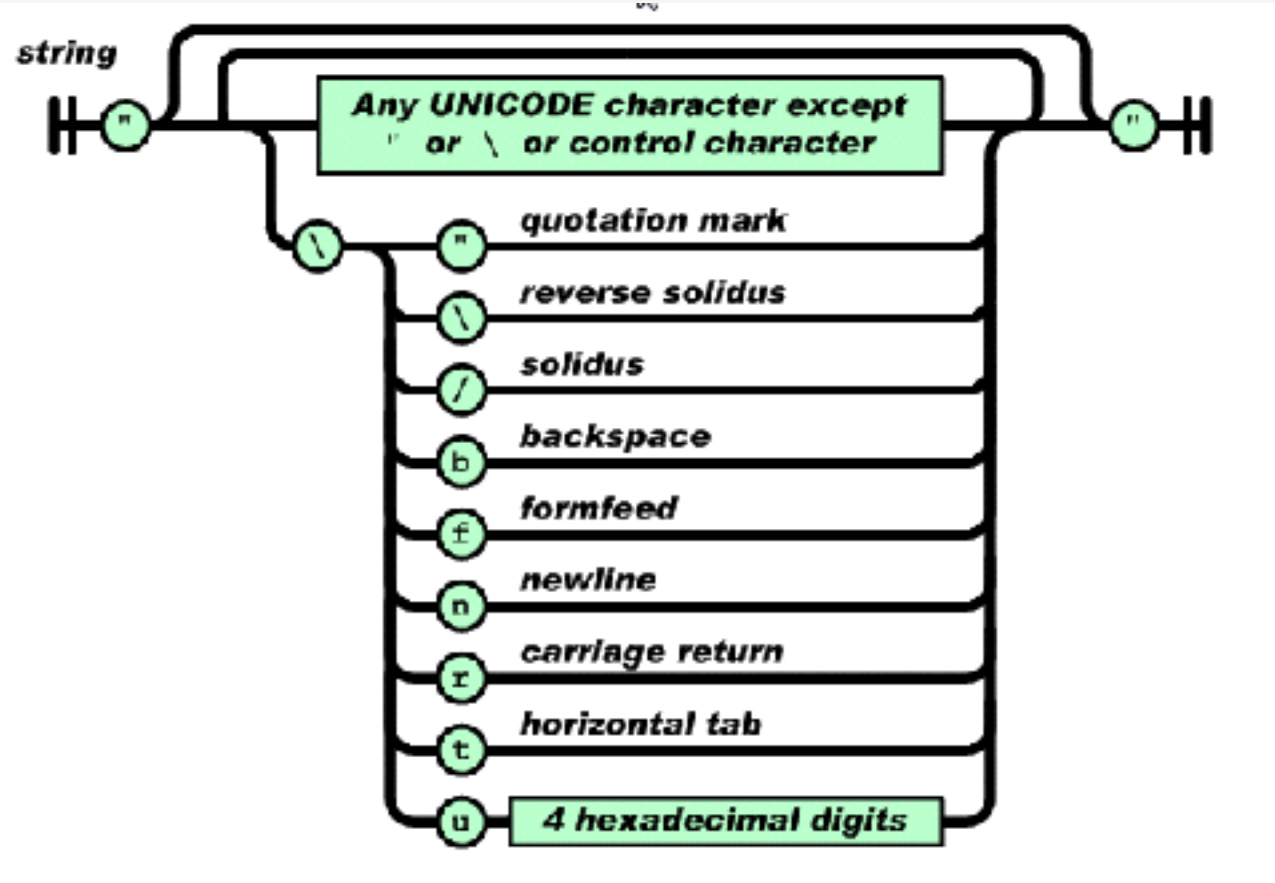
```
{
    NazwaKlienta: "Kowalski Sp.z.o.o",
    Zamowienie:
        {
            ZamowienieID:  234,
            NazwaProduktu:  "śruby 20mm",
            Ilosc: 5
        }
}
```

JSON - tablice

- Tablice definiowane są przez nawiasy kwadratowe. Elementy przedzielone są przecinkami.

```
{ ksiazki: [
        {
          id: 1,
          tytul: "Powstanie1944",
          autor_imie: "Władysław"
        },
        {
          id: 2,
          tytul: "Wesele",
          autor_imie: "Stanisław"
        }
    ]
}
```

JSON – podstawy, znaki specjalne



JSON - narzędzia

- Edytor – dowolny taki aby było podświetlanie np. Notepad++
- Wsparcie do JSON w InteliJ
- Walidowanie i formatowanie JSON na sieci
  https://jsonformatter.curiousconcept.com/

Ćwiczenie 1:

Sprawdzić jak się wyświetla zawartość pliku „cwiczenie1.json"

JSON - przykład

Ćwiczenie 2:

Stworzenia JSON’a przechowującego artykuł na bloga.

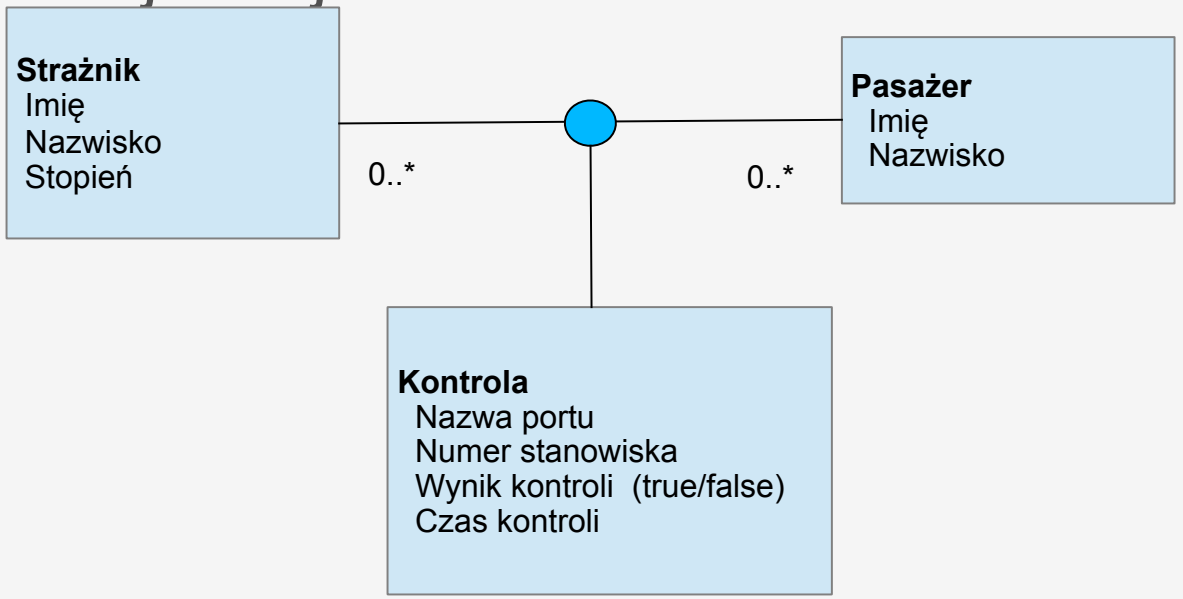
Pojedynczy wpis powinien zawierać:

- Unikalny identyfikator
- Tytuł
- Treść artykułu
- Data publikacji
- Lista tagów
- Ilość polubień
- Informacje o autorze – imię i nazwisko

JSON

Ćwiczenie 3

Stworzyć JSON przechowujący informacje o kontrolach na lotnisku. Wpisać dane dla jednej kontroli.



MongoDB – podstawy

RDBMS	MongoDB
Baza danych (Database)	Baza danych (Database)
Tablica (Table)	Kolekcja (Collection)
Wiersz (Row)	Dokument (Document)
Kolumna (Column)	Pole (Field)

MongoDB – cechy bazy

- Każda baza zawiera kolekcje dokumentów.
- Każdy dokument w kolekcji może być inny - różna liczba pól.
- Struktura dokumentu może być zgodny z strukturą klas.
- Schemat dokumentów nie musi być tworzony wcześniej - tworzony jest w momencie dodawania dokumentu do kolekcji.
- Brak jest złożonych połączeń (ang.join).

MongoDB – instalacja Windows

1) Wybieramy community server, system pobieramy z: https://www.mongodb.com/download-center/community
2) Instalujemy „Complete" , **bez** Compass
3) Przenosimy z *C:\Program Files\MongoDB\Server\4.x\bin* do *C:\MongoDB*
4) Uruchamiamy *cmd* i wchodzimy do katalogu *C:\MongoDB*
   C:\Users\mszymans>cd c:\MongoDB
5) Tworzymy katalog C:\MongoDB\Data
   c:\MongoDB>md data
6) c:\MongoDB> mongod.exe --dbpath "c:\MongoDB\data"
   2018-06-19T08:35:17.599+0200 I CONTROL [initandlisten] MongoDB starting : pid=8
   .....
   2018-06-19T08:35:18.870+0200 I NETWORK [initandlisten] **waiting for connectionson** port 27017
7) W oddzielnym okienku odpalamy *mongo.exe*
8) Instalujemy klienta https://robomongo.org/download

MongoDB Shell

MongoDB Shell - interaktywny interfejs dostępu do MongoDB, który pozwala zadawać zapytania, aktualizować dane i wykonywać operacje administracyjne.

MongoDB – db.stats()

Wyświetlenie informacji o aktualnej bazie, ilość kolekcji itp: **db.stats()**

```
{
        "db" : "test",
        "collections" : 0,
        "views" : 0,
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "numExtents" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "fileSize" : 0,
         ..........
        "ok" : 1
}
```

MongoDB – tworzenie kolekcji / dokumentów

O czym warto pamiętać:

- Dane w dokumencie powinny być całością potrzebnych danych, tak żeby nie trzeba byłą łączyć dane z różnych dokumentów. Jeśli dane są rozłączne to trzymamy w oddzielnych dokumentach.
- Dane w dokumentach mogą być duplikowane, pamięć dyskowa jest tania.
- Jeśli są potrzebne łączenia danych (bazodanowe JOIN) to robimy je przy zapisie a nie przy odczytach.
- Optymalizujemy schemat dla najczęściej występujących zapytań.

MongoDB – operacje na bazie

Tworzenie bazy / podłączenie do istniejącej

```
use  mojabaza
```

Wyświetlenie istniejących baz (o ile jest przynajmniej jeden dokument):

```
show dbs
```

Usunięcie bazy do której jesteśmy połączeni :

```
db.dropDatabase()
```

MongoDB – dodawanie danych

Tworzenie kolekcji:

```
db.createCollection(name, options)
db.createCollection("mojakolekcja")
```

Wyświetlenie listy kolekcji:

```
show collections
```

Kasowanie kolekcji:

```
db.mojakolekcja.drop()
```

Ćwiczenie 3a - tworzenie i kasowanie bytów w MongoDB

MongoDB – dodawanie danych

Dodawanie dokumentu (przy okazji dodaje collection):

```
db.ksiazki.insert({" tytul" : "Powstanie 1944"})
```

Ćwiczenia 4a

Założyć bazę „Kurs", kolekcje „Uczestnicy" i umieścić jeden dokument z kursantem (imię, nazwisko)

MongoDB – podstawowe typy danych

- Double
- String
- Integer np. *{ilosc:23}*
- Arrays
- Binary – z użyciem *BinData()*
- Undefined np. *{tytul:undefined}*
- Boolean - true / false
- Null
- Date np. *{datazalozenia: new Date("2018-01-01")}*

MongoDB – klucz główny

_id - 12 bitowa szesnastkowa liczba, która gwarantuje unikalność każdego dokumentu w kolekcji. Jeśli nie zostanie podana w trakcie wstawiania zostanie automatycznie wstawiona (kombinacja aktualnego czasu, identyfikatora noda, id procesu i kolejnego numeru).

```
> db.Samoloty.find().pretty()
{
        "_id" : ObjectId("5d274778d1a764339548d77d"),
        "numer_rejestracyjny" : "DABV",
        "typ" : "Airbus"
}
```

MongoDB – wyszukiwanie danych

Ćwiczenie 4

Wstaw dane w oparciu o dane z pliku C*wiczenie4.json* do bazy *kurs*

MongoDB - find()

Podstawowa konstrukcja wyszukiwania:

```
db.kontrole.find().pretty()
```

odpowiednika *SELECT * FROM kontrole*

MongoDB – wyszukiwanie danych, podstawy

Operacja	Składnia	Przykład	Odpowiednik SQL
Przyrównanie	{<key>:<value>}	db.kontrole.find({"wynik_kontroli":"true"})	WHERE wynik_kontroli=true
Przyrównanie		db.kontrole.find({"straznik.imie":"Tomasz"})	Zapytanie ze złączeniem
Mniej niż	{<key>:{$lt:<value>}}	db.kontrole.find({"czas_kontroli":{$lt:50}})	WHERE czas_kontroli<50
Mniej lub równo	{<key>: {$lte:<value>}}	db.kontrole.find({"czas_kontroli":{$lte:50}})	WHERE czas_kontroli<=50
Większe niż	{<key>: {$gt:<value>}}	db.kontrole.find({"czas_kontroli":{$gt:50}})	WHERE czas_kontroli>50
Różne	{<key>: {$ne:<value>}}	db.kontrole.find({"czas_kontroli":{$ne: 50}})	WHERE czas_kontroli<>50

MongoDB – wyszukiwanie danych z AND

Składnia AND

```
db.collection.find( { $and: [
                                {key1: value1},
                                {key2: value2}
                              ]
                    } )
```

Przykład

```
db.kontrole.find( {  $and:[
                            {"straznik.imie":"Tomasz"},
                            {"pasazer.imie": "Jan"}
                           ]
                  }).pretty()
```

MongoDB – wyszukiwanie danych z OR

Składnia OR

```
db.collection.find( { $or: [
                                {key1: value1},
                                {key2: value2}
                               ]
                    } )
```

Przykład

```
db.kontrole.find ({  $or:[
                                {"straznik.imie": "Tomasz"},
                                {"straznik.imie": "Jan"}
                               ]
                   })
```

MongoDB – wyszukiwanie danych z OR i AND

Przykład AND oraz OR

```
db.kontrole.find({
                    $and:[   {"czas_kontroli":{$gt:70}},
                             {$or:[
                                    {"straznik.imie":"Tomasz"},
                                    {"straznik.imie": "Jan"}
                                ]
                             }
                         ]
                })
```

MongoDB – find cd..

Sprawdzanie czy jest wartość null albo gdzie nie ma pola ocena:

```
db.kontrole.insert( {"ocena": null } )
db.kontrole.find( {"ocena": null } )
```

Wyszukanie dokumentów które nie mają wybranego pola

```
db.kontrole.find( {"ocena": { $exists: false } } )
```

Kursor w MongoDB

Kursor pozwala na przeglądanie zwróconej listy wyników po elemencie.

```
var kursorKontrole = db.kontrole.find( { czas_kontroli : { $gt:2 }});

while(kursorKontrole.hasNext()) {
  print(tojson(kursorKontrole.next()));
}
```

MongoDB – replaceOne

Składnia

```
    db.COLLECTION_NAME.replaceOne(SELECTION_CRITERIA,
REPLACEMENT_DATA)
```

Przykłady

```
db.kontrole.find({"test" :  23})
db.kontrole.replaceOne({"nazwa_portu" :  "Warszawa" }, {"test":23})
db.kontrole.find({"test" :  23})
```

Wiele

```
var bulk = db.items.initializeUnorderedBulkOp();
bulk.find( { item: "abc123" } ).replaceOne( { item: "abc123", status:
"P", points: 100 } );
bulk.execute();
```

MongoDB – update

Składnia

```
db.COLLECTION_NAME.updateMany(SELECTION_CRITERIA, UPDATED_DATA)
```

Przykład

```
db.kontrole.updateMany(
            {"pasazer.imie": "Jan"},
            {$set:{"czas_kontroli":23}}
   )
db.kontrole.find({"pasazer.imie": "Jan"})
```

MongoDB – update

Bardziej złożony update

```
db.kontrole.updateMany(
        {$and:[
                {"straznik.imie":"Tomasz"},
                {"pasazer.imie": "Jan"}
                ]},
        {$set:{"czas_kontroli":21}}
    )
db.kontrole.find({$and:[{"straznik.imie":"Tomasz"},
{"pasazer.imie": "Jan"}]}).pretty()
```

MongoDB – remove

Składnia

```
db.COLLECTION_NAME.remove(SELECTION_CRITERIA)
```

Przykłady

```
db.kontrole.find({"pasazer.imie" : "Jan"})
db.kontrole.remove({"pasazer.imie" : "Jan"})
db.kontrole.find({"pasazer.imie" : "Jan"}).pretty()
```

MongoDB – projections

„Projections" pozwala wybrać tylko określone pola zamiast zwracania całego dokumentu.

```
db.COLLECTION_NAME.find (SELECTION_CRITERIA, RETURN_VALUES)
```

Przykłady, zwracane dwa pola (jedynka oznacza wyświetl)

```
db.kontrole.find({"pasazer.imie":"Stanislaw"}, {"KEY":1,
"nazwa_portu":1} )
```

MongoDB – indeksy

Wyszukiwanie bez użycia indeksów jest nieefektywne należy używać indeksów.

Indeks może być nałożony na jedno albo więcej pól indeksów.

Indeks tworzony jest z użyciem komendy *createIndex()*

1 – indeks porządek rosnący, istotne przy sortowaniu i range queries

-1 - indeks porządek malejący

Przykłady

```
db.kontrole.createIndex({"pasazer.imie":1})
db.kontrole.createIndex({„pasazer.imie":1},{"pasazer.nazwisko":1})
```

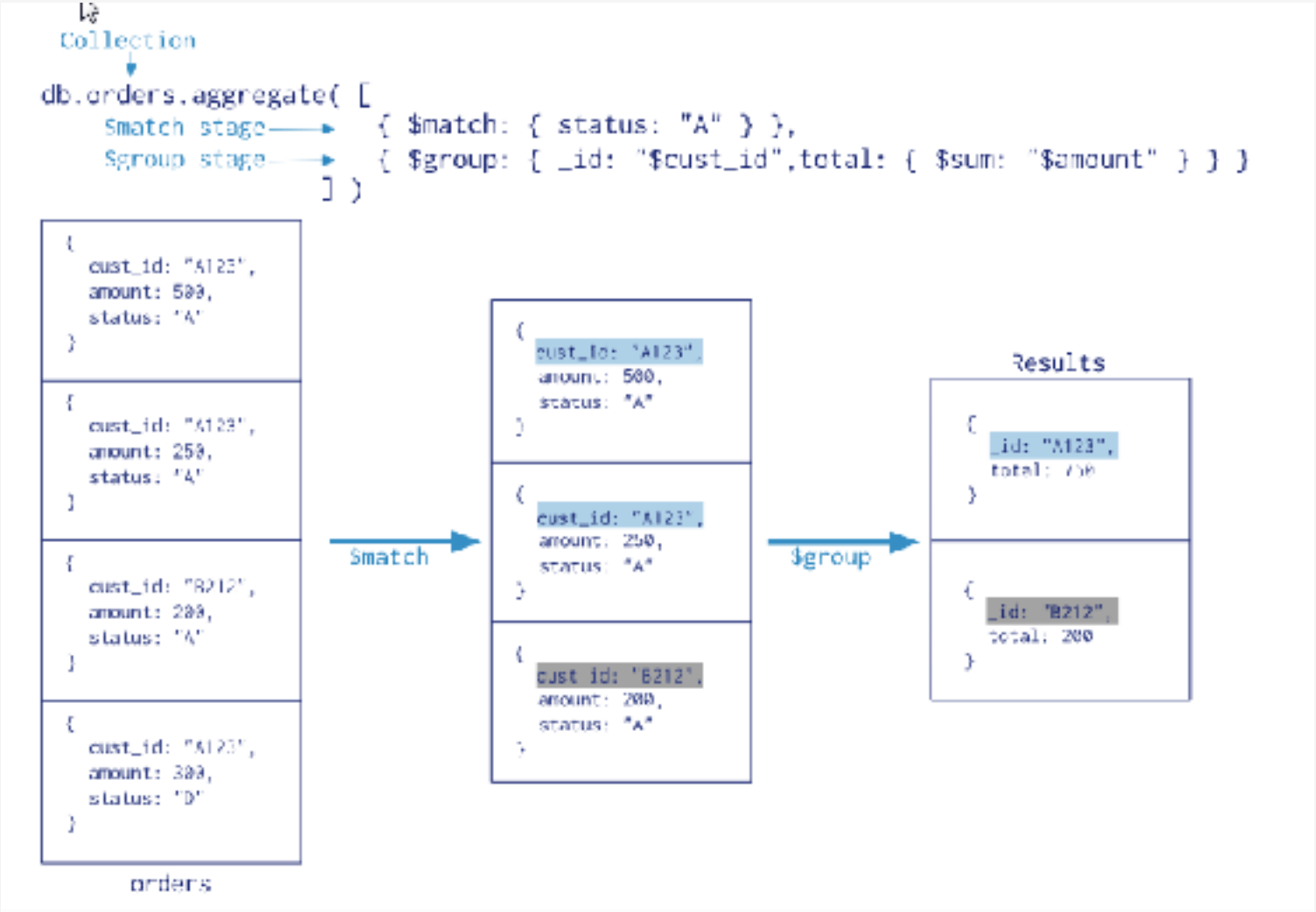
MongoDB – aggregation

Agregacja operacja działająca podobnie do operacji agregujących w SQL.

Wyrażenie	Description	Przykład
$sum	Sumuje wartości z wybranych dokumentów.	db.kontrole.aggregate([{$group : {_id : "$nazwa_portu", total: {$sum : "$czas_kontroli"}}}])
$avg	Liczy średnią z wybranych dokumentów.	db.kontrole.aggregate([{$group : {_id : "$nazwa_portu", total: {$avg: "$czas_kontroli"}}}])
$min / $max	Minimum i maksymim wartości z wybranych dokumentów.	db.kontrole.aggregate([{$group : {_id : "$nazwa_portu", total: {$min: "$czas_kontroli"}}}])

MongoDB – aggregation

Bardziej złożona składnia



MongoDB – sortowanie

Do sortowania używamy metody sort() gdzie podana jest lista pól po których będziemy sortować i dodatkowo z informacją czy sortowanie jest rosnące czy malejące.

1 – kolejność rosnąca

-1 – kolejność malejąca

Przykład

```
db.kontrole.find({}).sort({"nazwa_portu":1})
```

MongoDB – limit , skip

Ograniczenie listy wyników zapytania

```
db.kontrole.find().limit(3)
```

„Pomijanie” wyników zapytania

```
db.kontrole.find().skip(2)
```

Limit i skip razem

```
db.kontrole.find().skip(2).limit(1)
```

MongoDB – ćwiczenia

Ćwiczenie 5

Opis ćwiczenia znajduje się w pliku cwiczenie5.json

MongoDB – ćwiczenia

Ćwiczenie 6

Zaprojektowanie schematu przechowującego informacje o książkach w systemie podobnym do lubimyczytac.pl . Baza powinna zawierać dane o

- Podstawowe informacje o książce: imię i nazwisko autora, data wydania,kategoria, tagi opisujące książke.
- Informacje o wystawionych komentarzach dla książki (tablica obiektów), informacja kto wystawił i kiedy, jaką dał ocenę (w skali 1-10)
- Informacja o wydawcy (to jako oddzielny obiekt): nazwa wydawcy, data rozpoczęcia działalności wydawcy, miasto gdzie ma centralę wydawca.
- Baza w której trzymamy dokumenty powinna mieć nazwę „lubimyczytac" a kolekcja „ksiazki"
- Wprowadzić 3 przykładowe dokumenty i zaproponować przynajmniej 4 zapytania które mogą się przydać w działaniu platformy (tak żeby było użyte wyszukiwanie, sortowanie czyli czym więcej konstrukcji o której mówiliśmy tym lepiej)

MongoDB –

MongoDB – bezpieczeństwo

- Zdefiniowanie kontroli dostępu - stworzenie użytkowników, logowanie oparte o wybraną metodę uwierzytelniania.
- Zdefiniowanie ról dostępu - użytkownik przypisany jest do ról
- Zastosowanie szyfrowania do np.TSL / SSL do komunikacji między klientem a serwerem.
- Włączenie funkcjonalności audytu
- Uruchamianie MongoDB z dedykowanego użytkownika

MongoDB – backup

Możliwe metody wykonania backup'u:

- **Kopiowanie plików danych MongoDB**
- **Użycie mongodump / mongorestore , tworzy plik BSON (binary JSON)**
  https://docs.mongodb.com/manual/reference/program/mongodump/

MongoDB – monitoring

Narzędzia do monitoringu bazy:

- **mongostat - ilość operacji (insert, query, update) które są wykonywane w danym momencie na serwerze**

```
$ mongostat
connected to: 128.0.1.1
insert  query update  delete  getmore  command  flushes  mapped  vsize  res  faults  locked db  idx miss %  qr|qw  ar|aw  netIn  netOut  conn
  1000      1   1103    1805       1      1|0        0    320m   808m  54m      32  test:84.6%           0    0|0    0|1   321k    91k     2
  2000      1   1229    1187       1      1|0        0    320m   808m  54m     102  test:83.9%           0    0|0    0|1   279k    54k     2
  1000      1    964     995       1      1|0        0    320m   808m  54m      24  test:85.2%           0    0|0    0|1   243k    49k     2
```

- **mongotop - czas zapisów / odczytów do kolekcji**

```
$ mongotop
connected to: 128.0.1.1
                    ns        total        read        write
                                                    2016-03-15T14:43:23
test.monitoringExample      902ms        1ms         901ms
test.system.users           0ms          0ms         0ms
test.system.namespaces      0ms          0ms         0ms
test.system.js              0ms          0ms         0ms
test.system.indexes         0ms          0ms         0ms
```

MongoDB – tworzenie użytkownika

Tworzenie użytkownika z shella:

```
db.createUser(
{  user: "Guru99",
   pwd: "password",

   roles:[{role: "userAdminAnyDatabase" , db:"admin"}]})
```

Dostępne domyślne role:
https://docs.mongodb.com/manual/reference/built-in-roles/

Podstawowe:

- read
- readWrite
- dbAdmin

MongoDB – Replikacja

Co to jest?

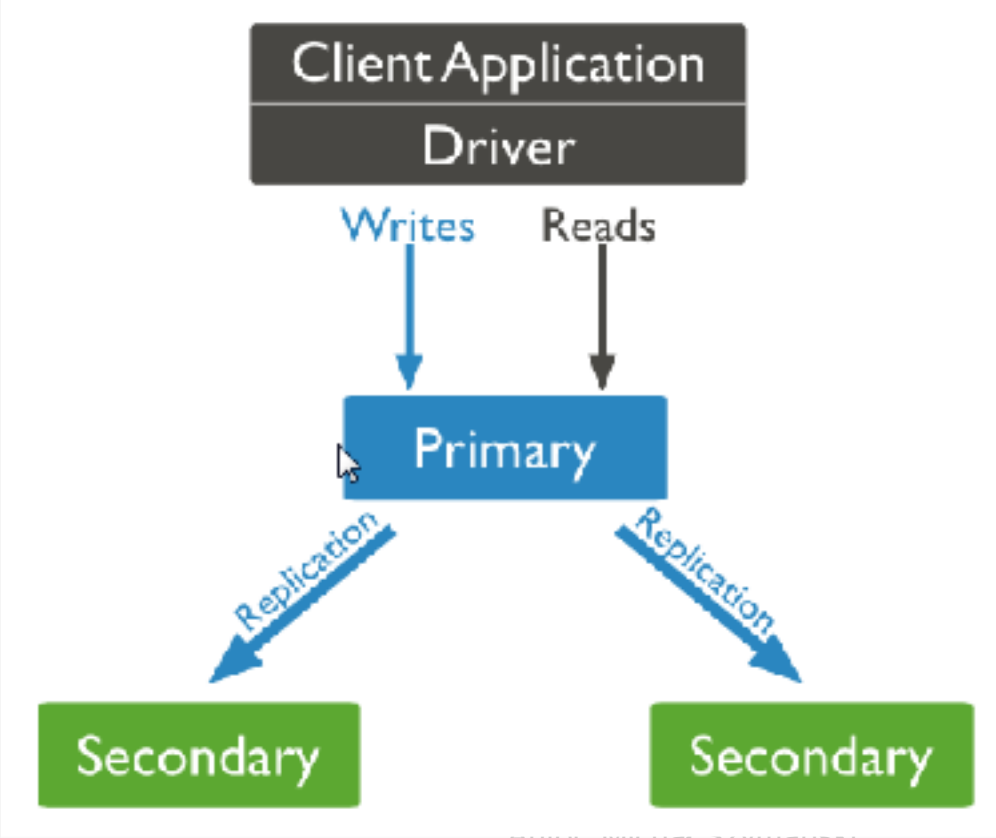
Proces synchronizacji danych pomiędzy różnymi serewera.

Co zyskujemy?

- Ze względów bezpieczeństwa, awaria pojedynczego serwera nie powoduje utraty danych i utraty dostępności.
- Zwiększenie wydajności odczytów.
- Dodatkowa kopia może być użyta np. do tworzenia raportów albo być traktowana jako backup.

MongoDB – Replikacja, jak to działa?

- Grupa serwerów MongoDB (mongod) przechowujących te same dane.
- Jeden node otrzymuje wszystkie zapisy i dystrybuje dane do reszty.



MongoDB – Replikacja, jak to działa?

- Każda instalacja replikacji może składać się z dowolnej ilości nodów
- Jest tylko jeden node główny („Primary")
- Działa mechanizm automatic failover
- Działa mechanizm automatic recovery

MongoDB – Sharding

Co to jest?

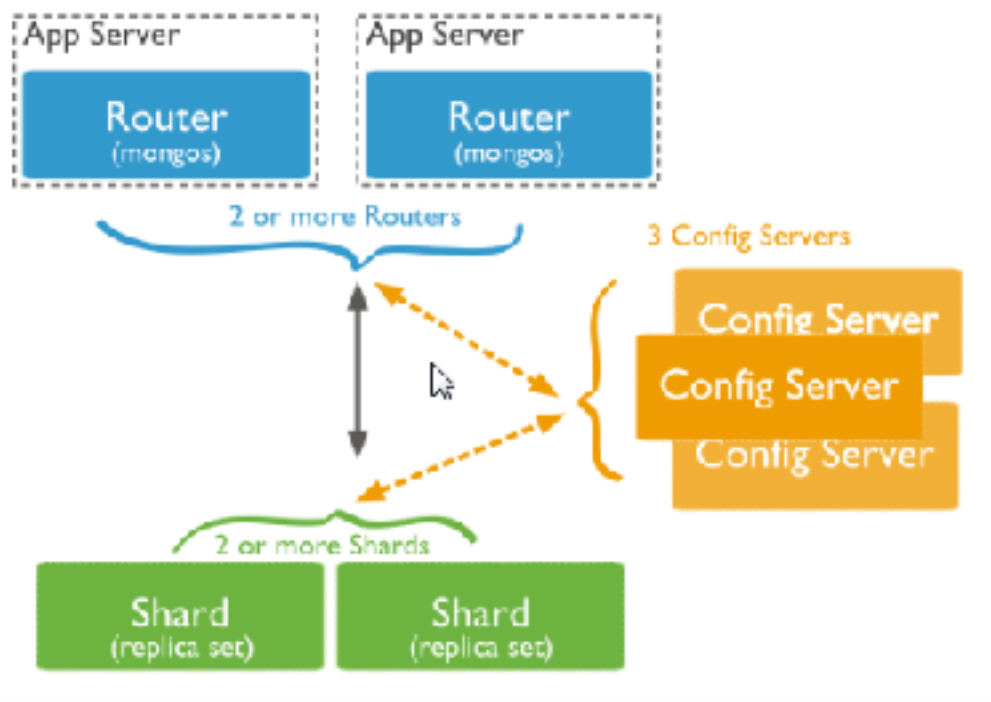
Sharding proces przechowywania danych na wielu serwerach. Wykorzystywane głównie w sytuacji jak mamy dużo danych i chcemy utrzymać wydajność. Jest to skalowanie poziome.

- Przy replikacji wszystkie zapisy trafiają do wszystkich węzłów, w shardingu zapis trafia do jednego węzła.
- Dane dla których opóźnienia zapisu są istotne w dalszym ciągu mogą trafić do node ‚master’

MongoDB – Sharding

Elementy instalacji:

- Shards – każdy shard jest oddzielnym serwerem
- Config Server – w konfigu jest przechowywana informacja gdzie sa które ane
- Query router – odpowidzialny za przekierowanie zapytań do odpowiedniego shard'a.



MapReduce: Dlaczego prędkość ma znaczenie..

1990

Średnia wielkość dysku	1370 MB
Prędkość odczytu	4.4 MB/s,
Czas odczytu całego dysku	5 min

2010

Średnia wielkość dysku	1 TB (747x większy)
Prędkość odczytu	100 MB/s (25x szybszy)
Czas odczytu całego dysku	**3 godz. (36x dłużej!!!)**

MapReduce – przykład

Znaleźć liczbę INT8 w zbiorze zawierającej 100 mld wartości w postaci cyfr INT8 w zbiorze tekstowym:

343443

454545

5776576

….

Szacowanie wielkości (najgorszy scenariusz)

Jedna liczba to 20 znaków + znak końca lini = 21

Czyli wszystkie liczby to 21 mld znaków = 2 100 000 000 000 bajtów a to 2.1TB (zakładając że kilobajt to 1000B)

Dobry dysk SATA 3 w trybie Burst read – 200MB/s czyli 0.0002 TB/s. (realia około ¼ tej prędkości)

Czyli przeczytanie całość zajmuje 2.1 / 0.002 = 10 500 s = 2.91 godziny (a bardziej realne 12 godzin) !

Nie wygląda to dobrze ☹

MapReduce – przykład

Rozwiązanie – przechowywać plik pocięty na kawałki na 1000 serwerach i analizowanie go w częściach !

Algorytm:

- Na każdej części znajdujemy liczbę
- Z wszystkich znalezionych liczb składamy końcową listę

Czas działania programu skracamy do 10 sekund..

NoSQL MapReduce porównanie do RDBMS

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear