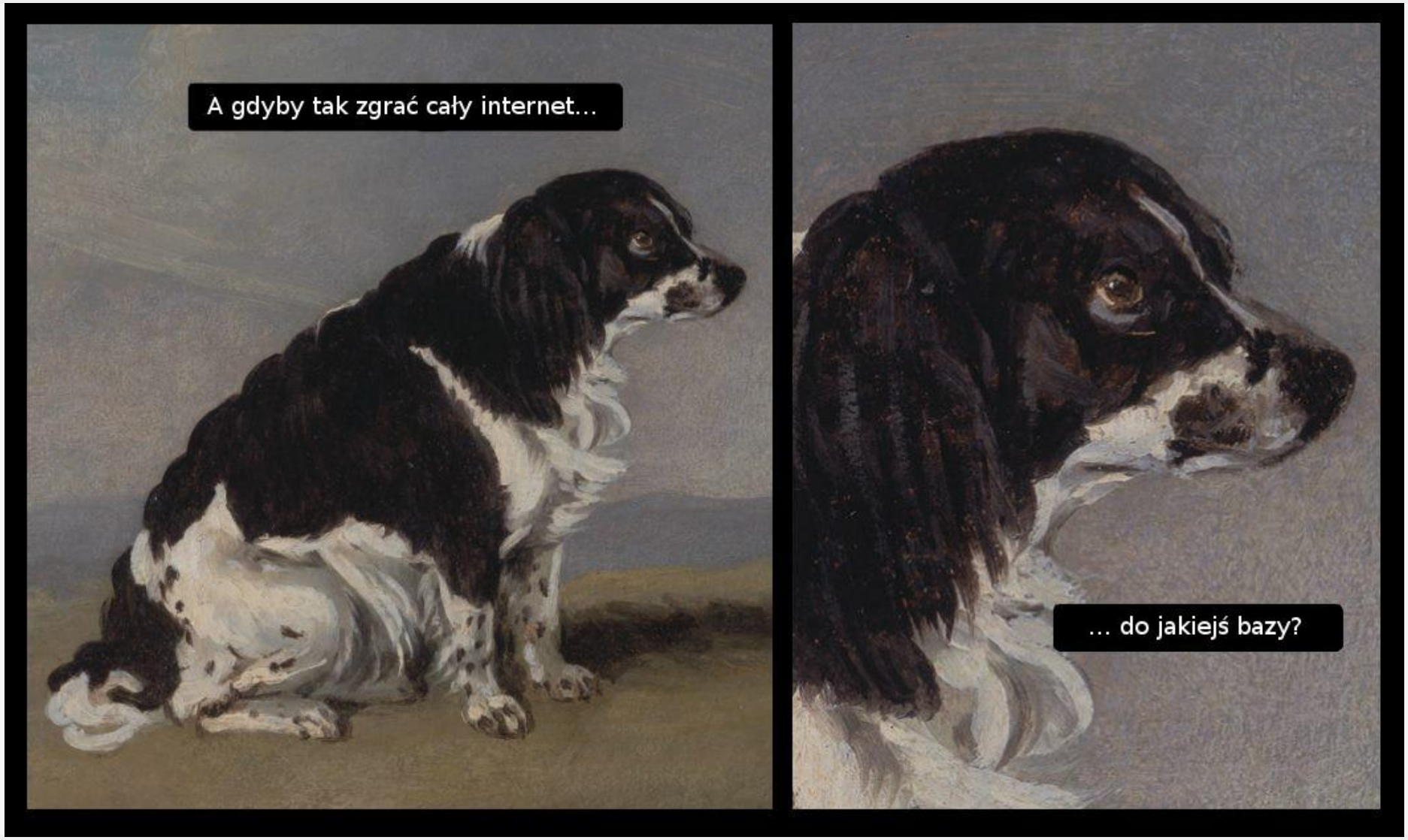
Bazy danych noSQL

Krystian Rybarczyk

Po co noSQL?



noSQL

- noSQL = not only SQL
- wyjście poza ograniczenia SQL
  - inne ograniczenia - brak ACID

Rozpraszanie danych

?

BASE zamiast ACID

BASE zamiast ACID

Ba Basic Availability — gwarancja odpowiedzi zgodnej z CAP

BASE zamiast ACID

Ba Basic Availability — gwarancja odpowiedzi zgodnej z CAP

Soft-state — zapisy nie muszą być spójne na poziomie zapisu
ani różne repliki nie muszą być ciągle spójne

BASE zamiast ACID

Ba Basic Availability	gwarancja odpowiedzi zgodnej z CAP
Soft-state	zapisy nie muszą być spójne na poziomie zapisu ani różne repliki nie muszą być ciągle spójne
Eventual consistency	zapisy zapewniają spójność w jakimś momencie w przyszłości (np. leniwie podczas odczytu)

Rozpraszanie danych

?

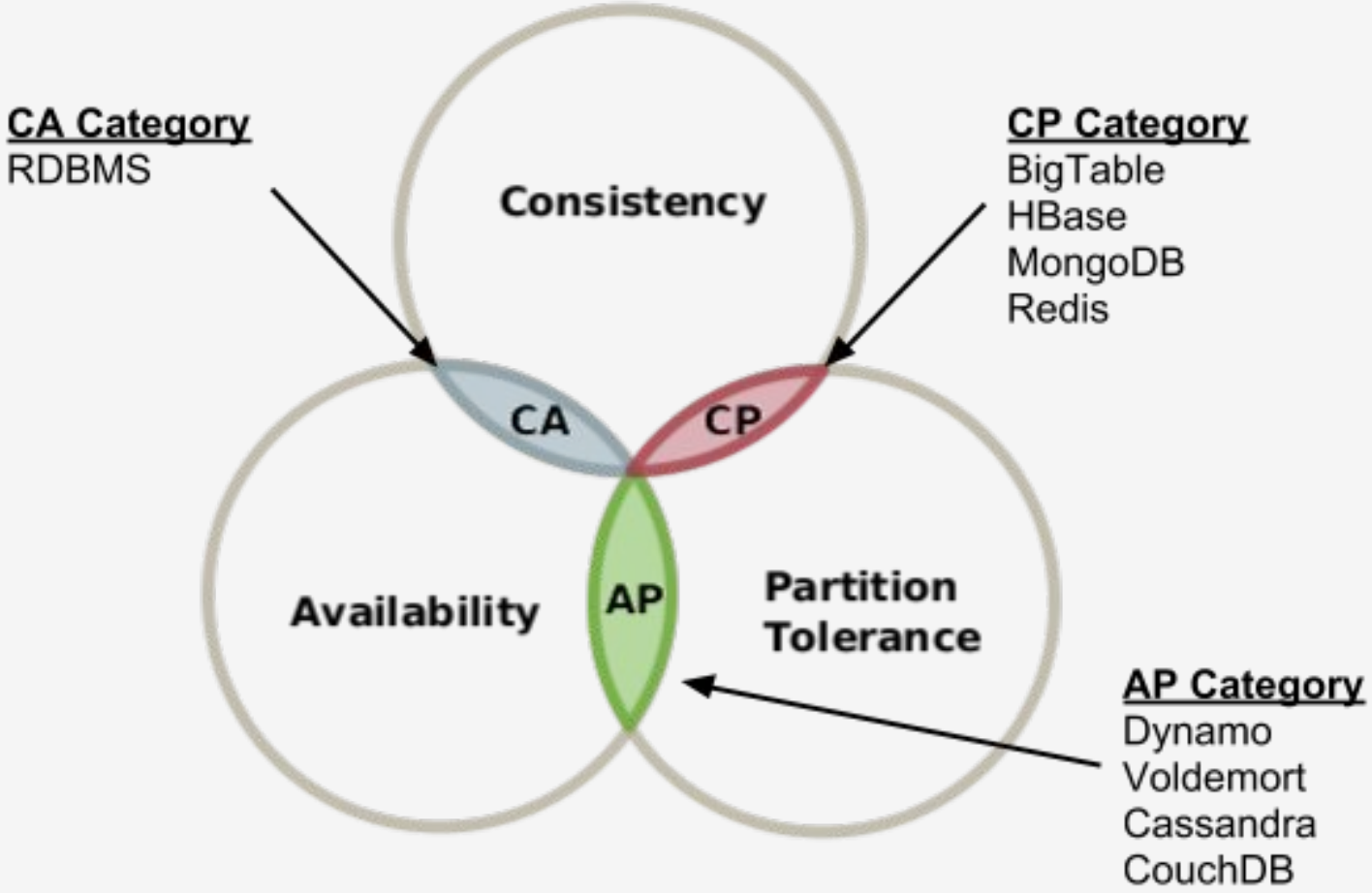
Rozpraszanie danych

Twierdzenie CAP

- Consistency
  - Każdy odczyt otrzymuje najnowszy zapis albo błąd
- Availability
  - Każde zapytanie otrzymuje (nie-błędną) odpowiedź - bez zapewnienia, że zawiera najnowszy zapis
- Partition tolerance
  - System działa pomimo dowolnej ilości zgubionych (lub opóźnionych) przez sieć pomiędzy węzłami wiadomości. Niedostępność pojedynczego węzła nie powinna powodować awarii całego systemu.

Rozpraszanie danych

Twierdzenie CAP



Rodzaje baz noSQL

- Kolumnowe (wide column): Cassandra, Accumulo
- Key-value: DynamoDB, Redis
- Dokumentowe: MongoDB, Elastic (poprzednio ElasticSearch)
- Grafowe: Neo4J, TITAN

Elastyczny model danych

Elastyczny model danych

Dokumentowa baza danych

mongoDB®

Model relacyjny

Model dokumentowy

mongoDB®

Model relacyjny

Dane przechowywane w wierszach

Model dokumentowy

Dane przechowywane w dokumentach

mongoDB®

Model relacyjny	Model dokumentowy
Dane przechowywane w wierszach	Dane przechowywane w dokumentach
Jedna encja może rozciągać się na kilka tabel	Jedna encja (dokument) to jeden rekord

mongoDB®

Model relacyjny	Model dokumentowy
Dane przechowywane w wierszach	Dane przechowywane w dokumentach
Jedna encja może rozciągać się na kilka tabel	Jedna encja (dokument) to jeden rekord
Jednorodna struktura tabeli (schemat)	Dokumenty nie muszą mieć jednorodnej struktury

mongoDB®

Model relacyjny

Model dokumentowy

mongoDB®

Model relacyjny	Model dokumentowy
Wiersz	Dokument

mongoDB®

Model relacyjny	Model dokumentowy
Wiersz	Dokument
Tabela	Kolekcja

mongoDB®

Model relacyjny	Model dokumentowy
Wiersz	Dokument
Tabela	Kolekcja
Klucz: dowolny atrybut	Klucz: pole _id

mongoDB®

Model relacyjny	Model dokumentowy
Wiersz	Dokument
Tabela	Kolekcja
Klucz: dowolny atrybut	Klucz: pole _id
Indeks	Indeks

Przechowywanie danych

JSON

- Dokumenty przechowywane w formacie JSON
- JSON czyli JavaScript Object Notation
  - Format wymiany danych
  - Czytelny dla ludzi
  - Łatwy do generowania/parsowania przez maszyny
  - Niezależny od języka

BSON

- Binarny JSON
  - dodatkowe typy danych
  - sortowane pola

JSON - dane w postaci nazwa - wartość

```
{
    "age": 31,
    "city": "New York",
    "name": "John"
}
```

Typy danych JSON

Typy danych JSON

"name":"John"	string

Typy danych JSON

"name":"John"	string
“alive”:false	boolean

Typy danych JSON

"name":"John"	string
“alive”:false	boolean
“age”:30	liczba

Typy danych JSON

"name":"John"	string
“alive”:false	boolean
“age”:30	liczba
“surname”:null	null

Typy danych JSON

"name":"John"	string
“alive”:false	boolean
“age”:30	liczba
“surname”:null	null
“names”:[“John”, “David”, “”]	lista

Typy danych JSON

"name":"John"	string
“alive”:false	boolean
“age”:30	liczba
“surname”:null	null
“names”:[“John”, “David”, “”]	lista
“person”:{“name”:”John”, “age”:30}	obiekt

Typy danych

Typy danych dostępne w MongoDB:
https://docs.mongodb.com/manual/reference/bson-types/

JSON

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

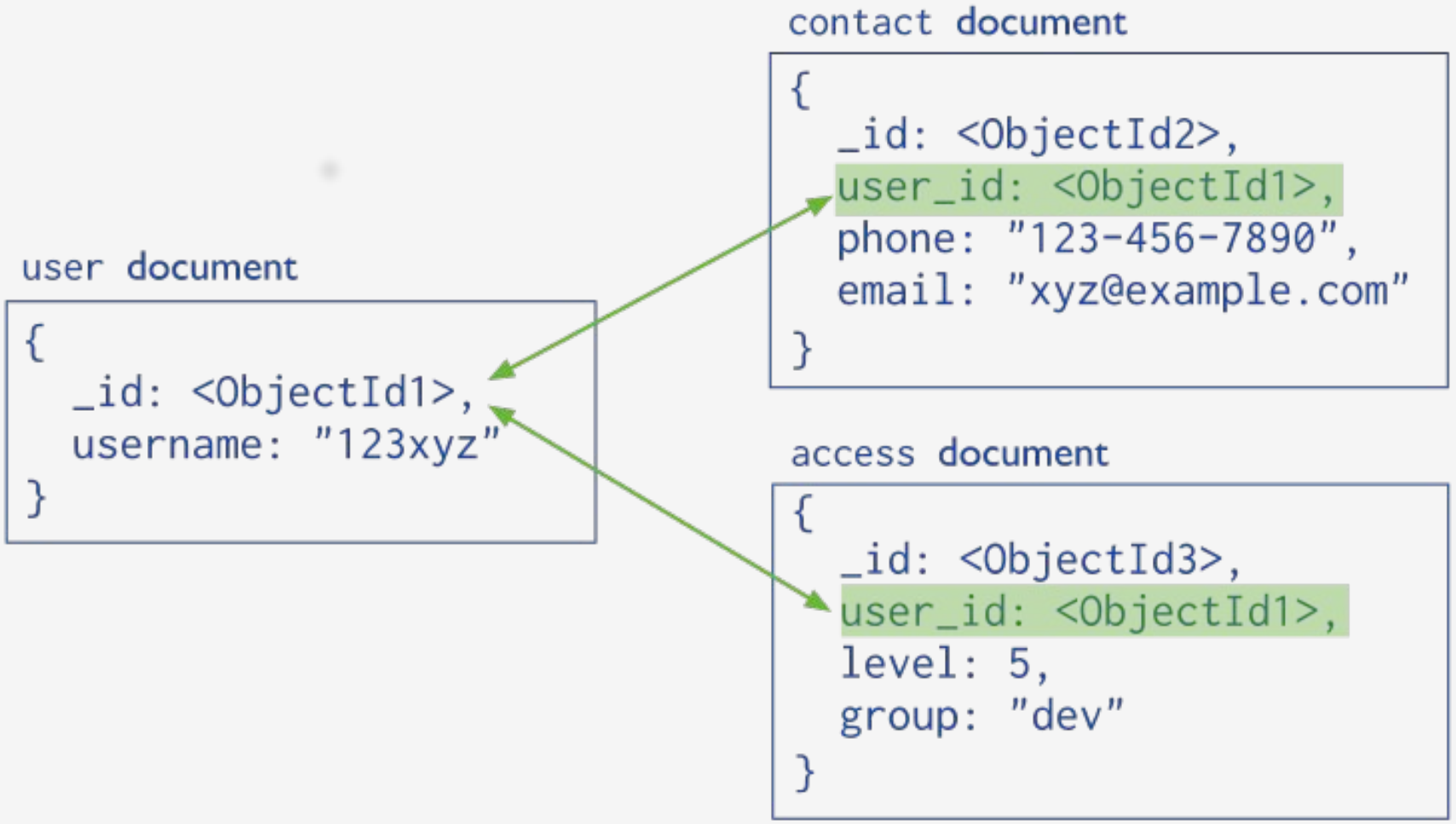
mongoDB®

Modelowanie danych

Embedding


{
_id: <ObjectId1>,
username: "123xyz",
contact: {
phone: "123-456-7890",
email: "xyz@example.com"
},
access: {
level: 5,
group: "dev"
}
}
Embedded sub-document
Embedded sub-document

Normalized


contact document
{
_id: <ObjectId2>,
user_id: <ObjectId1>,
phone: "123-456-7890",
email: "xyz@example.com"
}
user document
{
_id: <ObjectId1>,
username: "123xyz"
}
access document
{
_id: <ObjectId3>,
user_id: <ObjectId1>,
level: 5,
group: "dev"
}

Modelowanie danych

Embedded data model

- Większa wydajność
- Używane w relacjach 1-1
- Duplikacja danych
- Wymaga tylko jednego zapytania do bazy

Normalized data model

- Używaj jeśli korzyści wydajności embeddingu nie przeważają kosztu duplikacji danych
- Duże hierarchiczne zbiory danych
- Reprezentacja bardziej złożonych relacji wiele do wielu
- Wymaga kilku zapytań do bazy

CRUD

Create, Read, Update, Delete

_id

- Unikalny identyfikator, pozwala rozróżnić dokumenty
- Tworzony automatycznie jeśli nie jest podany
- Jest indeksem
- Nie można go usunąć
- Można użyć go, aby odtworzyć datę utworzenia
  ObjectId(_id).getTimestamp();

JSON

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

Create

Wstawianie danych:

```
db.animals.insert({"type" : "hen"})
```

```
INSERT INTO animals
VALUES(‘hen’);
```

Read

Znajdź wszystkie dokumenty:

db.animals.find()	SELECT * FROM animals

Read

Znajdź wszystkie dokumenty:

db.animals.find()	SELECT * FROM animals

Ogranicz liczbę znalezionych dokumentów:

db.animals.findOne() db.animals.find().limit(1)	SELECT TOP 1 * FROM animals

Read

Znajdź wszystkie dokumenty:

db.animals.find()

SELECT * FROM animals

Ogranicz liczbę znalezionych dokumentów:

db.animals.findOne()
db.animals.find().limit(1)

SELECT TOP 1 * FROM animals

Znajdź dokument z wartością:

db.animals.find({“type”:”cat”})

SELECT * FROM animals WHERE type=’cat’

Read

Znajdź dokument spełniający kilka kryteriów:

```
db.animals.find({“type”:”cat”},
{“name” : “whiskas”})
```

```
SELECT * FROM animals WHERE
type=’cat’ AND name=’whiskas’
```

Read

Znajdź dokument spełniający jedno z podanych kryteriów:

```
db.animals.find({
  $or : [ {"type" : "cat"}, {"type":"dog"} ]
})
```

```
SELECT * FROM animals WHERE
type = ‘cat’
OR type=’dog’
```

Read

Znajdź dokument spełniający jedno z podanych kryteriów:

```
db.animals.find({
  $or : [ {"type" : "cat"}, {"type":"dog"} ]
})
```

```
SELECT * FROM animals WHERE
type = ‘cat’
OR type=’dog’
```

Znajdź dokument z jedną z wartości w jednym polu

```
db.animals.find({"type" :
  { $in: ["cat", "dog"] }
})
```

```
SELECT * FROM animals WHERE
type IN (‘cat’, ‘dog’)
```

Read

Wyświetlanie/ukrywanie wybranych pól:

```
db.animals.find(
  {"type" : "monkey"} ,
  {"age" : 1}
)
```

1 - pokaż; 0 - ukryj

SELECT age FROM type = ‘monkey’

Read

Sortowanie wyników:

db.animals.find().sort({"type" : 1})
1 - rosnąco; -1 - malejąco

SELECT * FROM animals
ORDER BY type ASC

Pomijanie dokumentów:

db.animals.find().skip(1)

Liczenie dokumentów:

db.animals.find().count()

SELECT COUNT(*) FROM animals

Operatory logiczne:

$and, $or, $not, $nor

Operatory dotyczące pól:

$exists, $type

Operatory porównania:

$eq, $gt, $gte, $lt, $lte, $ne, $in, $nin

Operatory ewaluacyjne:

$mod, $regex, $text, $where

Operatory tablicowe:

$all, $elemMatch, $size

Przykłady wykorzystania wybranych operatorów

$ne - not equal:

```
db.animals.find({"type" : {$ne: "dog"}})
```

$lt - less than:

```
db.animals.find({"age" : { $lt: 20}})
```

$or:

```
db.animals.find({
  $or : [ {"type" : "cat"}, {"type":"dog"} ]
})
```

$and:

```
db.animals.find({
  $and : [ {"type" : "cat"}, {"type":"dog"} ]
})
```

Wyszukiwanie wewnątrz JSON

db.people.findOne()

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

Przykłady wykorzystania wybranych operatorów

$elemMatch - znajdź dokument, który zawiera listę z elementem, spełniającym wszystkie kryteria:

```
db.people.find({
  “awards” : {
    $elemMatch : {
      “by” : “IEEE Computer Society”,
      “year” : { $lt : 1960 }
      }
    }
  })
```

Bardzo pomocna dokumentacja dotycząca wyszukiwania
https://docs.mongodb.com/manual/reference/operator/query/

Update

Aktualizowanie:

```
db.animals.update(
  {"type":"cat"},                    //wszystkie dokumenty, gdzie type = ‘cat’
  {$set: {"name":"whiskas"}},        //ustaw ‘whiskas’ w name
  {upsert: true}                     //utwórz, jeśli nie istnieje
)

db.animals.update(
  {"type":"cat"},                    //wszystkie dokumenty, gdzie type = ‘cat’
  {$set: {"name":"meow"}},           //ustaw ‘whiskas’ w name
  {multi: true}                      //zaktualizuj wiele dokumentów
)
```

Delete

Usuń rekordy o kryteriach:

```
db.animals.remove({"type" : null})
```

```
DELETE FROM animals
WHERE type IS NULL
```

Delete

Usuń rekordy o kryteriach:

```
db.animals.remove({"type" : null})
```

```
DELETE FROM animals
WHERE type IS NULL
```

Usuń kolekcję:

```
db.animals.drop()
```

```
DROP TABLE animals
```

MongoDB może wykonywać dowolne polecenia JavaScript.

```
var cursor = db.students.find();

while ( cursor.hasNext() ) {
   printjson( cursor.next() );
}
```

Tworzenie zmiennych:
var number = 3

Korzystanie z listy:
var list = [1, 2, 3]
list[1]

result: 2

Korzystanie z obiektów JSON:
var json = {"name" : "Krystian",
"surname" : "Rybarczyk"}
json["name"]

result: Krystian

Korzystanie ze złożonych obiektów JSON:

```
var json = {
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    }

json["name"]
```

```
result:
 /* 1 */
 {
     "first" : "John",
     "last" : "Backus"
}
```

Korzystanie ze złożonych obiektów JSON:

```
var json = {
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    }

json["name"][“first”]
```

result:
John

Wyszukiwanie wewnątrz JSON

db.people.findOne()

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

Wyszukiwanie wewnątrz JSON

db.people.find({“name.first” : “John”})

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

Demo

```
var firstResult = db.animals.find()[0]                  //pobierz pierwszy element z listy wyników

print(firstResult["type"], firstResult["name"])         //wyświetl pola “type” oraz “name”
```


mongoDB®
Shell

Podstawy Mongo

- Atomowy zapis dokumentu
- Baza jest dostępna przez interfejs HTTP i shell
- Struktury są tworzone po wstawieniu danych

Podstawowe polecenia

Sprawdź jakiej bazy używasz:

db

Pokaż wszystkie bazy

show dbs

Przełącz do innej bazy/stwórz nową

use <nazwa>

Pokaż istniejące kolekcje

show collections

Pokaż pomoc

help

OLAP

Czyli agregacje

db.database.aggregate[]

- Umożliwia przetwarzanie danych
- Wykorzystywane do tworzenia statystyki na podstawie danych

Liczenie grup:

```
db.getCollection('ocean_data').aggregate([{
$group: {
  _id : "$name",
  products: { $sum : 1 }
  }
}])
```

Warunkowe przetwarzanie danych - match

```
db.getCollection('ocean_data').aggregate([
{ $match: { "station_id" : 8461490 } },
{ $group: {
      _id : "$name",
      products: { $sum : 1 }
      }
}
])
```

Przetwarzanie elementów listy

Dekonstruuje listę i dla każdego z elementów tworzy nowy dokument.

Przetwarzanie elementów listy

db.people.aggregate([{ $unwind : "$awards"}])

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }
}
```

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
}
```

Przetwarzanie elementów listy

Dekonstruuje listę i dla każdego z elementów tworzy nowy dokument.

```
db.ocean_data.aggregate([
{ $unwind : "$products"},
{ $match: { "products.name" : "water_temperature"}},
{ $group: {_id: "$products.name", count: { $sum: 1}} }
])
```

Szukanie geograficzne

Dokumentacja dotycząca wyszukiwania geograficznego
https://docs.mongodb.com/manual/reference/operator/aggregation/geoNear/