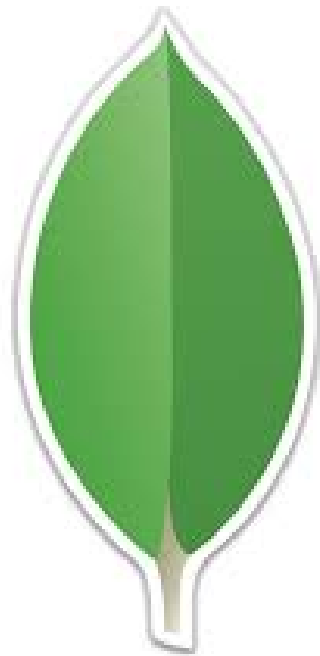


```
{  
  "database": "MongoDB",  
  "type": ["NoSQL", "schema-free",  
           "document-oriented",  
           "high-performance"]  
}
```

Przemysław Grzesiowski
2 grudnia 2018

Najważniejsze to mieć plan !

1. dane liczbowe, badania rynku
2. moja pierwsza baza, mongo shell
3. mongo vs baza relacyjna – pojęcia
4. json – typy danych
5. kolekcje w mongo
6. zapytania z poziomu mongo shell – ćwiczenia
7. mongo podstawy - podsumowanie
8. model danych, podstawy projektowania bazy
9. plugin dla IJ
10. sharding, replication
11. wady mongo
12. linki, poczytaj mi mamo



Let's start mongo server



```
cd ~
```

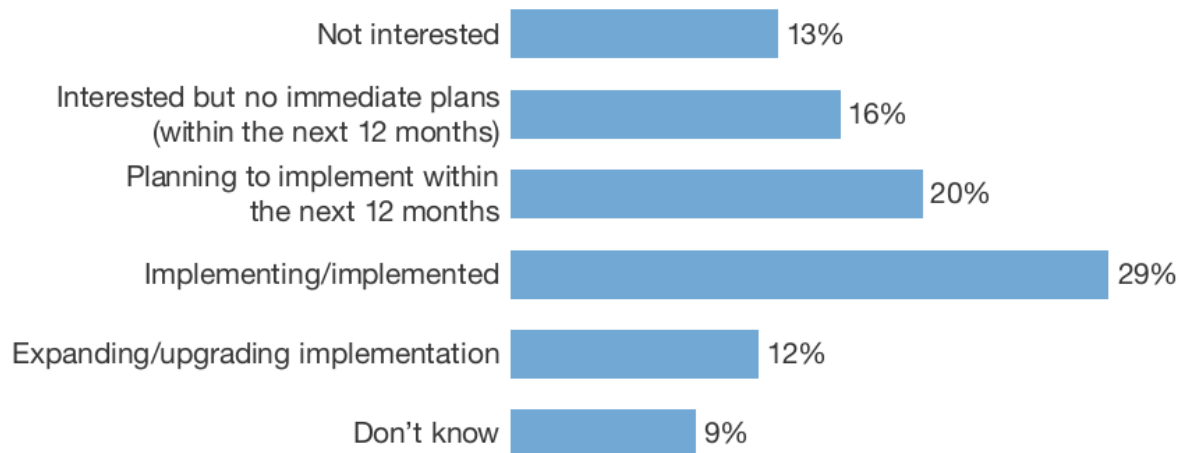
```
mkdir mongo
```

```
docker run -p 27017:27017 --name my_mongo -v ~/mongo:/data/db mongo
```

NoSql? - czemu nie ...

FIGURE 1 Enterprise Architects Are Seeing The Opportunity In NoSQL

“What are your firm’s plans to use distributed NoSQL databases?”

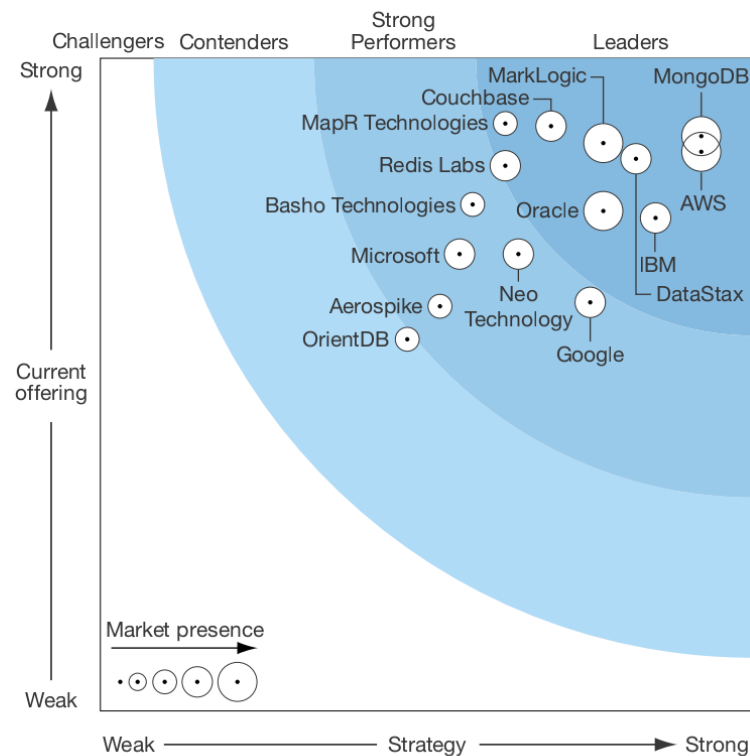


Base: 2,094 data and analytics technology decision-makers
(percentages do not total 100 due to rounding)

Source: Forrester's Global Business Technographics® Data And Analytics Survey, 2016

Mongo liderem baz noSql

FIGURE 3 Forrester Wave™: Big Data NoSQL, Q3 2016



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

DB-Engines Ranking

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

327 systems in ranking, May 2017

| Rank | | | DBMS | Database Model | Score | | |
|----------|----------|----------|------------------------|-------------------|----------|----------|----------|
| May 2017 | Apr 2017 | May 2016 | | | May 2017 | Apr 2017 | May 2016 |
| 1. | 1. | 1. | Oracle + | Relational DBMS | 1354.31 | -47.68 | -107.71 |
| 2. | 2. | 2. | MySQL + | Relational DBMS | 1340.03 | -24.59 | -31.80 |
| 3. | 3. | 3. | Microsoft SQL Server + | Relational DBMS | 1213.80 | +9.03 | +70.98 |
| 4. | 4. | ↑ 5. | PostgreSQL + | Relational DBMS | 365.91 | +4.14 | +58.30 |
| 5. | 5. | ↓ 4. | MongoDB + | Document store | 331.58 | +6.16 | +11.36 |
| 6. | 6. | 6. | DB2 + | Relational DBMS | 188.84 | +2.18 | +2.88 |
| 7. | 7. | ↑ 8. | Microsoft Access | Relational DBMS | 129.87 | +1.69 | -1.70 |
| 8. | 8. | ↓ 7. | Cassandra + | Wide column store | 123.11 | -3.07 | -11.39 |
| 9. | 9. | 9. | Redis + | Key-value store | 117.45 | +3.09 | +9.21 |
| 10. | 10. | 10. | SQLite | Relational DBMS | 116.07 | +2.27 | +8.81 |

Migracje ze świata sql to nosql mongo

| Organization | Migrated From | Application |
|---------------------|----------------------|--------------------------------------|
| eHarmony | Oracle & Postgres | Customer Data Management & Analytics |
| Shutterfly | Oracle | Web and Mobile Services |
| Cisco | Multiple RDBMS | Analytics, Social Networking |
| Craigslist | MySQL | Archive |
| Under Armour | Microsoft SQL Server | eCommerce |
| Foursquare | PostgreSQL | Social, Mobile Networking Platforms |
| MTV Networks | Multiple RDBMS | Centralized Content Management |
| Buzzfeed | MySQL | Real-Time Analytics |
| Verizon | Oracle | Single View, Employee Systems |
| The Weather Channel | Oracle & MySQL | Mobile Networking Platforms |

Źródło: "RDBMS to MongoDB Migration Guide"

Badania rynku

- niezależne badanie: MongoDB provides greater performance than Couchbase or Cassandra in all the tests, in some cases by as much as 25x.

(<https://www.mongodb.com/collateral/comparative-benchmarks-mongodb-vs-couchbase-vs-cassandra>)

- 70% oszczędności w stos. do baz relacyjnych (Oracle):
(<https://www.mongodb.com/collateral/total-cost-ownership-comparison-mongodb-oracle>)

Mongo shell

1. Podpinamy się pod terminal i startujemy mongoDB shell:

```
docker exec -it my_mongo bash  
mongo
```

2. Poruszanie się po bazach

- pokaż bazy danych

```
show dbs
```

- pokaż aktualną bazę danych

```
db
```

- przełącz do innej bazy

```
use <db_name>
```

**{ “ćwiczenie”: “Moja
pierwsza baza mongo” }**



{ name: mongo, type: DB }



Moja pierwsza kolekcja w mongoDB

```
>use pierwsza_baza_mongo  
>db.piosenki.insert({"tytuł":"Autobiografia",  
                    "zespół":"Perfect"})
```

*alias do
obecnej
bazy*

*nazwa
kolekcji*

dokument json



Moja pierwsza kolekcja w mongoDB

```
>use pierwsza_baza_mongo  
>db.piosenki.insert({"tytuł":"Autobiografia",  
                    "zespół":"Perfect"})
```

*alias do
obecnej
bazy*

*nazwa
kolekcji*

dokument json

```
> show dbs  
> show collections
```

{“polecenie”: ”Zamknij mongo shell (exit) i wróć do swojej nowo stworzonej bazy”}

Moja pierwsza baza - find

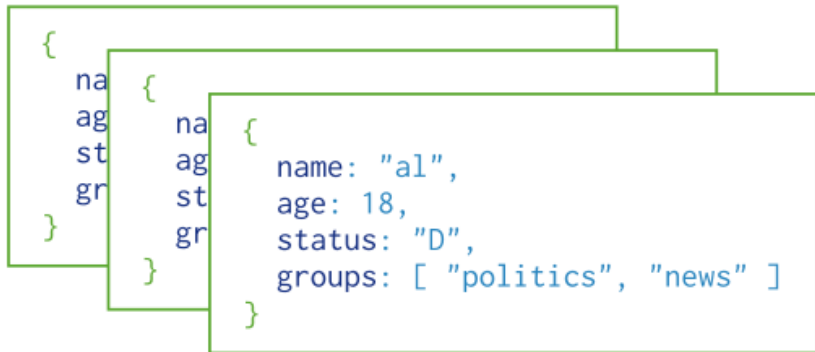


{“polecenie1”: ”wstaw kolejne 3 dokumenty do kolekcji piosenki, w tym jedną z nich zespołu Perfect”}

{“polecenie2”: ”wykonaj następujące polecenia:”}

```
>use pierwsza_baza_mongo
>db.piosenki.find()
>db.piosenki.find().pretty()
>db.piosenki.findOne()
>db.piosenki.find().limit(1)
>db.piosenki.find().limit(2)
>db.piosenki.find({"zespół":"Perfect"})
>db.piosenki.find().sort({"tytuł":-1})
>db.piosenki.find().sort({"tytuł":1})
>db.piosenki.find().sort({"zespół":1}).limit(1)
```

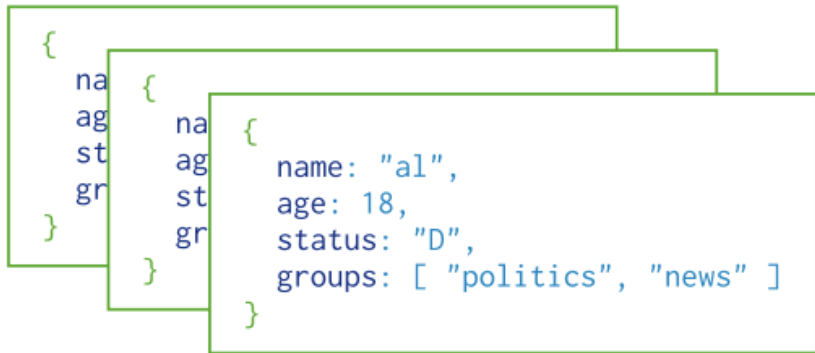
{“zapamiętaj”：“Kolekcja w NoSQL jest podobna do tabeli w bazie relacyjnej”}



Collection

- kolekcja to zbiór podobnych dokumentów json

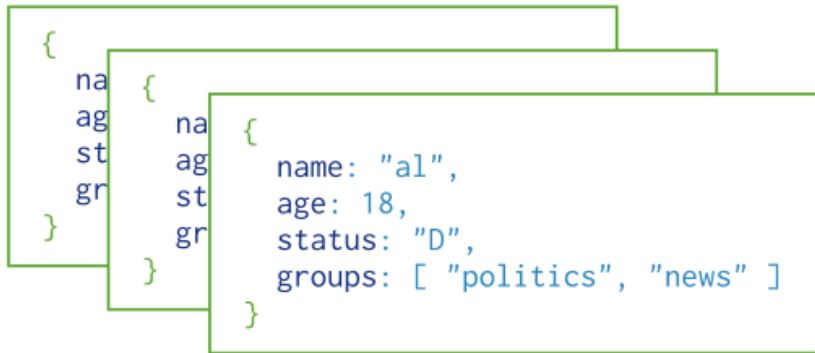
{“zapamiętaj”：“Kolekcja w NoSQL jest podobna do tabeli w bazie relacyjnej”}



Collection

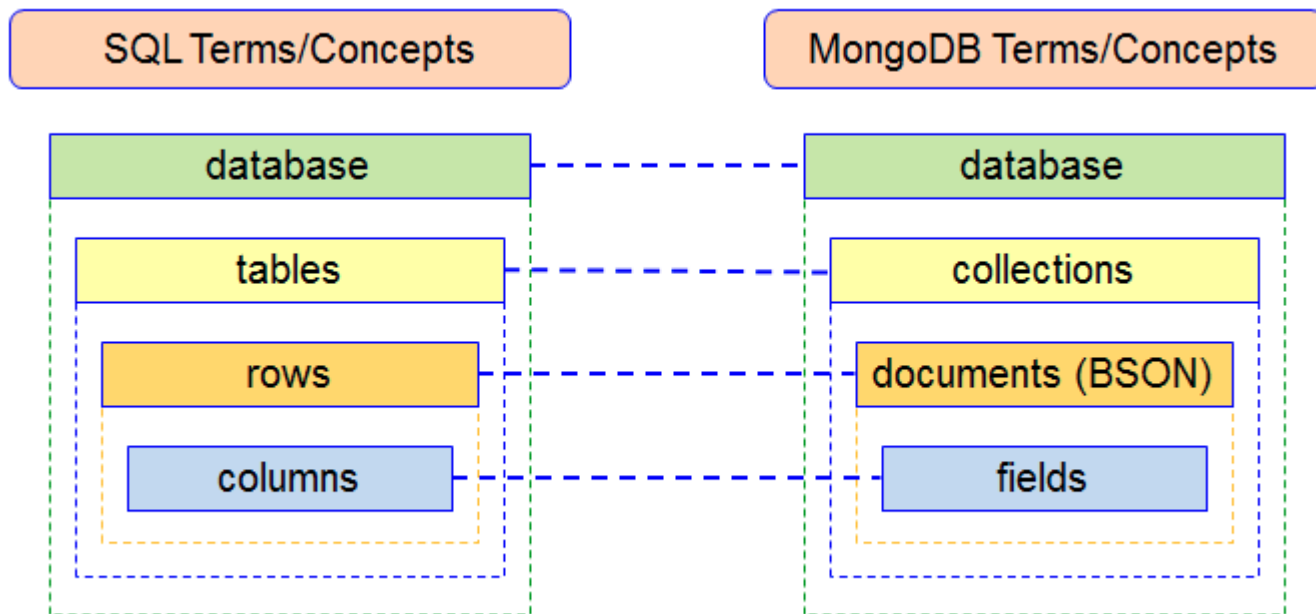
- kolekcja to zbiór podobnych dokumentów json
- dokument reprezentuje wiersz, a pola to kolumny

{“zapamiętaj”:”Kolekcja w NoSQL jest podobna do tabeli w bazie relacyjnej”}



Collection

- kolekcja to zbiór podobnych dokumentów json
- dokument reprezentuje wiersz, a pola to kolumny
- jedna baza danych zawiera 1 lub więcej kolekcji



| id | user_name | email | age | city |
|----|---------------|-----------------|-----|-------------|
| 1 | Mark Hanks | mark@abc.com | 25 | Los Angeles |
| 2 | Richard Peter | richard@abc.com | 31 | Dallas |



```
{
  "_id": ObjectId("5146bb52d8524270060001f3"),
  "age": 25,
  "city": "Los Angeles",
  "email": "mark@abc.com",
  "user_name": "Mark Hanks"
}
{
  "_id": ObjectId("5146bb52d8524270060001f2"),
  "age": 31,
  "city": "Dallas",
  "email": "richard@abc.com",
  "user_name": "Richard Peter"
}
```

JSON



{JSON} ? a co to takiego ... ?

JSON



JavaScript Object Notation.

```
{  
  "property1": "value1",  
  ...  
  "propertyN": "valueN",  
}
```

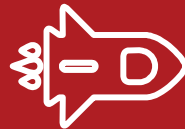
JSON



In JSON, values must be one of the following data types:

- ✓ a string
- ✓ a number
- ✓ an array
- ✓ a boolean
- ✓ null
- ✓ an object (JSON object)

String/number/array/ boolean/null



```
{ "name":"John" }
```

```
{ "age":30 }
```

```
{  
  "employees":[ "John", "Anna", "Peter" ]  
}
```

```
{ "sale":true }
```

```
{ "middlename":null }
```

Json object



```
{  
  "whiskey": { "name":"Bushmills", "age":30,"country":"Northern Ireland" }  
}
```

↑
field
(property)
(key)

Value (embedded json)

Moja pierwsza baza - dodajemy drugą kolekcję



```
{  
  "polecenie": "utwórz drugą kolekcję w Twojej bazie – o nazwie: aktorzy",  
  "szczegóły": {  
    "dodaj pola": ["imię", "nazwisko", "płeć(K/M)", "rok urodzenia", "filmy"],  
    "min ilość dokumentów w kolekcji aktorzy" : 3,  
    "min ilość filmów dla każdego aktora": 2,  
    "uwagi": ["wstawiamy tylko atrakcyjne aktorki/aktorów",  
              "filmy xxx są dozwolone"]  
  }  
}
```

```
> show collections
```


Moja pierwsza baza - zapytania



```
{  
  "polecenie": "używając komendy find przeszukaj kolekcję aktorów",  
  "szczegóły": "użyj dokumentacji mongo: db.collection.find() oraz tzw. Query  
Selectors (https://docs.mongodb.com/manual/reference/operator/query/ )",  
  "przykłady przeszukiwań": ["wszyscy aktorzy (mężczyźni)", "wszyscy aktorzy z  
    imieniem xxx", "wszyscy aktorzy urodzeni po 1950", "wszyscy aktorzy z  
    filmu f"],  
  "inne": "w razie potrzeby wstaw więcej dokumentów do kolekcji aktorzy, żeby  
    wyszukiwanie miało sens"  
} > db.aktorzy.find(...)
```

Moja pierwsza baza - zapytania



```
db.aktorzy.find({queryjson})
```

```
> db.aktorzy.find({"imię":"Cezary"})
```

```
> db.aktorzy.find({"rok_urodzenia":{"$gt":1980}})
```

```
> db.aktorzy.find({"płeć":"K", "data_ur":{"$gt":1985,  
$lt:1999}})
```

```
> db.aktorzy.find({"filmy":{"$in":["film2","film5"]}})
```

```
db.aktorzy.find({queryjson}, {projection})
```

```
> db.aktorzy.find({"płeć":"K", "data_ur":{"$gt":1985}},  
{"nazwisko":1})
```

{“cechy”: [“flexible data model”,
“schema-less”]}

```
{“polecenia”: [  
    “wstaw piosenkę do kolekcji aktorów”,  
    “wstaw aktorów z zupełnie inną strukturą dokumentu (np. nazwy pól po  
        angielsku) do kolekcji aktorzy”  
    “a teraz spróbuj to zrobić w bazie mySQL :-)”  
}]
```

{“cechy”: [“flexible data model”,
“schema-less”]}

- istnieje możliwość walidacji schematu (→ “Document Validation” mongo 3.2)
- czy frywolność zawsze płaci?

Łatwość zmian

```
{  
  "title": "Przewodnik po mongo DB",  
  "year": 2010,  
  "ISBN": 1234,  
  "author": "Jan Kowalski"  
}
```

Łatwość zmian

```
{  
  "title": "Przewodnik po mongo DB",  
  "year": 2010,  
  "ISBN": 1234,  
  "author": "Jan Kowalski"  
}  
  
{  
  "title_en": "mongo DB guide",  
  "title": "Przewodnik po mongo DB",  
  "year": 2010,  
  "ISBN": 1234,  
  "author": "Jan Kowalski"  
}
```

Łatwość zmian

```
{
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}

{
  "title_en": "mongo DB guide",
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}
// "title" as embedded object example:
{
  "title": [{
    "en": "mongo DB guide"
  }, {
    "pl": "Przewodnik po mongo DB"
  }],
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}
```

Łatwość zmian

```
{
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}

{
  "title_en": "mongo DB guide",
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}

// "title" as embedded object example:
{
  "title": [{
    "en": "mongo DB guide"
  }, {
    "pl": "Przewodnik po mongo DB"
  }],
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}
```

| title | year | ISBN | author |
|----------------------------------|------|------|--------------|
| Przewodnik po <u>mongo DB</u> | 2010 | 1234 | Jan Kowalski |

| title | year | ISBN | author |
|---|------|------|--------------|
| en= <u>Mongo DB</u> Guide, pl= Przewodnik po <u>mongo DB</u> | 2010 | 1234 | Jan Kowalski |

Łatwość zmian

```
{
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}

{
  "title_en": "mongo DB guide",
  "title": "Przewodnik po mongo DB",
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}

// "title" as embedded object example:
{
  "title": [{
    "en": "mongo DB guide"
  }, {
    "pl": "Przewodnik po mongo DB"
  }],
  "year": 2010,
  "ISBN": 1234,
  "author": "Jan Kowalski"
}
```

| title | year | ISBN | author |
|-------------------------------|------|------|--------------|
| Przewodnik po <u>mongo DB</u> | 2010 | 1234 | Jan Kowalski |

| title | year | ISBN | author |
|--|------|------|--------------|
| en= <u>Mongo DB</u> Guide, pl= <u>Przewodnik po mongo DB</u> | 2010 | 1234 | Jan Kowalski |

books:

| pk | year | ISBN | author |
|----|------|------|--------------|
| 1 | 2010 | 1234 | Jan Kowalski |



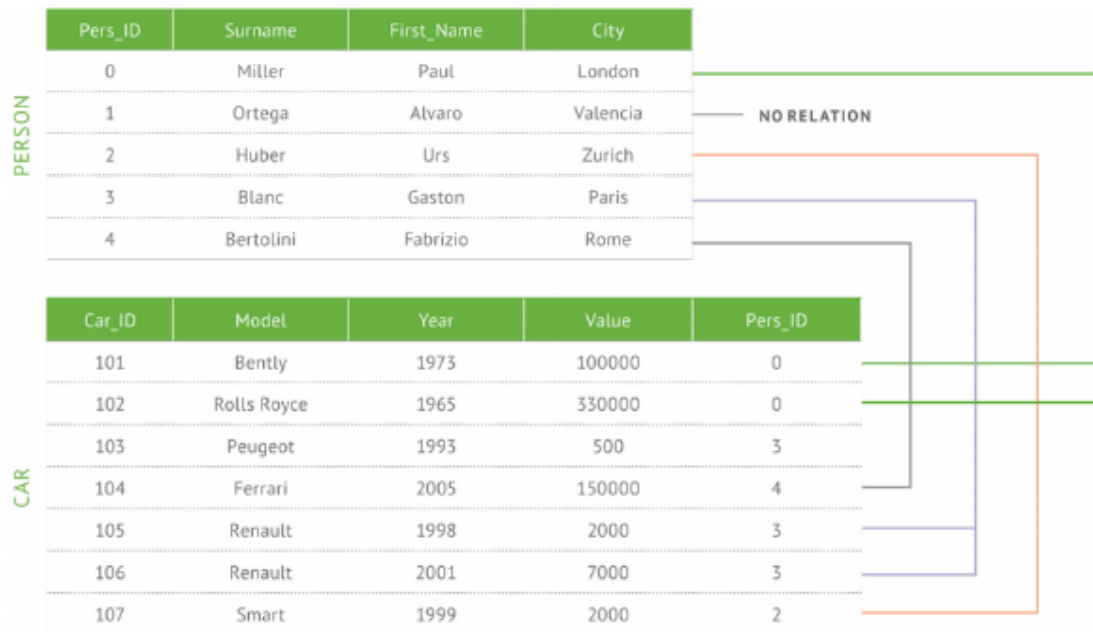
titles:

| book_fk | lang | title |
|---------|------|-------------------------------|
| 1 | en | <u>Mongo DB</u> Guide |
| 1 | pl | Przewodnik po <u>Mongo DB</u> |

Data model in

- MongoDB: all data for a given record in a single document
- RDB: information for a given record is usually spread across many tables

```
{
  first_name: "Paul",
  surname: "Miller",
  city: "London",
  location: [45.123, 47.232],
  cars: [
    {
      model: "Bentley",
      year: 1973,
      value: 100000, ...},
    {
      model: "Rolls Royce",
      year: 1965,
      value: 330000, ... }
  ]
}
```





```
{  customer_id : 1,
    first_name : "Mark",
    last_name  : "Smith",
    city       : "San Francisco",
    phones: [ {
                type : "work",
                number: "1-800-555-1212"
            },
            {
                type : "home",
                number: "1-800-555-1313",
                DNC: true
            },
            {
                type : "home",
                number: "1-800-555-1414",
                DNC: true
            }
        ]
    }
```



Mongo podstawy - podsumowanie

- kolekcja jest zbiorem podobnych dokumentów (obiektów JSON)



Mongo podstawy - podsumowanie

- kolekcja jest zbiorem podobnych dokumentów (obiektów JSON)
- każdy dokument json składa się z 1+ par: <key>:<value>



Mongo podstawy - podsumowanie

- kolekcja jest zbiorem podobnych dokumentów (obiektów JSON)
- każdy dokument json składa się z 1+ par: <key>:<value>
- value to najczęściej
 - ObjectId
 - tekst (string),
 - liczba,
 - null,
 - boolean (true/false),
 - tablica,
 - inny zagnieżdżony obiekt json,
 - referencja do obiektu z innej kolekcji (ObjectId lub DBRef),
 - zmienna typu geolocation,
 - timestamp lub date;

Data model in mongo



- przyjęty model danych warunkuje w większości przypadków znacznie szybsze odczyty (mniej operacji łączenia (JOIN) informacji z wielu tabel)
- *The MongoDB document is physically stored as a single object, requiring only a single read from memory/disk. On the other hand, RDBMS JOINS require multiple reads from multiple physical locations.*



Data model in mongo

- przyjęty model danych warunkuje w większości przypadków znacznie szybsze odczyty (mniej operacji łączenia (JOIN) informacji z wielu tabel)
 - *The MongoDB document is physically stored as a single object, requiring only a single read from memory/disk. On the other hand, RDBMS JOINS require multiple reads from multiple physical locations.*
- bazę należy projektować pod kątem jakie zapytania będą wykonywane najczęściej (tabele, które są zazwyczaj używane wspólnie, powinny wylądować w jednej kolekcji)

PRICE OF 1GB STORAGE

| | | |
|------|---|-----------|
| 1981 | → | \$300,000 |
| 1990 | → | \$10,000 |
| 1994 | → | \$1,000 |
| 2000 | → | \$10 |
| 2012 | → | \$0.10 |



Data model in mongo



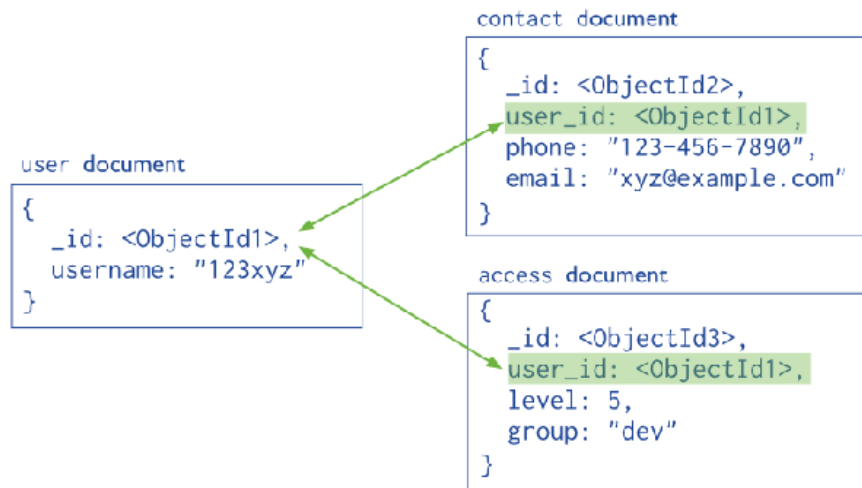
- rozmiar danych na dysku nie jest już tak istotny jak w latach 70-tych
- czas pracy i “time to market” jest dziś dużo droższy/ważniejszy niż przestrzeń dyskowa

Data model in mongo



- normalizacja vs duplikacja danych
- nie jesteśmy ograniczeni modelem danych:
 - relacje 1:n można zapisać w tej samej kolekcji (np. w formie tablicy),
 - zagnieżdżone dokumenty dają szerokie możliwości
 - możemy wciąż korzystać z indeksów

{ "jak_projektować": "Zagnieżdzać czy nie?" }



{“jak_projektować”:”Zagnieżdżać czy nie?”}

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach

NIE zagnieżdżaj gdy:

{ “jak_projektować”: ”Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)

NIE zagnieżdżaj gdy:

{ “jak_projektować”: ”Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)
- ✓ gdy “dzieci” są modyfikowane razem z rodzicem (tranzakcyjność)

NIE zagnieżdżaj gdy:

{ “jak_projektować”: ”Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)
- ✓ gdy “dzieci” są modyfikowane razem z rodzicem (tranzakcyjność)

NIE zagnieżdżaj gdy:

- x uniknięcie duplikacji danych jest ważne, a zwiększony koszt(czas) odczytu nie ma znaczenia

{ “jak_projektować”: “Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)
- ✓ gdy “dzieci” są modyfikowane razem z rodzicem (tranzakcyjność)

NIE zagnieżdżaj gdy:

- ✗ uniknięcie duplikacji danych jest ważne, a zwiększony koszt(czas) odczytu nie ma znaczenia
- ✗ skomplikowane relacje “many to many”

{ “jak_projektować”: “Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)
- ✓ gdy “dzieci” są modyfikowane razem z rodzicem (tranzakcyjność)

NIE zagnieżdżaj gdy:

- x uniknięcie duplikacji danych jest ważne, a zwiększony koszt(czas) odczytu nie ma znaczenia
- x skomplikowane relacje “many to many”
- x pola “dzieci” są rzadko używane i/lub są duże

{ “jak_projektować”: ”Zagnieżdżać czy nie?” }

Zagnieżdżaj:

- ✓ gdy możemy zyskać na odczytach
- ✓ gdy pola - “dzieci” mają sens tylko w kontekście rodzica (“one to one”, “one to many”)
- ✓ gdy “dzieci” są modyfikowane razem z rodzicem (tranzakcyjność)

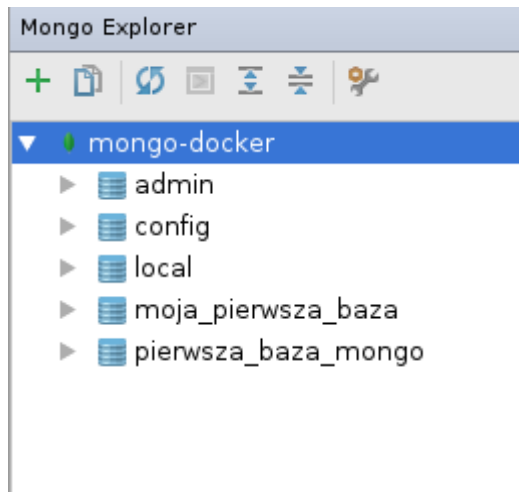
NIE zagnieżdżaj gdy:

- x uniknięcie duplikacji danych jest ważne, a zwiększony koszt(czas) odczytu nie ma znaczenia
- x skomplikowane relacje “many to many”
- x pola “dzieci” są rzadko używane i/lub są duże
- x istnieje ryzyko przekroczenia limitu dokumentu (16MB)

{“next”:“IntelliJ support”}



```
{  
  "ŻeCoMamZrobić?": "zainstaluj plugin",  
  "ŻeNibyJak?": "(File -> Settings -> Plugins)",  
  "jakiPlugin?": "Mongo Plugin",  
  "iCoDalej?": ["uruchom Mongo Explorer",  
                 "dodaj nowego aktora z poziomu IJ"]  
}
```



Mongo Explorer

+ [Icons]


- ▼ mongo-docker
 - ▶ admin
 - ▶ config
 - ▶ local
 - ▼ moja_pierwsza_baza
 - aktorzy
 - piosenki**

| Property | Value |
|----------------|-----------|
| size | 208 bytes |
| count | 3 |
| avgObjSize | 69 bytes |
| storageSize | 36 KB |
| capped | false |
| nindexes | 1 |
| totalIndexSize | 36 KB |
| indexSizes | |
| _id_ | 36 KB |

Live example - kolekcje “people” i “companies”

```
{
  "_id" : ObjectId("57d7a121fa937f710a7d486e"),
  "first_name" : "Yvonne",
  "last_name" : "Pham",
  "quote" : "Aliquam est reiciendis alias neque ad.",
  "job" : "Counselling psychologist",
  "ssn" : "401-31-6615",
  "address" : {
    "city" : "Burgessborough",
    "street" : "83248 Woods Extension",
    "zip" : "47201"
  },
  "company_id" : ObjectId("57d7a121fa937f710a7d486d"),
  "employer" : "Terry and Sons",
  "birthday" : ISODate("2011-03-17T11:21:36Z"),
  "email" : "murillobrian@cox.net"
}
```

```
{
  "_id" :
  ObjectId("57d7a121fa937f710a7d486d"),
  "sector" : "Wholesale",
  "name" : "Terry and Sons",
  "mission" : "implement frictionless
systems",
  "address" : {
    "city" : "Lake Meaganton",
    "state" : "Idaho",
    "street" : "211 Diane Shoals",
    "zip" : "10914-3394"
  },
  "logo" :
  "http://dummyimage.com/687x376"
}
```



Restore/Dump

```
mongodump -- help  
mongorestore --help
```

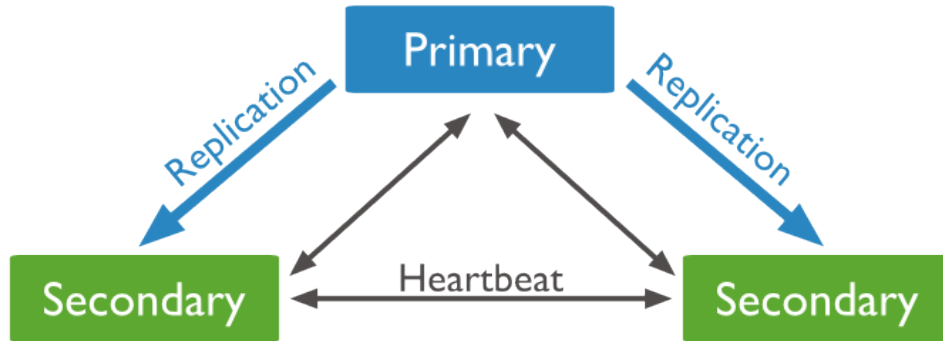
Tworzymy kopię naszej bazy



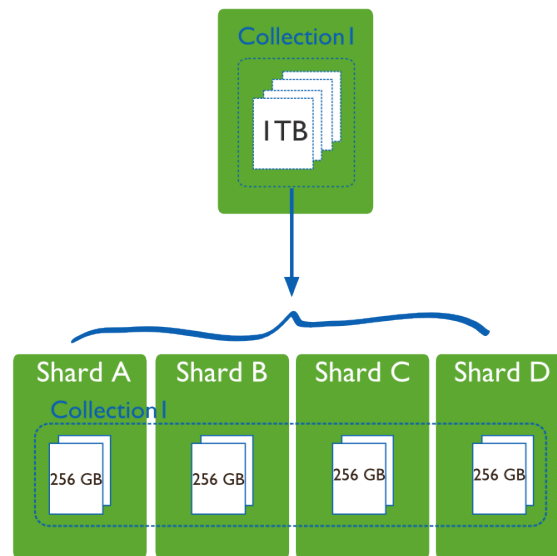
```
{  
  "polecenie1": "używając komendy mongodump dokonaj wykonania zrzutu  
kolekcji aktorzy",  
  "szczegóły": "użyj dokumentacji mongo: mongodump -help",  
  "polecenie2": "prześlij plik zrzutu do kolegi (wykorzystaj docker cp a potem  
slacka)"  
  "polecenie3": "używając mongorestore przywróć bazę danych kolegi na swojej  
maszynie, przejrzyj jego ulubionych aktorów"  
}
```


Mongo na produkcji

replication



sharding



WADY MONGO

- ✓ brak transakcji
- ✓ większy rozmiar bazy danych (każde pole jest przechowywane)
- ✓ ciągle nowe porównując do baz relacyjnych

“

Much of the data we use today has complex structures that can be modeled and represented more efficiently using JSON (JavaScript Object Notation) documents, rather than tables. With sub-documents and arrays, JSON documents also align with the structure of objects at the application level. This makes it easy for developers to map the data used in the application to its associated document in the database. By contrast, trying to map the object representation of the data to the tabular representation of an RDBMS slows down development.

LINKI

- <https://www.mongodb.com/mongodb-architecture>
- <https://docs.mongodb.com/manual/reference/sql-comparison/>
- <https://www.mongodb.com/nosql-explained>

CHCESZ WIEDZIEĆ WIĘCEJ?

- skalowanie
- rodzaje baz noSql
- Indeksy
- BSON
- SQL query
- aggregation pipeline
- widoki (views)
- Geospatial Indexes and Queries



Wnioski, skargi, reklamacje, opinie

- co zmieniłbyś w dzisiejszych zajęciach?
- powiedz teraz albo zachowaj milczenie



Wnioski, skargi, reklamacje, opinie

- co zmieniłbyś w dzisiejszych zajęciach?
- powiedz teraz albo zachowaj milczenie do czasu ankiety





Koniec