Bazy Danych NoSQL

na podstawie MongoDB

Autor: Piotr Chudzik

Agenda

- Czym są bazy NoSQL?
- Do czego wykorzystujemy NoSQL?
- System rozproszony / Rozproszona baza danych
- Teorie CAP oraz BaSE
- Rodzaje baz NoSQL
- Format plików JSON i jego rodzaje
- Baza danych MongoDB
  - Typy danych
  - Sposoby przechowywania danych (kolekcje, bazy danych, dokumenty)
  - Przetwarzanie danych (modyfikacja danych)

Autor: Piotr Chudzik

Czym są bazy danych NoSQL?

Nazwa **NoSQL** nie oznacza braku języka SQL. Jego pełne rozwinięcie to **N**ot **O**nly **SQL** (potocznie mówiąc: baza nierelacyjna).

Termin **NoSQL** został po raz pierwszy został użyty przez Carlo Strozziego w 1998 roku jako nazwa dla lekkiej relacyjnej bazy Strozzi NoSQL.

Można spotkać się również z określeniem **NoREL** (brak relacji).

Czym są bazy danych NoSQL?

Bazy Danych typu NoSQL charakteryzują się:

- Brakiem modelu relacyjnego
- Istnieje możliwość rozproszenia danych
- Systemy NoSQL mogą skalować się w poziomie
- Większość baz danych NoSQL są typu open source

Autor: Piotr Chudzik

Czym są bazy danych NoSQL?

Wykorzystując NoSQL zyskujemy:

- Brak ograniczeń SQL – reguła ACID nie obowiązuje NoSQL
- Zyskujemy lepszą elastyczność i dostępność do danych kosztem spójności tych danych
- Wysoka wydajność zapisu
- Prostsza obsługa dużego wolumenu danych

Autor: Piotr Chudzik

Czym są bazy danych NoSQL?

Baza NoSQL nie posiada:

- Wbudowanej integralności danych
- Skomplikowane zapytania wymagają dużo czasu
- Mało która baza NoSQL dostarcza interfejs SQL
- Nie ma standaryzacji danych
- Bazy NoSQL nie są przystosowane do złożonych danych. Taki typ danych może mocno spowolnić działanie bazy.

Autor: Piotr Chudzik

Czym są bazy danych NoSQL?

Do głównych cech NoSQL możemy zaliczyć:

- Brak narzuconego schematu pracy z danymi
- Łatwiejszy w obsłudze oraz automatyzacji co jest równoznaczne z mniejszym kosztem utrzymania/administracji
- Prostszy w migracji
- Dla programistów łatwiejszy w obsłudze

Autor: Piotr Chudzik

Do czego wykorzystujemy NoSQL?

Kilka przykładów systemów dla NoSQL:

- Facebook generuje ponad **100 miliardów wiadomości miesięcznie** (Rozmiar wiadomości: około 1MB).
- Twitter przechowuje ponad **6TB** Danych dziennie.
- Aby zapisać **6TB** danych na dysku przy standardowym łączu (około 70MB/s) będziemy potrzebowali **cały dzień**.

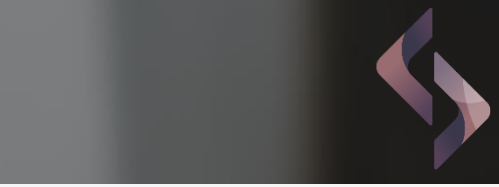
Autor: Piotr Chudzik

Do czego wykorzystujemy NoSQL?

Gdzie jeszcze możemy spotkać NoSQL?:

- W systemach czasu rzeczywistego, gdzie wynik funkcji jest zależny od czasu np. systemy pomiaru prędkości.
- Gry MMO
- Systemy cechujące się szybką reakcją (płatności zbliżeniowe)
- Bazy o wysokiej częstotliwości odczytu

Autor: Piotr Chudzik



Baza Danych SQL

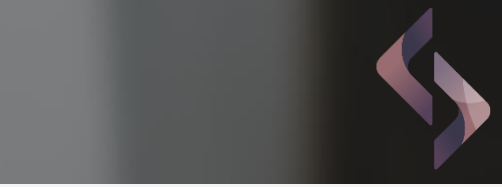
Tabela pracowników

ID	Imie	Nazwisko	Oddzial
1	Adam	Nowak	1
2	Ewa	Kowalska	2

Tabela telefonów

ID	Pracownik	Telefon
1	1	400-213-980
2	2	123-900-455
3	2	400-300-955

Autor: Piotr Chudzik



Baza Danych NoSQL

ID: 1	Imie: Adam	Nazwisko: Nowak	Telefony: ["400-213-980"]
ID: 2	Imie: Ewa	Nazwisko: Kowalska	Telefony: ["123-900-455","400-300-955"]

Autor: Piotr Chudzik

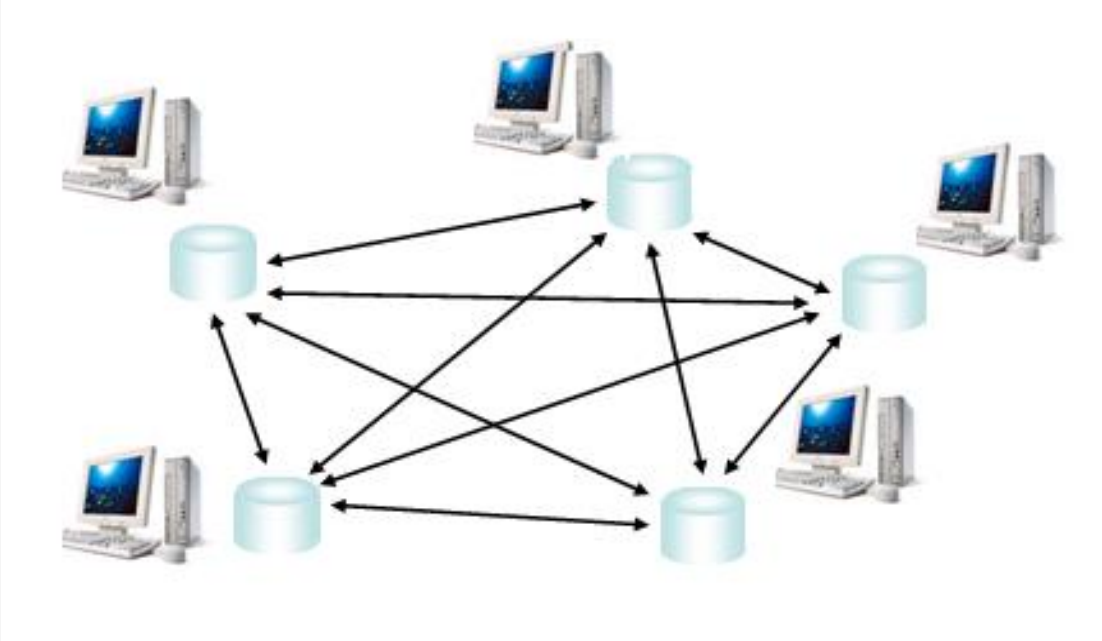
Czym jest rozproszona baza danych?

Rozproszona baza danych to zespół baz danych połączonych w całość, zlokalizowany na różnych serwerach.

Użytkownik nie wie, czy dane pochodzą z pojedynczej bazy i komputera, czy z wielu komputerów (system dla użytkownika jest przezroczysty).

System rozproszony

- Istnieje ona na dwóch lub więcej komputerach
- Jest przezroczysta dla aplikacji (traktowana jako całość)
- Zmiany na pojedynczej maszynie są uwzględniane na pozostałych
- Zwiększona wydajność w przetwarzaniu danych
- Zbiór komputerów tworzących system rozproszony nazywamy **klastrem** (*ang. cluster*).



System rozproszony – czym jest dokładnie klaster?

Technologia umożliwiająca łączenie kilku systemów komputerowych w jedność. Użytkownik nie wie, czy komunikuje się z pojedynczym serwerem, czy farmą maszyn.

Podstawowe podejścia:

- Zduplikowanie węzłów sieci i zapewnienie ciągłego działania systemu. Awaria pojedynczego serwera nie powoduje przerwanie pracy (*ang. fail-over system*). Wszystkie informacje są zapisywane na każdym pracującym równolegle serwerze (z pewnym opóźnieniem). W przypadku awarii użytkownicy są kierowani na działającą maszynę.
- Klient (np. aplikacja) jest połączony do klastra na podstawie zaprogramowanych reguł:
  - Balans obciążenia (*ang. load balancer*) – łączymy się z maszyną o najmniejszym opóźnieniu.
  - Serwer dedykowany (*ang. dedicated servers*) – łączymy zawsze się z pierwszym serwerem z listy. Jeżeli jest niedostępny, wybieramy kolejny z listy.
  - Na podstawie algorytmu Round-Robina – wybieramy losowy serwer z listy.

Autor: Piotr Chudzik

System rozproszony

Wyróżniamy dwa sposoby na stworzenie rozproszonej bazy danych:

- Decentralizacja (*ang. decentralization*) – pojedyncza baza danych zostanie podzielona na części, które zostaną umieszczone w różnych węzłach sieci.
- Scalenie (*ang. consolidation*) – zbiór baz danych łączymy w taki sposób, aby korzystać z niego jako pojedynczej bazy danych.

Autor: Piotr Chudzik

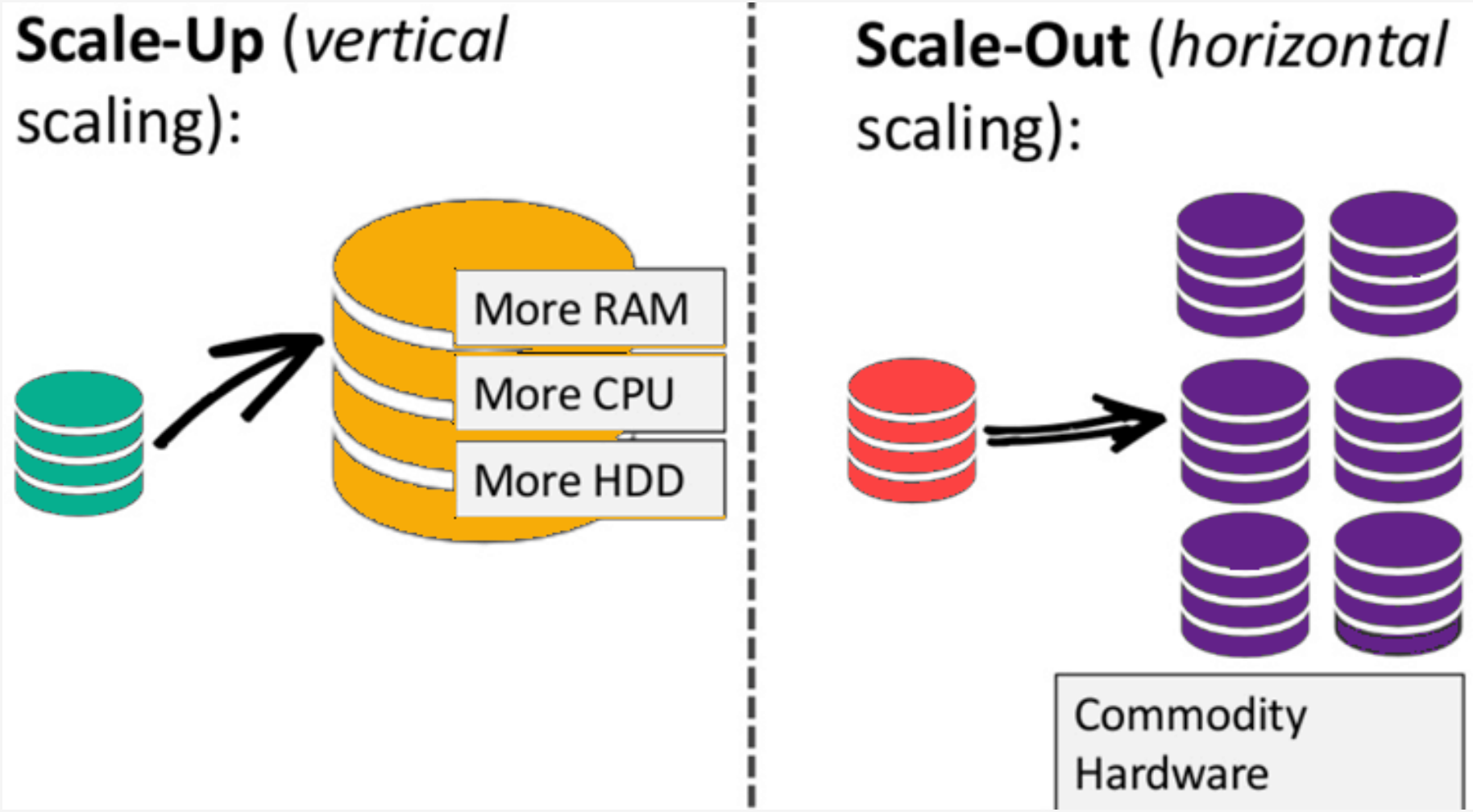
System rozproszony

Podział danych w poszczególnych węzłach w rozproszonej bazie danych następuje poprzez fragmentację lub replikację danych:

- Podział (*ang. partition*) – dane zostanę rozłożone pomiędzy węzłami bazy danych. W razie awarii nie mamy dostępu do części danych.
- Replikacja *(ang. replication)* – danę są kopiowane pomiędzy węzłami. W razie awarii nie utracimy dostępu do żadnych danych kosztem zwiększonym czasem zapisu.

Autor: Piotr Chudzik

System rozproszony



Teoria CAP oraz BaSE

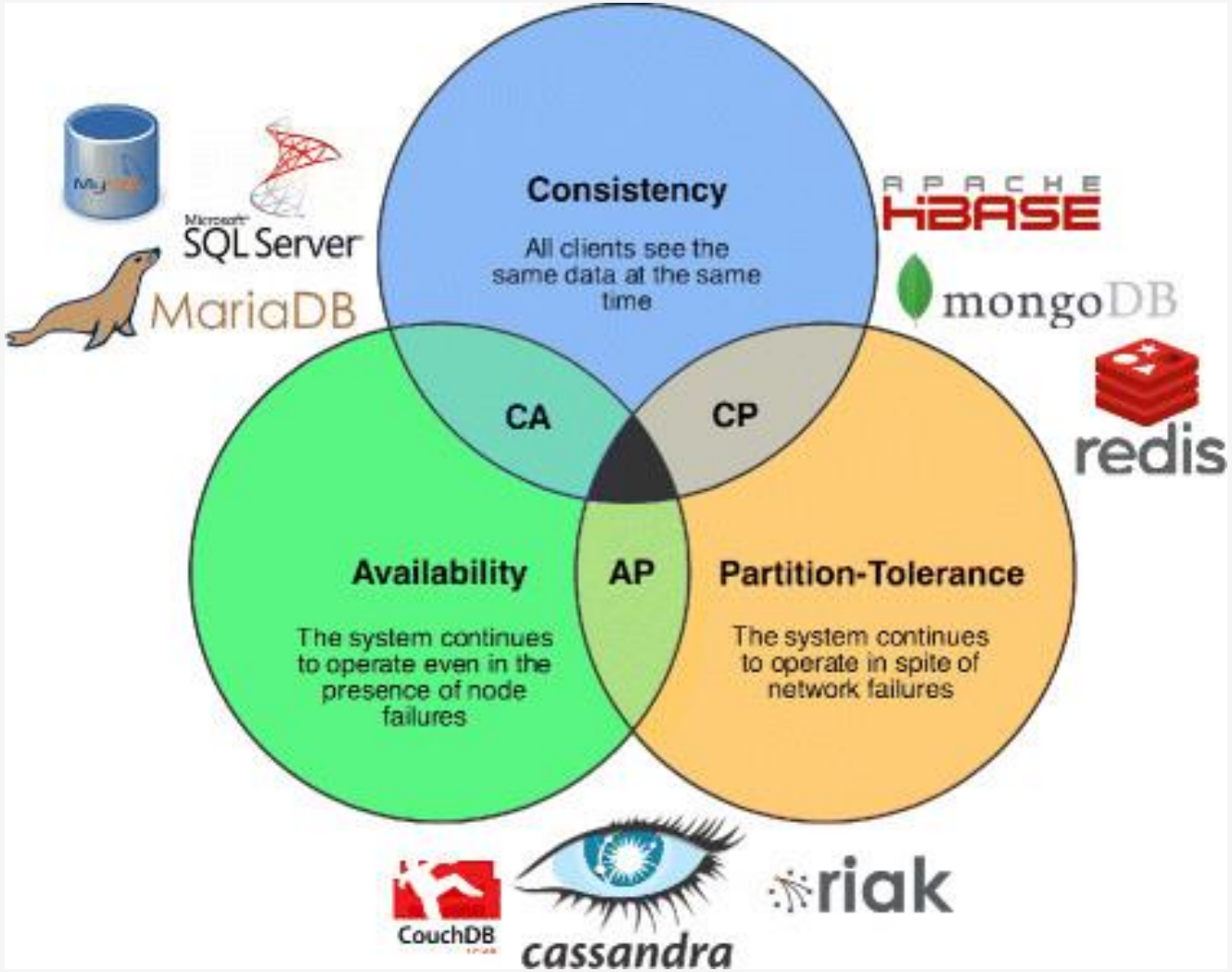
Teoria CAP to warunki, które mogą spełniać bazy danych:

- Spójność (*ang. constistency*) – każdy odczyt zwraca najnowszy zapis lub błąd
- Dostępność (*ang. avaibility*) – każde zapytanie zwraca niebłędną odpowiedź. Nie mamy pewności, że zwróci najnowszy zapis do bazy danych
- Możliwość podziału (*ang. parition tolerance*) – system działa pomimo dowolnej ilości zgubionych lub opóźnionych przez sieć pomiędzy węzłami wiadomości. Niedostępność pojedynczego węzła nie powoduje awarii całego systemu.

Według teorii CAP każdy silnik bazy danych może spełniać tylko dwa warunki.

Autor: Piotr Chudzik

Teoria CAP oraz BaSE



Autor: Piotr Chudzik

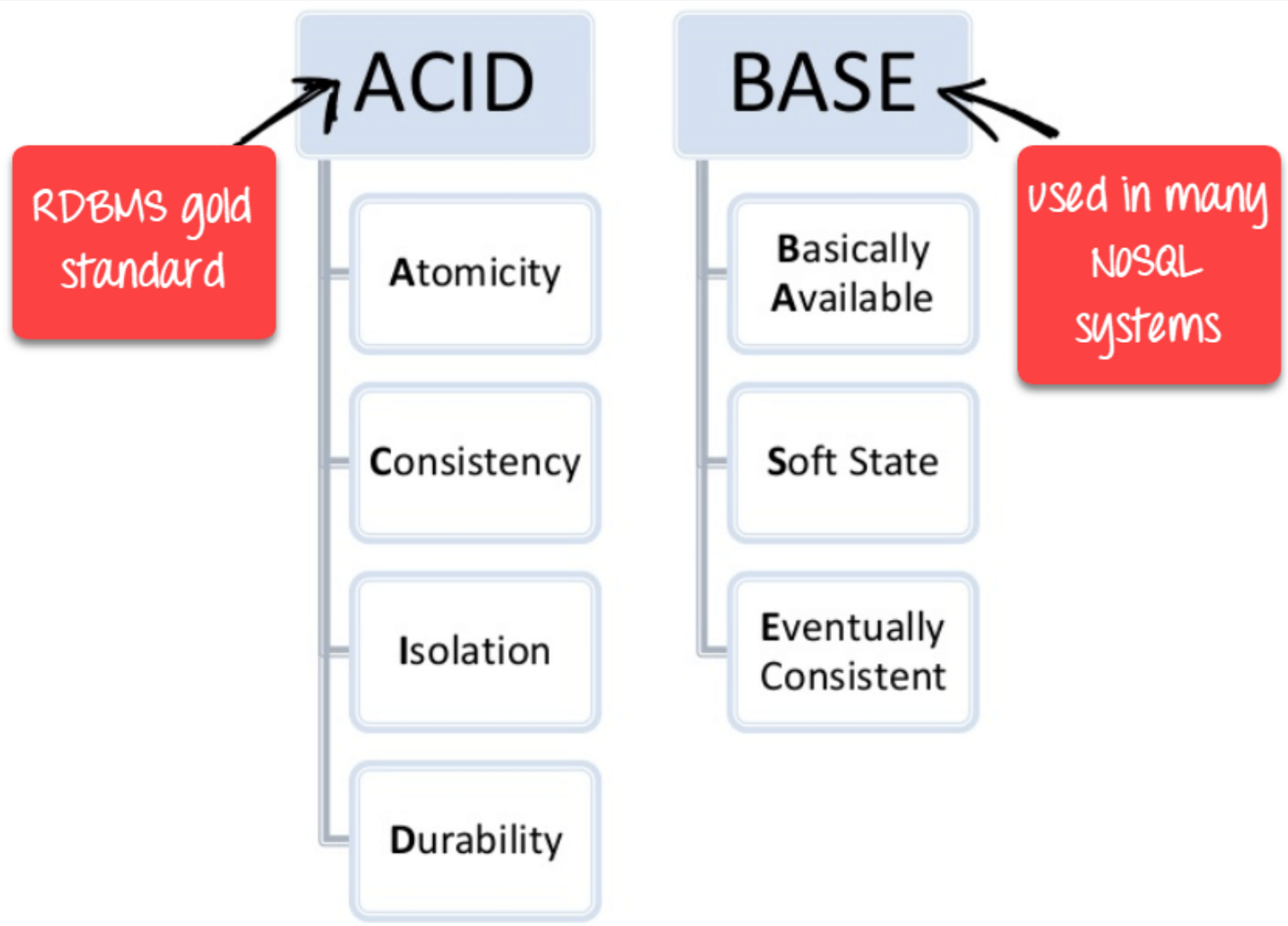
Teoria CAP oraz BaSE

Bazy danych NoSQL spełnieją kompromis **BaSE**:

- **B**asically **A**vailable – gwarancja odpowiedzi zgodnej z teorią CAP
- **S**oft State – zapisy w bazie nie muszą być spójnie na poziomie zapisu ani różne repliki nie muszą być ciągle spójne
- **E**ventually **C**onsistent – zapisy zapewniają spójność z opóźnieniem

Autor: Piotr Chudzik

Teoria CAP oraz BaSE



Autor: Piotr Chudzik

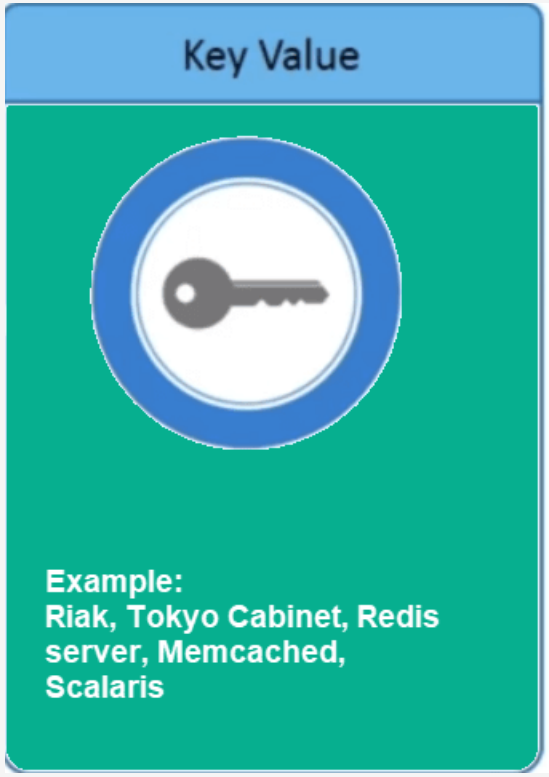
Rodzaje baz NoSQL

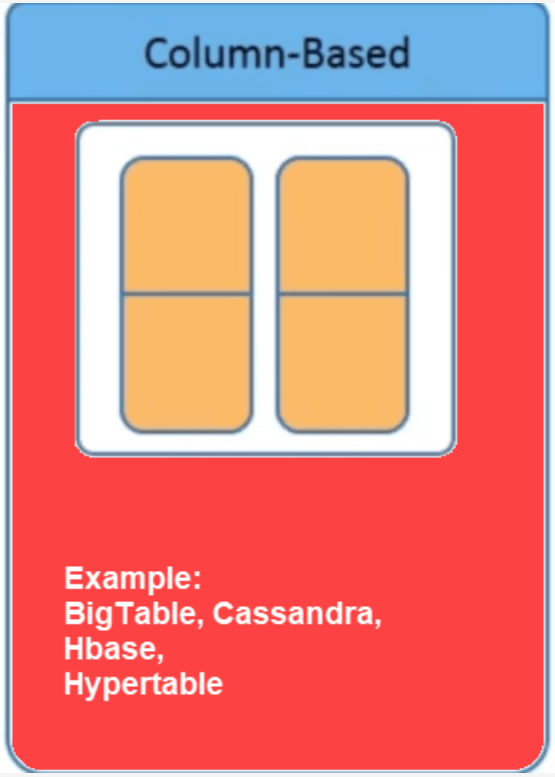
Bazy danych NoSQL możemy podzielić na kategorie:

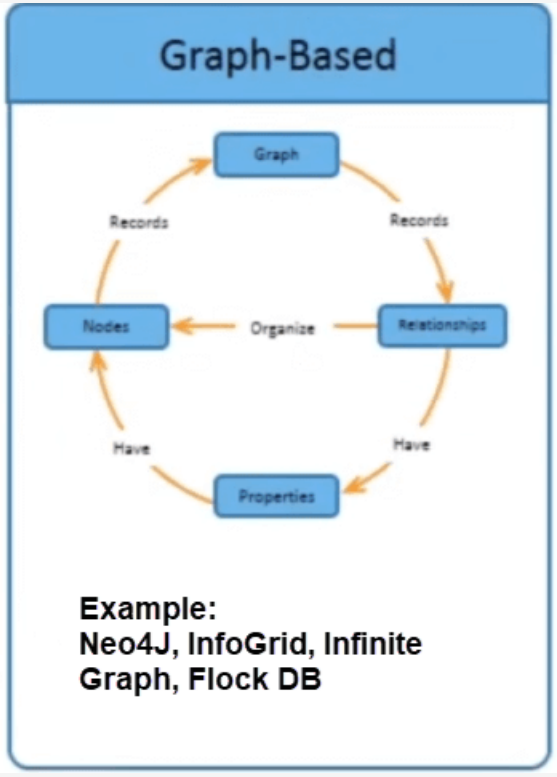
- Asocjacyjne: Redis, Voldemort
- Dokumentowe: MongoDB, RavenDB
- Kolumnowe: Cassandra, Hbase
- Grafowe: Titan, Neo4j

Autor: Piotr Chudzik

Rodzaje baz NoSQL







Autor: Piotr Chudzik

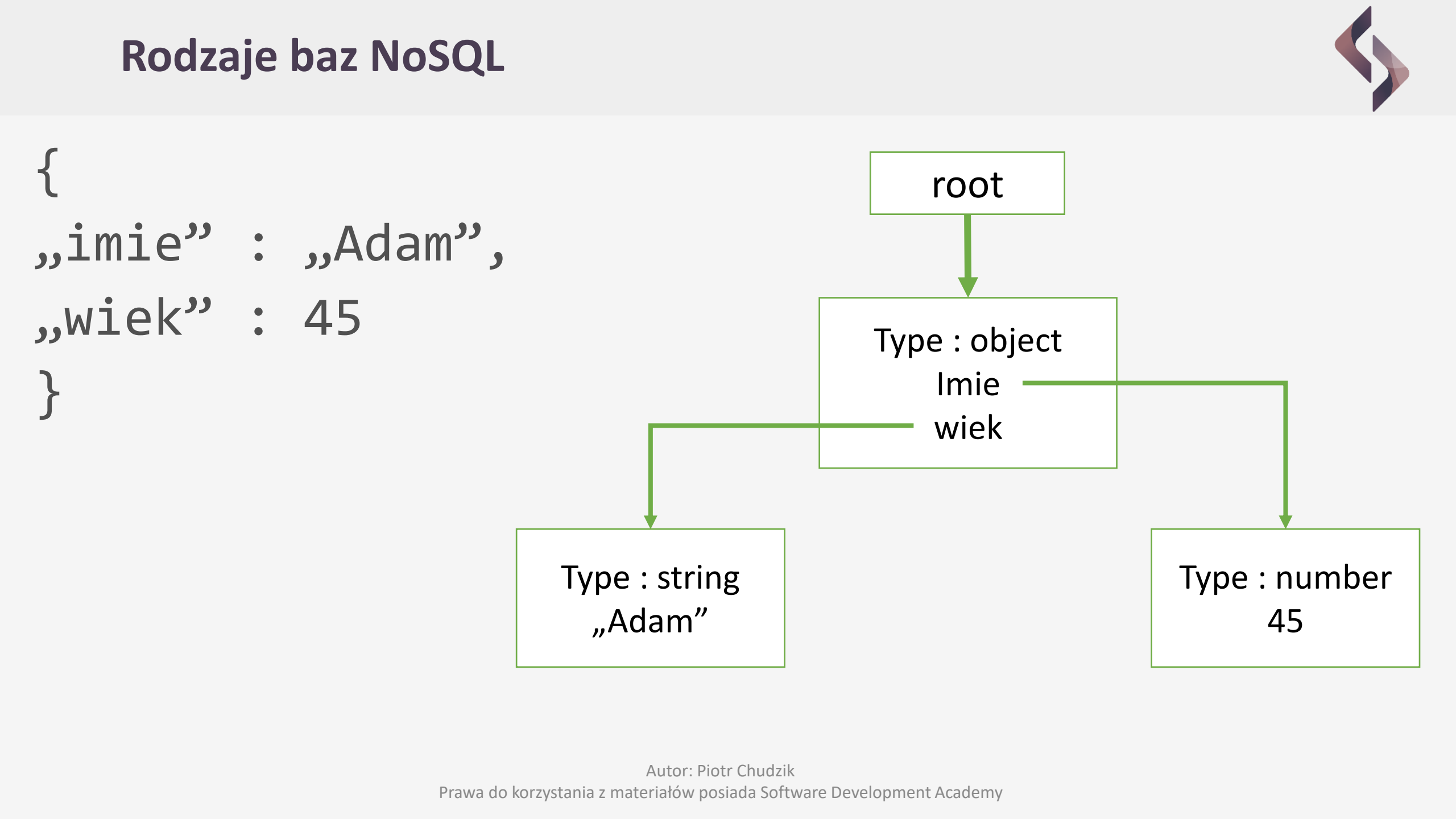
Rodzaje baz NoSQL

Główne cechy baz asocjacyjnych:

- Jest to najprostszy rodzaj baz NoSQL
- Opierają się na parze klucz:wartość
- Pod atrybutem trzymamy wartość określonego typu.

Autor: Piotr Chudzik

Rodzaje baz NoSQL



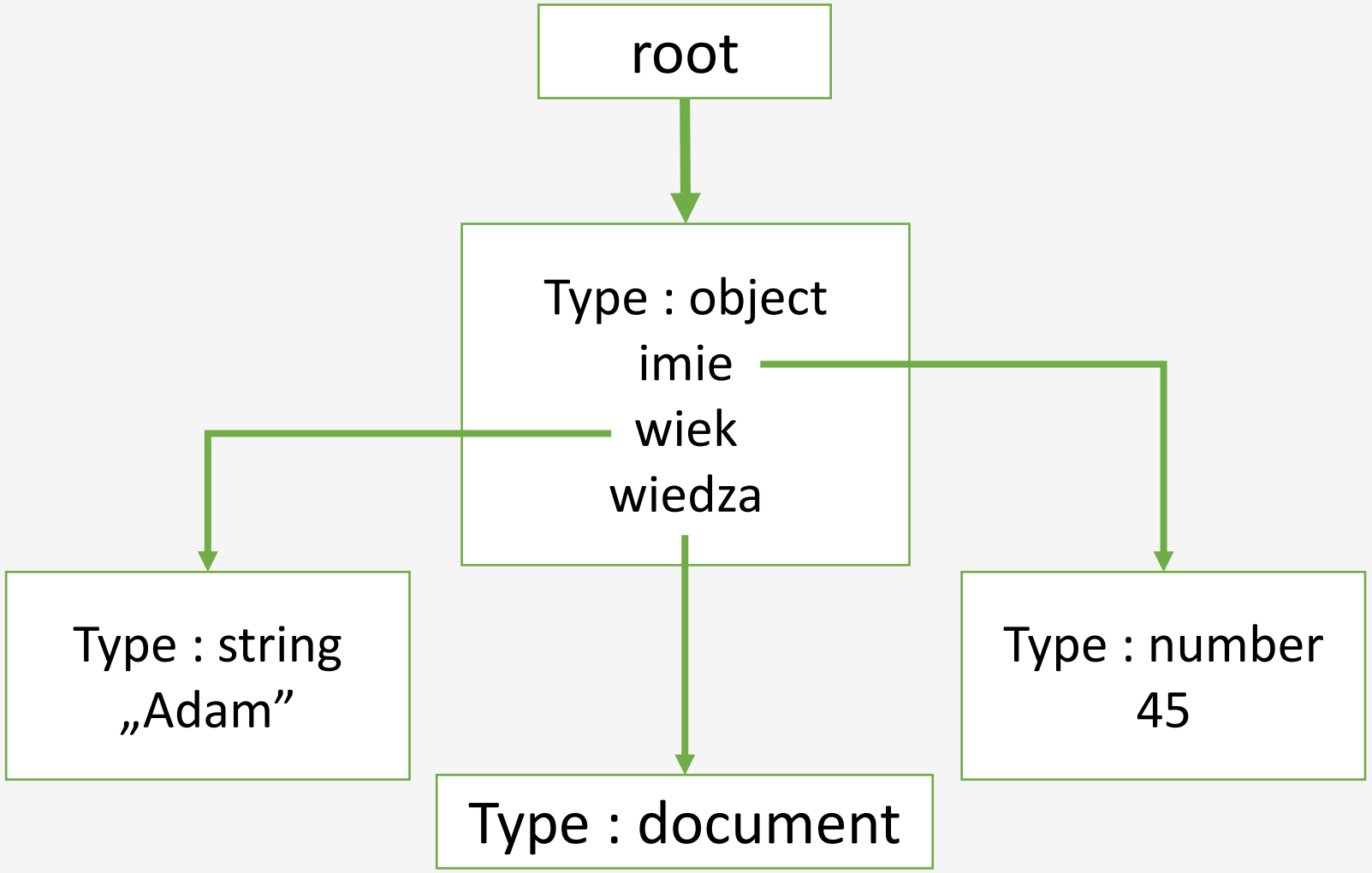
Rodzaje baz NoSQL

Główne cechy baz dokumentowych:

- Opiera się na parze klucz:dokument (dokument to mapa obiektów)
- Dokument może mieć zagnieżdżone kolejne dokumenty.

Rodzaje baz NoSQL

```
{
„imie” : „Adam”,
„wiek” : 45,
„wiedza” :
        {
        „SQL” : „Ekspert”,
        „Java” : „Podstawy”
        }
}
```



Autor: Piotr Chudzik

Rodzaje baz NoSQL

Główne cechy baz kolumnowych:

- Opierają się na systemie mapowania
- Wykorzystują dużą ilość kolumn, zamiast wierszy

Rodzaje baz NoSQL

Główne cechy baz grafowych:

- Przechowują informacje o sieciach
- Przeznaczony do struktur grafowych
- Często dostarczane ze specjalnym językiem zapytań

Autor: Piotr Chudzik

Format plików JSON i jego rodzaje

JSON (*ang. JavaScript Object Notation*) to lekki format wymiany danych komputerowych. Wiele języków programowania wykorzystuje go poprzez dodatkowe biblioteki

Najważniejsze jego cechy to:

- Wykorzystuje tablice asocjacyjne
- Pozwala na wykorzystywanie jako wartości: typy proste, tablice, obiekty, kolejne JSONy
- Istnieją jego dwa dodatkowe rodzaje: BSON (wersja binarna JSONa) oraz JSONC (pozwala na umieszczenie komentarzy w strukturze)
- Wyróżniamy dwie struktury dokumentów: referencja (ang. references) lub zagnieżdżenia (*ang. embedded*)

Autor: Piotr Chudzik

Format plików JSON i jego rodzaje

Przykład danych w formacie JSON:

```
{
      „name” : „Adam”,
      „last” : „Kowalski,
      „age” : 35,
      „kids” : [„Agnieszka”, „Jakub”]
      „isMarried” : true,
      „language” : { „name” : „ENG”, „level” : „C1” }
}
```

Autor: Piotr Chudzik

Format plików JSON i jego rodzaje

Przykład zagnieżdżenia dokumentu (MongoDB – denormalizacja)

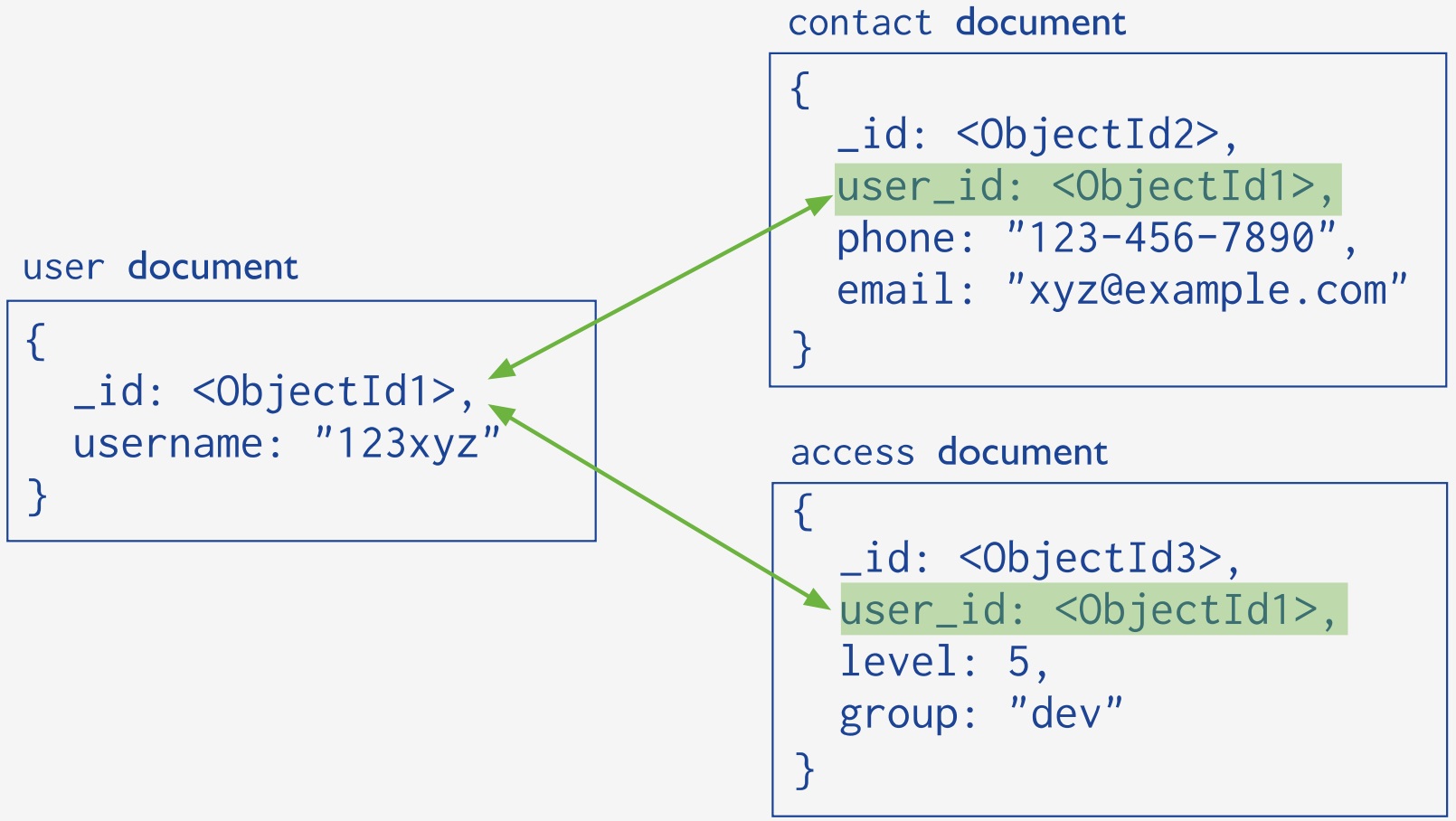
```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

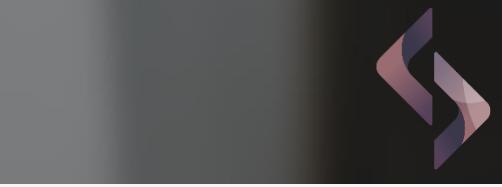
Embedded sub-document

Embedded sub-document

Format plików JSON i jego rodzaje

Przykład referencji do dokumentu (MongoDB - normalizacja)





Format plików JSON i jego rodzaje

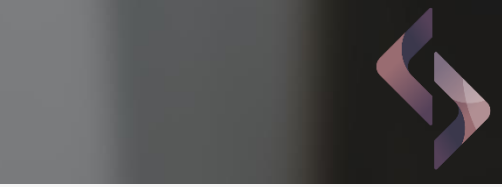
Embedded data model

- Wysoka wydajność
- Używany w relacjach 1:1
- Powoduje duplikacje danych
- Wymaga jednego zapytania, aby pozyskać wszystkie dane

References data model

- Używany, jeżeli korzyści wydajności pierwszej opcji nie przeważają kosztu duplikacji danych
- Duże, hierarchiczne zbiory danych
- Reprezentacja bardziej złożonych relacji wielu do wielu
- Wymaga kilku zapytań do bazy danych

Autor: Piotr Chudzik



Baza Danych MongoDB

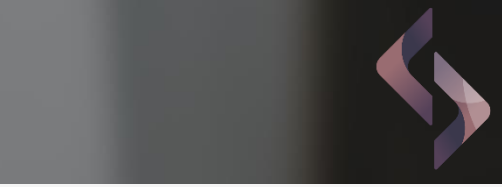
Model relacyjny

- Dane przechowywane w wierszach
- Jeden PK może tworzyć wiele relacji (być powiązany z kilkoma tabelami)
- Jednorodna struktura tabeli (schematu)

Model dokumentowy

- Dane są przechowywane w dokumentach typu JSON
- Jeden dokument (zawierający wiele encji) to jeden rekord
- Dokumenty nie muszą mieć jednorodnej struktury

Autor: Piotr Chudzik



Baza Danych MongoDB

Model relacyjny

- Wiersz
- Tabela
- Klucz – dowolny typ danych, może składać się z kilku kolumn

Model dokumentowy

- Dokument
- Kolekcja
- Klucz – typ danych _id

Autor: Piotr Chudzik

Baza danych MongoDB

Klucz typu _id to unikalny identyfikator pozwalający rozróżnić dokumenty. Nie możemy go usunąć i zostanie on automatycznie utworzony jeżeli nie został on podany.

Autor: Piotr Chudzik

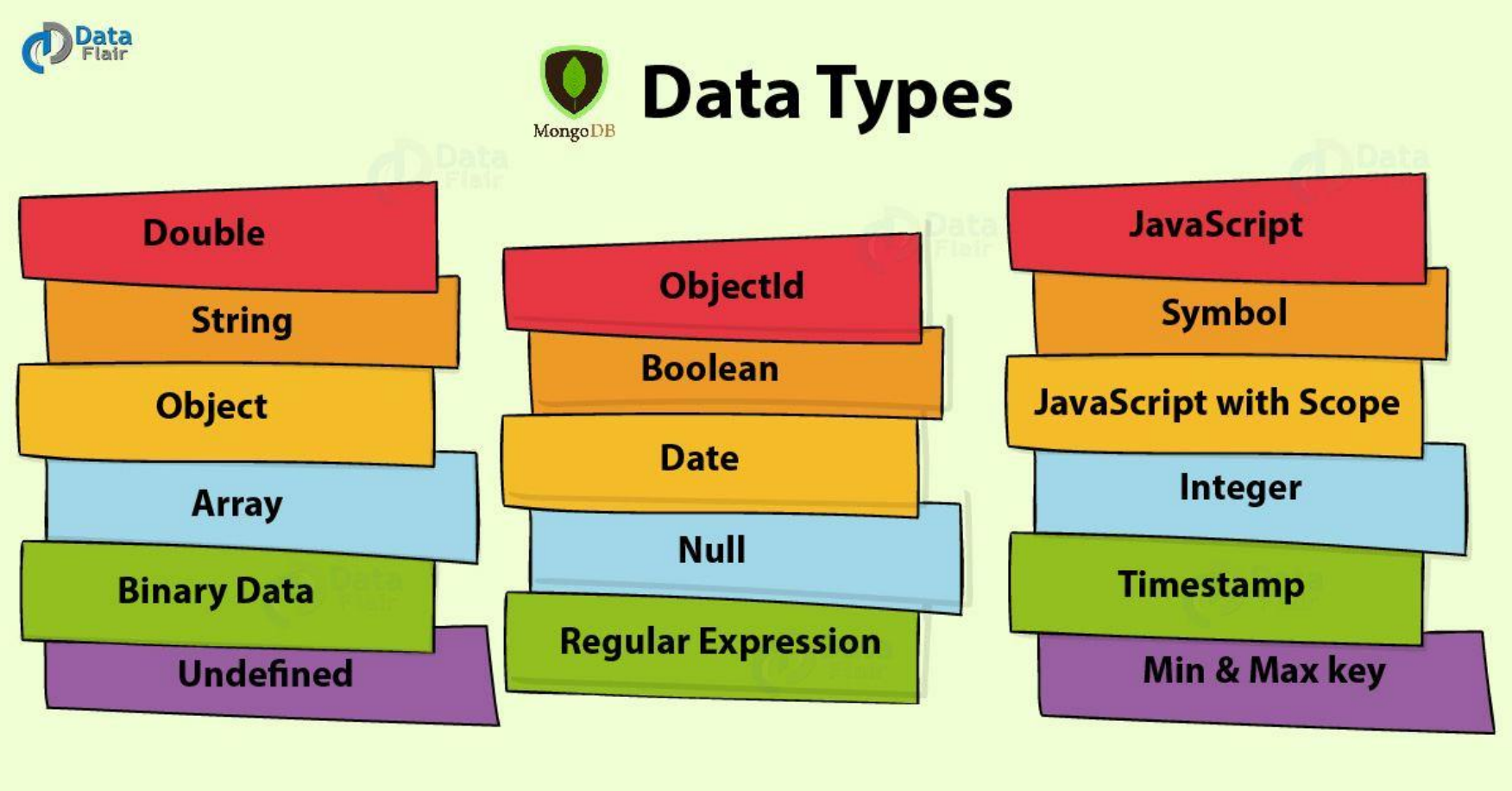
Baza danych MongoDB

W MongoDB istnieje specjalny ObjectID. Jest to 12 bajtowy identyfikator generowany według schematu:

- Pierwsze 4 bajty oznaczają czas w formacie UNIX
- Kolejne 3 bajty to ID maszyny (np. nazwa węzła)
- Następne 2 bajty to ID procesu, który tworzy dokument
- Ostatnie 3 bajty tworzą wartość losową

Autor: Piotr Chudzik

Baza danych MongoDB



Źródło: https://data-flair.training/blogs/mongodb-data-types/

Baza danych MongoDB

Podstawowe polecenia MongoDB:

- Której bazy danych aktualnie używamy: `db`
- Pokaż wszystkie bazy danych: `show dbs`
- Przełącz bazę danych/stwórz nową: `use <nazwa_db>`
- Pokaż kolekcje bazy danych: `show collections`
- Wyświetl pomoc: `help`

Autor: Piotr Chudzik

Baza danych MongoDB | Dodawanie dokumentu

MongoDB

```
db.<nazwa_kolekcji>.insert(<dokument>)
db.animals.insert({ „type” : „cat” })
```

MySQL

```
INSERT INTO animals VALUES (‚cat’);
```

Autor: Piotr Chudzik

Baza danych MongoDB | Dodawanie kilka dokumentów

MongoDB

```
db.<nazwa_kolekcji>.insertMany([<dokument1>,<dokument2>])
db.animals.insertMany([{ „type” : „cat” }, {„type” : „dog”}])
```

MySQL

```
INSERT INTO animals VALUES (‚cat’), (‚dog’);
```

Autor: Piotr Chudzik

Baza danych MongoDB | Dodawanie dokumentu

Ćwiczenie:

- Utwórz bazę danych o nazwie **sda**.
- Dodaj zawartość dokumentu *Employee01.json* za pomocą funkcji insert do kolekcji **workers**.
- Dodaj pozostałą zawartość dokumentów za pomocą funkcji insertMany do kolekcji **workers**.

Autor: Piotr Chudzik

Baza danych MongoDB | Odczytywanie danych

MongoDB

```
db.<nazwa_kolekcji>.find()
db.animals.find()
```

MySQL

```
SELECT * FROM animals;
```

Autor: Piotr Chudzik

MongoDB

```
db.<nazwa_kolekcji>.findOne()          <- Wyświetl tylko jeden
db.<nazwa_kolekcji>.find().limit(n)    <- Wyświetl n dokumentów
```

```
1. db.animals.findOne()
2. db.animals.find().limit(4)
```

MySQL

```
1. SELECT * FROM animals LIMIT 1;
2. SELECT * FROM animals LIMIT 4;
```

Autor: Piotr Chudzik

MongoDB

```
{
   „name” : „Jan”,
   „address” : { „street” : „Piotrkowska”, „nr” : 35 }
}
```

Aby „dostać się” do imienia: name

Aby „dostać się” do nazwy ulicy: address.street

MongoDB

```
db.<nazwa_kolekcji>.find().skip(n)        <- Pomiń n dokumentów
db.animals.find().skip(1)
```

MySQL

```
1. SELECT * FROM animals LIMIT 2,100;
```

Autor: Piotr Chudzik

Baza danych MongoDB | Odczytywanie danych

Ćwiczenie:

- Wyświetl pierwszy dokument z kolekcji **workers**.
- Wyświetl trzy dokumenty z kolekcji **workers**.
- Pomiń pierwsze 2 dokumenty podczas wyświetlenia z kolekcji **workers**.

Autor: Piotr Chudzik

Baza danych MongoDB | Operatory

- 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**

Autor: Piotr Chudzik

MongoDB

```
1. db.animals.find({„type” : „cat”})
2. db.animals.find({$and : [{„type” : „cat”}, {„name” : „Meow”}]})
3. db.animals.find({$or : [{„nr” : 1}, {„nr” : 2}]})
4. db.animals.find({„type” : {$in : [„cat”, „dog”]}})
```

MySQL

```
1. SELECT * FROM animals WHERE type = ‚cat’;
2. SELECT * FROM animals WHERE type = ‚cat’ AND name = ‚Meow’;
3. SELECT * FROM animals WHERE nr = 1 OR nr = 2;
4. SELECT * FROM animals WHERE type IN (‚cat’,‚dog’);
```

Autor: Piotr Chudzik

Baza danych MongoDB | Operatory

Ćwiczenie:

- Wyświetl dokumenty, gdzie wiek jest równy 30 lub 24.
- Wyświetl dokumenty, gdzie wynagrodzenie jest większe niż 5500
- Wyświetl osobę o imieniu Agnieszka albo o nazwisku Smith.

Autor: Piotr Chudzik

Baza danych MongoDB | Wyświetlanie pozycji dokumentu

MongoDB

```
db.<nazwa_kolekcji>.find({„klucz” : wartosc},{„klucz” : 1|0}, …)
1 – pokaż | 0 - ukryj

db.animals.find({ „type” : „cat” }, {„age” : 1})
```

MySQL

```
SELECT age FROM animals WHERE type = ‚cat’;
```

Autor: Piotr Chudzik

Baza danych MongoDB | Sortowanie

MongoDB

```
db.<nazwa_kolekcji>.find().sort({klucz : 1|-1})
1 – rosnąco | -1 – malejąco

db.animals.find().sort({“type” : 1})
```

MySQL

```
SELECT * FROM animals ORDER BY type;
```

Autor: Piotr Chudzik

Baza danych MongoDB | Sortowanie & Wyświetlanie

Ćwiczenie:

- Wyświetl jedynie imiona i nazwiska z dokumentów w kolekcji workers.
- Pomiń wyświetlenie pozycji wynagrodzenia z dokumentów.
- Posortuj dokumenty według wieku malejąco, nie wyświetlając go.

Autor: Piotr Chudzik

Baza danych MongoDB | Count

MongoDB

```
db.<nazwa_kolekcji>.find().count()
db.animals.find().count()
```

MySQL

```
SELECT COUNT(*) FROM animals;
```

Autor: Piotr Chudzik

Baza danych MongoDB | $elemMatch

$elemMatch pozwala na szukanie dokumentu, który zawiera dokument zagnieżdzony, który spełnia wszystkie warunki.

```
db.people.find({
                „awards” : {
                      $elemMatch : {
                                  „by” : „IEE Computer Society”,
                                  „Year” : { $lt : 1960 }
                                  }
                }
                })
```

Baza danych MongoDB | $elemMatch

Ćwiczenie:

- Znajdź w kolekcji **bios** osoby co dostały od ACM w 2001 roku.

Autor: Piotr Chudzik

Baza danych MongoDB | Aktualizacja danych

MongoDB

```
db.<nazwa_kolekcji>.update({klucz : wartość},{$set : { klucz : nowa_wartość }} [,{upsert : true}] [,{multi : true}])
```

upsert – utwórz dokument z tym kluczem, jeżeli nie istnieje żaden dokument.
Domyślnie: false
multi – aktualizacja wielu dokumentów.
Domyślnie: false

```
db.animals.update({„type” : „cat”}, {$set : {„name” : „Kitty”}},{multi : true})
```

MySQL

```
UPDATE animals SET name = ‚Kitty’ WHERE type = ‚cat’;
```

Autor: Piotr Chudzik

Baza danych MongoDB | Aktualizacja danych

Ćwiczenie:

- Dodaj klucz „isZeroPit” o wartości **true** dla osoby, która mam mniej niż 26 lat.
- Ustaw wynagrodzenie na 10000 dla osoby, której wynagrodzenie wynosi 12000

Autor: Piotr Chudzik

Baza danych MongoDB | Usuwanie danych

MongoDB

```
db.<nazwa_kolekcji>.remove(klucz : wartość, …) ← usuwanie dokumentów
db.<nazwa_kolekcji>.drop()                     ← usuwanie kolekcji
```

1. `db.animals.remove({„type” : null})`
2. `db.animals.drop()`

MySQL

1. `DELETE FROM animals WHERE type IS NULL;`
2. `DROP TABLE animals;`

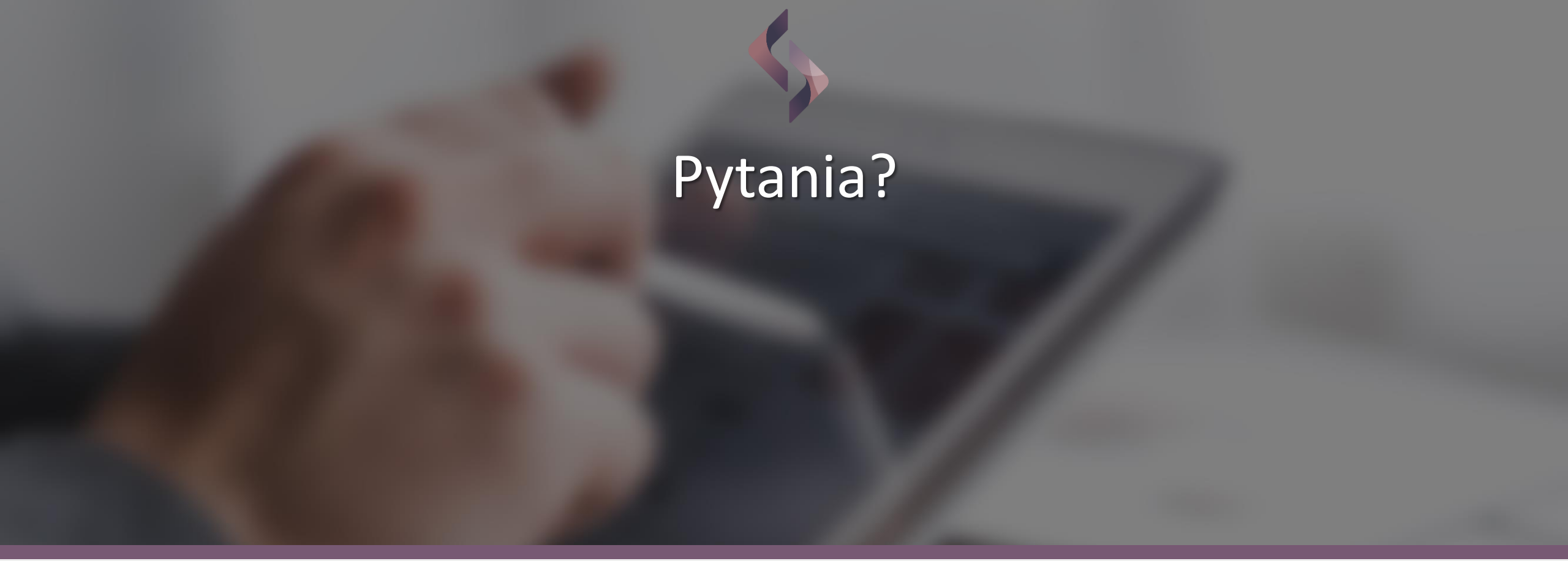
Autor: Piotr Chudzik

Baza danych MongoDB | Usuwanie dokumentów

Ćwiczenie:

- Usuń pracownika z kluczem nationality i wartością USA.
- Usuń kolekcję workers.

Autor: Piotr Chudzik



Pytania?

Autor: Piotr Chudzik