NIERELACYJNE ROZWIĄZANIA BAZODANOWE

WYKŁAD 1

PROGRAM ZAJĘĆ

Wykład

- Wprowadzenie do nierelacyjnych baz danych (1h)
- Bazy dokumentowe, grafowe, klucz-wartość oraz kolumnowe (2h)
- Modele nierelacyjnych baz danych oraz przegląd oprogramowania (3h)
- Polecenia tworzące, wybierające, aktualizujące i usuwające (3h)
- Rozwiązania Big Data dla nierelacyjnych baz danych uczenie maszynowe, klasyfikowanie danych, web scraping (6h)

Ćwiczenia

- Zakładanie baz danych, struktura dokumentów JSON i ich charakterystyka (2h)
- Instrukcje tworzące kolekcje i dokumenty, definicja pól danych, typy danych (2h)
- Dokumenty zagnieżdżone, indeksowanie dokumentów (2h)
- Importowanie i eksportowanie danych pomiędzy systemami relacyjnymi, nierelacyjnymi oraz częściowo ustrukturyzowanymi oraz internetowe źródła danych web scraping oraz media społecznościowe (4h)
- Aspekty jakości danych w nierelacyjnych bazach danych dane ze stron internetowych, z mediów społecznościowych i deduplikacja danych (2h)
- Klasyfikowanie treści w nierelacyjnych bazach danych (3h)
- Wykorzystanie rozwiązań Big Data w nierelacyjnych bazach danych - studia przypadków obejmujące web scraping, media społecznościowe, uczenie maszynowe (30h)

LITERATURA

Podstawowa lista lektur	1. Wrycza S., Maślankowski J. (red.) Informatyka ekonomiczna. Teoria i zastosowania., PWN, 2019 (rozdział Bazy danych. Big Data.) 2. Guy H., NoSQL, NewSQL i BigData. Bazy danych następnej generacji, Helion, 2019 3. Materiały zamieszczone na Portalu Edukacyjnym UG: http://pe.ug.edu.pl.
Uzupełniająca lista lektur	1. Dokumentacja bazy MongoDB (http://mongodb.com) 2. Dokumentacja bazy ElasticSearch (https://www.elastic.co/guide/index.html) 3. Dokumentacja języka Python (http://python.org) 4. Dokumentacja języka Java (https://docs.oracle.com/en/java/) **5**. Bierer D., Learn MongoDB 4.x: A guide to understanding MongoDB development and administration for NoSQL developers, Packt Publishing, 2020 6. Sadalage P.J., Fowler M., NoSQL. Kompendium wiedzy, Helion, 2015 7. Sullivan D., NoSQL. Przyjazny przewodnik, Helion, 2015

KRYTERIA ZALICZENIA

Sposób oceniania (składowe)	Próg zaliczeniowy	Składowa oceny końcowej
kolokwium - samodzielne rozwiązanie problemu postawionego przez prowadzącego	51%	40%
projekt - system bazodanowy	51%	20%
egzamin - test	51%	40%

AGENDA

- Bazy NoSQL
- Praktyczne aspekty NoSQL

BAZY NOSQL

- Magazyn dokumentów
  - Standardowy format (XML, JSON itp.)
  - Klucze unikalne
- Grafowe
- Magazyny danych typu klucz-wartość (key-value)
  - Rekordy są parami klucz-wartość (definiowane rekursywnie)
  - Obsługują duże tabele

MODEL BIGTABLE

- Magazyn pamięci rozwijany przez Google
- BigTable to rozproszona, stała wielowymiarowa posortowana kolekcja typu mapa (mapuje klucz wartość)

PRZYKŁAD HBASE

Implementacja BigTable, która używa silnika HDFS

Row Key	Time Stamp	ColumnFamily `contents`	ColumnFamily `anchor`
"com.cnn.www"	t9		`anchor:cnnsi.com` = "CNN"
"com.cnn.www"	t8		`anchor:my.look.ca` = "CNN.com"
"com.cnn.www"	t6	`contents:html` = "<html>..."	
"com.cnn.www"	t5	`contents:html` = "<html>..."	
"com.cnn.www"	t3	`contents:html` = "<html>..."	

BAZY NOSQL – MODELE DYSTRYBUCYJNE

- Pojedynczy serwer
- Współdzielenie
- Replikacja master-slave
- Replikacja peer-to-peer

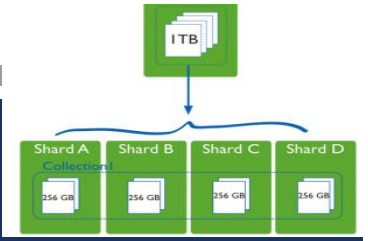
DWA SPOSOBY OBSŁUGI DANYCH ROZPROSZONYCH

- **Sharding** dzieli dane na serwery, aby każdy serwer był źródłem dla określonego podzbioru danych.
- **Replikacja** wykonuje kopie danych na serwerach, umieszczając każdą część
  w różnych miejscach.

POJEDYNCZY SERWER

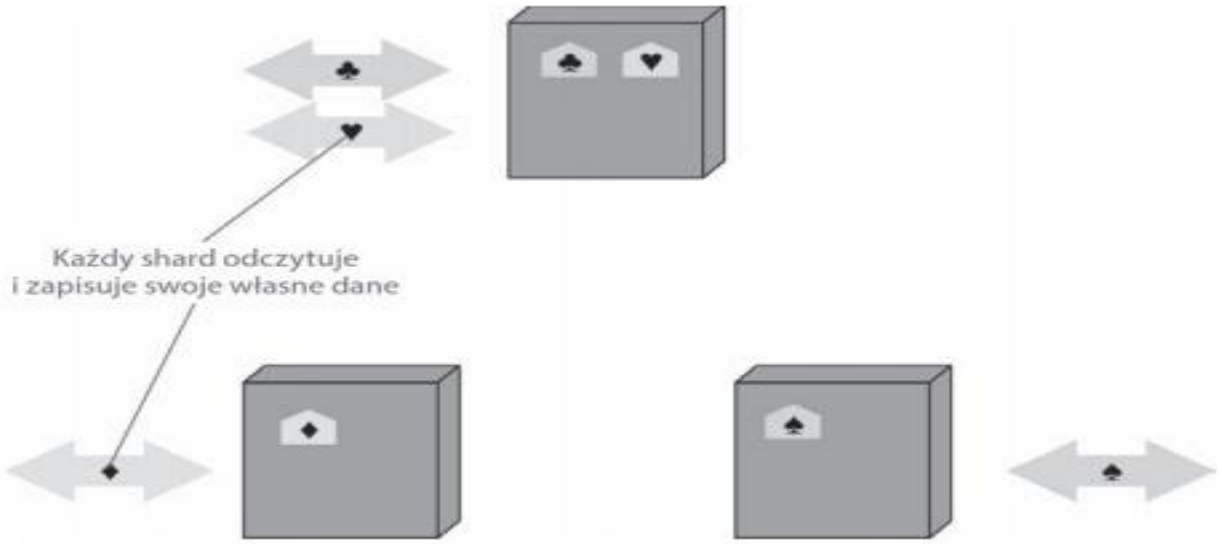
- Instalowany na pojedynczej maszynie
- Pozwala na całe spektrum dostępu dostępu do magazynu dokumentów, w tym również klucz-wartość
- Zalecany do przetwarzania prostych agregacji

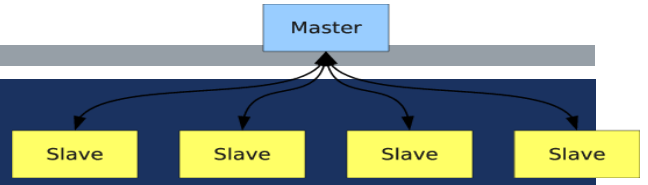
WSPÓŁDZIELENIE



- Różne osoby mogą dostępować do różnych części magazynu danych
- Wówczas magazyn danych może się zapychać
- Sharding oznacza skalowanie poziome, tj. umieszczanie różnych części danych na osobnych serwerach
- Powinien być zaimplementowany przed załadowaniem danych

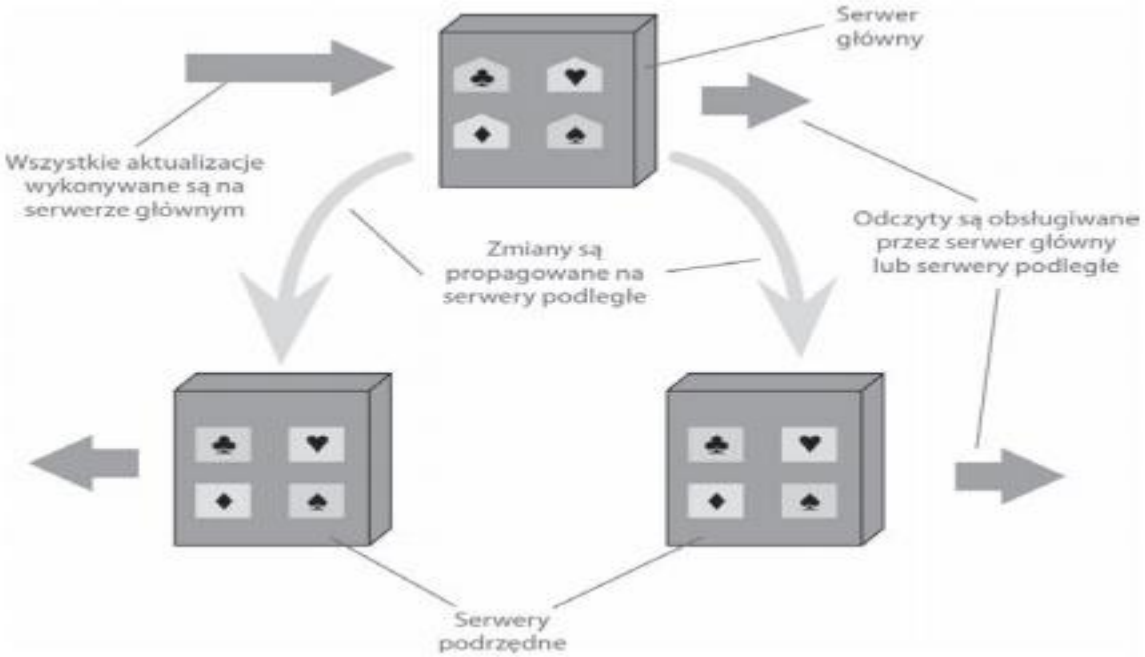
PRZYKŁAD PODZIAŁU DANYCH





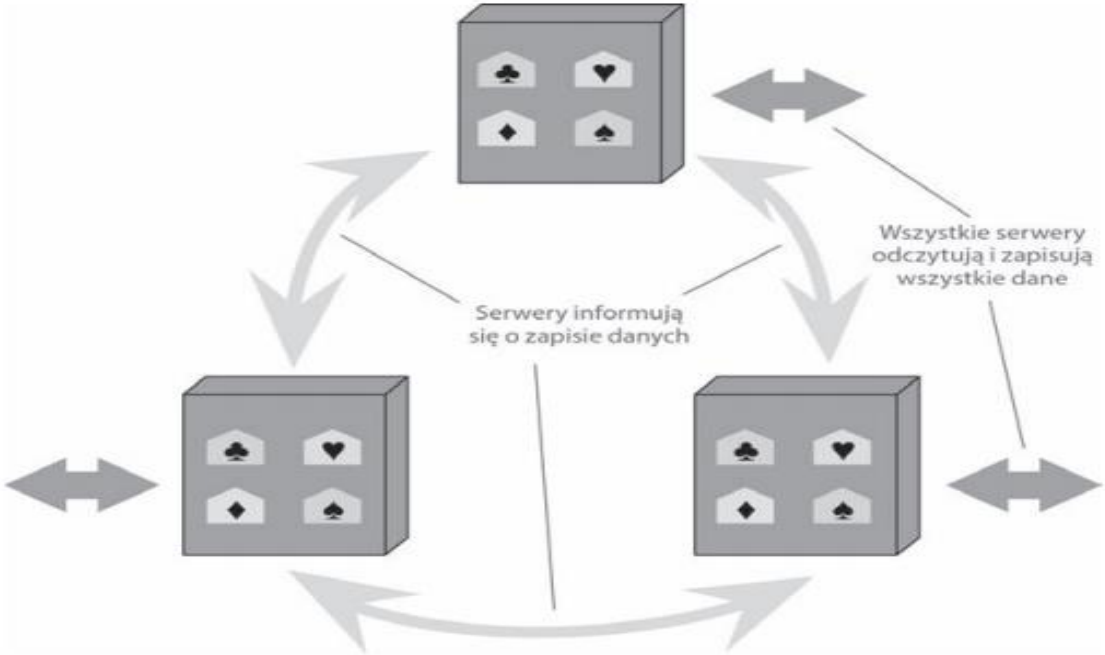
REPLIKACJA MASTER-SLAVE

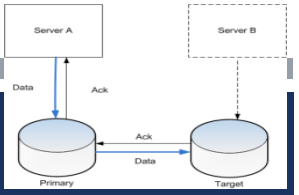
- Serwer **master** jest autorytatywnym serwerem odpowiedzialnym za podział danych
- **Slave** to serwery podległe, które są synchronizowane podczas procesu replikacji
- Serwery podległe odpowiadają na żądania, nawet gdy serwer główny przestanie odpowiadać na zapytania
- Rezultatem może być brak spójności
- Replikacja master-slave pomaga w wydajności odczytu, nie zapisu danych



REPLIKACJA PEER-TO-PEER

- Brak serwera głównego
- Wszystkie repliki mają równe prawa
- Brak spójności – w tym samym czasie może być ten sam zapis wykonywany





PODSUMOWANIE REPLIKACJI

- **master-slave** – serwer główny obsługuje zapis i rozgłasza dane do serwerów podległych, które obsługują odczyt danych
- **peer-to-peer –** dowolny serwer obsługuje zapis koordynując synchronizację danych.

BAZY NOSQL JAKO BAZY DOKUMENTÓW

- dokument,
- kolekcja,
- dokument osadzony,
- brak schematu,
- schemat polimorficzny.

DOKUMENT

- Zestaw trzyelementowy
  - { 'foo': 'a', 'bar': 'b', 'baz': 'c'}
- Torba
  - {'foo': 'a', 'bar': 'b', 'baz': 'c', 'foo': 'a'}

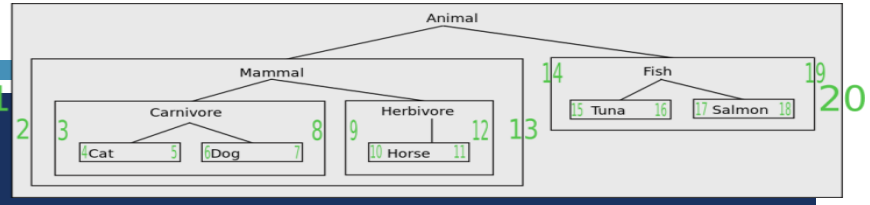
- Brak kolejności
  - { 'foo': 'a', 'bar': 'b', 'baz': 'c'}
- jest równoważny z:
  - {'baz': 'c', 'foo': 'a', 'bar': 'b'}

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

KLUCZE I WARTOŚCI

- {
  - 'nazwaPracownika' : 'Julia Kowalska',
  - 'dzial' : 'Wytwarzanie oprogramowania',
  - 'dataRozpoczecia' : '10/02/2010',
  - 'kodyZakonczonychProjektow' : [189847, 187731, 176533, 154812]
- }

DOKUMENT ZAGNIEŻDŻONY



- { 'nazwaPracownika' : 'Julia Kowalska',
- 'dzial' : 'Wytwarzanie oprogramowania',
- 'dataRozpoczecia' : '10/02/2010',
- 'zakonczoneProjekty' : {
  - { 'kodProjektu' : 189847, 'nazwaProjektu' : 'System rekomendacji produktów', 'menadzerProjektu' : 'Jagoda Dulska' },
  - { 'kodProjektu' : 187731, 'nazwaProjektu' : 'Wymiana danych finansowych, wersja 3', 'menadzerProjektu' : 'Jakub Rosiewicz'},
  - { 'kodProjektu': 176533, 'nazwaProjektu' : 'Uwierzytelnianie klientów', 'menadzerProjektu' : 'Michał Krawiec'},
  - { 'kodProjektu': 154812, 'nazwaProjektu' : 'MiesiÚczny raport sprzedaĝy', 'menadzerProjektu': 'Boĝena Rondel'}
- }

KOLEKCJA

	ArrayList	LinkedList
get()	O(1)	O(n)
add()	O(1)	O(1) amortized
remove()	O(n)	O(n)

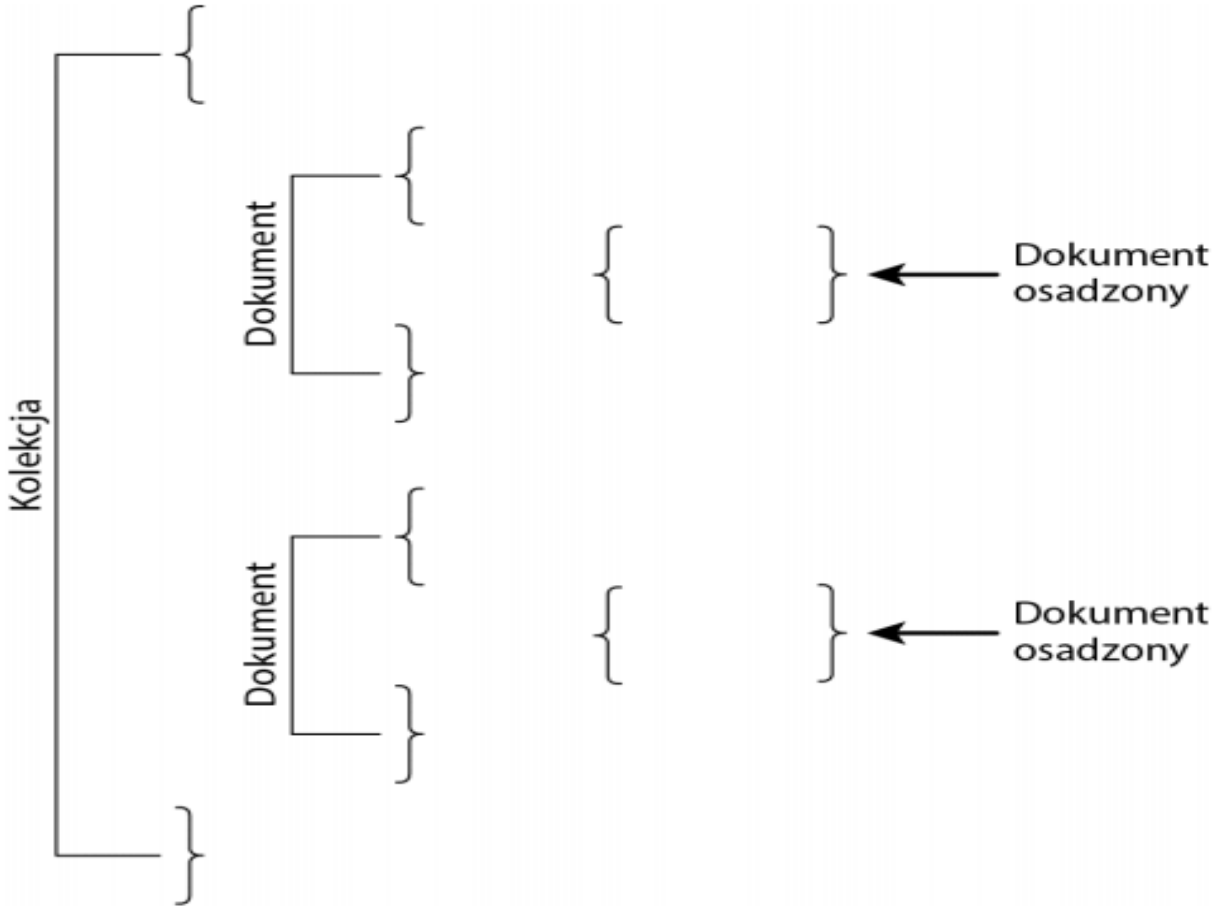
- Kolekcja to grupa dokumentów
- Dokumenty są spokrewnione, np. encją
- Zawierają indeksy zwiększające wydajność

- Przykład kolekcji:
  - Pracownik:[Dokument1, Dokument2, Dokument3]

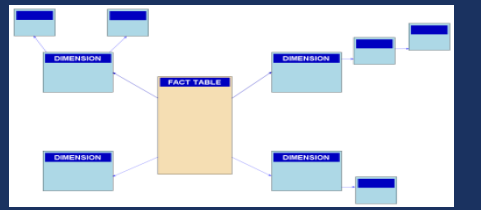
DOKUMENT OSADZONY

- Przechowuje spokrewnione dane w jednym dokumencie
- Unika się wówczas złączeń jak w tabelach relacyjnych, np. student -> przedmioty
- Podczas odczytywania danych pobierane są również dane spokrewnione, bez konieczności ich odczytywania z innej tabeli

DOKUMENT OSADZONY – PRZYKŁAD



BRAK SCHEMATU

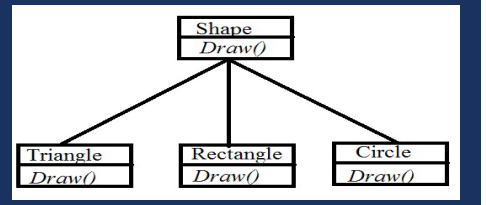


- Schematy znane z baz relacyjnych zwykle definiują następujące komponenty:
- tabele, kolumny, klucze główne, klucze obce, ograniczenia.
- Brak schematu ma dwie cechy:
  - większa elastyczność,
  - większa odpowiedzialność.

ZMIANY PAR KLUCZ-WARTOŚĆ

- { 'nazwaPracownika' : 'Anna Kowalska', 'dzial' : 'Wytwarzanie oprogramowania', 'dataRozpoczecia' : '10/02/2024', 'kodyZakonczonychProjektow' : [189847, 187731, 176533, 154812] }
- { 'nazwaPracownika' : 'Krzysztof Kowalski', 'dzial' : 'Finanse', 'dataRozpoczecia' : '21/05/2024', 'certyfikaty' : 'CPA' }

SCHEMAT POLIMORFICZNY



- Dokumenty zawarte w kolekcjach mogą przyjmować wiele różnych form:
  - 3 elementy, 4 elementy
  - Różne klucze, różne wartości

```
{
    {'a' : 1,
     'b' : 2,
     'c' : 3
    }
    {'a' : 7,
     'b' : 8
     'd' : 10
    }
    {'a' : 20,
     'e' : 30,
     'f' : 40,
     'g' : 50
    }
}
```

POPULARNE FORMATY ZAPISU DANYCH

- CSV
- JSON
- XML

STRUKTURA PLIKU JSON

- {"menu":
  - { "id": "plik",
    - "value": "Plik",
    - "popup":
    - { "menuitem": [{"value": "Nowy", "onclick": "NowyDokument()"}, {"value": "Otwórz", "onclick": "OtwórzDokument()"}, {"value": "Zamknij", "onclick": "ZamknijDokument()"}] }
  - }
- }

STRUKTURA PLIKU XML

- **<menu** id="plik" value="Plik"**>**
  - **<popup>**
    - **<menuitem** value="Nowy" onclick="NowyDokument()" **/>**
    - **<menuitem** value="Otwórz" onclick="OtwórzDokument()" **/>**
    - **<menuitem** value="Zamknij" onclick="ZamknijDokument()" **/>**
  - **</popup>**
- **</menu>**

PRAKTYCZNE ASPEKTY NOSQL

SQL VS. NOSQL (MONGODB)

database	database
table	collection
row	document lub BSON document
column	field
index	index
table joins	$lookup, zagnieżdżone dokumenty
primary key	_id
agregacje	pipeline

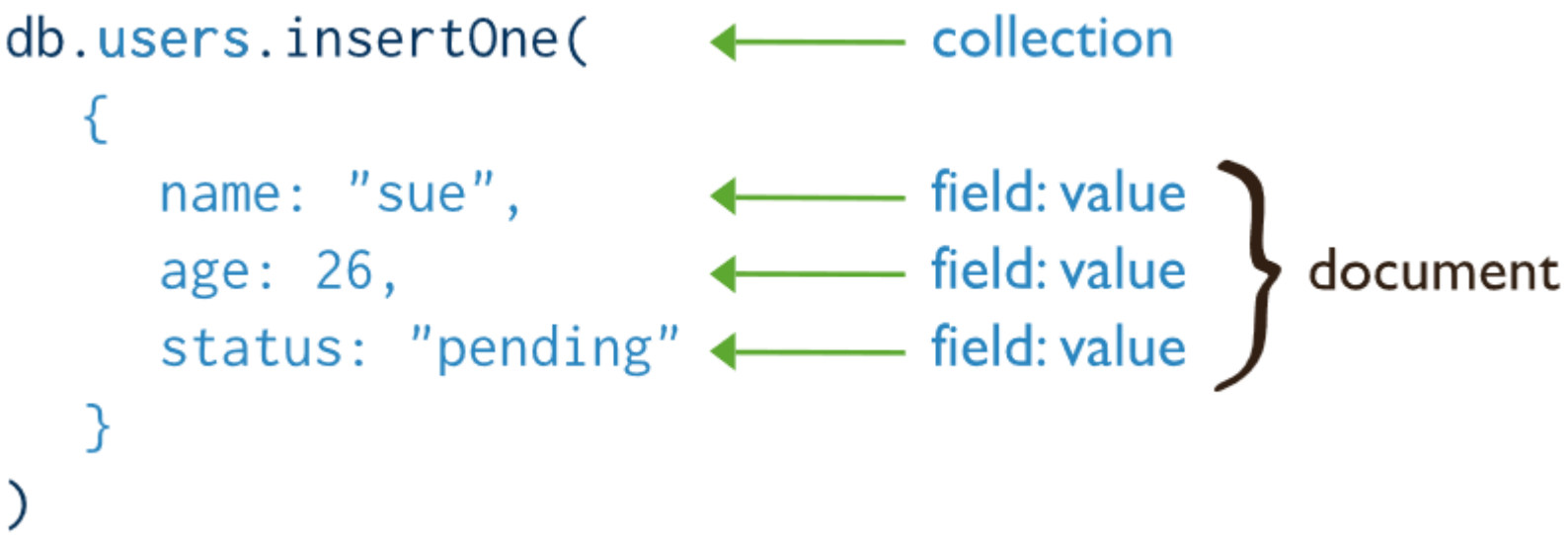
PRZEGLĄDANIE BAZ DANYCH

- > show databases;
  - admin (empty)
  - local 0.078GB
- > use admin
  - switched to db admin
- > use test
  - switched to db test

DOKUMENTY – WSTAWIANIE

- db.collection.insertOne()
- db.collection.insertMany()
- db.collection.find()

WSTAWIANIE DANYCH (DOKUMENTACJA MONGODB)



AKTUALIZACJA DANYCH

- db.collection.update(
- <query>, (warunek)
- <update>, (aktualizacja)
- {
- upsert: <boolean>, (tworzenie nowego dokumentu jeżeli nie istnieje)
- multi: <boolean>, (aktualizuje wiele dokumentów)
- writeConcern: <document>, (alternatywne nadpisanie)
- collation: <document>, (ustawienia kodowania znaków itp.)
- arrayFilters: [<filterdocument1>, ...] (ustawienia filtrowania tablic)
- }
-)

- db.kolekcja.update(
- { name: "A" },
- {
- name: "B",
- rating: 1,
- score: 1
- },
- { upsert: true }
-)

- db.kolekcja.update({ "_id.name": "AiB", "_id.uid": 0 },
- { "categories": ["poet", "playwright"] },
- { upsert: true })

ZAPISYWANIE KOLEKCJI

- db.collection.update()
- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.findAndModify()
- db.collection.findOneAndUpdate()
- db.collection.findOneAndReplace()
- db.collection.save()
- db.collection.bulkWrite()

WYŚWIETLANIE DANYCH

- db.kolekcja.find({})
- SELECT * FROM kolekcja

- db.kolekcja.find({ status: "D" })
- SELECT * FROM kolekcja WHERE status = "D"

- db.kolekcja.find({ status: { $in: ["A", "D"] } })
- SELECT * FROM kolekcja WHERE status in ("A", "D")

- db.kolekcja.find({ status: "A", qty: { $lt: 30 } })
- SELECT * FROM kolekcja WHERE status = "A" AND qty < 30

- db.kolekcja.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })
- SELECT * FROM kolekcja WHERE status = "A" OR qty < 30

- db.kolekcja.find({
- status: "A",
- $or: [{ qty: { $lt: 30 } }, { item: /^p/ }]
- })
- SELECT * FROM kolekcja WHERE status = "A" AND (qty < 30 OR item LIKE "p%")

AKTUALIZACJA KOLEKCJI

- db.collection.updateOne(<filter>, <update>, <options>)
- db.collection.updateMany(<filter>, <update>, <options>)
- db.collection.replaceOne(<filter>, <replacement>, <options>)

AKTUALIZACJA KOLEKCJI – POJEDYNCZA, WIELOKROTNA

```
db.kolekcja.updateOne(
   { item: "paper" },
   {
     $set: { "size.uom": "cm", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

```
db.kolekcja.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

ZASTĘPOWANIE DOKUMENTÓW

- db.inventory.replaceOne(
- { item: "paper" },
- { item: "paper", instock: [{ warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 }] }
-)

USUWANIE DOKUMENTÓW

- db.collection.deleteMany()
- db.collection.deleteOne()

USUWANIE DOKUMENTÓW

- db.kolekcja.deleteMany({})
- db.kolekcja.deleteMany({ status : "A" })
- db.kolekcja.deleteOne({ status: "D" })

SQL VS. MONGODB

```
CREATE TABLE people (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

```
db.people.insertOne( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

```
db.createCollection("people")
```

SQL VS. MONGODB

- ALTER TABLE people
- ADD join_date DATETIME

- db.people.updateMany(
- { },
- { $set: { join_date: new Date() } }
-)

SQL VS. MONGODB

- ALTER TABLE people
- DROP COLUMN join_date

- db.people.updateMany(
- { },
- { $unset: { "join_date": "" } }
-)

SQL VS. MONGODB

- CREATE INDEX idx_user_id_asc
- ON people(user_id)

- db.people.createIndex({ user_id: 1 })

SQL VS. MONGODB

- CREATE INDEX
- idx_user_id_asc_age_desc
- ON people(user_id, age DESC)

- db.people.createIndex(
  { user_id: 1, age: -1 })

SQL VS. MONGODB

- DROP TABLE people
- db.people.drop()

PRAKTYCZNE ZAGADNIENIA BAZ NOSQL

PRACA W NOSQL – WSTAWIANIE DOKUMENTU

```
mongo
use test
> db.kolekcja.insert(
... {
... "adres":{
... "ulica":"Piaskowa 9",
... "miasto":"Sopot",
... "kod":"81-864" },
... "pokoje":[21,33,35]
... }
... )
```

```
db.kolekcja.insert(
{
"adres":{
"ulica":"Armii Krajowej 101",
"miasto":"Sopot"
},
"pokoje":[213,215],
"liczbainstytutow":"1"
}
)
```

PRZEGLĄDANIE DANYCH

- db.kolekcja.find()
- db.kolekcja.find({ "adres.miasto"="Sopot" })
- db.kolekcja.find({"pokoje":213 })
- db.kolekcja.find({"pokoje": { $gt:100 }})
- db.kolekcja.find({"pokoje": { $lt:100 }})
- db.kolekcja.remove ({ "adres.miasto": "Sopot" })

ĆWICZENIE.

Imię	Nazwisko	Wiek	Płeć
Anna	Nowak		F
Jan	Kowalski	19	M
Ewa	Nowak	21	F

- Zaproponować strukturę i dodać wartości.

PYTANIE

- Odpowiednikiem kolumny z bazy relacyjnej w bazie NoSQL jest:
- baza (database)
- pole (field)
- kolekcja (collection)