Wyższa Szkoła Ekonomii i Informatyki w Krakowie

MongoDB

Ryszard Brzegowy

Kraków, 2024

SQL vs NoSQL

SQL

- Tabele
- Wiersze
- Kolumny

NoSQL (Dokument)

- Kolekcje
- Dokumenty

WSEI

SQL vs NoSQL

Przykładowe mechanizmy

SQL

- MySQL/MariaDB
- SQLServer
- Oracle
- PostgreSQL
- SQLite
- Cloud*

NoSQL

- MongoDB
- Google Firestore
- Azure Cosmos DB
- Redis (klucz-wartość)
- MemCache
- Cassandra
- SimpleDb
- HBase
- Cloud*

Cloud - m.in. AWS, Azure, Google Cloud - produkty własne i hostowane

WSEI

NoSQL

Najczęstsze sposoby przechowywania danych

- Klucz - wartość
- Document store
- Data structures server
- Tuple store
- Bazy obiektowe
- Szerokolumnowe
- Grafy

Część baz NoSQL pracuje jako mix kilku technik.

WSEI

NoSQL

Zalety

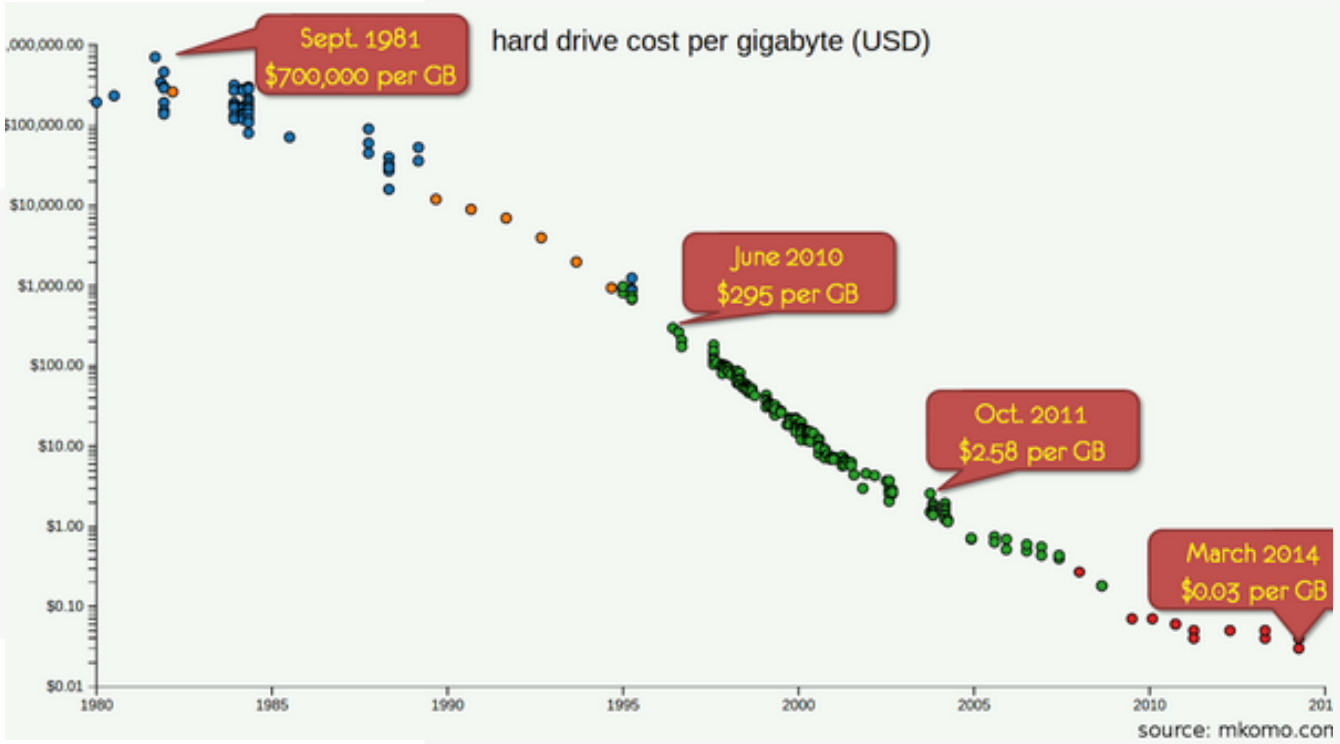
- elastyczność - brak wymaganej "sztywnej" struktury danych
- łatwość skalowania w poziomie (sharding danych)
- mniejsze ryzyko utraty całości danych/czasowego braku dostępu w wyniku awarii (sharding)
- wydajność (w szczególności: odczyt)
- łatwość przechowywania dużych zbiorów danych (skalowanie poziome)

WSEI

NoSQL

Wady

- trudniejszy dostęp przy złożonych modelach danych
- potencjalny problem z ilością miejsca zajmowanego przez dane (but who cares today?)



*sharding - podział zbiorów danych na mniejsze porcje

WSEI

MongoDB

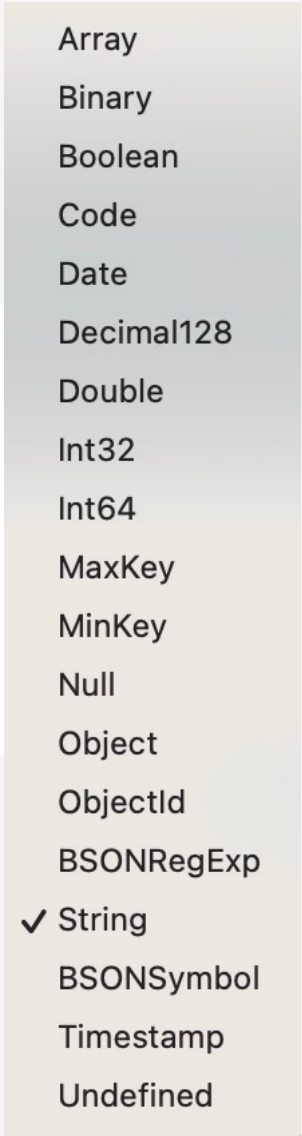
- Baza NoSQL oparta na kolekcjach i dokumentach
- Dokumenty korzystają z formatu BSON
- Dostęp do danych następuje przez API
- Baza danych
  - własny serwer - Community Edition lub Enterprise Edition
  - cloud - MongoDB Atlas, Enterprise Advanced
- Narzędzia
  - MongoDB Shell (CLI)
  - MongoDB Compass (click, click)

WSEI

MongoDB

Dokumenty

- Pojemnik na dane
- Dostęp do dokumentu adresowany jest przez unikalny (dla bazy) klucz
- Dostęp jest realizowany przez zdefiniowane przez bazę API
- Wbudowane typy danych w dokumencie: patrz obok:)



WSEI

MongoDB

Kolekcje

- Grupy powiązanych logicznie dokumentów (np. notatki, tagi)
- Kolekcję można luźno (bardzo luźno;)) potraktować jako odpowiednik tabeli
- Do kolekcji można przypisać Scheme pilnujący i definiujący model danych w dokumentach
- Kolekcje obsługują indeksy (jedno i wiele pól dokumentu)

WSEI

NoSQL

Projekt:

- złożone obiekty - zagnieżdżanie danych w dokumentach (np. tagi w notatce)
- zwyczajowo - brak normalizacji
- sharding danych - podział zbiorów danych na mniejsze porcje
- projekt specyficzny do zastosowania - np. bazy oparte o grafy
- najczęstsze formaty danych: JSON (i odmiany. np. BSON), XML, YAML, prymityw (klucz-wartość)

WSEI

MongoDB HowTo

- usługa MongoDB Atlas
- Kroki skorzystania:
  - konto
  - klaster
  - użytkownik
  - baza (jest testowa wstępnie wypełniona)
  - nadanie praw użytkownikowi do bazy

WSEI

MongoDB - implementacja

- Node - natywny klient - biblioteka mongodb
- Web app - usługa Atlas App Services
  - Aplikacja po stronie serwera MongoDB
  - Klient - biblioteka realm-web
  - Klient może:
    - wykonać funkcje zdefiniowane w aplikacji serwerowej
  - skorzystać z wbudowanych gotowych funkcji do pracy z bazą - Realm Query Api

WSEI

MongoDB - ORM, ODM

- Zalety
  - Typowanie danych
  - Walidacja
  - Cache
  - Obsługa zdarzeń (w czasie rzeczywistym)
  - Zarządzanie połączeniami z serwerem
  - Wydzielenie abstrakcji komunikacji z bazą (umożliwia szybką migrację na inny rodzaj bazy lub sposób połączenia z bazą)
- Najpopularniejsze w JS/TS
  - Mongoose
  - Prisma

WSEI

Mongoose

- Biblioteka ODM (Object-Document Mapping)
- Bazuje na schematach - opisują strukturę danych (typy, wymagalność)
- Oraz modelach - obiektach scalających strukturę i sposób pracy z dokumentami.
- Model to API do kolekcji po stronie kodu

WSEI

Mongoose

Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test');

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

WSEI

Mongoose

- Instalacja - pakiet npm - mongoose
- Połączenie z bazą:
  - .connect(connectionURI): Promise
- Tworzenie Schema:
  - const schema = new mongoose.Schema(schemaObj, opts)
- Tworzenie Model
  - const model = mongoose.Model(modelName, schema)

WSEI

Mongoose

- SchemaObject dla notatki:

```
{
    title: {type: String, required: true},
    content: {type: String, required: true},
    public: Boolean
    […]
}
```

- options: {timestamps: true}

WSEI

Mongoose

- ModelObject dla notatki:

  ```
  const Note = new mongoose.model('Note', noteSchema)
  ```

- Tworzenie obiektów
  const newNote = new Note(noteObject)
- Pobieranie kolekcji
  Note.find()
- Pobieranie dokumentu
  Note.findOne(), Note.findById()
- Usuwanie dokumentu
  Note.delete()

WSEI

Mongoose query helpers

```
// With a JSON doc
Person.
  find({
    occupation: /host/,
    'name.last': 'Ghost',
    age: { $gt: 17, $lt: 66 },
    likes: { $in: ['vaporizing', 'talking'] }
  }).
  limit(10).
  sort({ occupation: -1 }).
  select({ name: 1, occupation: 1 }).
  exec(callback);
```

```
// Using query builder
Person.
  find({ occupation: /host/ }).
  where('name.last').equals('Ghost').
  where('age').gt(17).lt(66).
  where('likes').in(['vaporizing', 'talking']).
  limit(10).
  sort('-occupation').
  select('name occupation').
  exec(callback);
```

WSEI

Wyższa Szkoła Ekonomii
i Informatyki w Krakowie