Node.JS ile Web Programlama
Rapor - 11

Mahire Zühal Özdemir

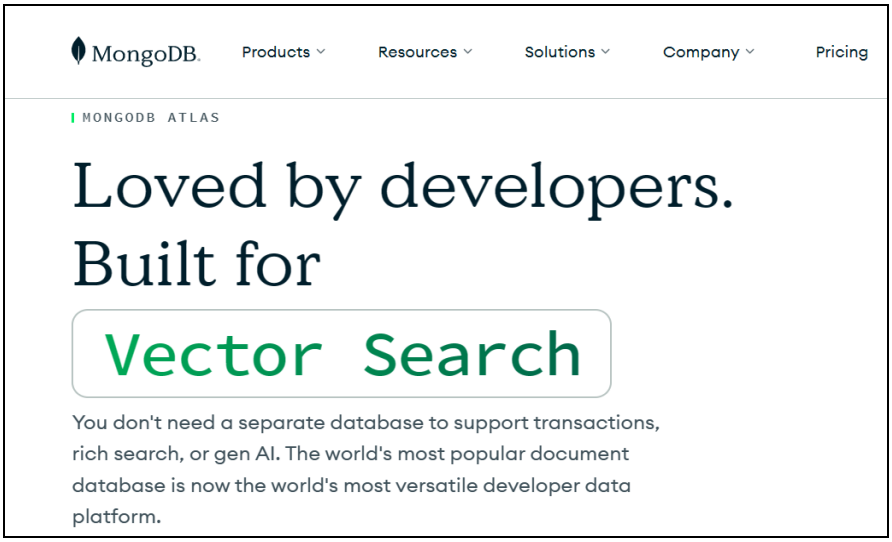
İÇİNDEKİLER

ŞEKİLLER DİZİN

MONGODB

MongoDB, NoSQL (Not Only SQL) veritabanı olarak sınıflandırılan bir veritabanı sistemidir. İsmi "humongous" (çok büyük) ve "database" kelimelerinin birleşiminden gelir. İlk olarak 2009 yılında geliştirilmeye başlanmıştır.

- Geleneksel ilişkisel veritabanı sistemlerinden farklı olarak, MongoDB belge-tabanlı bir veritabanıdır. Veriler JSON benzeri bir formatta belgelendirilir. Her bir belge, birbirinden farklı alanlar ve veri tipleri içerebilir. Bu, veri modelinin daha esnek olmasını ve dinamik veri yapısının daha iyi yönetilmesini sağlar.



Şekil 1: MongoDb

SQL ve NoSQL Veritabanı Farkı

Veri Yapısı:

- **SQL:** İlişkisel veritabanlarında, veri tabloları, sütunlar ve satırlar şeklinde yapılandırılır. Veriler düzenli bir yapıda saklanır.
- **NoSQL:** NoSQL veritabanları, belge-tabanlı, sütun-tabanlı, anahtar-değer tabanlı veya grafik-tabanlı gibi farklı veri modellerini destekler. Veri yapısı, veritabanı türüne bağlı olarak değişebilir.

Veri Modeli:

- **SQL:** İlişkisel veritabanları, önceden tanımlanmış bir şema kullanır. Veri yapısı ve ilişkiler, şema tarafından belirlenir.
- **NoSQL:** NoSQL veritabanları, şemasız veya esnek bir şema modeline sahip olabilir. Veri modeli, uygulama tarafından dinamik olarak belirlenebilir.

Ölçeklenebilirlik:

- **SQL:** Geleneksel ilişkisel veritabanları genellikle dikey ölçeklenir, yani tek bir sunucunun gücü artırılarak daha fazla işlemci veya bellek eklenir.
- **NoSQL:** NoSQL veritabanları genellikle yatay ölçeklenebilirlik sunar, yani daha fazla sunucu ekleyerek sistemin kapasitesi genişletilebilir.

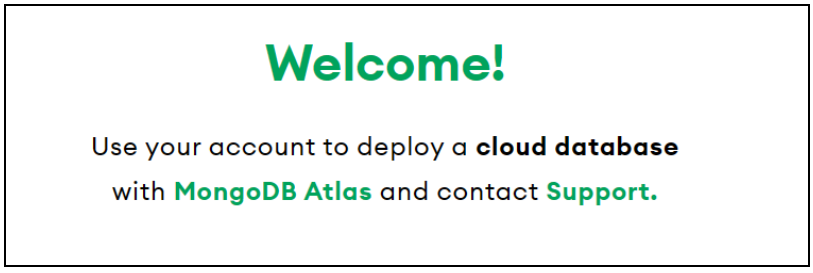
ACID Uyumluluğu:

- **SQL:** İlişkisel veritabanları genellikle ACID (Atomicity, Consistency, Isolation, Durability) uyumlu işlemleri destekler. Bu, veritabanı işlemlerinin güvenilir ve tutarlı olmasını sağlar.
- **NoSQL:** NoSQL veritabanları, ACID uyumlu olmayan bazı işlemleri destekleyebilir veya temel ACID özelliklerini esnetebilir. Bu, bazı uygulamalarda daha esnek bir veritabanı davranışı sağlar.

MongoDB Kurulum

MongoDb resmi sitesinden kayıt işlemi gerçekleştirilir.

Bulut depoda mondogb kullanmak için hesap açılır.



Şekil 2: MongoDB Kurulum

MongoDB Kullanarak Task Manager Uygulaması

Hesap açma işleminden sonra mongodb kullanarak yeni bir uygulama geliştiriyoruz. Task-Manager adında bir klasör oluşturup aşağıdaki kodları çalıştırıyoruz.

```
npm init -y

npm install mongodb
```

Mongo.js dosyası oluşturup aşağıdaki kodları kullanarak mongodb'yi import ediyoruz.

```
const mongodb = require('mongodb')
const MongoClient = mongodb.MongoClient
```

Şekil 3: MongoDB import işlemi

MongoDB web sitesi üzerinden deployment oluşturarak bize verilen URL'i kopyalıyoruz.

Veri Tabanı Bağlantısı

Verilen URL'i ve aşağıdaki kodu kullanarak veritabanı bağlantısı yapılır.

```
const uri = "mongodb+
const client = new MongoClient(uri);
const database_name = 'task-manager'

async function run() {
  try {
    const database = client.db(database_name)
    console.log('Veritabanına bağlandı.');
  }catch{
    console.log('Veritabanına bağlanamadı.');
  }finally{
    await client.close();
  }
}
run().catch(console.dir);
```

Şekil 4: Veritabanı Bağlantısı

```
PS C:\Users\zuhal\Desktop                    node .\mongodb.js
Veritabanına bağlandı.
```

Şekil 5: Veritabanı Bağlantısı

Veritabanına Veri Ekleme

Veritabanına veri eklemek için js dosyamızı kullanabiliriz. Koleksiyon adı ve veritabanını belirledikten sonra veri eklemek için aşağıdaki kod kullanılır.

```
const uri = "mongodb+srv://                                                    "
const client = new MongoClient(uri);
const database_name = 'task-manager'
const collectionName = 'users'; // Koleksiyon adı tablo adı

async function run() {
  try {
    const database = client.db(database_name);
    console.log('Veritabanına bağlandı.');

    const users = database.collection(collectionName);
    const user = {
      name:'Ahmet',
      lastname:'Yılmaz',
      age:32
    };
    const result = await users.insertOne(user);
    console.log(user, 'user eklendi.');

  } catch{
    console.log('Veritabanına bağlanamadı.');
  }finally{
    await client.close();
  }
}
run().catch(console.dir);
```

Şekil 6: Koleksiyona Veri Ekleme

Kodu çalıştırdığımız zaman çıktı aşağıdaki gibi olur.

```
PS C:\Users\zuhal\Desktop                    node .\mongodb.js
Veritabanına bağlandı.
{
  name: 'Ahmet',
  lastname: 'Yılmaz',
  age: 32,
  _id: new ObjectId('663b32c59d3ca7d5ac969176')
} user eklendi.
```

Şekil 7: User Ekleme

MongoDB üzerinden koleksiyonu görüntülediğimizde verinin eklenmiş olduğunu görürüz.

Şekil 8: MongoDb Çıktısı

Veritabanında Veri Arama

Veritabanında belirli bir değere göre veri aramak için *findOne* komutu kullanılır. Bu komutla birlite veritabanında ilgili özelliğin bulunup bulunmadığı kontrol edilir.

```
async function run() {
    try {
      const database = client.db(database_name);
      database.collection(collectionName).findOne({name:'Ahmet'}, function(err, result) {
        if (err) throw err;
        console.log(result.name);
        database.close();
      })
    }catch{
      console.log('Veritabanına bağlanamadı.');
    }finally{
      await client.close();
    }
  }
  run().catch(console.dir);
```

Şekil 9:Veri Arama