Veritabanı Yönetim Sistemleri
1906003022015

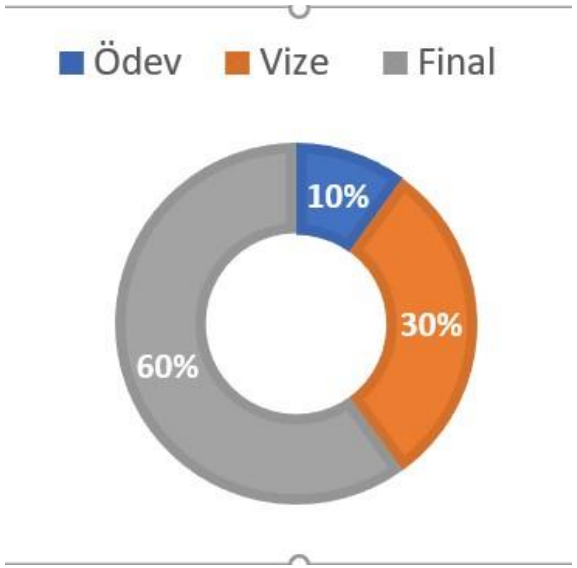
Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

Giriş

Ders Günü ve Saati:

Pazartesi: 09:15-12:45

- Devam zorunluluğu %70
- Uygulamalar MS SQL ve MongoDB üzerinde gerçekleştirilecektir.



HAFTA	KONULAR
Hafta 1	VT ve VTYS'ne giriş
Hafta 2	ER Veri Modeli
Hafta 3	İlişkisel Modeller, İlişkisel model tasarımı
Hafta 4	İlişkisel Cebir ve Hesaplamalar
Hafta 5	İlişkisel Sorgular, SQL giriş
Hafta 6	SQL ile veri tabanı programlama
Hafta 7	SQL-Kısıtlar:Veri-tipi,birincil-anahtar,ikinci-anahtar,
Hafta 8	Vize
Hafta 9	İlişkisel Veri Tabanı Tasarımı ve Normalizasyon
Hafta 10	yarı-yapısal veri modelleri, XML
Hafta 11	JSON
Hafta 12	İlişkisel olmayan DB, NoSQL
Hafta 13	NoSQL
Hafta 14	DBMS -Eşzamanlılık (Concurrency) Kontrolü

İÇERİK

MongoDB

MongoDB'yi oluşturmanın birincil amacı:

- Ölçeklenebilirlik
- Verim
- Yüksek kullanılabilirlik
- Tek sunucu dağıtımlarından büyük, karmaşık çok siteli mimarilere ölçeklendirme.
- MongoDB'nin kilit noktaları
- Daha Hızlı Geliştirme
- Dağıtımı Daha Kolay
- Daha Büyük Ölçekleme

MongoDB

MongoDB , yüksek performans, yüksek kullanılabilirlik ve otomatik ölçekleme sağlayan açık kaynaklı bir belge veritabanıdır.

Basit bir deyişle, şunu söyleyebilirsiniz - Mongo DB, belge odaklı bir veritabanıdır.

MongoDB'deki bir kayıt, alan ve değer çiftlerinden oluşan bir veri yapısı olan bir belgedir. MongoDB belgeleri JSON nesnelerine benzer. Alanların değerleri diğer belgeleri, dizileri ve belge dizilerini içerebilir.

MongoDB

Koleksiyonlar/Görünümler/Talep Üzerine Gerçekleştirilmiş Görünümler

MongoDB, belgeleri koleksiyonlarda saklar . Koleksiyonlar, ilişkisel veritabanlarındaki tablolara benzer.

Koleksiyonlara ek olarak, MongoDB şunları destekler:

- Salt Okunur Görünümler (MongoDB 3.4'ten itibaren)
- İsteğe Bağlı Gerçekleştirilmiş Görünümler (MongoDB 4.2'den itibaren).

Ana Özellikler

Yüksek performans

MongoDB, yüksek performanslı veri kalıcılığı sağlar. Özellikle,

- Gömülü veri modelleri desteği, veritabanı sistemindeki G/Ç etkinliğini azaltır.
- Dizinler daha hızlı sorguları destekler ve katıştırılmış belgelerden ve dizilerden anahtarlar içerebilir.

Sorgu API'sı

MongoDB Sorgu API'si, okuma ve yazma işlemlerini (CRUD) ve ayrıca şunları destekler:

- Veri toplama
- Metin Arama ve Jeo-uzaysal Sorgular .

Ana Özellikler

Yüksek kullanılabilirlik

MongoDB'nin replika seti adı verilen replikasyon tesisi şunları sağlar:

- otomatik yük devretme
- veri yedekleme.

Bir çoğaltma kümesi , aynı veri kümesini koruyan, artıklık sağlayan ve veri kullanılabilirliğini artıran bir grup MongoDB sunucusudur.

Ana Özellikler

Yatay Ölçeklenebilirlik

MongoDB, temel işlevselliğinin bir parçası olarak yatay ölçeklenebilirlik sağlar :

- Parçalama , verileri bir makine kümesi arasında dağıtır.
- 3.4'ten itibaren MongoDB, parça anahtarına dayalı veri bölgeleri oluşturmayı destekler . Dengeli bir kümede, MongoDB bir bölgenin kapsadığı okuma ve yazma işlemlerini yalnızca bölge içindeki parçalara yönlendirir. Daha fazla bilgi için Bölgeler kılavuz sayfasına bakın.

Ana Özellikler

Çoklu Depolama Motorları Desteği

MongoDB, birden çok depolama motorunu destekler :

- WiredTiger Storage Engine (Hareket Halinde Şifreleme desteği dahil)
- Bellek İçi Depolama Motoru .

Ayrıca MongoDB, üçüncü tarafların MongoDB için depolama motorları geliştirmesine olanak tanıyan takılabilir depolama motoru API'sı sağlar.

SQL'den MongoDB'ye Eşleme Tablosu

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index

Documents

MongoDB stores data records as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON. For the BSON spec, see bsonspec.org

```
{
  _id: ObjectId("509a8fb2f3f4948bd2f983a0"),
  user_id: "abc123",
  age: 55,
  status: 'A'
}
```

Documents

Bir alanın değeri, diğer belgeler, diziler ve belge dizileri dahil olmak üzere BSON veri türlerinden herhangi biri olabilir. Örneğin, aşağıdaki belge, değişen türlerde değerler içerir:

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

Documents

Type	Number	Alias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
Undefined	6	"undefined"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"

Type	Number	Alias
Regular Expression	11	"regex"
DBPointer	12	"dbPointer"
JavaScript	13	"javascript"
Symbol	14	"symbol"
JavaScript code with scope	15	"javascriptWithScope"
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"

SQL'den MongoDB'ye Eşleme Tablosu

```
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"
} )
```

Veritabanları ve Koleksiyonlar

MongoDB, veri kayıtlarını koleksiyonlarda bir araya getirilen belgeler (özellikle BSON belgeleri) olarak saklar . Bir veritabanı , bir veya daha fazla belge koleksiyonunu depolar.

MongoDB'de veritabanları bir veya daha fazla belge koleksiyonunu tutar.

```
use myDB
```

Bir veritabanı yoksa, MongoDB, o veritabanı için verileri ilk depoladığınızda veritabanını oluşturur. Bu nedenle, var olmayan bir veritabanına geçebilir ve aşağıdaki işlemi 'de gerçekleştirebilirsiniz

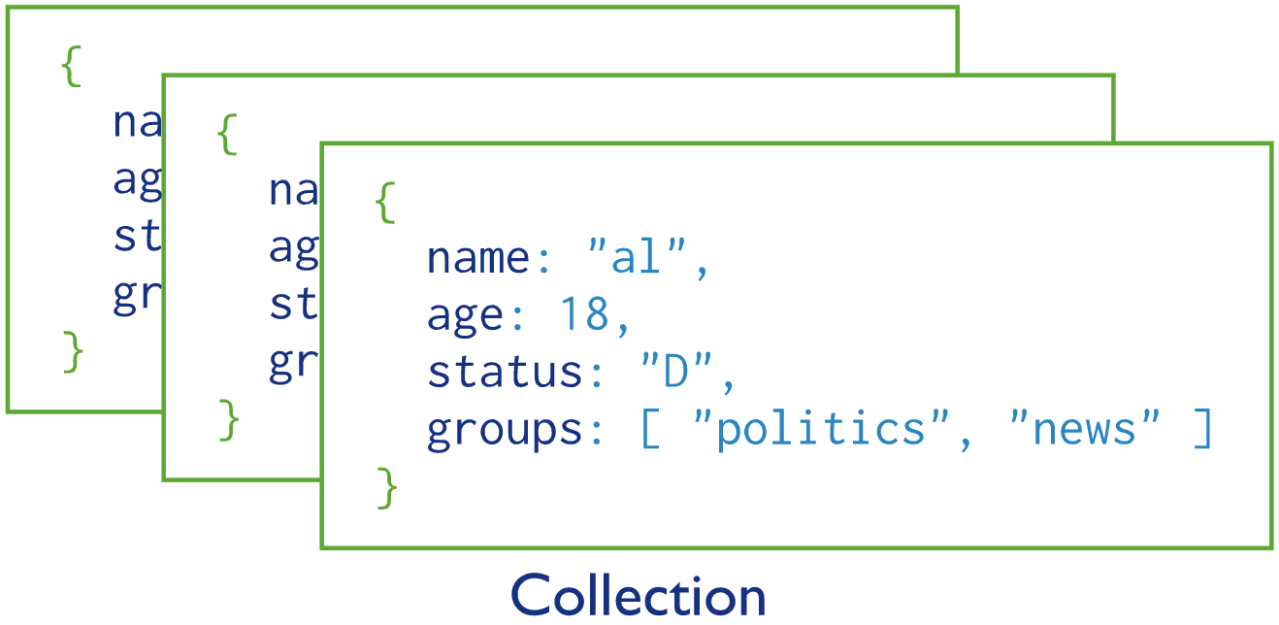
```
use myNewDB

db.myNewCollection1.insertOne( { x: 1 } )
```

Veritabanları ve Koleksiyonlar

Koleksiyonlar

MongoDB, belgeleri koleksiyonlarda saklar. Koleksiyonlar, ilişkisel veritabanlarındaki tablolara benzer.



Veritabanları ve Koleksiyonlar

Koleksiyonlar

Bir koleksiyon yoksa, MongoDB, o koleksiyon için verileri ilk depoladığınızda koleksiyonu oluşturur.

```
db.myNewCollection2.insertOne( { x: 1 } )
db.myNewCollection3.createIndex( { y: 1 } )
```

Hem ve insertOne()hem de createIndex()işlemler, zaten mevcut değilse, ilgili koleksiyonlarını oluşturur. Koleksiyon adının MongoDB Adlandırma Kısıtlamalarına uygun olduğundan emin olun .

Veritabanları ve Koleksiyonlar

Koleksiyonlar

Açık Oluşturma

MongoDB, db.createCollection() maksimum boyutu veya belge doğrulama kurallarını ayarlamak gibi çeşitli seçeneklerle açıkça bir koleksiyon oluşturma yöntemini sağlar. Bu seçenekleri belirtmiyorsanız, koleksiyonlar için verileri ilk depoladığınızda MongoDB yeni koleksiyonlar oluşturduğundan koleksiyonu açıkça oluşturmanız gerekmez.

Veritabanları ve Koleksiyonlar

Koleksiyonlar

Son 24 saatin hava durumu verilerini yakalayan bir zaman serisi koleksiyonu oluşturmak için şu komutu verin:

Varsayılan olarak, bir koleksiyon, belgelerinin aynı şemaya sahip olmasını gerektirmez; yani, tek bir koleksiyondaki belgelerin aynı alan kümesine sahip olması gerekmez ve bir alanın veri türü, bir koleksiyon içindeki belgeler arasında farklılık gösterebilir.

```
db.createCollection(
    "weather24h",
    {
       timeseries: {
          timeField: "timestamp",
          metaField: "data",
          granularity: "hours"
       },
       expireAfterSeconds: 86400
    }
)
```

Veritabanları ve Koleksiyonlar

Görüntüleme

Bir MongoDB görünümü, içeriği diğer koleksiyonlarda veya görünümlerde bir bileşik (agrigation) ardışık düzeni tarafından tanımlanan sorgulanabilir bir nesnedir. MongoDB, görüntüleme içeriğini diske devam ettirmez. Bir görünümün içeriği, bir istemci görünümü sorguladığında isteğe bağlı olarak hesaplanır . MongoDB, istemcilerin görünümü sorgulama iznine sahip olmasını gerektirebilir. MongoDB, görünümlere karşı yazma işlemlerini desteklemez.

Veritabanları ve Koleksiyonlar

Görüntüleme

Görünüm Oluşturma

```
db.createCollection(
  "<viewName>",
  {
    "viewOn" : "<source>",
    "pipeline" : [<pipeline>],
    "collation" : { <collation> }
  }
)
```

```
db.createView(
  "<viewName>",
  "<source>",
  [<pipeline>],
  {
    "collation" : { <collation> }
  }
)
```

Veritabanları ve Koleksiyonlar

Desteklenen İşlemler

Komutlar	Yöntemler
create	db.createCollection() db.createView()
collMod	
	db.getCollection() db.getCollectionInfos() db.getCollectionNames()
find distinct count	db.collection.aggregate() db.collection.find() db.collection.findOne() db.collection.countDocuments() db.collection.estimatedDocumentCount() db.collection.count() db.collection.distinct()

MongoDB CRUD İşlemleri

CRUD işlemleri belgeleri *oluşturur* , *okur* , *günceller* ve *siler*.

Create Operations

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

```
db.users.insertOne(          <-- collection
   {
      name: "sue",           <-- field: value
      age: 26,               <-- field: value      } document
      status: "pending"      <-- field: value
   }
)
```

MongoDB CRUD İşlemleri

Create Operations

insertOne()

The insertOne() method has the following syntax:

```
db.collection.insertOne(
   <document>,
   {
      writeConcern: <document>
   }
)
```

Parametre	Tip	Tanım
document	belge	Koleksiyona eklenecek bir belge.
writeConcern	belge	İsteğe bağlı. Yazma endişesini ifade eden bir belge . Varsayılan yazma endişesini kullanmayı atlayın. Bir işlemde çalıştırılıyorsa, işlem için yazma endişesini açıkça ayarlamayın. İşlemlerle ilgili yazma endişesini kullanmak için, bkz . İşlemler ve Yazma Endişesi .

MongoDB CRUD İşlemleri

Create Operations

insertMany()

Yöntem insertMany() aşağıdaki sözdizimine sahiptir:

```
db.collection.insertMany(
   [ <document 1> , <document 2>, ... ],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```

Parametre	Tip	Tanım
document	belge	Koleksiyona eklenecek bir dizi belge.
writeConcern	belge	İsteğe bağlı. Yazma endişesini ifade eden bir belge . Varsayılan yazma endişesini kullanmayı atlayın. Bir işlemde çalıştırılıyorsa, işlem için yazma endişesini açıkça ayarlamayın. İşlemlerle ilgili yazma endişesini kullanmak için, bkz . İşlemler ve Yazma Endişesi .
ordered	boole	İsteğe bağlı. mongodÖrneğin sıralı mı yoksa sırasız bir ekleme mi gerçekleştireceğini belirten bir boole . Varsayılan olarak true.

MongoDB CRUD İşlemleri

Create Operations

db.createCollection()

Yöntem insertMany() aşağıdaki sözdizimine sahiptir:

```
db.createCollection(name, options)
```

Parametre	Tip	Tanım
name	sicim	Oluşturulacak koleksiyonun adı. Bkz . Adlandırma Kısıtlamaları .
options	belge	İsteğe bağlı. Sınırlı bir koleksiyon oluşturmak, yeni bir koleksiyonda önceden alan tahsis etmek veya bir görünüm oluşturmak için yapılandırma seçenekleri.

```
1.  >use dataflair
2.  switched to db dataflair
3.  >db.createCollection("mongodb",
4.  { capped:true, size:1000000, max:2})
5.  { "ok" : 1 }
```

MongoDB CRUD İşlemleri

Read Operations

db.collection.find()

Okuma işlemleri bir koleksiyondan belgeleri alır; yani belgeler için bir koleksiyon sorgulayın. MongoDB, bir koleksiyondaki belgeleri okumak için aşağıdaki yöntemleri sağlar:

```
db.collection.find(query, projection)
```

Parametre	Tip	Tanım
query	belge	İsteğe bağlı. Sorgu operatörlerini kullanarak seçim filtresini belirtir . Bir koleksiyondaki tüm belgeleri döndürmek için bu parametreyi atlayın veya boş bir belge ({}) iletin.
projeksiyon	belge	İsteğe bağlı. Sorgu filtresiyle eşleşen belgelerde döndürülecek alanları belirtir. Eşleşen belgelerdeki tüm alanları döndürmek için bu parametreyi atlayın. Ayrıntılar için, bkz . Projeksiyon .

```
db.users.find(                          ← collection
   { age: { $gt: 18 } },                ← query criteria
   { name: 1, address: 1 }              ← projection
).limit(5)                              ← cursor modifier
```

MongoDB CRUD İşlemleri

Query Documents

```
{ <field1>: <value1>, ... }
```

```
{ <field1>: { <operator1>: <value1> }, ... }
```

```
{ status: { $in: [ "A", "D" ] } }
```

```
{ status: "A", $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ] }
```

```
{ size: { h: 14, w: 21, uom: "cm" } }
```

MongoDB CRUD İşlemleri

Query Documents

Query an Array

```
db.inventory.find( { tags: ["red", "blank"] } )
```

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```

```
db.inventory.find( { dim_cm: { $gt: 25 } } )
```

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

```
db.inventory.find( { "tags": { $size: 3 } } )
```

MongoDB CRUD İşlemleri

Update Documents

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

```
db.collection.updateOne(
   <filter>,
   <update>,
   {
     upsert: <boolean>,
     writeConcern: <document>,
     collation: <document>,
     arrayFilters: [ <filterdocument1>, ... ],
     hint:  <document|string>        // Available starting in MongoDB 4.2.1
   }
)
```

MongoDB CRUD İşlemleri

Update Documents

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

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

MongoDB CRUD İşlemleri

Delete Documents

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

```
db.collection.deleteOne(
   <filter>,
   {
      writeConcern: <document>,
      collation: <document>,
      hint: <document|string>
   }
)
```

Parameter	Type	Description
filter	document	Specifies deletion criteria using query operators. Specify an empty document { } to delete the first document returned in the collection.
writeConcern	document	Optional. A document expressing the write concern. Omit to use the default write concern. Do not explicitly set the write concern for the operation if run in a transaction. To use write concern with transactions, see Transactions and Write Concern.

```
db.inventory.deleteOne( { status: "D" } )
```