

# MONGODB

---

NAZIM EMRE ŞAVKLI

# İÇİNDEKİLER

<b>BÖLÜM 1: VERİTABANI VE NoSQL</b>	<b>1</b>
Veritabanı Nedir?	2
NoSQL Nedir?	6
Veritabanı Tasarımı (Modelleme)	7
SQL Mi? NoSQL Mi?	9
Neler Öğrendik?	10
<b>BÖLÜM 2: MONGODB VERİTABANINA GİRİŞ</b>	<b>13</b>
MongoDB	14
MongoDB Kurulumu	16
MongoDB – İlk Adım	21
İlk Veritabanımızı Oluşturuyoruz	21
Mongod ve Mongo Ne İşe Yarıyor?	22
Mongo Shell	23
MongoDB Temel Kavramlar	24
Database	25
Collection	25
Document	27
MongoDB Veritabanında Veri Modelleme	27
Embedded Document (Gömülü Döküman) Kullanımı	27
Referenced Documents (Normalize Edilmiş İlişkisel Veriler) Kullanımı	28
Şimdi Pratik Yapma Zamanı - I	30
Neler Öğrendik?	30
<b>BÖLÜM 3: İŞTE BAŞLIYORUZ...</b>	<b>33</b>
MongoDB Veritabanları ile Çalışma	34
Yeni Bir Collection Oluşturalım	34

MongoDB Insert Document – Yeni Kayıt Ekleme	35
Yeni Bir Kayıt Ekleme	35
Birden Fazla Kaydı Tek Sorguda Ekleme	42
MongoDB Update Document – Kayıt Güncelleme	44
MongoDB Kayıt Güncelleme ve \$set Operatörü	45
\$unset Operatörü ve upsert Komutu Kullanımı	46
\$inc Operatörü Kullanımı	48
MongoDB Remove Document – Kayıt Silme	50
Şimdi Pratik Yapma Zamanı - II	51
Neler Öğrendik?	52

## **BÖLÜM 4: MONGODB KAYIT SORGULAMA** **53**

Giriş	54
find() Fonksiyonu ile Sorgulama	54
Tüm Kayıtları Listeleme	55
pretty() Fonksiyonu Kullanımı	55
Kayıtları Filtreleme (Filtering)	56
Karşılaştırma Operatörleri ile Veri Filtreleme	57
Veri Sıralama - sort() Fonksiyonu	59
Mantıksal Operatörler ile Veri Sorgulama	63
\$exists Operatörü Kullanımı	66
\$in ve \$nin Operatörü Kullanımı	67
limit() Fonksiyonu Kullanımı	69
skip() Fonksiyonu Kullanımı	71
\$where Operatörü ile Sorgu içi JavaScript Çalıştırma	71
\$type Operatörü Kullanımı	74
Şimdi Pratik Yapma Zamanı – III	79
Dizi İçeren Kayıtları Sorgulama	79

\$all Operatörü Kullanımı	82
\$elemMatch Operatörü Kullanımı	83
\$size Operatörü Kullanımı	85
\$push ve \$addToSet Operatörleri ile Update İşlemi	85
Şimdi Pratik Yapma Zamanı – IV	91
Neler Öğrendik?	92

## **BÖLÜM 5: MONGODB AGGREGATION (KÜMELEME) İŞLEMLERİ** **95**

Giriş	96
Aggregation Pipeline Üyelerini Kullanarak Kümeleme İşlemleri	96
aggregate() Fonksiyonu Kullanımı	97
Şimdi Pratik Yapma Zamanı – V	103
count() ve distinct() Fonksiyonlarının Kullanımı	104
map-reduce Fonksiyonları ile Kümeleme İşlemleri	106
Şimdi Pratik Yapma Zamanı – VI	111
Neler Öğrendik?	111

## **BÖLÜM 6: MONGODB INDEX KULLANIMI** **113**

Index Nedir?	114
MongoDB Veritabanında Index Oluşturma	114
Unique Index Nedir? Unique Index Oluşturma	116
Compound Field Index Oluşturma	118
Text Index Oluşturma	118
\$text Operatörü ile Text Search İşlemleri	120
Birden Fazla Kelimeyi Aramak	121
Bir Kelime Grubunu Aramak	122
Arama İfadesinden Bir Kelimeyi Hariç Tutmak	124
Case Sensitive Arama Yapmak	125
Şimdi Pratik Yapma Zamanı – VII	126
Neler Öğrendik?	126

**BÖLÜM 7: MONGODB REGULAR EXPRESSION KULLANIMI 129**

Regular Expression Tanımlama	130
Telefon Numarası Kontrolü	132
Tarih Kontrolü	132
MongoDB’de Regular Expression Kullanımı	134
Neler Öğrendik?	136

**BÖLÜM 8: C# MONGODB DRIVER İLE BAĞLANTI 139**

C# MongoDB Driver Kurulumu	140
C# MongoDB Veritabanı Bağlantısı	143
Yeni Bir Document Ekleme	149
MongoDB Document Güncelleme	152
Document Silme	156
C# MongoDB Driver ile Veri Sorgulama	158
Neler Öğrendik?	176

**BÖLÜM 9: MONGODB JAVA DRIVER İLE BAĞLANTI 179**

Java MongoDB Driver Kurulumu	180
Java MongoDB Veritabanı Bağlantısı	185
Yeni Bir Document Ekleme	188
Document Güncelleme	193
Document Silme	200
Java MongoDB Driver ile Veri Sorgulama	204
Dizin	219

# VERİTABANI VE NoSQL

## BU BÖLÜMDE

Veritabanı Nedir?	2
NoSQL Nedir?	6
Veritabanı Tasarımı (Modelleme)	7
SQL Mi? NoSQL Mi?	9
Neler Öğrendik?	10

Bu bölümde, veritabanı kavramına giriş yaparak başlıyoruz. Ardından pek çoğunuzun merak ettiğini tahmin ettiğim NoSQL kullanırsak SQL'e hayır mı diyoruz. Yoksa farklı bir teknoloji mi kullanıyoruz sorusuna yanıt mı arıyoruz. Genel anlamda "Veritabanı Tasarımı ve Modelleme Teknikleri" üzerinde düşünüp konuştuktan sonra ise SQL ve NoSQL kavramlarının karşılaştırmalı incelemesini yapacağız. Bakalım hangisi daha iyi...

## VERİTABANI NEDİR?

Belki de pek çoğunuz bu kitabı alırken, yalnızca popülerliğinden haberdar olduğunuz MongoDB'yi öğrenmeniz gerektiği düşüncesine kapılarak kitap raflarına yöneldiniz. Evet MongoDB günümüzde pek popüler bir veritabanı ancak bu kitabın yazılış amacı sadece MongoDB veritabanının bizlere sunduğu özellikleri ele almak değil, daha da ötesine geçerek veri ile nasıl baş edilir bunu kavramanızı sağlayabilmektir.

İlk olarak derseniz şu veri kavramını bir aradan çıkaralım. Veri nedir? Bir parça da standart tanımın dışına çıkarak söyleyebiliriz ki; herhangi bir kuruluşun, işletmenin, hatta ve hatta bir tek kişinin bile yaptığı her işlemde, attığı her adımda gördükleri, yaptıkları, söyledikleri, yazdıkları, alıp sattıkları ve daha pek çok şeydir veri. Günümüz dünyası öyle bir hal aldı ki; bir X firmasının ürünlerini pazarlarken yapacağı reklamların içeriğini belirleyebilmek için, hedef kitlesi olarak belirlediği insanların günlük rutinlerinde neler yaptıklarını, nelerden hoşlandıklarını, ne yiyip ne içtiklerini dahi öğrenmelerini gerektirir duruma gelindi.

Peki, bunca insana nasıl ulaşacaksınız? En bilindik yöntemlerden biri kişilerin e-posta ve cep telefonu numaralarını, ad soyadlarını, hatta belki de doğum tarihlerini kayıt altına alarak reklamları doğrudan ulaştırmaktır. Promosyon kodları tanımlayarak dönemlik indirimler yapmak, belki de kişiye doğum gününde ekstra indirimler ve bazı fırsatlar sunmak bugün en yaygın kullanılan pazarlama yöntemlerinden biri haline gelmiştir.

Her bir müşterinin kendi kişisel üyelik hesabı olacak, kullanıcı adı ve şifre bilgileri bu hesaplarda saklanacak. Dahası e-posta adresi, telefon, adres, ad ve soyad, doğum tarihi gibi bilgiler de kayıt edilecektir. Günümüzde bir şekilde teknoloji ile temas kurmuş herkes bu işlerin böyle yürütüldüğünü bilir. İşte bu bilgilerin her birine biz veri diyoruz.

Veriyi kayıt altına almak artık kaçınılmaz surette bir ihtiyaç. Yukarıda verdiğimiz örneğin dışında; günlük hayatınızda yapmış olduğunuz işlemlerin pek çoğunda verilerinizi firmalara ya da şahıslara belli ölçülerde de olsa veriyorsunuz. Firmalara ya da şahıslara hakkınızda bilmeleri gereken bazı bilgiler sunuyorsunuz. İnsanlar bu bilgileri kayıt altına alabilmek için çeşitli yöntemler geliştiriyorlar. Eskiden bakkal defterleri vardı. Bakkal amcanız size verdiği malzemeleri bu deftere kayıt ederdi. Peki, neden veresiye ürün vermeye razı gelmişti bakkal amca? Çünkü ailenizin hakkında gerekli verilere sahipti. Ailenizin borcunu ödeyeceğine itimat etmişti. Güvenilirliğinize inanmıştı. Bakıldığı zaman yine sizinle ilgili veriler topladığını söyleyebilir miyiz? Evet ama başka bir yöntemle, verileri zihnine kayıt ederek.

Artık günümüzde büyük firmalar var. Ürünler buralardan satın alınıyor çoğunlukla. Bu firmalar sizinle ilgili verileri nerelerde saklıyor? Tabi ki veritabanında!

Veritabanı sistemlerinin gelişim aşamasında, standart bir text (metin) dosyasına yazılmış kısmen organize veri bloklarından oluştuğunu biliyor muydunuz? Tıpkı bir bakkal defteri gibi. Tabi ki gelişen teknoloji ve değişen dünya koşulları karşısında yetersiz kalındı. Verilerin sadece saklanması sorunları çözmekte yetersiz kaldı. Verileri bir bütün halinde, birbirleriyle ilişkilendirerek saklamak hem kayıt aşamasında hem de kayıtlı verilere daha sonra göz atmak için tekrar bakılması gerektiğinde büyük kolaylıklar sağlamaktaydı.

Bu noktada ilişkisel veritabanı dediğimiz kavram hayata geçirildi. Günümüzde de yoğun bir şekilde kullanılmakta olan bu veritabanı sistemleri, veriyi organize edilmiş bir biçimde saklamak için kullanılmaya başlandı.

Lafı daha fazla uzatmadan konuya giriş yapayım artık değil mi?

İlişkisel Veritabanı Yönetim Sistemleri, pek çoğunuzun bildiğinden emin olduğum üzere, MS SQL Server, Oracle, My SQL gibi pek çok veritabanı sistemi yazılımları sayesinde tasarlanıp geliştirilir durumda. Verileri buralarda oluşturduğumuz veritabanlarına kayıt ediyoruz. Yine kayıtlı verilerin sorgulanması ve yönetimi, veri kayıt performansını artıracak yöntemlerinin uygulamaya geçirilmesi, veri güvenliğini sağlama gibi operasyonel fonksiyonlar da yine bu ve benzeri yazılımlar aracılığı ile yönetilmekte.

İlişkisel Veritabanı Yönetim Sistemleri (kısaca VTYS), verinin doğru bir organizasyonla saklanmasını, veri denetimini ve güvenliğini ön planda tutar. İlişkisel veritabanları veriyi sadece kayıt etmez; aynı zamanda veri bütünlüğünü sağlar, veriyi denetim altında tutar, gerektiğinde size öğrenmek istediklerinizi sunar.

Her ne kadar pek çoğunuzun bu konu hakkında bilgi sahibi olduğunuzu düşünsem de yine de küçük bir örnekle ilişkisel veri mantığını ele almakta fayda olacağını düşünüyorum. Bir haber sitesi düşünün. Günlük haber kaynaklarından edindiği haberleri yayınlıyor bu sitemiz. Her bir haberin başlığını, içeriğini, haberle ilgili görselleri, haberin tarihini, haber ile ilişkili etiketleri ve haber kategorisini yayınlıyor olalım. Her bir haberin yalnız bir kategorisi olduğunu, bir haberin de birden fazla etikete sahip olabileceğini düşünelim.

Böyle bir sitenin veritabanı tasarımı işi önünüze geldiğinde ilk yapmanız gereken ihtiyaçları doğru belirleyip tek tek iş kurallarını tanımlamak olacaktır değil mi? Hangi verileri saklayacağız, verileri saklarken hangi kurallara dikkat etmeliyiz vb. sorular ile veritabanımızı tasarlamaya başlarız.



# 2

## MONGODB VERİTABANINA GİRİŞ

### BU BÖLÜMDE

MongoDB	14
MongoDB Kurulumu	16
MongoDB – İlk Adım	21
MongoDB Temel Kavramlar	24
MongoDB Veritabanında Veri Modelleme	27
Şimdi Pratik Yapma Zamanı - I	30
Neler Öğrendik?	30

İlk adımı MongoDB'yi bilgisayarımıza kurmayı öğrenerek atıyoruz. Şu Mongo Shell diyip durdukları neymiş bir göz attıktan sonra, tabi ki kısaca bakıp geçmeden, Mongo Shell'i biraz daha irdelleyeceğiz. Neler var, neler yapabiliyorum bir bakacağız. Nihayet de bölümün sonuna doğru MongoDB'de temel kavramları güzelce anlayacağız.

Tabi bölüm sonunda pratik yapmadan olmaz!

Birinci bölümde ele almaya çalıştığımız kavramların ardından artık MongoDB veritabanını incelemeye başlayabiliriz diye düşünüyorum.

## MONGODB

**MongoDB;** C++ programlama dili ile yazılmış, doküman tabanlı ve açık kaynak bir veritabanı uygulamasıdır. Bu noktada doküman tabanlı kavramını biraz daha genişletmenin iyi bir başlangıç olacağını düşünüyorum.

Anahtar-Değer ( Key-Value ) Veritabanları	Veriler anahtar-değer çiftleri şeklinde saklanır. Berkeley DB, Memcache DB vb. örnek olarak verilebilir.
Grafik Tabanlı Veritabanları ( Graph DB's )	Veriler arasındaki ilişkilerin de saklandığı veritabanlarıdır. Neo4J, FlockDB en bilindik graf veritabanı çeşitleridir.
Doküman Tabanlı Veritabanları	Veriler genellikle JSON formatında, Document adı verilen bloklar içinde saklanır. Kitabın konusu olan MongoDB haricinde, CouchDB, HBase örnek olarak verilebilir.

NoSQL Veritabanı Çeşitleri

Yukarıdaki tabloda da görüldüğü gibi, MongoDB doküman tabanlı bir NoSQL veritabanıdır. Doküman tabanlı veritabanları verileri çoğunlukla JSON data formatında saklamaktadırlar. Bu da hem veri yazma/okuma hızının artırılmasını hem de verilerin nesne tabanlı programlama prensiplerine daha yakın şekilde sorgulanabilir olmasını sağlamaktadır.

### Örnek BSON Veri Formatı

```
{
  id:1,
  ad: "Burak",
  soyad: "Şavklı",
  meslek: "Enerji Mühendisliği"
}
```

MongoDB verileri, yukarıdaki şekilde görüldüğü gibi, **BSON** (Binary JSON) adı verilen özel bir formatta saklar. Bu format JSON veri modeline çok benzemektedir. MongoDB verileri BSON formatında sakladığı için, JSON veri formatının desteklediği standart tiplere ek olarak, farklı pek çok türde veri tiplerini de desteklemektedir. Aşağıdaki tabloda JSON veri formatında desteklenen tipler görülmektedir.

## JSON (JavaScript Object Notation)

Number, String, Object, Boolean, Array, null

Şimdi BSON (Binary JSON) formatında desteklenen veri tiplerinin listesini inceleyelim:

Tip(Type)	Numara (Number)	Alias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary Data	5	"binData"
Undefined	6	"undefined" (artık kullanılmıyor)
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
Regular Expression	11	"regex"
DBPointer	12	"dbPointer"
JavaScript	13	"javascript"
Symbol	14	"symbol"
JavaScript (With Scope)	15	"javascriptWithScope"
32-bit Integer	16	"int"
Timestamp	17	"timestamp"
64-bit Integer	18	"long"
Min Key	-1	"minKey"
Max Key	127	"maxKey"

BSON Veri Tipleri <https://docs.mongodb.com/manual/reference/bson-types/> bağlantısından alınmıştır.

Yukarıdaki tabloda da görüldüğü gibi, BSON veri formatı JSON'dan farklı olarak pek çok tipi desteklemektedir. Bu özelliği sayesinde MongoDB veritabanı çok daha geniş bir yelpazede veri saklama becerisine sahip olmuştur.

## NOT

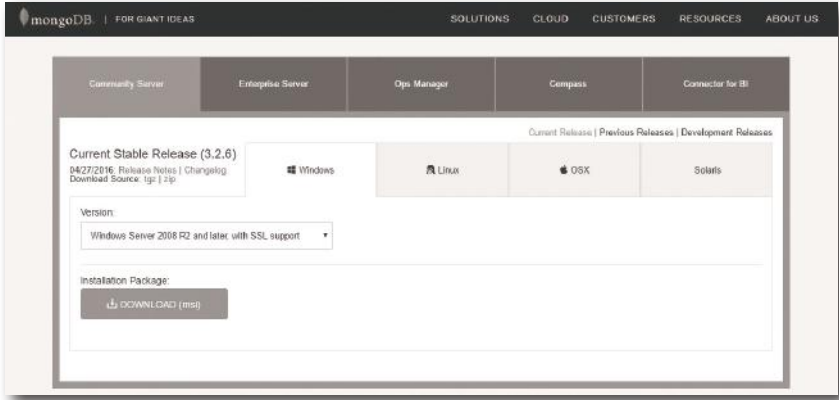
Tabloda görülen alanlardan **Number** alanı ve **Alias** alanında yer alan verilerin ne işe yaradıkları ve nasıl kullanılacakları, kitabın dördüncü bölümünde örnekler ile ele alınacaktır.

## MONGODB KURULUMU

Hadi artık kuralım şu MongoDB'yi dediğinizi duyar gibiyim. Sizleri daha fazla teoriyle sıkmamak için kurulumu başlatıyorum.

İlk olarak [www.mongodb.com/download-center](http://www.mongodb.com/download-center) bağlantısından MongoDB veritabanı dosyalarını indirmemiz gerekiyor.

İşletim sisteminizin türüne göre gerekli dosyalar bu bağlantı altında indirilebilir. Ben bu kitabı hazırlarken Windows işletim sistemi kullandığım için Windows üzerinde çalıştırılacak olan dosyaları indiriyorum.



MongoDB Download Sayfası

MongoDB hem yazılım geliştiricilere hem de veritabanı yöneticilerine yönelik pek çok doküman sunmanın yanı sıra, uzaktan eğitimlerle de kullanıcılarına hizmet vermektedir. Ancak tahmin edeceğimiz üzere tüm bu içerikler İngilizce sunulmaktadır. Yine de İngilizce bilgisi yönünden sorun yaşamayan okuyucularımızın <https://docs.mongodb.com/> bağlantısı üzerinden tüm bu içeriklere erişebileceğini de belirtmek istiyorum. .msi uzantılı kurulum dosyasını indirme işlemini tamamladıysanız kurulumla başlayabiliriz:

Next diyerek ilk adımı geçiyoruz.

Kurulum – Adım 1



# 5

## MONGODB AGGREGATION (KÜMELEME) İŞLEMLERİ

### BU BÖLÜMDE

Giriş	96
Aggregation Pipeline Üyelerini Kullanarak	
Kümeleme İşlemleri	96
Şimdi Pratik Yapma Zamanı – V	103
count() ve distinct() Fonksiyonlarının	
Kullanımı	104
map-reduce Fonksiyonları ile Kümeleme	
İşlemleri	106
Şimdi Pratik Yapma Zamanı – VI	111
Neler Öğrendik?	111

Hangi veritabanı ile çalışırsanız çalışın, mutlaka sizden analitik sorgular oluşturmanız beklenecektir.

Çünkü veritabanı kullanma amaçlarından biridir verileri analiz edebilmek. Dolayısıyla MongoDB veritabanları üzerinde kümeleme sorgularının nasıl oluşturulduğunu öğrenmek, bu kitabın okuyucuları olan sizlerin kaçınılmaz ödevlerinden biridir.

Bölüm boyunca sizlere; kümeleme kavramının ne anlama geldiğinden başlayarak adım adım gruplama ya da diğer bir ifadeyle kümeleme sorgularının çalışma mantığı, farklı ve detaylı örnekler üzerinde çalışılarak anlatılacaktır.

## GİRİŞ

Tüm veritabanı sistemlerinde kümeleme işlemleri, özellikle veri analizi aşamasında sıklıkla kullanılan operasyonel yöntemlerdir. Elde edilen verilerin işlenmesi ve bu verilerden çıkarımlar elde edilmesi aşamasında ilişkisel veritabanı sistemlerinde kullanılan gruplama işlemleri sayesinde, veri kümelerindeki kayıtların arasından, belirli kriterlerde gruplama ve sayma işlemleri gerçekleştirilebilmektedir.

Elbette bu konuda MongoDB de istisna oluşturmuyor. MongoDB, çok yüksek veri kümeleriyle çalışabilen bir sistem olduğu için kümeleme işlemleri sonucunda elde edilen çıkarımların, karar alıcılara sunacağı kolaylıklar hatırı sayılır ölçüdedir. Bu nedenle, MongoDB veritabanında elimizdeki verileri rahatlıkla işleyerek sayısal sonuçlar üretebilmemizi sağlamak amacıyla kümeleme yöntemleri geliştiriciler tarafından sunulmuştur.

## AGGREGATION PIPELINE ÜYELERİNİ KULLANARAK KÜMELEME İŞLEMLERİ

Microsoft SQL Server, Oracle, My SQL gibi ilişkisel veritabanı sistemleri üzerinde çalışan okuyucularım iyi bilirler ki SQL sorgu dilinde kullanılan group by komutu, bir veri kümesi üzerinde verileri gruplandırarak sayısal sonuçlar elde etmek için kullanılır. Daha önce veritabanı sistemleri ile hiç çalışmamış okuyucularım için giriş niteliğinde, diğer okuyucularım için de tekrar niteliğinde olsun diye kümeleme işlemlerinin genel mantığını bir örnek üzerinde kısaca açıklamakta yarar olduğunu düşünüyorum.

Bir öğretmen sınıftaki 40 öğrencinin ders başarı durumlarını görmek istiyor. Bir dersten başarılı olma koşulu ise o dersten en az 60 almak olsun. Öğretmen nasıl bir yöntem izler?

İlk olarak öğrencilerin ders notlarına bakar değil mi? Matematik dersinden 60 ve üstü notu olan öğrencileri sayar, bir kâğıda dersin adını ve o dersten başarılı olan öğrenci adedini yazar. Sonra Fizik dersine bakar ve aynı şekilde 60 ve üstü notu olan kaç öğrenci olduğunu yine yazarak ayırmış olur. Bu şekilde, örneğin 5 farklı ders için, başarılı öğrenci sayılarını tespit eder.

Öğretmenin bu yaptığı işlemin adı nedir peki?

Kesinlikle! Bu işlemin adı kümelemedir. Peki, kümeleme kriteri olarak ne kullanılmıştır? Tabi ki ders isimleri!

**DİKKAT**

Size vermiş olduğum örnek document yapısını, online json editöründe hazırladığım için veri formatı JSON data şeklinde gösterilmiştir. Ancak artık biliyorsunuz ki MongoDB BSON data type kullanmaktadır. O nedenle alan isimlerini çift tırnaklar arasında almak ya da sayısal verileri çift tırnaklar içinde tanımlamak zorunda değilsiniz! Yeter ki tarih-saat verisini, önceki bölümlerde kullandığımız yöntemle ekleyin.

İlk örneğimizde, hava durumu verileri içinde sıcaklık değeri 20'den küçük olan kayıtlara ait tüm bilgileri görüntülemek isteyelim. Aşağıdaki sorguyu inceleyin:

```
> db.veri.aggregate([ {$match:{deger:{$lt:20}}} ]).pretty()
```

**İşte bu!** dediğinizi duyar gibiyim. Burada görüldüğü üzere, koşul işlemi için \$match operatörü kullanılmıştır. Aggregate() fonksiyonu içinde, kümeleme kriterinin tanımlanmasında kullanılan bu operatörü biraz daha yakından inceleyelim.

### \$match Operatörü ile Koşul Tanımlama

Yazmış olduğumuz kümeleme sorgularında, kümeleme kriterini belirlemek için kullanılan operatördür. Sıcaklık değeri 20'den küçük olan kayıtları görüntülediğimiz aggregate() örneğimizde de hatırlayacağınız üzere, kümeleme işlemi için kriteri bu operatör ile tanımlamıştık. Şimdi dilerseniz \$match operatörünü kullanarak yeni bir örnek uygulama geliştirelim. Bu kez koşul olarak hava durumu Çok bulutlu olma kriterini kullanıyor olalım.

```
db.veri.aggregate([ {$match:{durum:"Çok bulutlu"}} ]).pretty()
```

Yukarıdaki örneğimizde match operatörünün kullanımına dikkat edelim.

Şu ana kadar yaptığımız iki örnek için de şu soruyu kendinize sordunuz mu?; *Biz bunları find() fonksiyonunu kullanarak da listeletebiliyoruz. Ancak siz bize aggregate() fonksiyonunun kullanım amacının kümeleme yapmak olduğunu söylediniz, hatta kümelemenin ne demek olduğunu da güzel bir örnek ile anlattınız. Ancak hala kümelenmiş bir veri seti göremiyoruz biz sonuç olarak. Bir sorun mu var? :*

Cevabınız çok basit arkadaşlar; henüz \$group operatörü ile tanışmadınız! Hadi o zaman tanışalım.

## \$group Operatörü ile Gruplama İşlemleri

\$group operatörü, kümelенmiş veri setinde sonuç olarak üretilecek verilerin tanımlanmasını sağlayan bir yardımcı operatördür arkadaşlar. Hava durumu için yaptığımız örnek üzerinde duralım yine. Aşağıdaki sorguyu inceleyip çalıştırın:

```
> db.veri.aggregate(  
... [{$group:{_id:"$durum",kayıtadet:{$sum:1}}} ] )
```

Sorguyu yazıp çalıştırdığınızda sonuç aşağıdaki ekran görüntüsünde olduğu gibi listelenecektir:



```
C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe  
> db.veri.aggregate(  
... [{$group:{_id:"$durum",kayıtadet:{$sum:1}}} ] )  
{ "_id" : "Açık", "kayıtadet" : 2 }  
{ "_id" : "Parçalı bulutlu", "kayıtadet" : 1 }  
{ "_id" : "Sağanak yağışlı", "kayıtadet" : 3 }  
{ "_id" : "Az Bulutlu", "kayıtadet" : 1 }  
{ "_id" : "Çok bulutlu", "kayıtadet" : 3 }  
{ "_id" : "Az bulutlu", "kayıtadet" : 2 }  
>
```

İşte şimdi gerçek anlamda bir **İşte bu!** deme zamanı! Gördüğünüz gibi veri seti, hava durumu bilgisine göre gruplanarak, gruplama sonucunda hava durumu çeşidine göre adet bilgisi listelenmiş oldu.

Bir başka örnek ile konuyu pekiştirelim. Bu örneğimizde de hava durumu verilerini şehir isimlerine göre gruplayarak, hangi şehre ait kaç adet kayıt olduğunu görelim:

```
> db.veri.aggregate(  
... {$group:{_id:"$şehir",kayıtadet:{$sum:1}}} )
```



```
C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe  
> db.veri.aggregate(  
... {$group:{_id:"$şehir",kayıtadet:{$sum:1}}} )  
{ "_id" : "Isparta", "kayıtadet" : 6 }  
{ "_id" : "Sakarya", "kayıtadet" : 6 }  
>
```



# MONGODB REGULAR EXPRESSION KULLANIMI

## BU BÖLÜMDE

Regular Expression Tanımlama	130
MongoDB'de Regular Expression Kullanımı	134
Neler Öğrendik?	136

Özellikle metin sorgulama yaparken (text search) olmazsa olmaz yöntemimiz regular expression kullanımıdır. Yazılım teknolojilerine aşına arkadaşların **regex** kısa adıyla da tanıdığı bu yöntemin uygulama alanlarından en yaygını validasyon işlemleridir.

Ancak bizi ilgilendiren kısım MongoDB'de nasıl regex kullanabileceğimiz olacak. Dolayısıyla önce regex kavramına genel bir bakış yapacağız, ardından günlük hayatta karşımıza çıkabilecek regex örneklerinden birkaçını kendimiz hazırlayacağız. Son olarak da MongoDB veritabanındaki koleksiyonlarda, veriler üzerinde regex ifadelerini nasıl çalıştırabileceğimizi kavrayacağız.

Kitabın sonuna doğru yaklaşıırken, kendinizi MongoDB'ye daha hakim hissediyor musunuz? Ya da yaptığınız örneklerin başarma yüzdesi arttıkça MongoDB'ye olan sevginiz bir kat daha arttı mı? Cevabınız evet ise, bir başka güzel konu olan Regular Expression kullanımı konusuna giriş yapabiliriz demektir.

Bu bölümde MongoDB'de düzenli ifadelerin (regular expression), metin aramalarında nasıl etkili biçimde kullanılabileceğini öğreniyor olacağız. Konu regular expression olunca tabi, bir zamanlar benim de hissettiğim o duyguyu bazıları'nızın da hissettiğini tahmin ediyorum. (ürperme)

MongoDB'de regular expression (kısaca regex) kullanımına geçmeden önce, regular expression ile düzenli ifadelerin nasıl tanımlandığını anlamaya çalışacağız.

## REGULAR EXPRESSION TANIMLAMA

Regular expression ifadeleri, bütüne baktığınızda anlaşılması zor olan ancak parçalara ayırarak incelediğinizde aslında tanımlaması çok da zor olmayan kalıplardır. Regular expression tanımı aslında bir çeşit kalıptır. Düzenli ifade tanımlamanın amacı, hazırladığınız kalıba uyan ifadeleri tespit edebilmektir.

Klasik yöntemler ile çok daha uzun ve zahmetli kodlar kullanarak, üstelik performanstan da kaybederek, yapacağınız kontrolleri regex kullanarak milisaniyeden daha kısa bir süre içinde kontrol ettirebilirsiniz. Akıllıca ve amaca doğru hizmet eden bir regular expression ifadesi ile bu mümkündür.

Düzenli ifadeler ile çalışırken kullandığımız temel işleçler ve görevlerine öncelikle bir göz atalım. Ardından da, bazı kural gerektiren ifadeler için nasıl regex tanımlayacağımızı örnek uygulamalar üzerinde inceleyelim.

### NOT

Hazırladığınız herhangi bir regex ifadesinin doğruluğunu test etmek için [www.regexpr.com](http://www.regexpr.com) ya da bir benzeri siteden yararlanabilirsiniz.

Aşağıdaki tabloda, temel regular expression işleçleri ve görevleri listelenmiştir:

İşleç	Görevi	Örnek Kullanım
.	(Nokta) Tek bir karakteri temsil eder.	Mo.go => Mongo, MoNgo, Morgo vb tüm stringleri kabul eder.
?	Kendinden önceki karakterin, string içinde olmasını ya da olmamasını kontrol eder.	Mon?go => Mongo ya da Mogo stringlerini kabul eder.

## MONGODB'DE REGULAR EXPRESSION KULLANIMI

Gelelim MongoDB ile regular expression ifadeleri nasıl kullanabildiğimize. MongoDB veritabanınızda kayıtlı olan veriler üzerinde regex ifadeleri filtreleyici olarak rahatlıkla kullanabiliyorsunuz. Düzenli ifadeler ile filtreleme yapabilmek için ise \$regex operatöründen yararlanıyoruz.

SQL ile veritabanı üzerine çalışan arkadaşlarımız bilirler, SQL dilinde like operatörü metinler içinde ile başlayan, ile biten gibi ifadeleri sorgulamada kullanılır. MongoDB'de de \$regex operatörü benzer bir işlev gerçekleştirir. Metinsel ifadeler içinde \$regex ile rahatlıkla filtreleme yapabilirsiniz.

Birkaç örnek ile pekiştirdiğimizde konunun tam olarak anlaşılacağını düşünüyorum.

Kitabın 6. bölümünde meteoroloji adında bir veritabanı oluşturmuştuk hatırlarsanız. Bu veritabanında veri isimli bir koleksiyon içinde Sakarya ve Isparta illeri için meteorolojik veriler saklıyorduk.

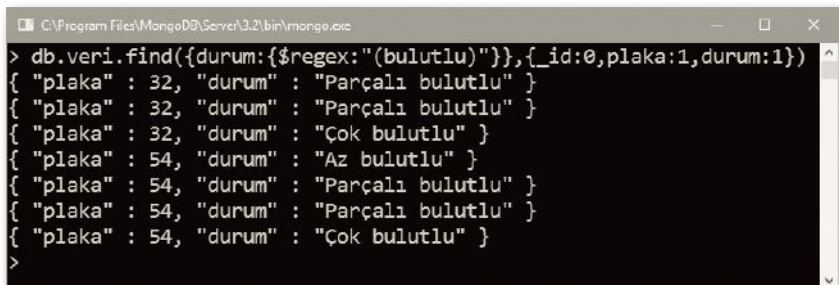
Örneğimizde; durum alanındaki verilerde içinde bulutlu kelimesinin geçtiği kayıtları filtrelemek istediğimizi düşünelim:

---

```
> db.veri.find({durum:{$regex:"(bulutlu)"},{_id:0,plaka:1,durum:1})
```

---

Kullanmamız gereken düzenli ifade, yukarıdaki gibi olacaktır. Ne kadar basit değil mi? Hiç öyle, bölüm başında yazdığımız gibi karmaşık tanımlama gerektirmiyor. Hatta oluşacak ekran çıktısı biraz daha sade ve anlaşılır olsun diye, gösterilecek alanların plaka ve durum bilgilerinden oluşacağını da belirtiyoruz. Komutu çalıştırdığımızda oluşacak ekran çıktısı aşağıdaki gibi olacaktır:

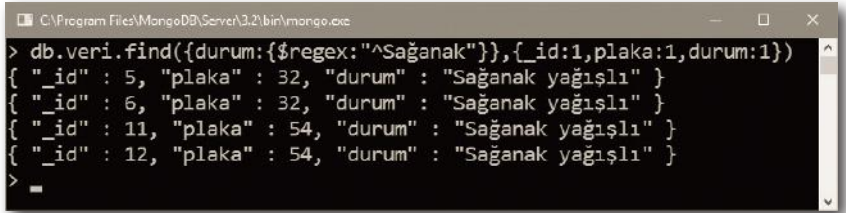


```
C:\Program Files\MongoDB\Server\3.2\bin>mongo.exe
> db.veri.find({durum:{$regex:"(bulutlu)"},{_id:0,plaka:1,durum:1})
{ "plaka" : 32, "durum" : "Parçalı bulutlu" }
{ "plaka" : 32, "durum" : "Parçalı bulutlu" }
{ "plaka" : 32, "durum" : "Çok bulutlu" }
{ "plaka" : 54, "durum" : "Az bulutlu" }
{ "plaka" : 54, "durum" : "Parçalı bulutlu" }
{ "plaka" : 54, "durum" : "Parçalı bulutlu" }
{ "plaka" : 54, "durum" : "Çok bulutlu" }
>
```

Bir başka örnek daha yapalım. Bu örneğimizde de yine durum verisi Sağanak kelimesi ile başlayan kayıtları filtreleyelim. Aşağıdaki komutu inceleyin:

```
> db.veri.find({durum:{$regex:"^Sağanak"}},{_id:1,plaka:1,durum:1})
```

Oluşacak ekran görüntüsü ise şu şekildedir:



```
C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe
> db.veri.find({durum:{$regex:"^Sağanak"}},{_id:1,plaka:1,durum:1})
{ "_id" : 5, "plaka" : 32, "durum" : "Sağanak yağışlı" }
{ "_id" : 6, "plaka" : 32, "durum" : "Sağanak yağışlı" }
{ "_id" : 11, "plaka" : 54, "durum" : "Sağanak yağışlı" }
{ "_id" : 12, "plaka" : 54, "durum" : "Sağanak yağışlı" }
>
```

Yine kitabımızın 5. bölümünde kullanmış olduğumuz okul isimli bir veritabanımız vardı hatırlarsınız. Mutlaka o veritabanı da elinizin altındadır, değil mi? Eğer sildiyse de, 5. bölüme geri dönerek verileri yeniden ekleyebilirsiniz.

Bu örneğimizde de, okul veritabanı içindeki ogrenci isimli koleksiyonda yer alan kayıtlardan, adı sessiz harf ile başlayan kayıtları, düzenli ifadelerden yararlanarak elde etmeye çalışalım. Aşağıdaki komutu inceleyin:

```
> db.ogrenci.find({ad:{$regex:"^[^aeııöüü]"},$options:"i"},{_id:1,ad:1})
```

Yukarıdaki örneğimizde; [ ] içine sesli harfleri tek tek tanımlayarak başlarına satır başı simgesi olan ^ karakterini ekliyor ve bu sesli harflerin dışındakileri ifade ettiğimizi bildiriyoruz. Köşeli parantezler dışına da yine ^ karakterini ekleyerek de sessiz harf ile başlanacağını bildiriyoruz.

Burada \$options parametresinden bahsetmek gerekirse; \$options parametresine atanan i değeri sayesinde, case sensitive bir arama gerçekleştirmiş oluyoruz. Yalnız tabi ki MongoDB'nin Türkçe karakter sorunu burada karşımıza çıktığından, i harfinin büyüğü İ olarak algılanamıyor. O nedenle bu karakteri de regex dizisi içinde belirtiyoruz.

Oluşacak ekran çıktısı ise aşağıdaki şekilde karşımıza çıkıyor: