VERİ TABANI VE YÖNETİM SİSTEMLERİ

Dr. Öğretim Üyesi

Alper Talha KARADENİZ

GENEL DERS İÇERİĞİ

1. MongoDB Nedir?
2. MongoDB Özellikleri
3. Cluster Mimarisi
4. Temel CRUD İşlemleri
5. Veri Modelleme Stratejileri
6. İndeksleme ve Performans
7. Sorgu Optimizasyonu ve explain() Komutu
8. Uygulamalar

Lütfen derse gelmeden önce aşağıdaki linkler üzerinden gerekli belgeleri indiriniz.

https://www.mongodb.com/try/download/community

https://www.mongodb.com/try/download/shell

MongoDB Nedir?

MongoDB, modern uygulamalarda yaygın şekilde kullanılan, açık kaynaklı, belge yönelimli (document-oriented) bir NoSQL veritabanı sistemidir.

Verileri geleneksel satır-sütun yapılı tablolarda değil, JSON benzeri BSON (Binary JSON) formatında saklar. Bu yapı, esnek veri modeli sayesinde özellikle büyük veri, gerçek zamanlı analiz, içerik yönetimi ve mobil uygulamalar gibi alanlarda tercih edilmesini sağlar.

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

— no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB Document

```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
   { model: 'Bentley',
     year: 1973,
     value: 100000, ... },
   { model: 'Rolls Royce',
     year: 1965,
     value: 330000, ... }
  ]
}
```

Terminology

RDBMS		MongoDB
Table, View	→	Collection
Row	→	Document
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference
Partition	→	Shard

MongoDB Özellikleri

1. Açık Kaynak

MongoDB, açık kaynaklı bir yazılımdır; yani kaynak kodu herkesin erişimine açıktır. Bu sayede geliştiriciler istedikleri gibi düzenleyebilir, katkı sağlayabilir veya kendi ihtiyaçlarına göre uyarlayabilir.

2. Döküman Tabanlı

MongoDB, verileri belgeler (documents) içinde saklar. Bu belgeler, JSON benzeri BSON formatındadır. Bu yapı, karmaşık veri türlerini (iç içe listeler, objeler) daha doğal bir şekilde temsil etmeyi sağlar.

MongoDB Özellikleri

3. Schema Bağımsız Olması

MongoDB'de veriler şemasız (schema-less) olarak saklanabilir. Yani her dokümanın yapısı farklı olabilir; yeni alanlar eklemek veya çıkarmak için tabloyu güncellemeye gerek yoktur. Bu, esnek veri yönetimi sağlar.

4. Scalability (Ölçeklenebilirlik)

MongoDB büyük veri ile çalışırken yatay ölçeklenebilirlik (horizontal scalability) sağlar. Yani veriler birden çok sunucuya bölünebilir (sharding), böylece sistem performanslı kalır.

MongoDB Özellikleri

5. Replikasyon (Kopyalama)

MongoDB, replica set sayesinde verileri birden fazla sunucuya kopyalayabilir. Ana sunucuya bir şey olursa, yedek sunucular devreye girerek veri kaybını önler. Bu da yüksek erişilebilirlik (high availability) sağlar.

6. Load Balancing (Yük Dengeleme)

MongoDB çoklu sunucu ortamlarında gelen istekleri dengeli şekilde dağıtarak yük dengeleme yapar. Bu sayede sistem aşırı yüklenmez, performans korunur.

MongoDB Özellikleri

7. Sorgulama Kolaylığı

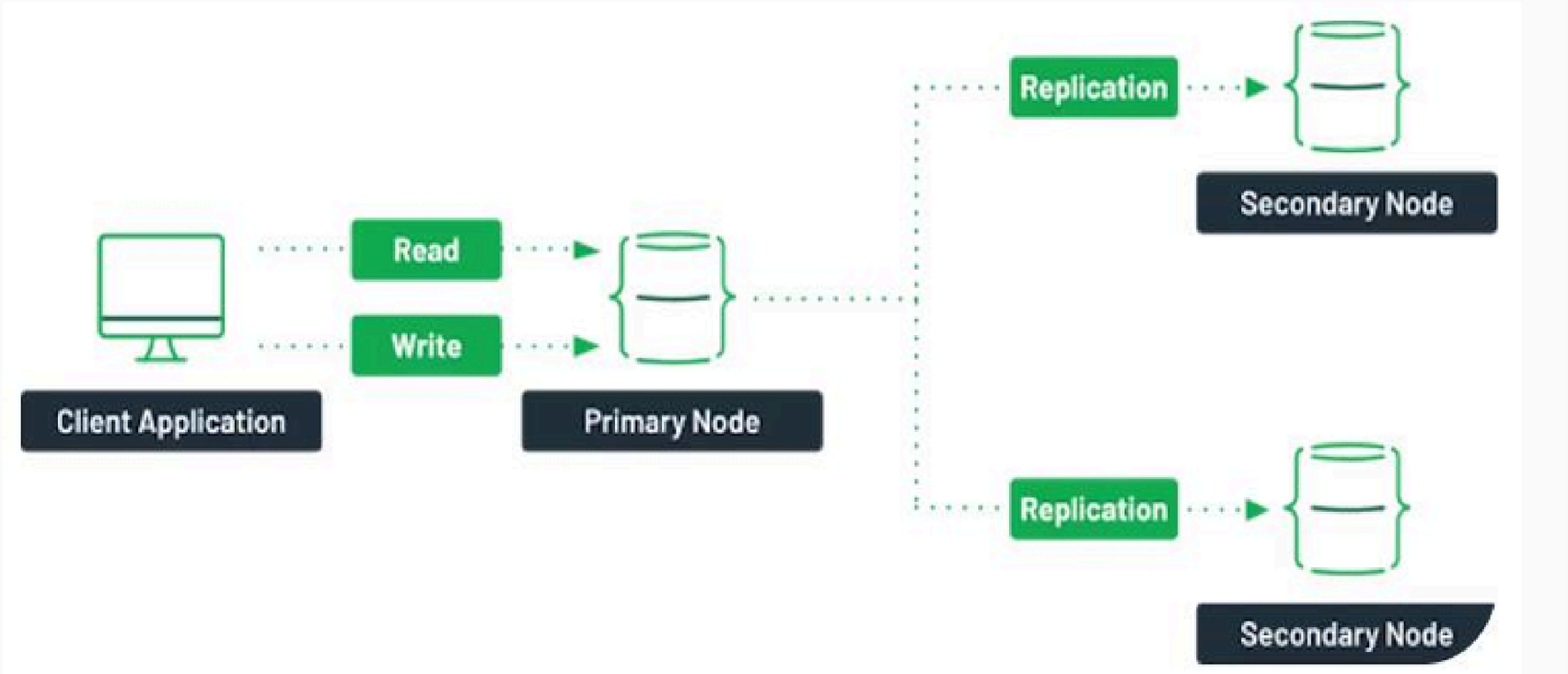
MongoDB, SQL'e benzer ama daha sade bir sorgu dili (Mongo Query Language - MQL) kullanır. JSON yapısında sorgular yazılır ve veriler kolayca filtrelenebilir.

8. Indexleme

MongoDB, sorgu performansını artırmak için indeks kullanır. Belirli alanlara indeks ekleyerek verilerin daha hızlı bulunmasını sağlar.

MongoDB Cluster Mimarisi

Bir MongoDB replica set, verilerin tam kopyasını içeren bir veya daha fazla sunucudan oluşan bir gruptur . Teknik olarak bir veya iki düğüme sahip olmak mümkün olsa da, önerilen minimum üç düğümdür. Bir birincil düğüm uygulamanızın okuma ve yazma işlemlerini sağlamaktan sorumluyken, iki ikincil düğüm verilerin bir kopyasını içerir



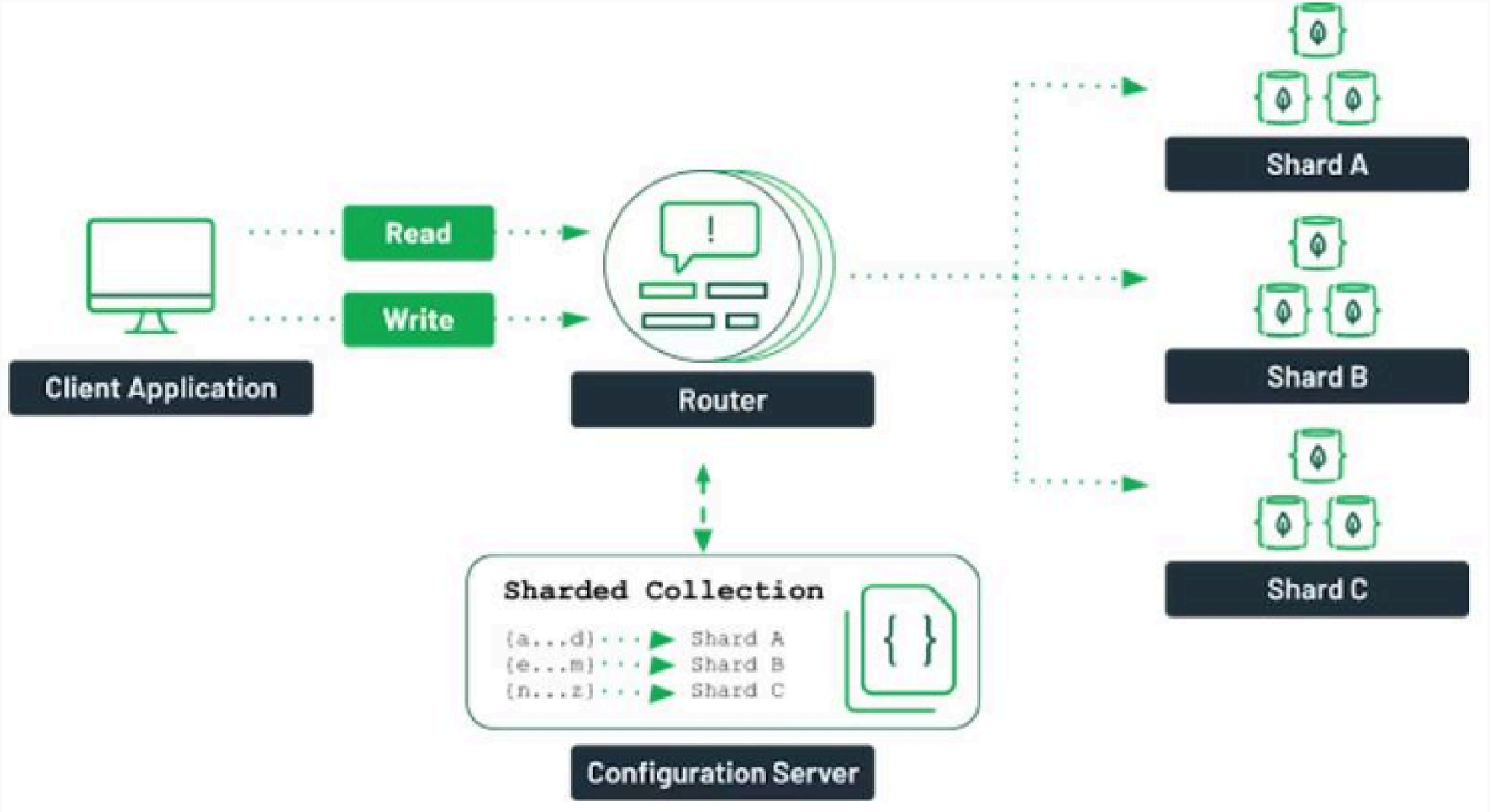
MongoDB Cluster Mimarisi

Birincil düğümün herhangi bir nedenle kullanılamaz hale gelmesi durumunda, bir seçim süreci ile yeni bir birincil düğüm seçilir. Bu yeni birincil düğüm artık okuma ve yazma işlemlerinden sorumludur. Birincil düğüm kullanılamaz durumdaysa, istemci uygulamasından gelen trafik yeni bir birincil düğüme yönlendirilir

MongoDB Cluster Mimarisi

Parçalanmış küme, verilerinizi birden fazla çoğaltma kümesine dağıtarak yatay olarak ölçeklendirmenin bir yoludur.
Bir koleksiyon üzerinde okuma veya yazma işlemi gerçekleştirildiğinde, istemci isteği bir yönlendiriciye (mongos) gönderir.
Yönlendirici daha sonra yapılandırma sunucusu aracılığıyla verilerin hangi parçada depolandığını doğrulayacak ve istekleri belirli kümeye gönderecektir

MongoDB Cluster Mimarisi



Temel CRUD İşlemleri

CRUD, veritabanı işlemlerinin temelini oluşturan 4 kavramın baş harflerinden oluşur:

1. Veri Ekleme (Create)

insert_one() metodu, belirtilen sözlük (dict) yapısını koleksiyona ekler.
Bu işlem MongoDB'de yeni bir doküman oluşturur.

```
collection.insert_one({"ad": "Veri Tabanları", "yazar": "Ali Yılmaz", "sayfa": 300})
```

2. Veri Sorgulama (Read)

find_one() fonksiyonu, belirtilen koşula uyan ilk belgeyi getirir.
Sonuç bir Python sözlüğü (dict) olur.

```
kitap = collection.find_one({"ad": "Veri Tabanları"})
```

Temel CRUD İşlemleri

3. Veri Güncelleme (Update)

update_one() fonksiyonu, belirtilen filtreyle eşleşen ilk belgeyi günceller.

```
collection.update_one({"ad": "Veri Tabanları"}, {"$set": {"sayfa": 320}})
```

- İlk parametre: Hangi belgeyi güncelleyeceğini belirleyen filtre ({"ad": "Veri Tabanları"})
- İkinci parametre: Ne şekilde güncelleme yapılacağıdır. Burada $set operatörü "sayfa" değerini 320 yapıyor.

4. Veri Silme (Delete)

delete_one() fonksiyonu, eşleşen ilk belgeyi koleksiyondan siler. Eğer başka "Veri Tabanları" belgeleri varsa onlar silinmez.

```
collection.delete_one({"ad": "Veri Tabanları"})
```

Gelişmiş Sorgu Operatörleri

MongoDB, sadece temel eşitlik sorguları değil; çok daha karmaşık ve güçlü sorgulama imkânları da sunar. Bu operatörler genellikle find() içinde kullanılır.

Karşılaştırma Operatörleri

$gt → "greater than" – büyük

{ yas: { $gt: 18 } } → 18 yaşından büyük olanları getirir.

$lt → "less than" – küçük

{ yas: { $lt: 30 } } → 30 yaşından küçük olanları getirir.

$ne → "not equal" – eşit olmayan

{ cinsiyet: { $ne: "E" } } → cinsiyeti "E" olmayanları getirir.

$in → Belirtilen değerlerden herhangi biri

{ sehir: { $in: ["Ankara", "İzmir"] } } → Bu şehirdekileri getirir.

Gelişmiş Sorgu Operatörleri

Mantıksal Operatörler

$and → Birden fazla koşul aynı anda sağlanmalı

{ $and: [{ yas: { $gt: 20 } }, { cinsiyet: "K" }] }

$or → En az bir koşul sağlanmalı

{ $or: [{ sehir: "Ankara" }, { sehir: "İzmir" }] }

$not → Koşulun sağlanmamasını ister

{ yas: { $not: { $gt: 30 } } } → 30'dan küçük olanlar

Gelişmiş Sorgu Operatörleri

Dizi İşlemleri

$elemMatch → Dizi içinde birden fazla koşulu aynı eleman için kontrol eder

{ notlar: { $elemMatch: { vize: { $gt: 50 }, final: { $gt: 60 } } } }

$size → Dizinin eleman sayısına göre sorgu yapar

{ hobiler: { $size: 3 } } → hobileri 3 tane olan belgeler

Veri Modelleme Stratejileri

MongoDB’de veri modelleme, uygulamanın hızını ve yapısını doğrudan etkiler. Başlıca iki yaklaşım vardır:

1. Embedded (Gömülü) Veri Modeli

- İlişkili veriler aynı doküman içinde tutulur.

Avantaj: Daha hızlı okuma sağlar çünkü tek sorgu yeterli.

Dezavantaj: Veri tekrarı olabilir, belge boyutu artabilir.

```
{
  "ad": "Ali",
  "siparisler": [
    { "urun": "Kalem", "adet": 3 },
    { "urun": "Defter", "adet": 2 }
  ]
}
```

Veri Modelleme Stratejileri

2. Referenced (Referanslı) Veri Modeli

- Veriler farklı koleksiyonlarda tutulur, ObjectId referansları ile bağ kurulur.

Avantaj: Veri tekrarını azaltır, esnek yapı sağlar.

Dezavantaj: Veriye ulaşmak için birden fazla sorgu gerekebilir (join benzeri lookup).

```
{
  "_id": 1,
  "ad": "Zeynep"
}
{
  "kullanici_id": 1,
  "urun": "Kalem",
  "adet": 5
}
```

// Sipariş

İndeksleme ve Performans

MongoDB'de indeksler, sorguların daha hızlı çalışmasını sağlar. Büyük veri koleksiyonlarında vazgeçilmezdir.

Tekil indeks: Belirli bir alan üzerinde oluşturulur. Örnek: { ad: 1 }

Bileşik indeks: Birden fazla alanı kapsar. Örnek: { ad: 1, soyad: -1 }

Metin indeksi: Metin arama yapmak için kullanılır. createIndex({ alan: "text" })

Sorgu Optimizasyonu ve explain() Komutu

explain() komutu bir sorgunun nasıl çalıştığını gösterir.
Kullanılarak sorgu süresi, kullanılan indeks ve taranan belge sayısı analiz edilebilir.

db.ogrenciler.find({ ad: "Ali" }).explain("executionStats")

Sonuç:

- nReturned: Kaç belge döndü
- executionTimeMillis: Kaç ms sürdü
- totalDocsExamined: Kaç belge tarandı (az olması iyidir)

UYGULAMALAR

Ekleme (insert_one / insert_many) İşlemi

```
collection.insert_one({"ad": "Veritabanları", "yazar": "Ali", "sayfa": 250})
# Tek kitap ekleme

collection.insert_many([
    {"ad": "Algoritmalar", "yazar": "Ayşe", "sayfa": 300},
    {"ad": "Yapay Zeka", "yazar": "Mehmet", "sayfa": 450}
])
 # Çoklu kitap ekleme
```

UYGULAMALAR

Ekleme (insert_one / insert_many) İşlemi

```
collection.insert_one({
 "ad": "Python",
 "etiketler": ["programlama", "veri", "otomasyon"]
}) # Liste içeren belge eklemek

collection.insert_one({
    "ad": "Zeynep",
    "notlar": {"vize": 75, "final": 85}
})  #  İç içe belge ile öğrenci ekleme
```

UYGULAMALAR

Bulma (find / find_one) İşlemi

```
collection.find_one({"ad": "Veritabanları"}) #  Belirli bir kitabı bul

collection.find({"sayfa": {"$gt": 300}}) # Sayfa sayısı 300'den büyük olanları bul

collection.find({"yazar": {"$in": ["Ali", "Ayşe"]}}) # Belirli yazarlara ait kitapları bul

collection.find({"notlar.vize": {"$gte": 70}}) # Belge içinde iç içe alanla bulma

collection.find({}, {"_id": 0, "ad": 1, "yazar": 1}) # Sadece "ad" ve "yazar" alanlarını getir
```

UYGULAMALAR

Sıralama (sort) İşlemi

```
collection.find().sort("sayfa", 1) # Sayfa sayısına göre artan sırala

collection.find().sort("ad", -1)  # Ada göre azalan sırala

collection.find().sort("tarih", -1) # Tarihe göre yeni olan en üstte

collection.find().sort([("yazar", 1), ("ad", 1)]) # Önce yazar, sonra ad sıralı getir

collection.find({"sayfa": {"$gt": 100, "$lt": 400}}).sort("sayfa", 1)
# Belirli sayfa aralığında olanları sıralayarak getir
```

UYGULAMALAR

Güncelleme (update_one / update_many) İşlemi

```
collection.update_one({"ad": "Veritabanları"}, {"$set": {"sayfa": 280}}) # Sayfa sayısını güncelle

collection.update_many({"yazar": "Ali"}, {"$set": {"yazar": "Ali Yılmaz"}}) # Bir yazarı topluca değiştir

collection.update_one({"ad": "Zeynep"}, {"$set": {"notlar.final": 90}}) # Notlara final notu ekle
```

UYGULAMALAR

Silme (delete_one / delete_many) İşlemi

```
collection.delete_one({"ad": "Veritabanları"}) # Tek belge sil

collection.delete_many({"yazar": "Ayşe"}) # Aynı yazara ait tüm kitapları sil

from datetime import datetime, timedelta
silinme_tarihi = datetime.now() - timedelta(days=30)
collection.delete_many({"tarih": {"$lt": silinme_tarihi}}) # Belirli tarihten önceki belgeleri sil
```

UYGULAMALAR

İç İçe Belge (Embedded Documents)

```
collection.insert_one({
    "ad": "Ahmet",
    "notlar": {"vize": 70, "final": 90}
})  # Öğrenci ve notları birlikte ekle

collection.insert_one({
    "siparis_no": 1001,
    "urun": {"ad": "Mouse", "fiyat": 150}
}) # Sipariş içinde ürün bilgisi
```

UYGULAMALAR

İç İçe Belge (Embedded Documents)

```
collection.insert_one({
    "ad": "Fatma",
    "adres": {"sehir": "Ankara", "posta_kodu": "06100"}
}) # Kullanıcı ve adres bilgisi

collection.insert_one({
    "urun_ad": "Laptop",
    "stok": {"adet": 15, "depo": "A1"}
}) # Ürün ve stok detayları
```

UYGULAMALAR

Karmaşık Sorgular

```
collection.find({"$and": [{"sayfa": {"$gt": 200}}, {"yazar": "Ali"}]})
# Hem sayfa > 200 hem de yazar "Ali"

collection.find({"$or": [{"sayfa": {"$lt": 200}}, {"yazar": "Ayşe"}]})
# Ya sayfa < 200 ya da yazar Ayşe

collection.find({"notlar": {"$elemMatch": {"final": {"$gt": 70}, "vize": {"$gt": 60}}}})
# Belge içinde hem final > 70 hem de vize > 60
```

UYGULAMALAR

Karmaşık Sorgular

```
collection.find({"$and": [{"sayfa": {"$gt": 100, "$lt": 300}}, {"ad": {"$regex": "Python"}}]})
# Sayfa sayısı 100 ile 300 arasında olanlar ve "Python" geçenler

collection.find({"etiketler": "veri"})
# "etiketler" dizisinde "veri" olan belgeler
```