

NoSQL

Temel Kavramlar ve Kısaltmaları

- RDBMS : Relational Database Management System
- ACID : Atomicity – Consistency – Isolation – Durability
- CAP : Consistency, Availability, Partition Tolerance
- SQL : Structured Query Language
- DB : Database
- JSON : JavaScript Object Notation

NoSQL













- İlişkisel olmayan (NoSQL) veri tabanı; 1998 yılında ilk olarak Carlo Strozzi tarafından öne sürülen bir kavramdır.
- NoSQL ilişkisel (RDBMS) (Relational Database Management System) veri tabanlarına alternatif olarak çıkan, nesneyi dikkate alan SQL'deki gibi belirli kolonlara ihtiyaç duymayan veri depolama yaklaşımıdır.
- İlişkisel olmayan veri tabanları yatay olarak ölçeklendirilen bir veri depolama sistemidir. NoSQL veritabanları, özellikle büyük veri hacmi, düşük gecikme süresi ve esnek veri modelleri gerektiren uygulamalar için optimize edilmiştir.
- Dünya'da NoSQL örneklerini incelediğimizde; sosyal ağlarda Digg'in 3 TB'lık çözümü, Facebook'un gelen postaları arama için 50 TB ve eBay'ın bütün verileri için 2 PB'lık çözümleri vardır.
- Günlük 7 TB'lık işlem hacmine sahip Twitter ve 10 TB'lık Facebook örneğindeki gibi, çok büyük verilerin depolanması ve yazılmasında ilişkisel veri tabanlarının eksik kaldığı hususlarda, yatay ölçekleme yapan dağıtık NoSQL çözümleri geliştirilmiştir.
- Veri tabanlarına ilişkin problemlerden biri olan ölçek sorununa, diğer çözümlerin içinde en iyi cevap veren NoSQL'dir.

Not:Yatay ölçekleme ve dikey ölçekleme:

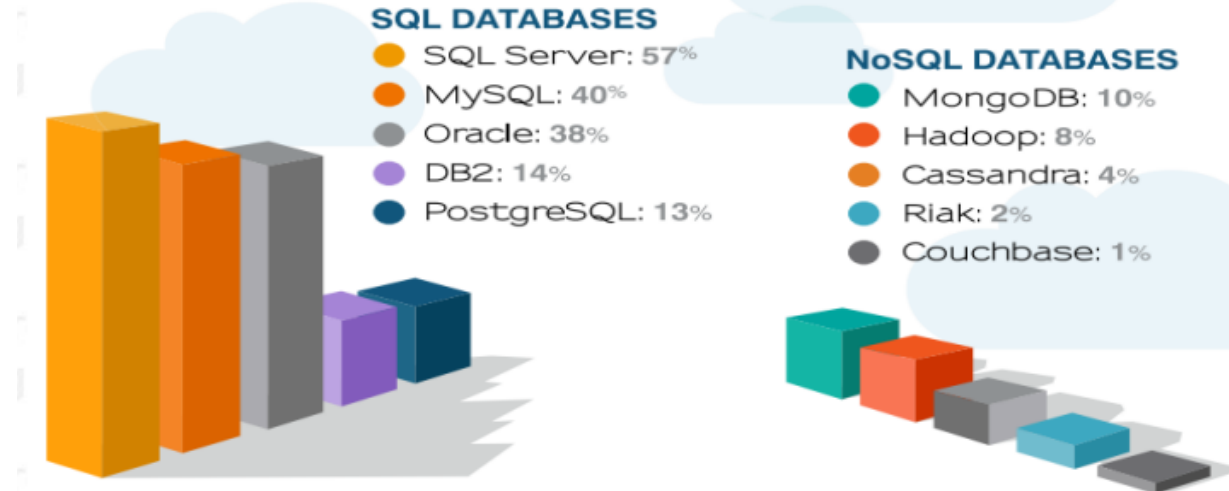
Dikey ölçeklendirme, mevcut makineye daha fazla kaynak eklemek anlamına gelmektedir.

Yatay ölçeklendirme, verileri işlemek için daha fazla makine eklemek anlamına gelir. Dikey ölçeklendirmenin uygulanması o kadar kolay değildir, öte yandan yatay ölçeklendirmenin uygulanması kolaydır.

- NoSQL veri tabanları göreceli olarak yeni bir kavram olarak gelişmektedir. Fakat e-ticaret, internet arama motorları ve sosyal ağlar gibi büyük ölçekli internet uygulamaları için güvenilirliğini kanıtlamıştır. Birçok kayıt saklama teknolojisi kendilerini NoSQL ürünü olarak sınıflandırmaktadır ve her biri kendilerine özgü karakteristiklere sahiptir.

Document Database	Graph Databases
 Couchbase  MarkLogic  mongoDB	 Neo4j  InfiniteGraph The Distributed Graph Database
Wide Column Stores	Key-Value Databases
 redis  amazon DynamoDB  AEROSPIKE  riak	 accumulo HYPERTABLE INC  Cassandra  APACHE HBASE Amazon SimpleDB

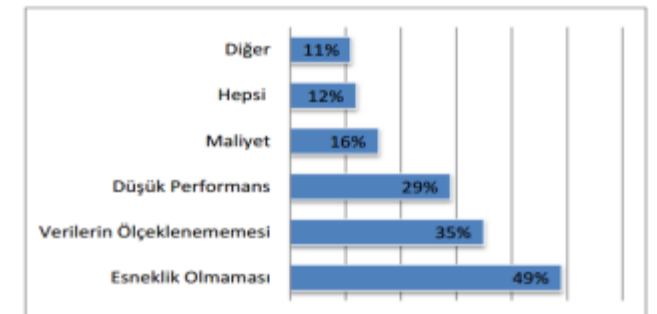
@cloudtxt <http://www.aryannava.com>



NoSQL sistemlerini genel olarak 4 grupta toplayabiliriz:

- **Döküman (Document) tabanlı:** Bu sistemlerde bir kayıt döküman olarak isimlendirilir. Verideki tüm alanlar üzerinden sorgulama imkânı sunarlar. Ayrıca farklı indeks seçenekleriyle (text, sparse, TTL, ...) sorgu performansını artırmak da mümkündür. Dökümanlar genelde JSON formatında tutulur. Bu dökümanların içerisinde sınırsız alan oluşturulabilir. Farklı formatlarda büyük verilerin saklandığı uygulamalar için uygundur. MongoDB, CouchDB, HBase, ve Amazon SimpleDB bunlara örnektir.
- **Anahtar / Değer (Key / Value) tabanlı:** Sadece anahtar üzerinden erişim sunarlar. Bazı ürünlerin ikincil indeks desteği olsa da bu destek sınırlıdır. Bu sistemlerde anahtara karşılık gelen tek bir bilgi bulunur. Yani kolon kavramı yoktur. Azure Table Storage, MemcacheDB ve Berkeley DB bunlara örnektir.
- **Grafik (Graph) tabanlı:** Veriler arasındaki ilişkiler ve bu ilişkilerin özellikleri üzerinden sorgulama imkânı sunar. Diğer sorgu modellerinde yüksek performans sunamadıkları için genel kullanımda tercih edilmezler. Diğerlerinden farklı olarak verilerin arasındaki ilişkiyi de tutan, Graph theory modelindeki sistemlerdir. BPM (Business Process Management) çözümleri ve sosyal ağ uygulamaları için uygundur. Neo4J, FlockDB bunlara örnektir.
- **Geniş Sütun tabanlı:** Anahtar – değer tabanlı veritabanları gibi veriye anahtar üzerinden erişim sağlarlar. Döküman tabanlı sistemler gibi verinin tüm alanlarından sorgulama yapmak mümkün değildir. Veriler anahtar & değer ikilisi şeklinde saklanır fakat anahtar birden fazla kolona işaret etmektedir. Çok büyük verilerin tutulduğu dağıtık sistemler için uygundur.

NoSQL TEMEL ÖZELLİKLERİ



Neden NoSQL gerekli

- **Kolay Ulaşılabilirlik (Basically Available):** Veri erişim sorunlarını ortadan kaldırmak için kopyaları kullanır ve paylaşılmış ya da bölümlenmiş veriyi birçok sunucudan alır.
- **Esnek Durum (Soft state):** ACID mantığında veri tutarlılığının olmazsa olmaz bir gereklilik olduğu savunulurdu fakat NoSQL sistemler tutarsız ve süreksiz verilerin barınmasına da izin verir.
- **Eninde sonunda Tutarlı (Eventually consistent):** Uygulamalar anlık tutarlılıkla ilgili olmasına rağmen, NoSQL sistemlerin gelecekte bir zamanda tutarlı olacağı farz edilir. ACID'in zorunlu tuttuğu tutarlılığa karşın NoSQL'de tanımlanmayan bir zamanda tutarlılığın oluşacağı garanti edilir.

	İlişkisel veritabanları	NoSQL veritabanları
En uygun iş yükleri	İşlemsel ve güçlü tutarlılığa sahip çevrimiçi işlem gerçekleştirme (OLTP) uygulamaları için tasarlanan ilişkisel veritabanları, çevrimiçi analitik işlem (OLAP) için uygundur.	NoSQL veritabanları, düşük gecikme süreli uygulamaları içeren çeşitli veri erişimi desenleri için tasarlanmıştır. NoSQL arama veritabanları, yarı yapılandırılmış veriler üzerinde analiz için tasarlanmıştır.
Veri modeli	İlişkisel model, verileri satır ve sütunlardan oluşan tablolar halinde normalleştirir. Tablolar, satırlar, sütunlar, dizinler, tablolar arasındaki ilişkiler ve diğer veritabanı öğeleri bir şema tarafından kesin bir şekilde tanımlanır. Veritabanı, tablolar arasındaki ilişkilerde başvurusal bütünlük uygular.	NoSQL veritabanları, performans ve ölçek için optimize edilmiş anahtar-değer, belge ve grafik gibi çeşitli veri modelleri sağlar.
ACID özellikleri	İlişkisel veritabanları bölünmezlik, tutarlılık, yalıtım ve dayanıklılık (ACID) özelliklerini sağlar: <ul style="list-style-type: none"> •Bölünmezlik, bir işlemin ya tamamen yürütülmesini ya da hiç yürütülmemesini gerektirir. •Tutarlılık, bir işlem gönderildiğinde verilerin veritabanı şemasına uygun olmasını gerektirir. •Yalıtım, eş zamanlı işlemlerin birbirinden bağımsız olarak yürütülmesini gerektirir. •Dayanıklılık, beklenmeyen bir sistem hatasından veya güç kesintisinden son bilinen duruma kurtarma becerisi gerektirir. 	NoSQL veritabanları yatay olarak ölçeklendirilebilen daha esnek bir veri modeli sağlamak için genellikle ilişkisel veritabanlarının bazı ACID özelliklerini esneterek bunlardan ödün verirler. Bu, tek bir bulut sunucusunun ulaşamayacağı derecede yatay ölçeklendirme gerektiren yüksek performanslı, düşük gecikme süreli kullanım örnekleri için NoSQL veritabanlarının mükemmel bir seçim olmasını sağlar.
Performans	Performans genellikle disk alt sistemine bağlıdır. En üst düzey performans için genellikle sorguların, dizinlerin ve tablo yapısının optimize edilmesi gerekir.	Performans genel olarak temel donanımın küme boyutu, ağ gecikme süresi ve çağrı yapan uygulama gibi etmenlerin birleşimine bağlıdır.
Ölçek	İlişkisel veritabanları genellikle donanımın işlem kapasitesini artırarak ölçeği artırır veya salt okunur iş yüklerine yönelik replikalar ekleyerek ölçeği genişletir.	Erişim desenleri aktarım hızını artırmak için neredeyse sınırsız ölçekte tutarlı performans sağlayan dağıtılmış mimariyi kullanarak ölçeği genişletebildiğinden, NoSQL veritabanları genellikle bölümlendirilebilen veritabanlarıdır.
API'ler	Veri depolama ve alma istekleri, yapılandırılmış sorgu diline (SQL) uygun sorgular kullanılarak iletilir. Bu sorgular ilişkisel veritabanı tarafından ayrıştırılır ve yürütülür.	Nesne tabanlı API'ler, uygulama geliştiricilerinin veri yapılarını kolayca depolamasına ve almasına imkan tanır. Bölüm anahtarları, uygulamaların anahtar-değer çiftlerini, sütun kümelerini veya seri hale getirilmiş uygulama nesneleri ve öznitelikleri içeren yarı yapılandırılmış belgeleri bulmasına imkan tanır.

- NoSQL sistemlerde tablo ve sütun kavramları bulunmamaktadır. NoSQL sistemlerde yeni bir alana ihtiyacınız olduğunda kaydı direk eklemeniz yeterli olmaktadır. Kayıtların maliyetsizce gerçekleşmesinin nedeni ise verilerin tablo ve sütunlarda saklanması yerine JSON formatına benzer yapıda saklanmasıdır.
- JSON, programlama dilinden bağımsız olan Javascript tabanlı veri değişim formatıdır. JSON'un amacı veri alışverişi yaparken daha küçük boyutlarda veri alıp göndermektir. Bu özellikleri sayesinde JSON ile çok hızlı uygulamalar oluşturabilmektedir. Aşağıdaki bulunan örnek bir JSON yapısını göstermektedir
- {
- "kitaplar": [- {
- "isim" : "Lost Symbol",
- "yazar" : "Dan Brown"
- },
- {
- "isim" : "Sherlock Holmes",
- "yazar" : "Sir Arthur Conan Doyle"
- }
-]
- }

Ne zaman bir NoSQL veritabanı seçilmemeli?

- NoSQL veritabanları tipik olarak normalize edilmemiş verilere dayanır; daha az tablo (veya kapsayıcı) kullanan ve veri ilişkileri referanslar kullanılarak modellenmeyen, daha çok gömülü kayıtlar (veya belgeler) olarak modellenen uygulama türlerini destekler. Finans, muhasebe ve kurumsal kaynak planlamadaki birçok klasik arka ofis iş uygulama yazılımı, veri anormalliklerini ve veri tekrarını önlemek için oldukça normalleştirilmiş verilere güvenir. Bunlar tipik olarak NoSQL Database için uygun olmayan uygulama yazılımı türleridir.
- NoSQL veritabanlarının diğer bir ayrımı sorgu karmaşıklığıdır. NoSQL veritabanları, tek bir tabloya yönelik sorgularla olağanüstü derecede iyi çalışır. Bununla birlikte, sorguların karmaşıklığı arttıkça, ilişkisel veritabanları daha iyi bir seçimdir. NoSQL veritabanı tipik olarak karmaşık birleştirmeler, alt sorgular ve sorguların bir WHERE (NEREDE) yan tümcesine yerleştirilmesini sağlamaz.
- Ancak bazen ilişkisel ve ilişkisel olmayan veritabanları arasında seçim yapılmasına gerek yoktur. Şirketler birçok durumda ilişkisel ve ilişkisel olmayan veri modellerinin bir kombinasyonunu kullanabilecekleri birleşik bir model sunan veritabanlarını tercih etmişlerdir. Bu hibrit yaklaşım, farklı veri türlerinin işlenmesinde artırılmış esneklik sunarken, aynı zamanda performansı düşürmeden okuma ve yazma tutarlılığı sağlar.



MongoDB

Belge Veri Tabanları

- En çok tercih edilen NoSQL veritabanı türüdür.
- Veriler, belgeler ve koleksiyonlar şeklinde yapılandırılmıştır. Belge bir PDF, Microsoft word belgesi, XML veya JSON dosyası olabilir.
- Sütunların ve veri türlerinin aksine bir belge, anahtar değer çiftlerini içerir. Her belgenin diğer belgelerle aynı yapıda olması gerekmez. Bu nedenle, ek veri eklemek için tüm veritabanının yapısını değiştirmek zorunda kalmadan daha fazla belge eklenebilir.
- Belgeler, ilişkisel bir tabloya benzer bir amaca hizmet eden koleksiyonlar halinde gruplanır.
- Belge veritabanları, belirli niteliklere sahip belge koleksiyonlarını aramak için bir sorgulama işlevi sağlar.
- Avantajlar arasında esnek veri modelleme yer alır. Belge tabanlı veritabanları ayrıca tahsilatların varlıklara (siparişler ve müşteri profilleri) göre ayrılmasını sağlar. Ancak gelişmiş sorgularla da sınırlıdır ve birleştirmelere izin vermezler.

MongoDB nedir?

- MongoDB verileri JSON biçiminde doküman olarak veritabanında saklayan NoSQL tabanlı bir veritabanıdır.
- 10gen firmasınınca 2007 yılında başlanan ve 2009 yılında AGPL lisansı ile açık kaynak projesine dönüştürülen MongoDB en popüler NoSQL yapılarından birisidir.
- MongoDB yüksek performans, yüksek kullanılabilirlik ve otomatik ölçeklendirme sağlayan bir açık kaynak belge veri tabanıdır.
- MongoDB yapı olarak dokümanları dikkate alır.
- NoSQL veri yapılarının geneli KEY ile sorgulamaya izin vermektedir. Fakat MongoDB QUERY desteği ile veriye ulaşılmasını sağlamaktadır.
- Sorgu oluşturulan alanlara performans artımını oluşturması adına secondary indekse destek vermektedir

Neden MongoDB ?

MongoDB yüksek performans sağlar . MongoDB'deki işlemlerin çoğu, ilişkisel veritabanlarına kıyasla daha hızlıdır.

MongoDB, bir arıza durumunda verileri hızlı bir şekilde kurtarmanıza olanak tanıyan otomatik çoğaltma özelliği sağlar.

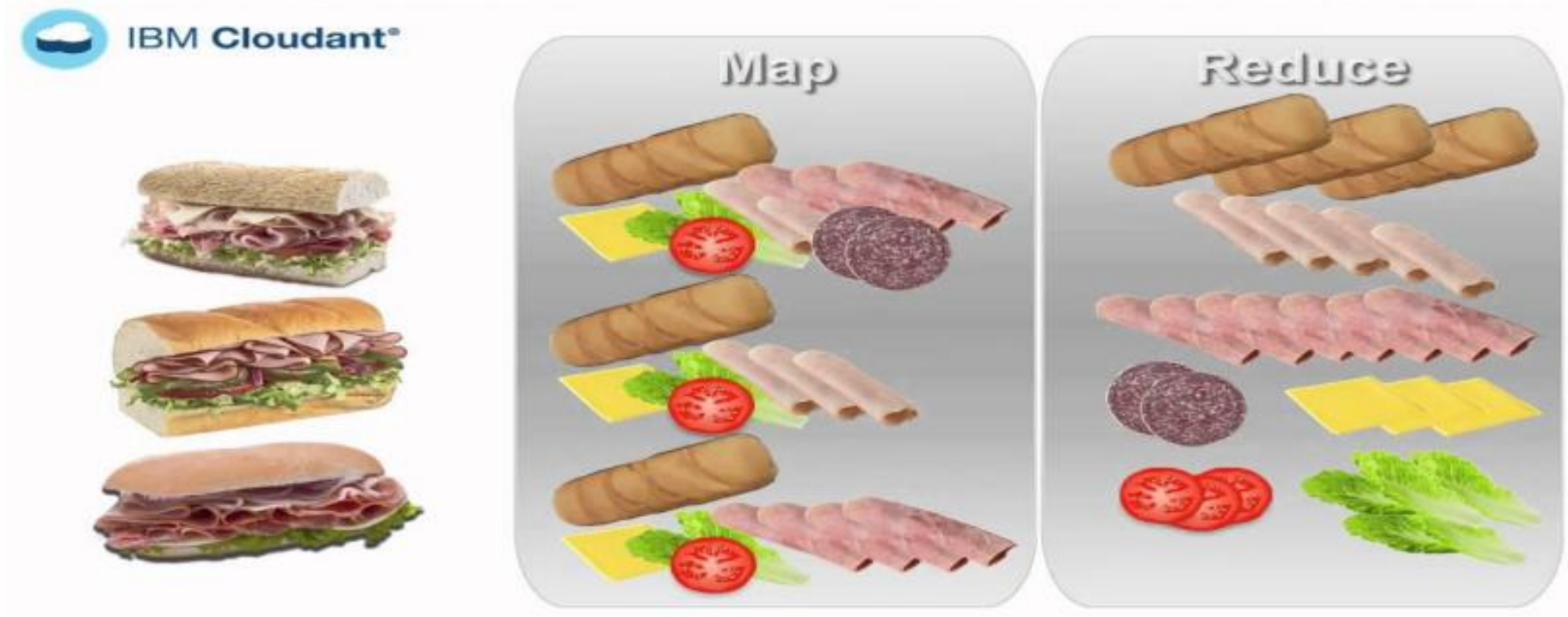
MongoDB'de paylaşım nedeniyle yatay ölçeklendirme mümkündür. Parçalama, verilerin bölümlenmesi ve verilerin sırasını koruyacak şekilde birden çok makineye yerleştirilmesi mümkündür.

Yük dengeleme : Yatay ölçeklendirme, MongoDB'nin yükü dengelemesine olanak tanır.

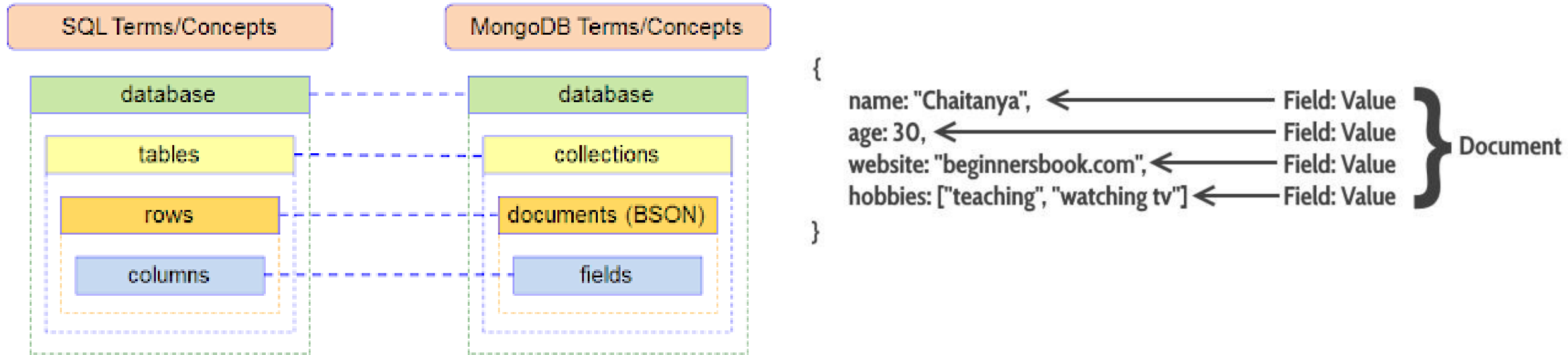
Yüksek Kullanılabilirlik : Otomatik Çoğaltma, MongoDB veritabanının kullanılabilirliğini iyileştirir.

İndeksleme: İndeks, belge içindeki tek bir alandır. Dizinler, bir MongoDB veritabanındaki her belgeyi aramak zorunda kalmadan verileri hızlı bir şekilde bulmak için kullanılır. Bu, MongoDB veritabanında gerçekleştirilen işlemlerin performansını artırır.

- MongoDB aynı zamanda MapReduce desteği sağlamaktadır. Oluşturacağınız analizleri görebilmenizi kolaylaştıracak bir sistemdir
- MapReduce ilişkisel veritabanı sistemleri ile (RDBMS) karşılaştırıldığında, Map select ifadesine ve where kısıtlarını belirlemeye, Reduce ise having, sum, count, average gibi hesaplamalar yapmamıza benzemektedir.



- MongoDB'deki koleksiyonlar , RDBMS'deki tablolara eşdeğerdir.
 - MongoDB'deki belgeler , RDBMS'deki satırlara eşdeğerdir.
 - MongoDB'deki alanlar , RDBMS'deki sütunlara eşdeğerdir.
-
- Alanlar (anahtar ve değer çiftleri) belgede, belgeler koleksiyonda ve koleksiyonlar veritabanında saklanır.



- Burada ilişkisel veritabanındaki bir tablonun MongoDB'de nasıl görüldüğünü göreceğiz. Gördüğünüz gibi, sütunlar anahtar/değer çiftleri (JSON Formatı) olarak temsil edilir, satırlar belgeler olarak temsil edilir. MongoDB her belgeye otomatik olarak benzersiz bir _id(12 bayt alan) alanı ekler, bu her belge için birincil anahtar görevi görür.
- MongoDB ile ilgili bir başka özellik, dinamik şemayı desteklemesidir; bu, bir koleksiyonun bir belgesinin 4 alana sahip olabileceği ve diğer belgenin yalnızca 3 alana sahip olabileceği anlamına gelir. İlişkisel veritabanında bu mümkün değildir.

Relational Database

Student_Id	Student_Name	Age	College
1001	Chaitanya	30	Beginnersbook
1002	Steve	29	Beginnersbook
1003	Negan	28	Beginnersbook



MongoDB

```
{
  "_id": ObjectId("....."),
  "Student_Id": 1001,
  "Student_Name": "Chaitanya",
  "Age": 30,
  "College": "Beginnersbook"
}
{
  "_id": ObjectId("....."),
  "Student_Id": 1002,
  "Student_Name": "Steve",
  "Age": 29,
  "College": "Beginnersbook"
}
{
  "_id": ObjectId("....."),
  "Student_Id": 1003,
  "Student_Name": "Negan",
  "Age": 28,
  "College": "Beginnersbook"
}
```


Yeni bir tablo nasıl oluşturulur?

SQL	MongoDB
<pre>CREATE TABLE people (id MEDIUMINT NOT NULL AUTO_INCREMENT, user_id Varchar(30), age Number, status char(1), PRIMARY KEY (id))</pre>	<pre>db.createCollection("people") db.people.insertOne({ user_id: "abc123", age: 55, status: "A" })</pre>

Yeni bir değer nasıl eklenir?

SQL	MongoDB
<pre>INSERT INTO people(user_id, age, status) VALUES ("bcd001", 45, "A")</pre>	<pre>db.people.insertOne({ user_id: "bcd001", age: 45, status: "A" })</pre>

Arama nasıl yapılır?

SQL	MongoDB
<code>SELECT *</code> <code>FROM people</code>	<code>db.people.find()</code>

Özelleştirilmiş arama nasıl yapılır?

SQL	MongoDB
<pre>SELECT id, user_id, status FROM people</pre>	<pre>db.people.find({}, { user_id: 1, status: 1 })</pre>
<pre>SELECT user_id, status FROM people</pre>	<pre>db.people.find({}, { user_id: 1, status: 1, _id: 0 })</pre>
<pre>SELECT * FROM people WHERE status = "A"</pre>	<pre>db.people.find({ status: "A" })</pre>

Güncelleme nasıl yapılır?

SQL	MongoDB
UPDATE people SET status = "C" WHERE age > 25	db.people.updateMany({ age: { \$gt: 25 } }, { \$set: { status: "C" } })
UPDATE people SET age = age + 3 WHERE status = "A"	db.people.updateMany({ status: "A" }, { \$inc: { age: 3 } })

Silme nasıl yapılır?

SQL	MongoDB
<code>DELETE FROM people WHERE status = "D"</code>	<code>db.people.deleteMany({ status: "D" })</code>
<code>DELETE FROM people</code>	<code>db.people.deleteMany({})</code>

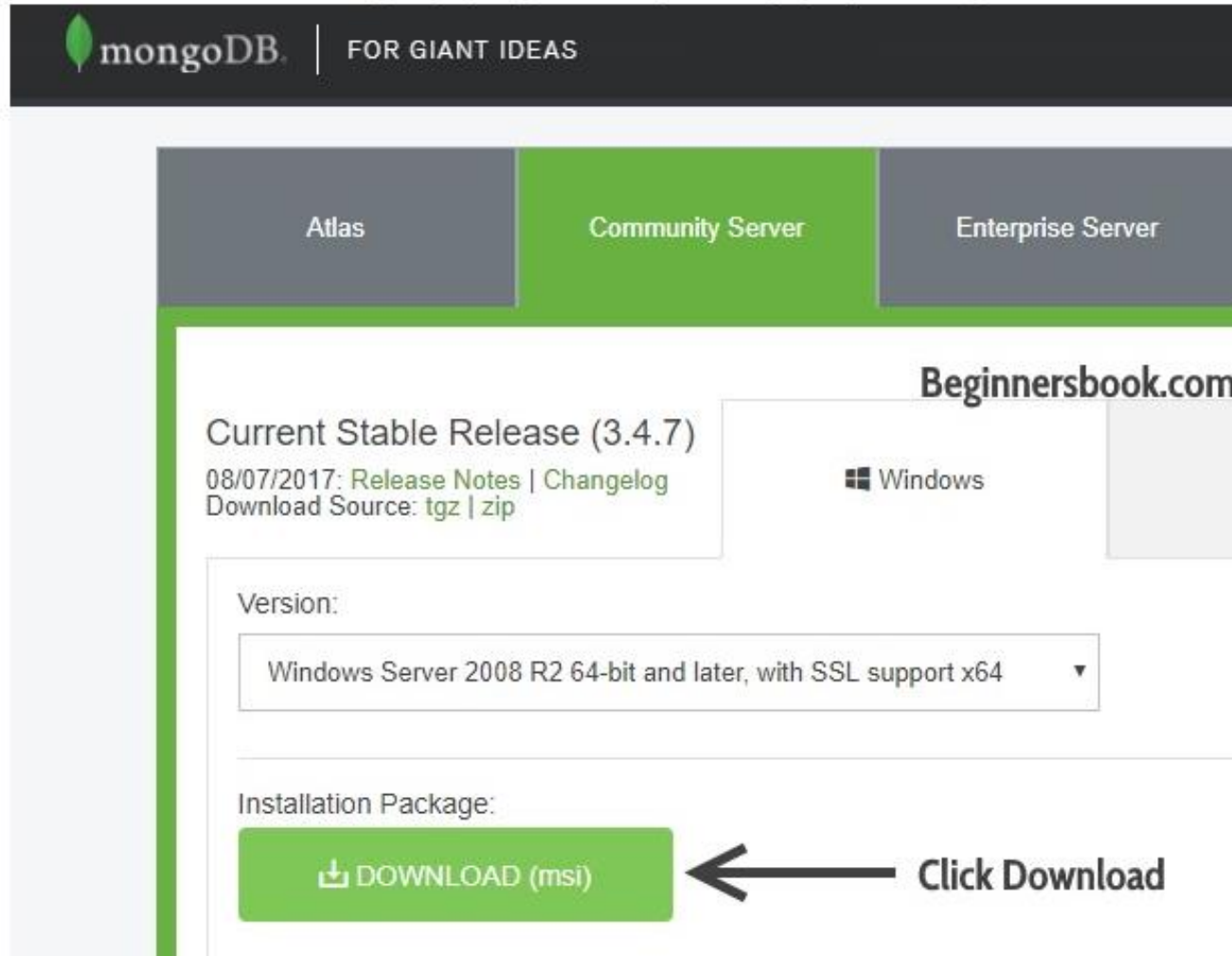
Örnek

- MongoDB’de her dokümanın benzersiz kimliklikleri olmak zorundadır. Benzersiz Kimlik alanı varsayılan olarak bu alan adı `_id` tanımlanmıştır. Eklenen document’a bir tekil anahtar atanmamışsa MongoDB ObjectId tipinde otomatik olarak benzersiz bir kimlik atar.

Soyadi	Adi	Yas
AYDIN	Onur Ekin	16
AYDIN	Efe Çınar	9
AYDIN	Gürkan	42

```
{
  "_id": ObjectId("124ega5c2b7d284dad101e4bc9"),
  "Soyadi": "AYDIN",
  "Adi": "Gürkan",
  "Yas": 42
},
{
  "_id": ObjectId("124efa8d2b7d284dad101e4bc8"),
  "Last Name": " AYDIN ",
  "First Name": "Onur Ekin",
  "Age": 16,
  "Adres": "Bulgurlu Mh. Karlıdere Cd.",
  "Sehir": "İstanbul"
},
{
  "_id": ObjectId("124efa8d2b7d284dad101e4bc9"),
  "Last Name": " AYDIN",
  "First Name": "Efe Çınar",
  "Age": 9,
  "Adres": "Bulgurlu Mh. Karlıdere Cd.",
  "Okul": "Faik Reşit Unat",
  "Sehir": "İstanbul"
}
```

1. Adım: MongoDB Community Server indirin.



The screenshot shows the MongoDB website's download page for the Community Server. The top navigation bar includes the MongoDB logo and the tagline 'FOR GIANT IDEAS'. Below this, three tabs are visible: 'Atlas', 'Community Server' (which is highlighted in green), and 'Enterprise Server'. The main content area is titled 'Beginnersbook.com' and displays the 'Current Stable Release (3.4.7)' with a date of '08/07/2017'. It provides links for 'Release Notes', 'Changelog', and 'Download Source' (tgz | zip). A 'Windows' button is also present. Under the 'Version:' section, a dropdown menu is set to 'Windows Server 2008 R2 64-bit and later, with SSL support x64'. In the 'Installation Package:' section, there is a green button labeled 'DOWNLOAD (msi)'. A large black arrow points from the text 'Click Download' to this button.

mongoDB. | FOR GIANT IDEAS

Atlas Community Server Enterprise Server

Beginnersbook.com

Current Stable Release (3.4.7)
08/07/2017: [Release Notes](#) | [Changelog](#)
Download Source: [tgz](#) | [zip](#)

Windows

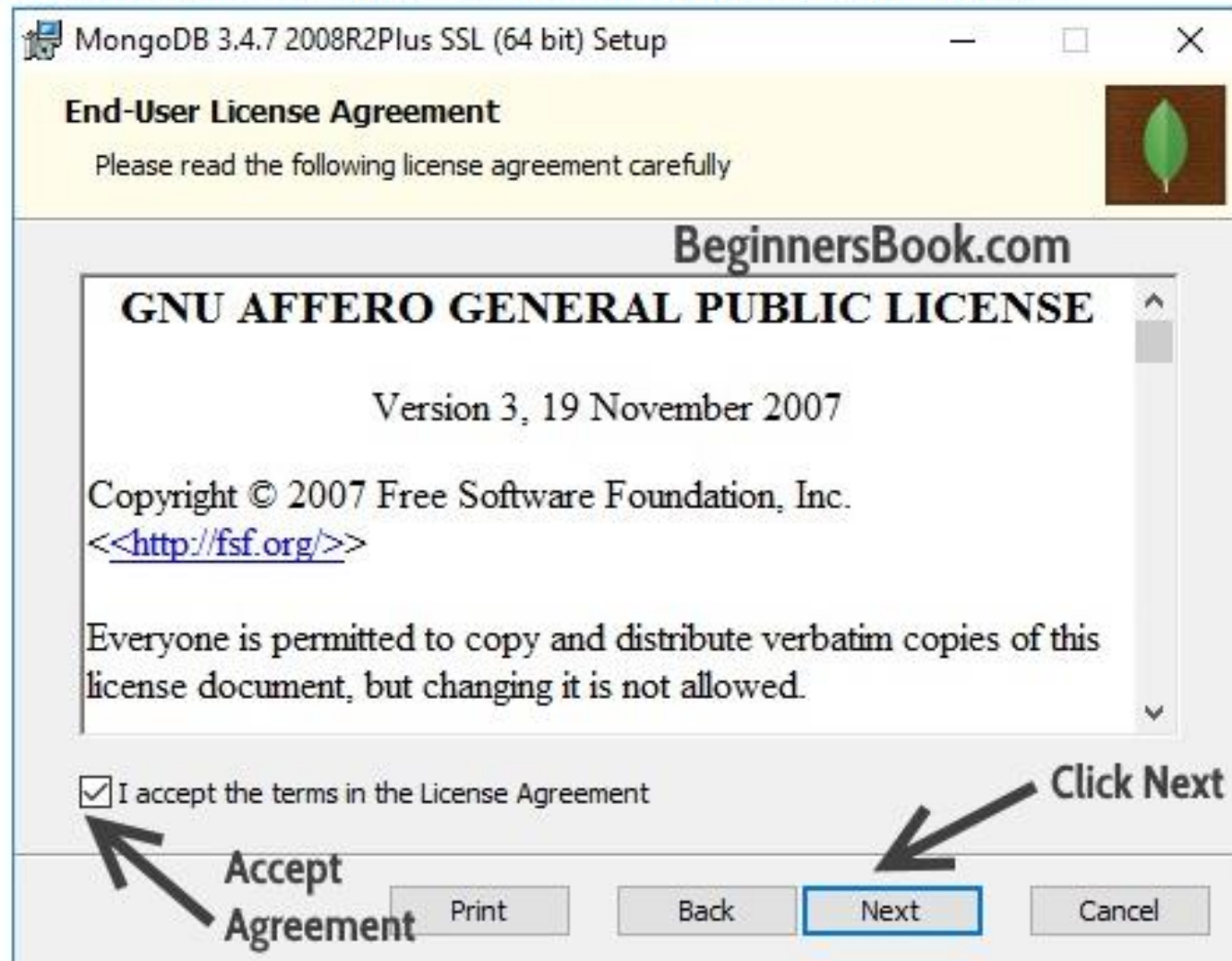
Version:
Windows Server 2008 R2 64-bit and later, with SSL support x64

Installation Package:
[DOWNLOAD \(msi\)](#) ← Click Download

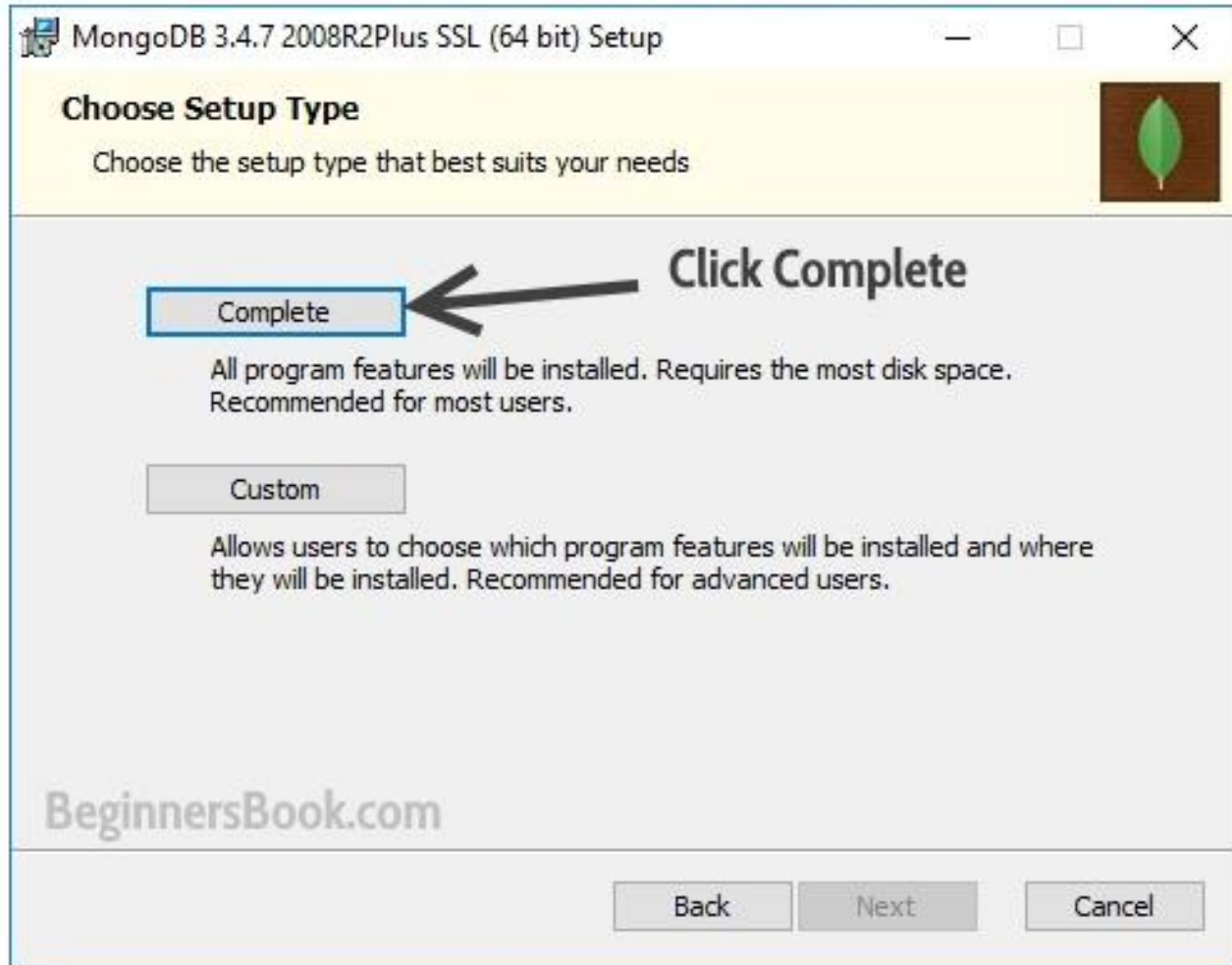
Adım 2: MongoDB kurulum pencereleri açıldığında İleri'ye tıklayın.



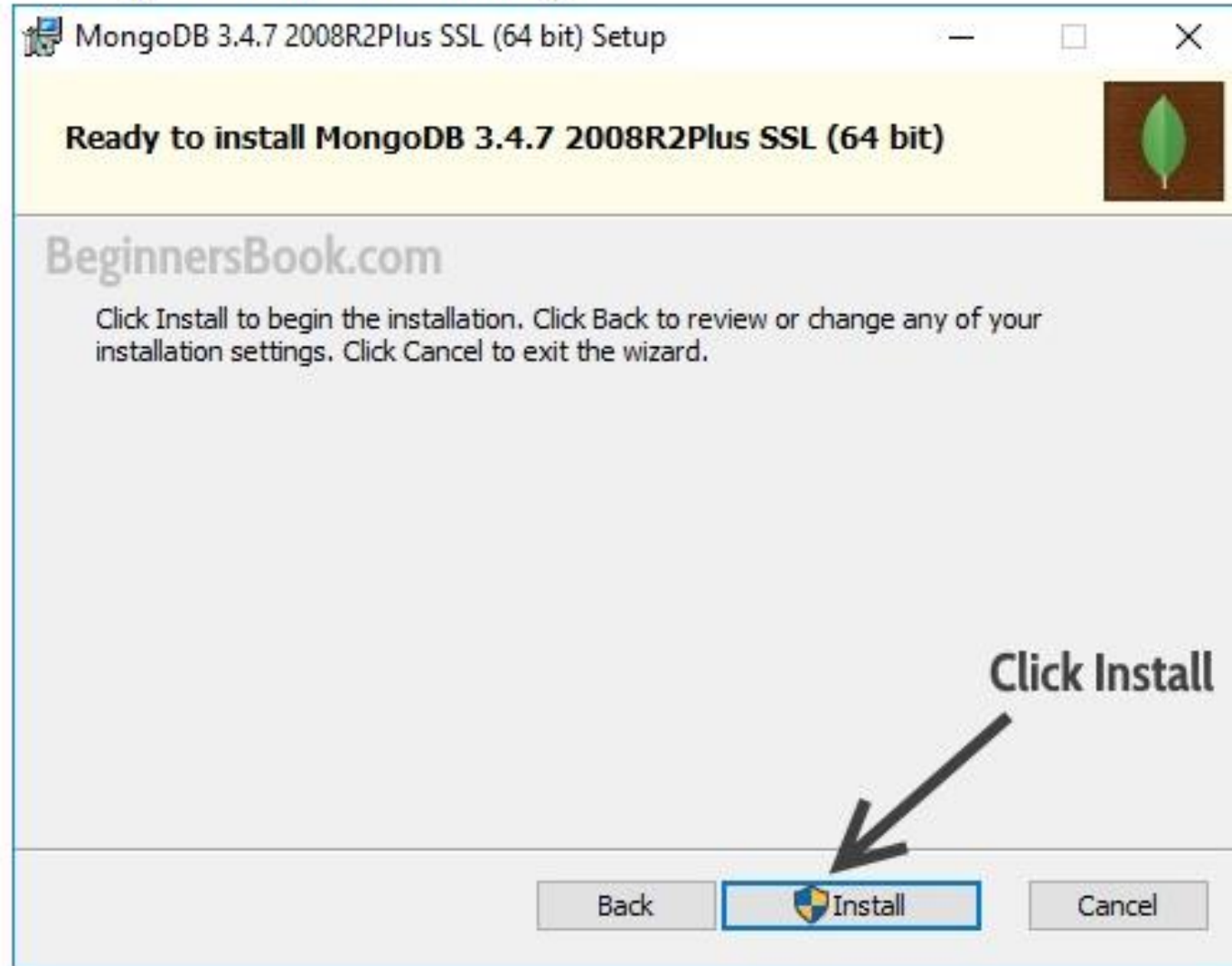
Adım 3: MongoDB kullanıcı Sözleşmesini kabul edin ve İleri'ye tıklayın.



Adım 4: Kurulum sizden Kurulum türünü seçmenizi istediğinde, Tamamla'yı seçin.



Adım 5: Kurulumu başlamak için Kur'a tıklayın.



MongoDB'yi başlatmak için cmd'den aşağıdaki kod satırını çalıştırın.

```
"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db"
```

Mongodb Çoklu Dil Desteği

- MongoDB'nin en önemli özelliklerinden biri, çoklu dil desteğidir. Python, PHP, Ruby, Node.js, C ++, Scala, JavaScript ve çok daha fazlası dahil olmak üzere popüler programlama dilleri için sürücü desteği sağlamaktadır.
- Bu programlama dilleri arasında en çok tercih edilenler ise Node.JS ve PHP dilleridir.

Node JS ile MongoDB kullanımı

- Bilgisayarınızda MongoDB mevcutsa bağlantıyı direkt lokal olarak sağlayabilirsiniz.
- Örneğimizde MongoDB yani belge odaklı bir veritabanı kullanmak için zaman mongoose adlı kütüphaneyi kullanmaktayız.

- Mongoose'u kuralım,

```
yarn add mongoose
```

- Kodumuzda mongoose'u çağıralım.

```
const mongoose = require('mongoose')
```

- Örnekte basit kullanıcı bilgilerini tutmak için `mongoose.Schema` metodu ile dökümanımıza kaydedeceğimiz satırların tiplerini ve özelliklerini belirtmemiz gerekir.

```
const Schema = new mongoose.Schema({
  username: { type: String, unique: true, required: true }, // username kısmı tekil ve zorunlu
  name: { type: String, unique: false, required: true }, // name kısmı tekil olmayan fakat zorunlu
  age: Number, // age kısmı ise sadece number tipinde olmalı, zorunlu veya tekil değil.
});
/*
Yukarıdaki örnekte basit bir schema (şema) çıkarttık.
*/
```

- Yukarıda hazırladığımız şemayı bir dökümana bağlayalım.

```
const Users = mongoose.model('Users', Schema); // Artık dökümanımızın metodlarına Users.find() gibi erişeceğiz.
```

- Veritabanımıza bağlanmak için `mongoose.connect` metodunu kullanacağız.

```
mongoose.connect('MONGODB_URI', err => console.log(err ? err : 'Mongo connected.'));
```


MongoDB—Node JS veri ekleme işlemi

```
const obj = {  
  username: 'veritabanıcılar',  
  name: 'Özgür',  
  age: 40  
}
```

`const user = new Users(obj)` // Yeni bir kullanıcı satırı oluşturalım.

`user.save((err, doc) => {` // Yeni oluşturduğumuz satırı işleyelim.

```
  if (err) {  
    console.error(err)  
  } else {  
    console.log(doc)  
  }  
})
```

MongoDB—Node JS veri arama işlemi

```
const username = 'Özgür'  
Users.findOne({username}, (err, doc) => {  
  if (err) {  
    console.error(err)  
  } else {  
    console.log(doc)  
  }  
})
```

MongoDB—Node JS güncelleme işlemi

`const id = '1234567abcd'` // Bu kısım veritabanından okunacaktır.

```
Users.findById(id, (err, user) => {  
  if (err) return handleError(err)
```

```
  user.age = 40;  
  user.save((err, updatedUser) => {  
    if (err) return handleError(err)  
    console.log(updatedUser)  
  })  
})
```

PHP ile MongoDB Kullanımı

- MongoDB sunucusuna bağlanabilmek için php'ye mongodb sürücüsü
- Eğer Linux kullanıyorsanız <https://github.com/mongodb/mongo-php-driver> github üzerinden kurulumu tamamlayabilirsiniz.
- Windows ortamında MongoDB ile çalışmak istiyorsanız öncelikle <https://github.com/mongodb/mongo-php-driver/downloads> adresinden çalışacağınız ortamın php sürümü, 32 / 64 bit gibi özelliklerini dikkate alarak dll dosyalarını indirin. İndirdiğiniz dll dosyaları arasında size uygun olanı seçin.
- Ardından seçtiğiniz dosyayı Php sürücülerinin bulunduğu dizine **php_mongo.dll** ismiyle kopyalayın. Son olarakta php.ini dosyanıza;
- **extension=php_mongo.dll** satırını ekleyin. Web sunucunuzu yeniden başlattığınızda mongodb sürücüsünün php ile kullanıma hazır hale gelecektir.

PHP-MongoDB sunucusuna bağlanma ve Veritabanı seçimi

```
<?php
try {
// Mongo Sunucusuna bağlanalım
$mongo = new MongoClient('mongodb://127.0.0.1:27017');
// Veritabanını Seçelim
$db = $mongo->selectDB('TestDb');
} catch(MongoConnectionException $e) {
die('Baglanti Kurulamadi : ' . $e->getMessage());
}
?>
```

PHP-MongoDB İle Kayıtları Listeleme

Uyeler collection'ı içerisinde yer alan kayıtları listeleyelim.

```
<?php
// Üyeler isimli Collection'ı Seçelim
$uyeler = new MongoClient($db, 'uyeler');
// Toplam Üye Sayısını Ekrana Basalım
printf('<p>Toplam Uye : %s</p>',$uyeler->count());
// Tüm uyeleri çekelim.
$uyeListesi = $uyeler->find();
// Çektiğimiz kayıtları listeleyelim.
foreach($uyeListesi as $uye) {
printf('Nick : %s | Web : %s <br>', $uye['nick'], $uye['web']);
}
?>
```

PHP-MongoDB İle Kayıt Ekleme

Varolan Collection içerisine yeni kayıtlar eklemek için;

```
<?php
// Üye bilgilerini tanımlıyorum
$yeniUye = array(
    'nick' => 'AHMETNACI',
    'web' => 'www.ahmetnaci.com'
);
try {
    $uyeler->insert($yeniUye);
} catch(MongoCursorException $e) {
    die('Yeni uye eklenirken teknik bir sorun olustu! > ' . $e->getMessage());
}
?>
```

PHP-MongoDB İle Kayıt Güncelleme

Collection içerisinde kaydı güncellemek istiyorsanız;

```
<?php
```

```
// Güncelleyeceğim kaydın koşulunu belirliyorum
```

```
$where = array('nick' => 'ahmetnaci');
```

```
// Güncellenecek alanını belirliyorum
```

```
$yeniBilgi = array('nick' => 'naci');
```

```
// Kaydı değiştiriyorum
```

```
$uyeler->update($where, $yeniBilgi);
```

```
?>
```

Mongodb İle Kayıt Silme

Varolan Collection içerisinde kaydı silmek istiyorsanız;

```
<?php
// Sileceğim kaydın koşullarını belirliyorum
$where = array('nick' => 'naci');
// Kaydı collection içerisinde siliyorum
$uyeler->remove($where);
?>
```

Kaynaklar

- <https://dergipark.org.tr/tr/download/article-file/297870>
- <https://www.oracle.com/tr/database/nosql/what-is-nosql/>
- <https://devveri.com/nosql-nedir>
- <https://www.ahmetcevahircinar.com.tr/wp-content/uploads/2017/11/nosql.pdf>
- <https://account.mongodb.com>
- <https://beginnersbook.com/2017/09/mapping-relational-databases-to-mongodb/>
- <https://cagatay.me/node-js-ile-veritaban%C4%B1-i%C5%9Flemleri-mongodb-9-1-76105f5e23a6>
- <https://www.cinarwbh.com/genel/php-ile-mongodb-kullanimi/>

İYİ ÇALIŞMALAR...