NoSql MongoDb

NoSQL Kavramı ve MongoDb Temelleri

NoSQL Nedir?

- NoSQL Veritabanı, sabit bir şema gerektirmeyen, ilişkisel olmayan bir Veri Yönetim Sistemidir. Birleştirmeleri önler ve ölçeklendirilmesi kolaydır.
- NoSQL veri tabanları sabit bir şemaya dayanmaz , bunun yerine esnek ve dinamik veri yapılarına izin verir. Bu esneklik, NoSQL veri tabanlarını sosyal medya gönderileri, günlük dosyaları ve sensör verileri gibi büyük hacimli yapılandırılmamış veya yarı yapılandırılmış verileri işlemek için çok uygun hale getirir.

NoSQL Nedir?

NoSQLilişkisel (RDBMS) (RelationalDatabase Management System) veri tabanlarına alternatif olarak çıkan, nesneyi dikkate alan SQL'deki gibi belirli kolonlara ihtiyaç duymayan veri depolama yaklaşımıdır.

NoSQLdendiğinde bilinen Hadoop, MongoDB, ElasticSearchvb. gibi birçok marka bulunmaktadır.

NoSQL Nedir? (Yanlış Bilinenler)

- NoSQL, “SQL yok” anlamına gelmez
- Açılımı: Not Only SQL (Sadece SQL değil)
- İlişkisel veritabanlarına alternatif değil, farklı bir yaklaşımdır
- NoSQL veritabanları ilişkisel model kurallarına uymaz
  - Hiçbir zaman düz sabit sütunlu kayıtlar içeren tablolar sağlamaz
  - Bağımsız kümeler veya BLOB'larla çalışma
  - Nesne-ilişkisel haritalama ve veri normalleştirme gerektirmez
  - Sorgu dilleri, sorgu planlayıcıları, referans bütünlüğü bağlantıları, ACID gibi karmaşık özellikler yok
-

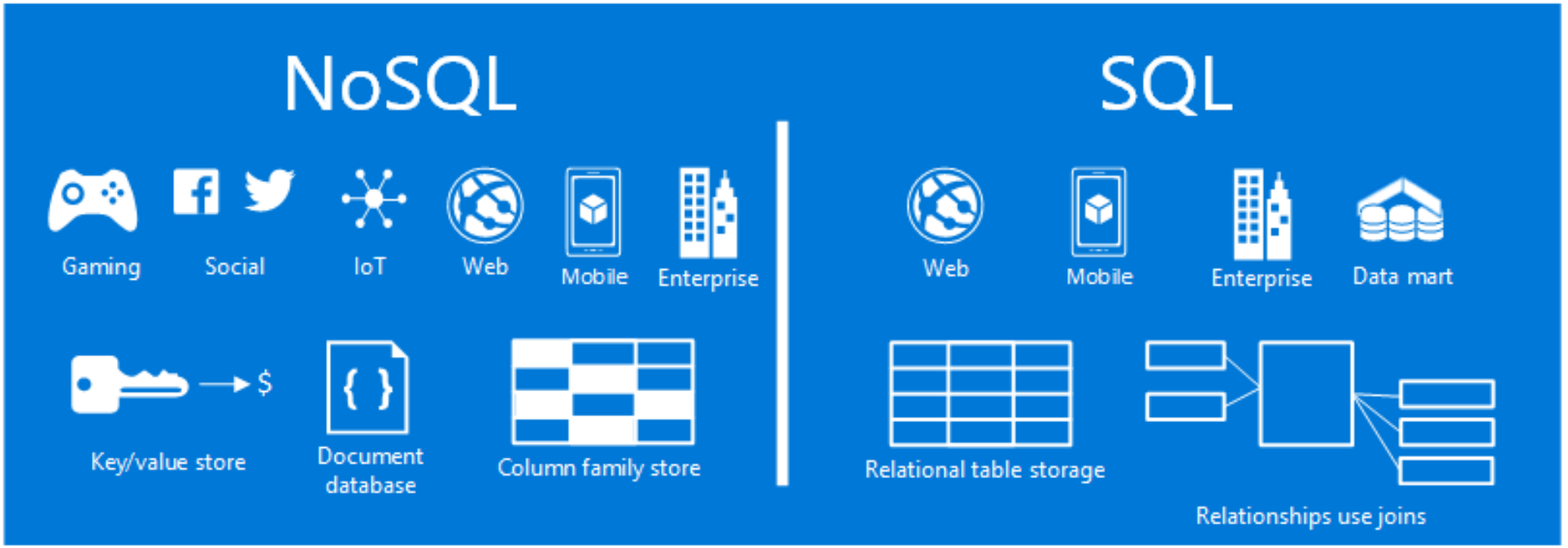
NoSQL Tarihçe

- 1998- Carlo Strozzi, hafif, açık kaynaklı ilişkisel veritabanı için NoSQL terimini kullandı
- 2000- Graf tabanlı veritabanı Neo4j başlatıldı
- 2004- Google BigTable kullanıma sunuldu
- 2005- CouchDB başlatıldı
- 2007- Araştırma makalesi Amazon Dinamo piyasaya sürüldü
- 2008- Facebook'un kaynakları açık Cassandra projesi
- 2009- NoSQL terimi yeniden kullanılmaya başlandı

NoSql Neden Ortaya Çıktı?

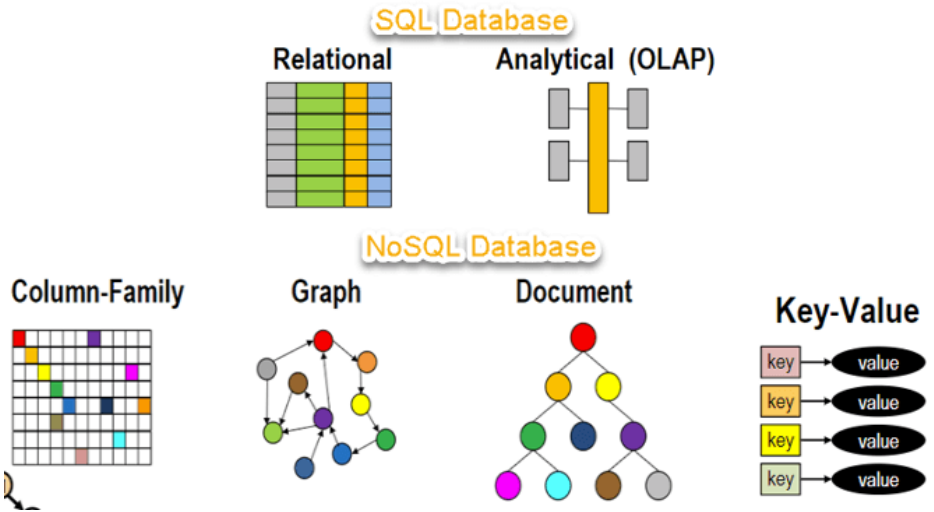
- Yoğun kullanıcı trafiği
- Yüksek işlem hacmi
- Artan maliyet ve performans ihtiyacı
- RDBMS’lerin bu senaryolarda zorlanması
- Organik veri yapısıyla veri saklama
- Yatay ölçekleme ihtiyacı

NoSql Ne- SQL


NoSQL
Gaming
Social
IoT
Web
Mobile
Enterprise
Key/value store
Document database
Column family store
SQL
Web
Mobile
Enterprise
Data mart
Relational table storage
Relationships use joins

Yaygın Yanlış Algı

- NoSQL = Yeni bir veritabanı ✗
- NoSQL = Sadece büyük veri ✗
- Gerçekte:
  - NoSQL bir veritabanı değil
  - Bir sistem, yaklaşım ve yöntemdir
  - Bu yaklaşımı kullanan farklı sistemler geliştirilmiştir



Büyük Veri Durumu

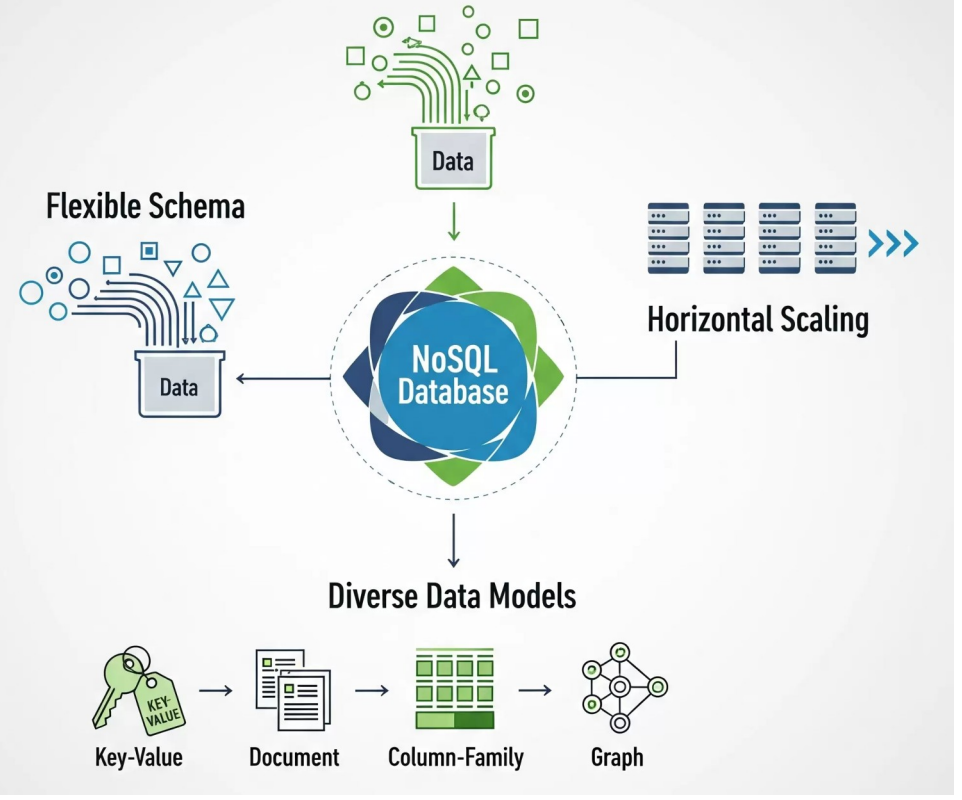
- Hız faktörü → NoSQL = Büyük veri algısı
  - Oysa:Günümüzde RDBMS'ler de milyarlarca kaydı yönetebiliyor
- NoSQL'in asıl gücü:
  - Şemasız (schema-less) yapı
  - Yatay ölçekleme
  - Normalize olmayan veriye hızlı erişim

.

NoSql Nasıl Bir Yaklaşımdır

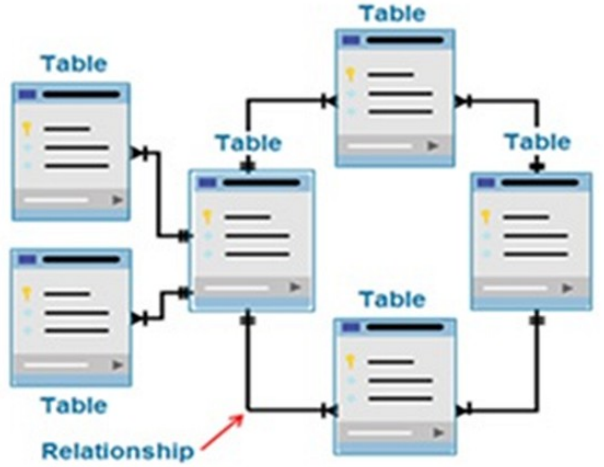
- Esnek şema yapısı
- Farklı veri saklama yaklaşımları (key-value, graph, document, column)
- Yatay ölçekleme
- Yüksek performans
- Normalize olmayan veri
- İlişkiler yok

.



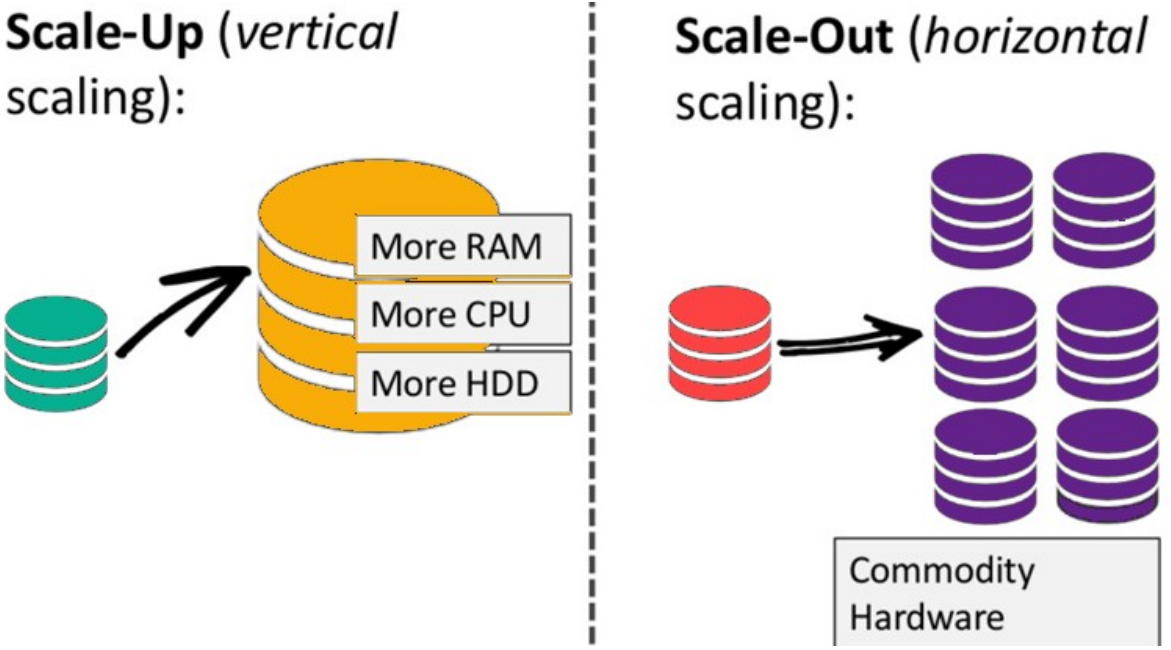
RDBMS Yapısı

- Önceden tanımlanmış
  - Tablolar
  - Kolonlar
  - Veri tipleri
- Tablolar arası katı ilişkiler
- Veri bütünlüğü ön plandadır



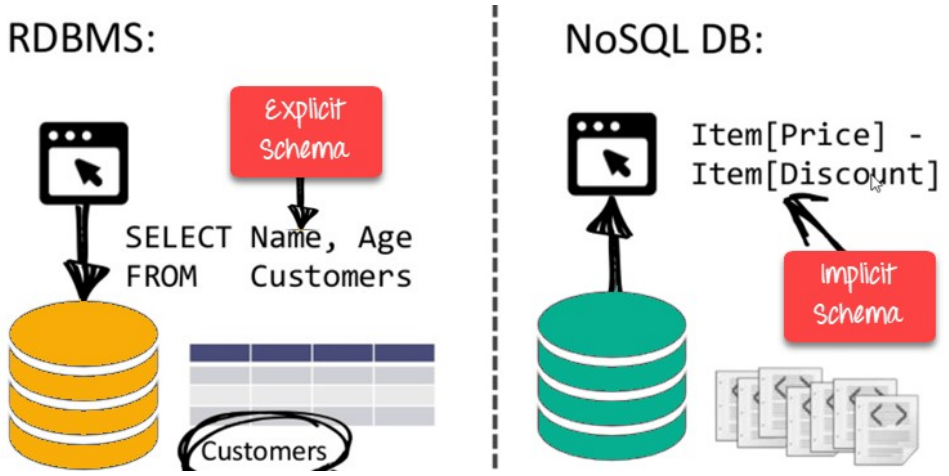
NoSql Yapısı

- Key / Value mantığı
- Veriler arasında fiziksel ilişki yok
- Bu sayede:
  - Daha hızlı erişim
  - Daha esnek yapı
- Yatay ölçekleme
- Esnek şema yapısı
- İlişki yok
- Veri bütünlüğü zamanla oluşur



Organik Veri Kavramı

- NoSQL = Organik veri
- Veri yapısı:
- Süreç içinde değişebilir
- Önceden kesin sınırlarla belirlenmez
- Bu yönüyle dinamik sistemler için uygundur



Şema Zorunluluğu Karşılaştırması

- RDBMS Şema değişirse:
  - Hata riski
  - Migrasyon gereksinimi
- NoSQL Şema Değişirse:
  - Şema zorunluluğu yok
- Yeni veri yapıları:
  - NoSql Eski verilerden bağımsız eklenebilir
  - RDBMS tasarım yeniden düzenlenir

.

Veri Saklama Yapısı

- RDBMS
  - Tablolar
  - Satır / Sütun yapısı
- NoSQL
  - Document (Döküman)
  - JSON tabanlı veri saklama

.

Esneklik Avantajı

- RDBMS
  - Tüm satırlar aynı kolonları uymalı
- NoSQL
  - Her doküman farklı alanlara sahip olabilir
  - Kolon ekleme/çıkarma serbesttir
  - Tüm verileri etkilemez

.

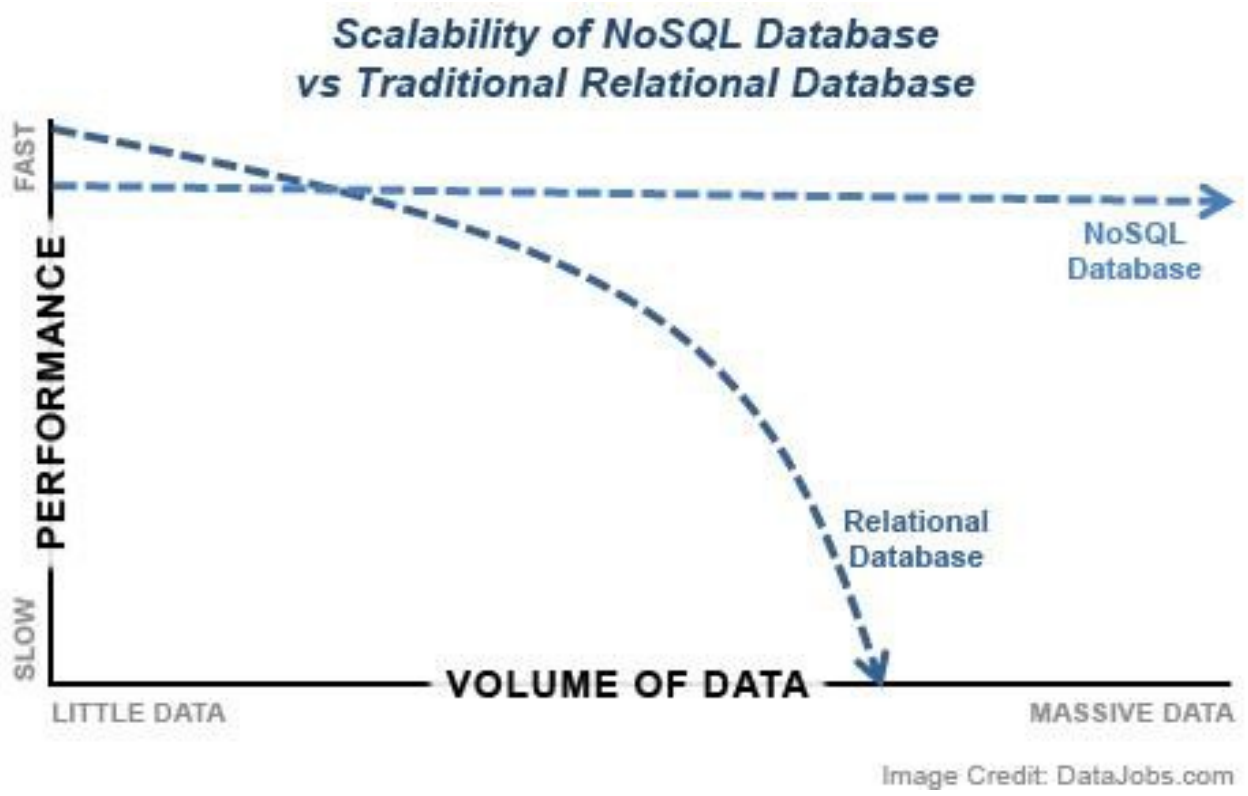
Performans ve Maliyet

- NoSQL Sistemleri:
  - Yüksek erişilebilirlik
  - Daha az maliyet
  - Daha yüksek performans
  - Dağıtık mimariye uygundur

Ölçeklenebilirlik

- RDBMS
  - Dikey ölçekleme
  - Daha güçlü donanım
- NoSQL
  - Yatay ölçekleme
  - Daha fazla sunucu
  - Dağıtık sistem avantajı

NoSQL RDBMS Karşılaştırma



Büyük Veri

Büyük veri, tek bir sunucu üzerinden yönetilemeyen, çok sayıda kullanıcının eş zamanlı olarak işlem yaptığı verilerdir.

Bir veriye, büyük veri diyebilmek için üç parametreye bakılır:

Volume: Verinin hacimsel büyüklüğü

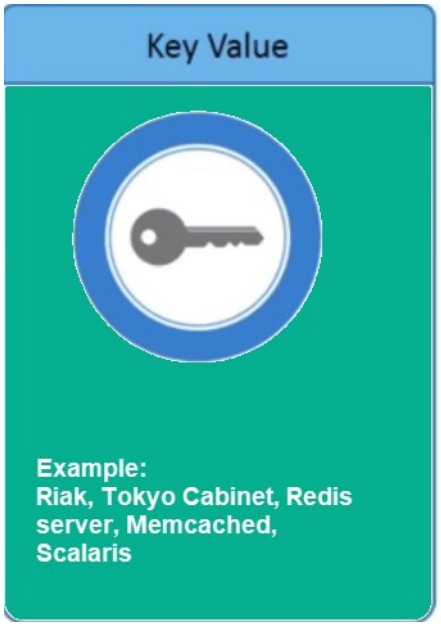
Velocity: Veri akışının hızı (sensörverileri, finansal işlemler, uygulama kullanım bilgileri -logkayıtları-gibi)

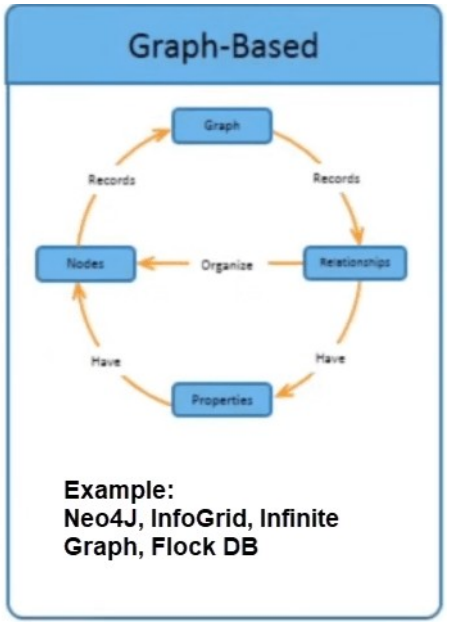
Variety: Verinin farklı formatlarda gelebilmesi

NoSQLveritabanları, bu üç özelliğe sahip verilerin işlenmesini sağlayan dağıtık, ölçeklendirilebilir ve yüksek erişime (highavailability) sahip veritabanlarıdır.

NoSQL Sistem Yaklaşımları

NoSQL sistemleri verileri dört kategoriden biri şeklinde saklar

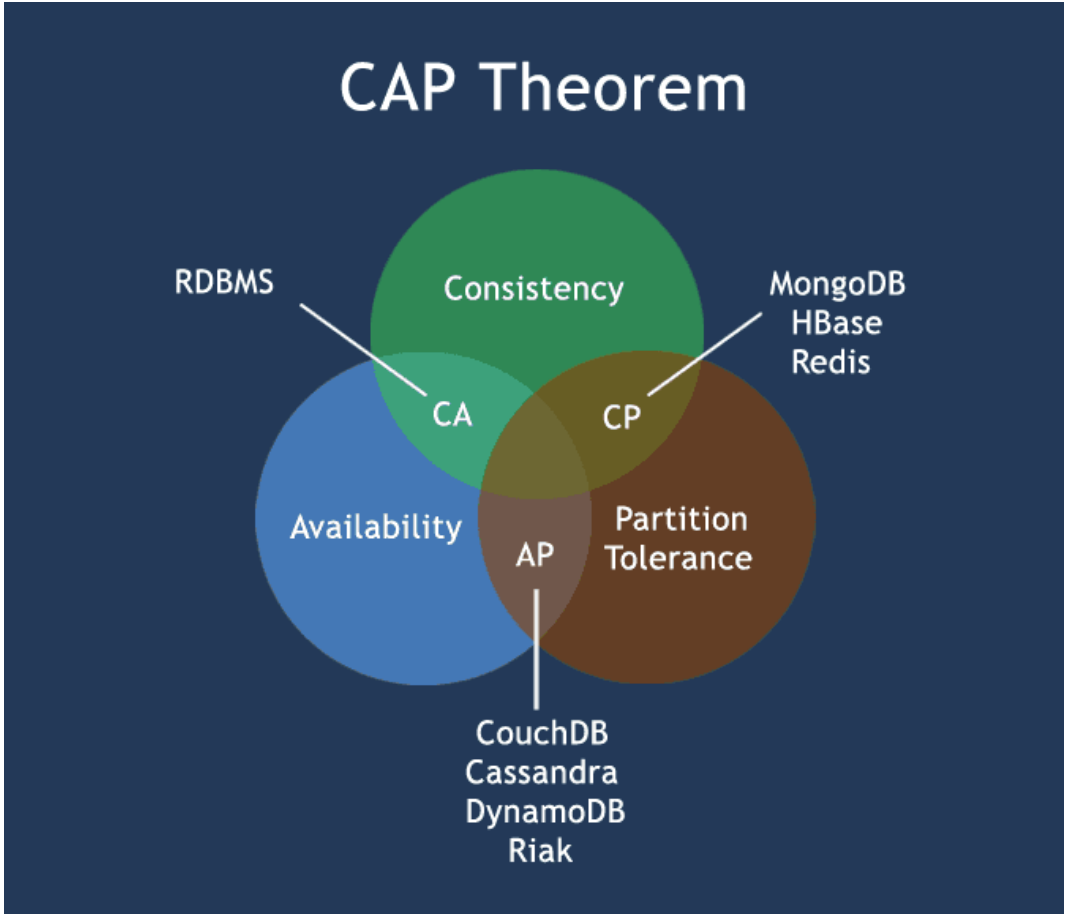




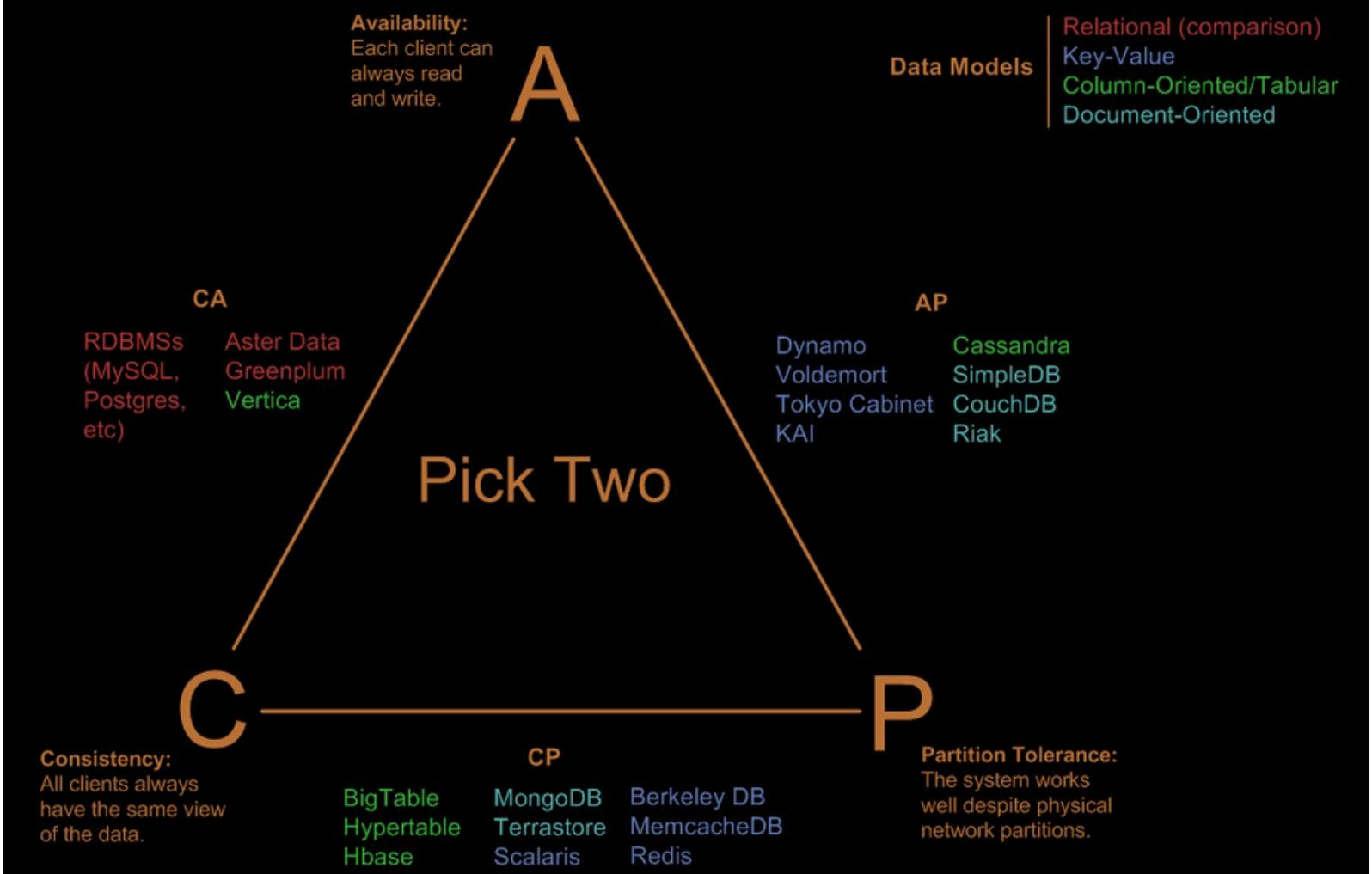
CAP Teoremi

- CAP teorisine göre herhangi dağıtık bir sistem aynı anda Tutarlılık (Consistency), Müsaitlik(Availability) ve Parçalanma payı (Partitiontolerance) kavramlarının hepsini birden asla sağlayamaz.
- Yani dağıtık bir sistemin her bir parçasından aynı veriye erişebilme, aynı anda bütün isteklere cevap verebilme, kaybedilen paketlere rağmen verinin bütününü kaybetmeme özelliklerine aynı anda sahip olamaz.
- Bunlardan mutlaka birinde problem çıkacaktır.
- NoSQLsistemlerde ise bu kavramlar esnetilerek yatayda büyüme ile performans kazancı amaçlanmıştır. Örneğin veriyi bölerek, belirli sayıdaki kopyalarını da dağıtık sistemin farkı parçalarına göndererek tutarlılık sağlanabilir.
- Bu sayede paket kaybı yaşansa da verinin tamamı kaybedilmez, aynı zamanda veri bölündüğü için her bir parçaya düşen yük dengelenmiş olur.

CAP Teoremi


CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
Partition
Tolerance
AP
CouchDB
Cassandra
DynamoDB
Riak

Visual Guide to NoSQL Systems



CAP

Not defterinizdeki müşteri sayfasına bilgi kaydedilir.

•Sorun: Müşteriler çoğalır, telefonda beklemesüresi artar. Memnuniyetsizlik. Hasta olunca o gün iş yapamıyorsunuz.

•Çözüm: Eşinizi de işe alıyorsunuz. Bir hat daha alıyorsunuz. Tek numarayı iki hatta yönlendiriyorsunuz. 2 farklı not defteri ile kayıt işi sürüyor.

•Sorun: Birinizdeki kayıt diğerinde yok! Dağıtık sisteminiz tutarlıdeğil!

•Çözüm: Yeni bilgi girildikçe karşılıklı güncelleme yapılacak.

•Sorun: Birisi işe gelemezse?

•Çözüm: Tutarlı ve ulaşılabilir olmak için o gün olmayana e-posta atıp, işe başlamadan yeni bilgileri güncellemesi istenecek.

•Sorun: Biriniz diğerini güncellemediği durumda ne olur?

•Sistem tutarlılık ve uygunlukaçısından çalışıyor olsa da bölünebilme toleransı bakımından zayıf kalıyor.

NoSQL Sistem Yaklaşımları

Sütun Tabanlı Sistemler

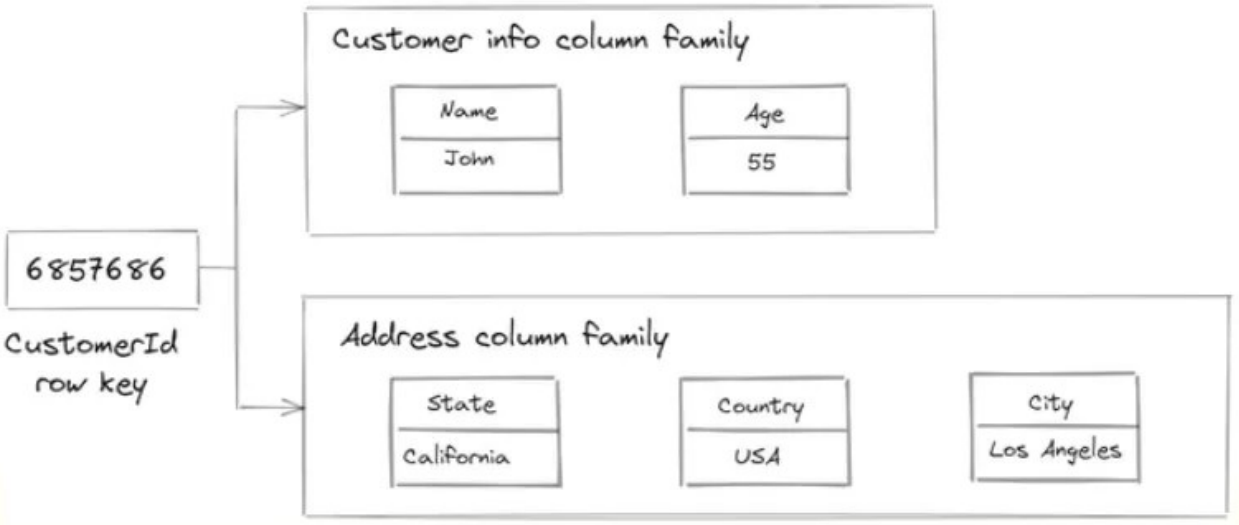
ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

- Veri tabanı sütunlar üzerinde çalışır
- Google Bigtable
- Her sütun ayrı ayrı ele alınır
- Veriler tek bir sutünda kolayca bulunabildiğinden SUM, COUNT, AVG, MIN vb. gibi toplama sorgularında yüksek performans sağlarlar.
- Veri ambarlarını yönetmek için yaygın kullanılır
- HBase, Cassandra, Hypertable, BigTable örnek sistemlerdir

Sütun Tabanlı Yaklaşım

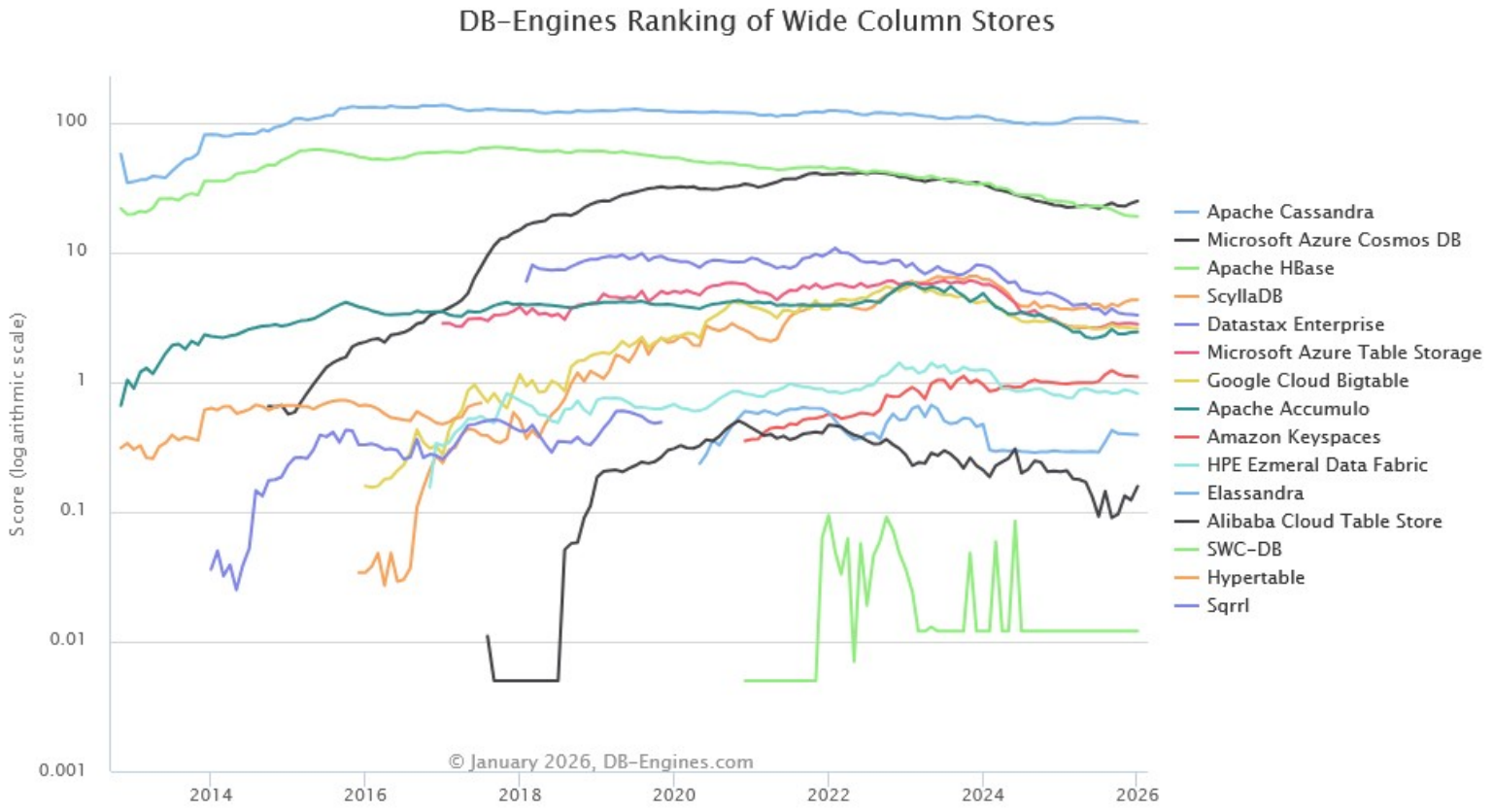
CustomerId	Name	Age	City	State	Country
6857686	John	55	Los Angeles	California	USA

RELATIONAL DATABASE



WIDE-COLUMN DATABASE

Sütun Tabanlı NoSql Veri Tabanları Eğilimleri



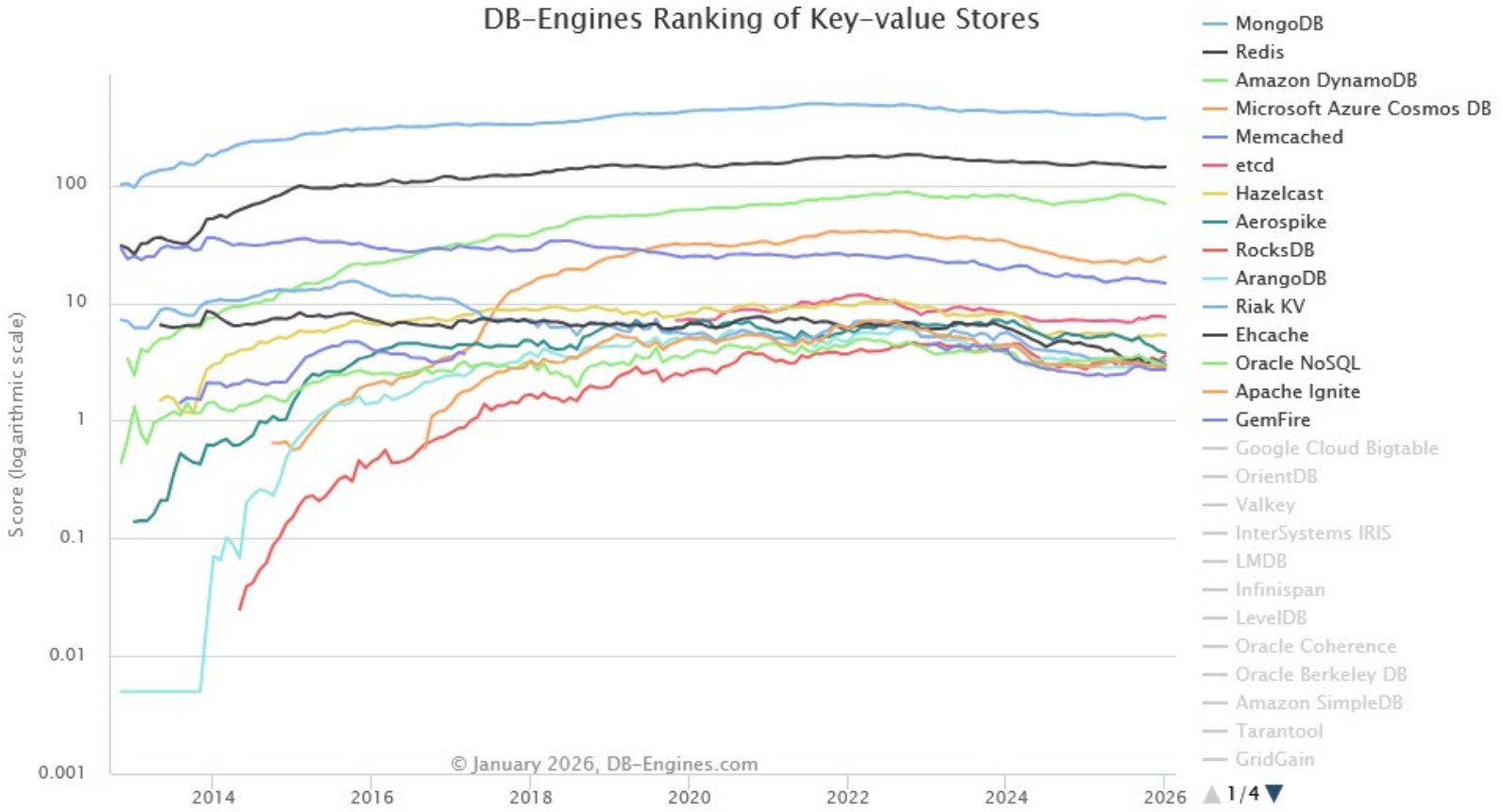
NoSQL Sistem Yaklaşımları

Key-Value Tabanlı Sistemler

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

- Veri tabanı sütunlar üzerinde çalışır
- Google Bigtable
- Her sütun ayrı ayrı ele alınır
- Veriler tek bir sutünda kolayca bulunabildiğinden SUM, COUNT, AVG, MIN vb. gibi toplama sorgularında yüksek performans sağlarlar.
- Veri ambarlarını yönetmek için yaygın kullanılır
- HBase, Cassandra, Hypertable, BigTable örnek sistemlerdir

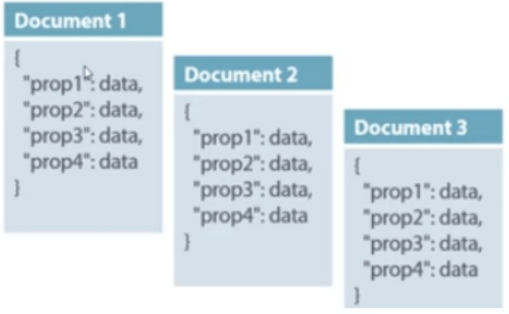
Key-Value Veri Tabanları Eğilimleri



NoSQL Sistem Yaklaşımları

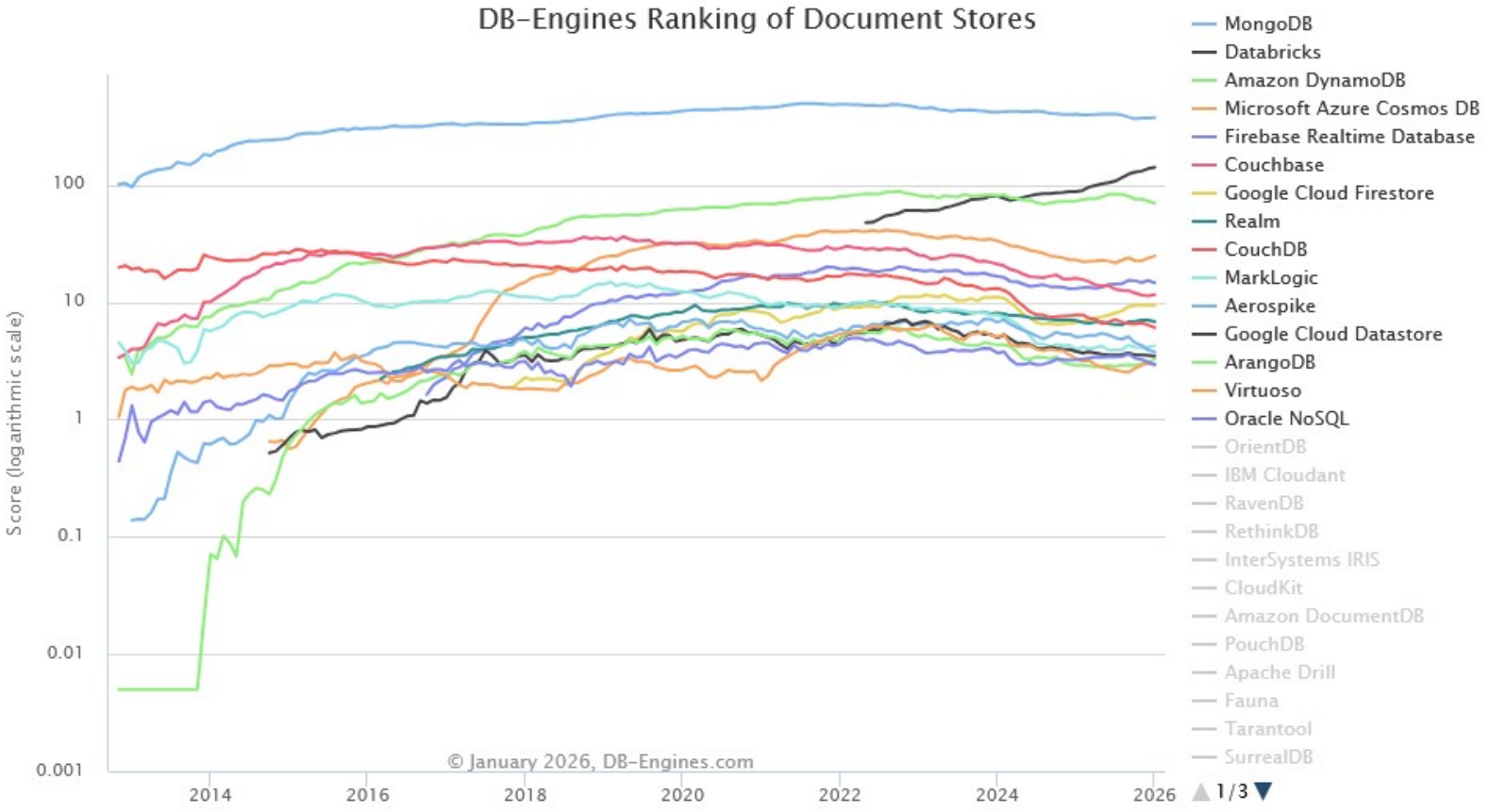
Doküman(Belge) Tabanlı Sistemler

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data



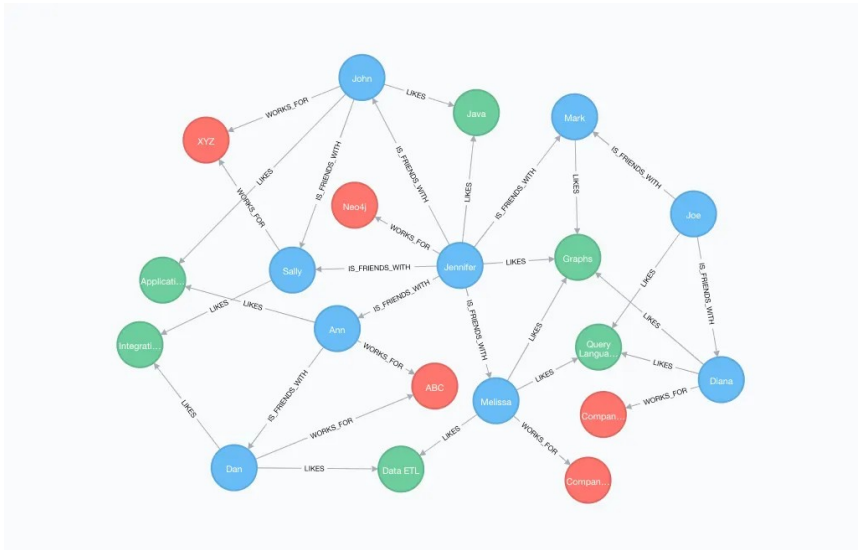
- Veriler anahtar-değer çifti ile saklanır
- Değer kısmı belge olarak depolanır
- Belge JSON veya XML formatta olur
- CMS sistemleri, blog platformaları, gerçek zamanlı analizler ve e-ticaret uygulamalarında kullanılır
- Amazon DynamoDB, CouchDB, MongoDB, Riak, Lotus Notes örnek verilebilir

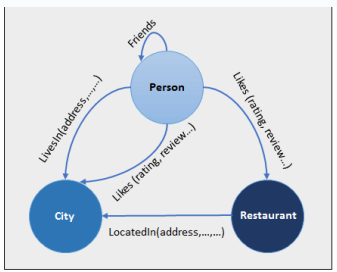
Doküman Tabanlı Sistem Eğilimleri



NoSQL Sistem Yaklaşımları

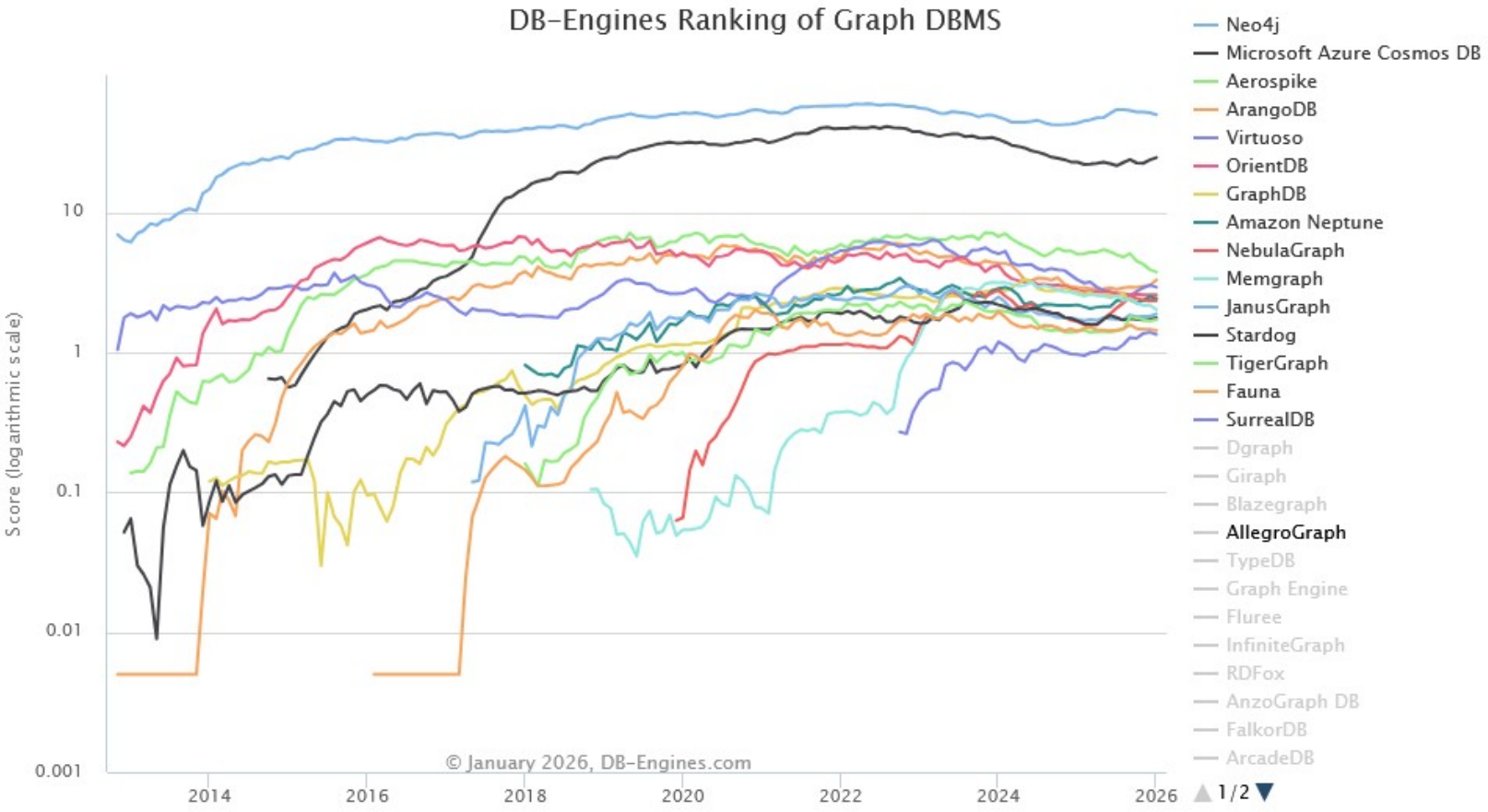
Graf Tabanlı Sistemler



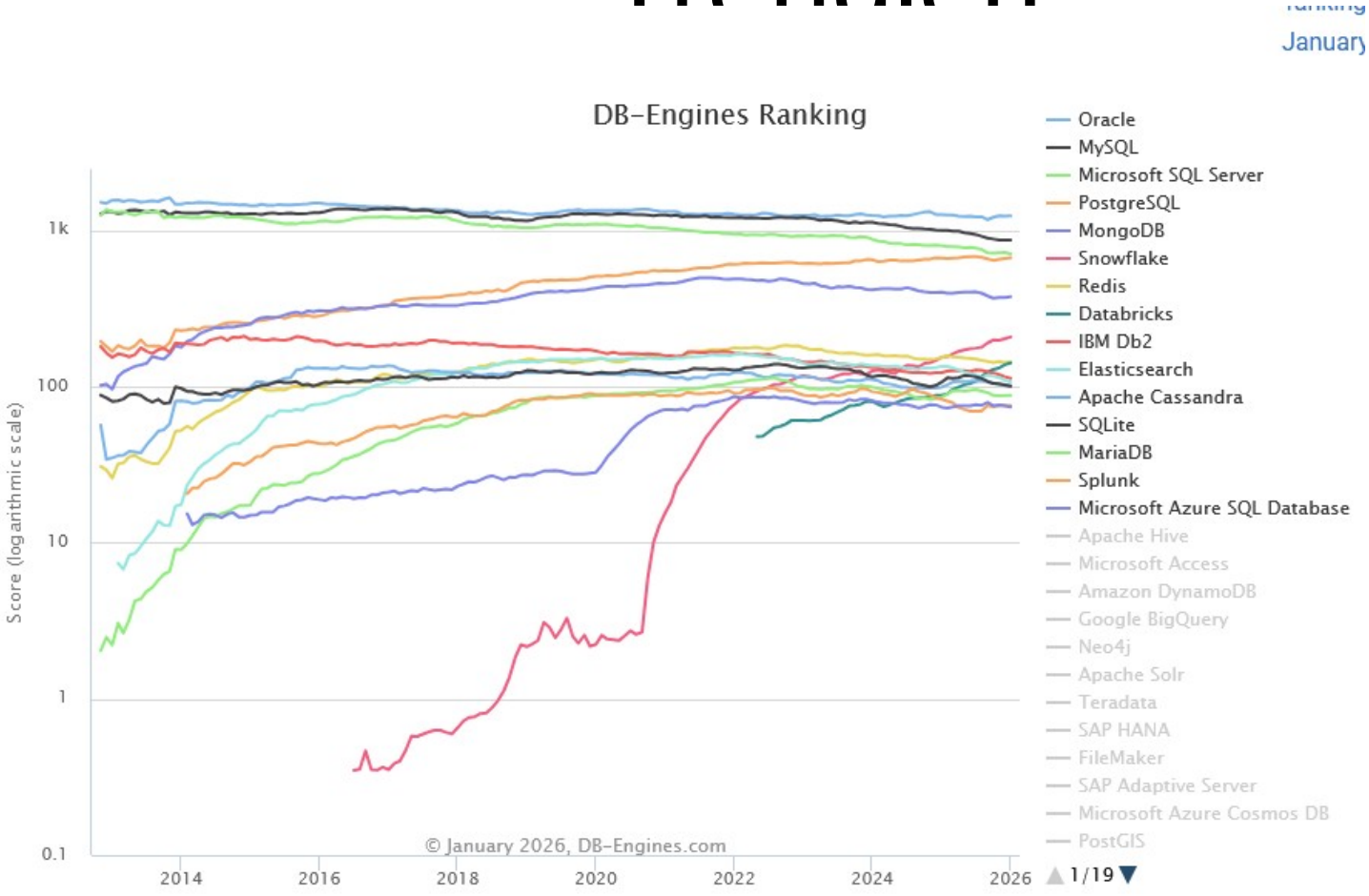


- Veriler düğüm yapısında tutulur
- Çok hızlı ilişki sorguları yazılabilir
- Düğümler özellik taşıyabilir
- Klasik join yerine doğrudan ilişkiler gezilir bu yüzden çok hızlıdır
- Derin ilişki sorgularında oldukça hızlıdır
- Sosyal ağlar, lojistik ve mekansal veriler için uygun yapıdadır
- Neo4J, OrientDB, FlockDB, Amazon Neptune, ArangoDB, JanusGraph örnek verilebilir

Graf Tabanlı Veri Tabanı Kullanımı



Veri Tabanları Genel Kullanım Trendleri



Özet

- NoSQL = Esneklik, Hız, Ölçeklenebilirlik
- Büyük veriden ziyade:
  - Organik ve değişken veri için uygundur

MongoDB Nedir?

- Doküman tabanlı NoSQL veri tabanı
- Şubat 2009’da ortaya çıkmıştır
- Collection içinde veriler saklanır
- Herhangi bir alana göre veya aralığa göre sorgu yazılabilir
- C++ dili ile yazılmıştır, Windows, Linux, Unix, BSD, OSX sistemlerinde çalışmaktadır
- Veriler BSON doküman yapısında saklanır
- Büyük ölçekli uygulamalar için tasarlanmıştır
- Geo-Spatial Queries (Harita tabanlı uygulamalarda konuma göre sorgulama)

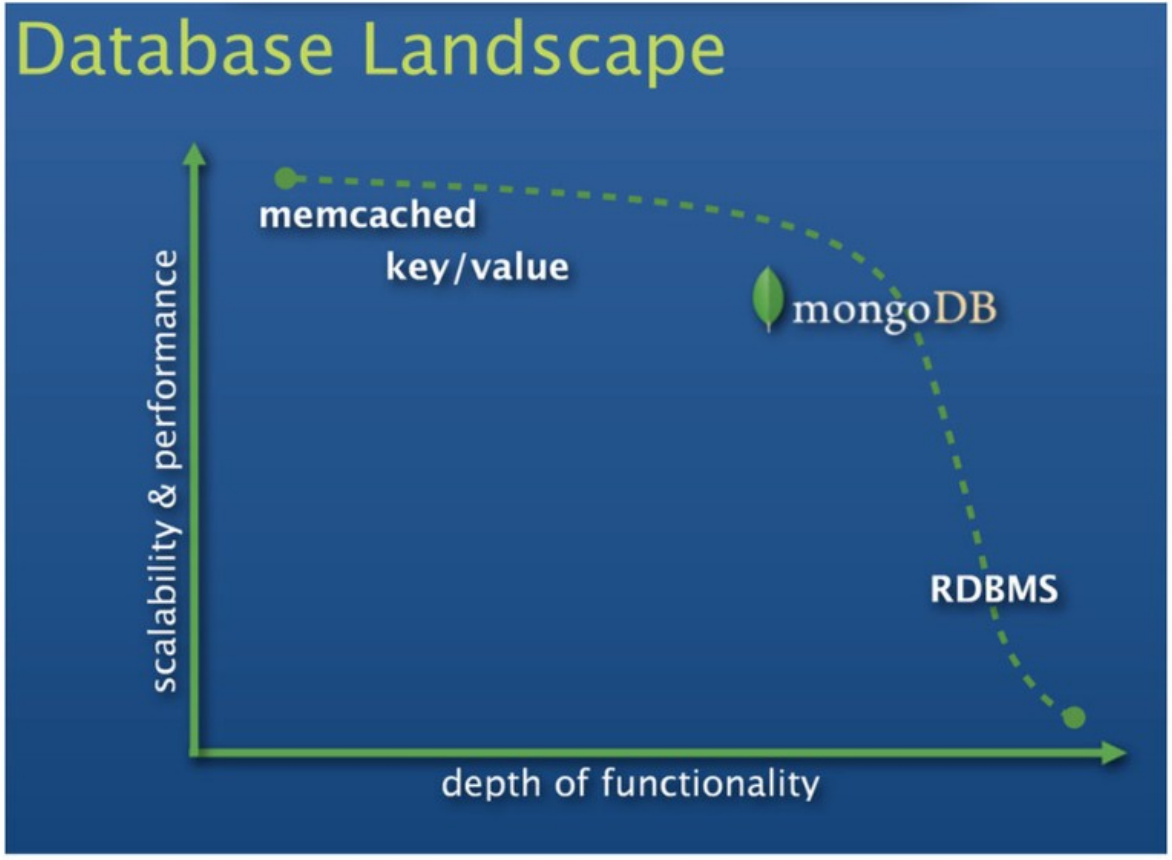
MongoDB Nedir?

- MongoDB ile İlişkisel veri tabanı karşılaştırma
- Collection ---> Tablo
- Document ---> Satır
- Sabit tablo şeması yoktur
- Karmaşık veri yapıları
- Join ve Foreign Key kullanılmaz
- Yatay ölçeklenebilir

NoSQL Sistemler

Veri Tabanı Dünyası

- Veri boyutu arttıkça, veri yapısı karmaşıklaştıkça RDBMS sistemlerinde ölçeklenibilirlik ve performans azalır



MongoDB Hangi Dillerde Çalışır?

- C, C#(.Net), C++, ERLANG, HASKELL, JAVA
- JavaScript, Ruby, Perl, Python, Node.js

RDBMS ve MongoDB Terminolojisi

RDBMS	MongoDB
Table	Collection
Row(s)	JSON Document
Index	Index
Join	Embedding & Linking
Partition	Shard
Partition Key	Shard Key

Temel Kavramlar

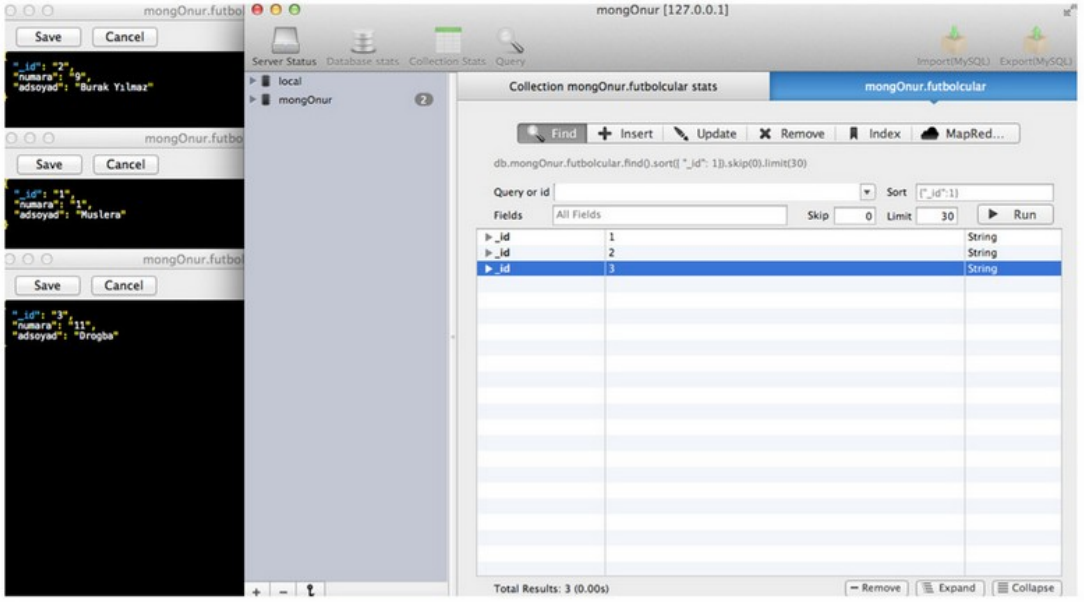
- RDBMS
- ACID:Atomicity, Consistency, Isolation, Durability
- CAP:Consistency, Availabilty, Partition Tolerance
- SQL
- DB
- JSON

MongoDB Kurulum

- İndirme adresi: https://www.mongodb.com/try/download/community
- İlgili adreste yukarıdaki ekran görüntüsünde olduğu gibi “Community Server” sekmesinde bilgisayarınıza uygun versiyonu belirttikten sonra “Download” butonuna tıklamanız yeterlidir.
- İndirdiğiniz programı çalıştırıp, kurulumu gerçekleştirdikten sonra “C:\Program Files\MongoDB\Server” dizisine göz atarsak MongoDB’nin kaynak dosyalarına erişebilmekteyiz.
- Dizin içerisinde versiyon isminde klasör altında “/bin” klasörü bizim MongoDB’ye erişmemizi sağlamaktadır.

MongoDB için GUI

- MongoHUB
- MongoVUE
- UMongo



Komut Satırı Arayüzü

- “Komut İstemi” penceresini açıp “C:\Program Files\MongoDB\Server\xxxversiyonumarası.0\bin” dizinine geçiniz.
- mongo –version
- mongo komutu MongoDb Komut Satırı Arayüzünü aktifleştirecektir. Bu komuttan sonra yazılan komutlar MongoDb’de yorumlanır ve çalıştırılır
- show dbs: Sunucudaki tüm veri tabanlarını getirir

Komut Satırı Arayüzü

- use[veri tabanı adı]:belirtilen veri tabanını kullanılabilir vaziyette seçer
- Show collections: üzerinde çalışılan veri tabanındaki collectionları getirir

Verileri Dizi Şeklinde Görünümü

Bir kişiye ait verileri tanımlayan basit bir json belgesi

```
{
  "_id" : 1 ,
  "isim" : {
    "ilk" : "Ada" ,
    "son" : "Lovelace"
  },
  "Başlık" : "İlk Programcı" ,
  "ilgi alanları" : [ "matematik" , "programlama" ]
}
```

MongoDB Sorgulama Araçları

- MongoDB Compass (Resmi Araç)
- Studio 3T (Profesyonel Seçim)
- Robo 3T (Hafif ve Hızlı)
- MongoDB Atlas "Browse Collections" (Web Tabanlı)
-
- Yeni başlıyorsan ve resmi destek istiyorsan: MongoDB Compass.
- SQL bilgin iyiyse ve bunu kullanmak istiyorsan: Studio 3T.
- Tarayıcıdan çıkmak istemiyorsan: Atlas Web Arayüzü.
-

MongoDb Sorguları

- MongoDB'de find() fonksiyonu, koleksiyonunuzdaki verileri sorgulamak için kullandığınız temel araçtır. Senin de belirttiğin gibi genel yapısı db.collection.find({sorgu}, {projeksiyon}) şeklindedir.
- Bu içteki süslü parantezlerin (nesnelerin) her biri, veritabanına "neyi aradığını" ve "sonucun nasıl görünmesini istediğini" söyleyen farklı birer filtredir.
- İlk parantez Query Filtresi
  - Bu bölüm, SQL'deki WHERE ifadesine karşılık gelir. Hangi dokümanların getirileceğini belirler. Eğer içi boşsa {} tüm dokümanlar gelir.
- MongoDB find fonksiyonu kayıtları filtrelemek için kullanılır. Fonksiyonun genel yapısı aşağıdaki gibidir
  - db.CollectionAdı.find({query},{projeksiyon},{})

MongoDb Sorgular

- İkinci Parantez: Projeksiyon (Projection)
  - Bu bölüm, SQL'deki SELECT ifadesine karşılık gelir. Gelen dokümanın içindeki hangi alanların (field) görünmesini, hangilerinin gizlenmesini istediğini belirler.
  - 1 (Göster): { name: 1, email: 1 } → Sadece isim ve e-posta alanlarını getir.
  - 0 (Gizle): { password: 0 } → Şifre alanı dışındaki her şeyi getir.
- Not: _id alanı varsayılan olarak her zaman gelir. _ide alanı görünmesi isteniyorsa { _id: 0 } yazılır.

MongoDb Sorgu Operatörleri

Operator	Karşılık	Açıklama	Örnek Kullanım
$eq	=		{ age: { $eq: 25 } }
$ne	!=		{ status: { $ne: "deleted" } }
$gt	>		{ price: { $gt: 100 } }
$gte	>=		{ score: { $gte: 50 } }
$lt	<		{ stock: { $lt: 10 } }
$lte	<=	Belirlenen değerden küçük veya eşit olanları seçer.	{ age: { $lte: 18 } }
$in	in	Belirtilen dizi (array) içindeki değerlerden birine sahip olanları seçer.	{ city: { $in: ["Ankara", "İzmir"] } }
$nin	Not in	Belirtilen dizi içindeki değerlerin hiçbirine sahip olmayanları seçer.	{ tag: { $nin: ["spam", "ads"] } }

Mantıksal Operatörler

Operator	Karşılık	Açıklama	Örnek Kullanım

$and	and	Tüm koşulların sağlanması gerekir.	db.products.find({
			  $and: [
			  { category: "Elektronik" },
			  { price: { $lt: 5000 } }
			  ]
			})

$or	or	Koşullardan en az birinin sağlanması yeterlidir.	db.users.find({
			  $or: [
			  { city: "İstanbul" },
			  { age: 30 }
			  ]
			})

$not	not	Kendisinden sonra gelen koşulun tersini alır.	db.products.find({ price: { $not: { $gt: 100 } } })
$exists		Bir alanın (field) döküman içinde var olup olmadığını kontrol eder.	({ phone: { $exists: true } })
$nor		Ne o, ne de o operatörüdür. Listedeki koşulların hiçbirinin sağlanmadığı dokümanları getirir	db.users.find({ $nor: [{ city: "Ankara" }, { status: "passive" }] })

Mantıksal Operatörler

- $and ve $or operatörleri birer dizi ([ ]) alır. Çünkü bu operatörlere birden fazla nesne (koşul) göndermeniz gerekir.

```
db.products.find({
 category: "Mutfak",
   $or: [
         { price: { $lt: 1000 } },
         { stock: 0 }
    ]
})
```

countDocuments

- countDocuments() :Bu fonksiyon, verdiğiniz sorgu filtresine uyan dokümanların kesin sayısını döndürür. Veritabanındaki koleksiyonu tarayarak gerçek zamanlı ve doğru sonuç verir.
  - Yapısı: db.collection.countDocuments({query}, {options})
  - Örnek: Yaşı 25'ten büyük olan kaç kullanıcı var?

    db.users.countDocuments({ age: { $gt: 25 } })
- estimatedDocumentCount() : Bu fonksiyon, koleksiyonun tamamındaki doküman sayısını tahmin etmek için meta verileri (istatistikleri) kullanır. Çok büyük koleksiyonlarda (milyonlarca satır) anında sonuç verir çünkü filtreleme yapmaz ve tüm koleksiyonu taramaz.
  - Yapısı: db.collection.estimatedDocumentCount()
  - Özellik: Filtre (query) alamaz. Sadece tablonun toplam büyüklüğünü hızlıca öğrenmek için kullanılır.
-

Gruplama Mantığı - aggregate

- SQL dünyasındaki GROUP BY mantığı, MongoDB'de çok daha güçlü ve esnek bir yapı olan Aggregation Framework (Toplulaştırma Çerçevesi) ile karşılanır.
- MongoDB'de gruplama yapmak için bir işlem hattı (pipeline) kurarız. Veriler bu işlem hattının aşamalarından geçer ve her aşamada bir işleme tabi tutulur. GROUP BY işleminin tam karşılığı bu hattın içindeki $group aşamasıdır.

Gruplama Mantığı - aggregate

SQL'deki SELECT ... FROM ... GROUP BY ... yapısının MongoDB karşılığı şöyledir:

```
db.collection.aggregate([
  {
    $group: {
      _id: "$gruplanacak_alan", // SQL'deki GROUP BY kısmı
      toplam: { $sum: "$hesaplanacak_alan" } // SQL'deki SUM, AVG vb. kısmı
    }
  }
])
```

MongoDb Sorgular - find

- Üçüncü Bölüm (Opsiyonel):
  - İmleç Yöntemleri (Cursor Methods)
- Aslında find() fonksiyonu doğrudan 3. bir parantez almaz. Ancak find() bir "Cursor" (İmleç) döndürdüğü için, parantezin dışına nokta koyarak ek ayarlar ekleriz. Bunlar sorgu sonucunun sırasını ve miktarını belirler.
- sort(): Sıralama yapar. { age: 1 } küçükten büyüğe, -1 büyükten küçüğe.
- limit(): Kaç tane kayıt geleceğini belirler.
- skip(): Belirli sayıda kaydı atlar (Sayfalama için kullanılır).

MongoDb Sorgular - find

- Diyelim ki bir "Kullanıcılar" tablosu var. Yaşı 20'den büyük olan kullanıcıların sadece isimlerini görmek istiyoruz ve en gençten başlayarak ilk 5 kişiyi getirmek istiyoruz:
  - ```
    db.users.find(
    { age: { $gt: 20 } },    // 1. Sorgu: Yaş 20'den büyük mü?
    { name: 1, _id: 0 }      // 2. Projeksiyon: Sadece ismi getir, ID'yi gizle
    ).sort({ age: 1 }).limit(5) // Ek Ayarlar: Yaşa göre sırala ve 5 kişiyle sınırla
    ```
-
    ```

MongoDb Sorguları

- Kullanici kolleksiyonundaki tüm kayıtları listelemek.
  - db.Kullanici.find()
- Sadece istenen kolonları listelemek
  - db.Kullanici.find({},{Ad:1, Soyad:1})
- Maaş alanında 5000 yazan kullanıcıları listelemek
  - db.Kullanici.find({Maas:1000},{Ad:1,Soyad:1})

SQL	MONGO	Example
SELECT	find()	db.Ornek.find() Aggregate, count ve sum gibi fonksiyonlarda kullanılabilir.
INSERT	insertOne()	db. Ornek. insertOne({ user_id: “ea0001”, name: “Enes” , age:40})

UPDATE
UPDATE Ornek SET
name = “METE”WHERE
age > 25
	UpdateMany()	db.Ornek.
		updateMany({
		age: { $gt: 25 } },
		{ $set: {
		name: “METE” } })

ALTER
ALTER TABLE Ornek
ADD join_date
DATETIME,ALTER TABLE
people
DROP COLUMN
join_date
	updateMany()	db.Ornek.
		updateMany({ },
		{ $set: {
		join_date:
		new Date()
		} })
		
		db.people.
		updateMany({ },
		{$unset: {
		“join_date”: “” } })

CREATE	insertOne() / createCollection()	insertOne, insertMany kullanıldığında colection oluşturulur. Yada db.createCollection (“TEST”) gibi direk oluşturabilirsiniz.

WHERE SELECT * FROM people WHERE age > 25	find() $ne : != $gt : > , $gte : >= $lt :< , $lte :<=	db.Ornek. find({ age: { $gt: 25 } })
LIKE like “%bc%” like “bc%” like ‘%bc’	Rexeg sorgularını aşağıdaki tag ile beraber kullanabilirsiniz. $regex tagı	db.Ornek. find({name:/bc/}) db.Ornek.find({name:/^bc/}) db.Ornek.find({name:/bc$/})
DELETE	deleteOne()/deleteMany()orremove()	db.Ornek. deleteOne({id: 11})
ORDER BY	sort() +1 asc, -1 desc	db.Ornek.find(). sort({join_date:-1})

GROUP BY	group() / aggregate($group)	db.Ornek.
		aggregate
		([{
		$group: {
		_id: “$name”}}])

JOIN	Left join için $lookup	db.[collection name]. aggregation ([{$lookup:{ from: “[foreign collection]”, localField : “[local field]”, foreignField : “[foreign collection field name]”, as : “[key name to appear in result]”}}])

MongoDB Atlas

- MongoDB Atlas, MongoDB’nin bulut tabanlı (cloud) yönetilen veritabanı hizmetidir.
- Yani MongoDB’yi kendi sunucuna kurup yönetmek yerine, AWS, Azure veya Google Cloud üzerinde hazır, güvenli ve ölçeklenebilir şekilde kullanmanı sağlar.
- Atlas Ne İşe Yarar?
  - Kurulum yok: Sunucu kurulma işi yapılmaz, birkaç tıkla MongoDB hazır olur
  - Güvenlik hazır: Otomatik yedekleme, şifreleme, IP kısıtlaması
  - Otomatik ölçeklenir: Veri ve trafik artınca kendisi büyür
  - Bakım derdi yok: Güncelleme, replikasyon, failover Atlas tarafından yapılır
  - Her yerden erişim: İnternet olan her yerden bağlanılabilir

MongoDb Atlas

- Kimler için uygun?
  - Web / mobil uygulama geliştiricileri
  - Mikroservis mimarileri
  - Hızlı prototip ve üretim ortamları
  - “Veritabanıyla değil, uygulamayla uğraşayım” diyenler

.

MongoDB Atlas

The multi-cloud developer data platform available on AWS, Azure, and Google Cloud

Free

$0/hour

Explore Atlas

Free forever

For learning and exploring MongoDB in a cloud environment.

- 512MB of storage
- Shared RAM
- Shared vCPU

Recommended

Dedicated

$0.08/hour

Scale Your Project

Starts at $56.94/month

For production applications with sophisticated workload requirements.

- 10GB to 4TB of storage
- 2GB to 768GB RAM
- 2vCPUs to 96vCPUs

View dedicated pricing

Flex

$0.011/hour

Build a Prototype

Up to $30/month

For application development and testing, with on-demand burst capacity for unpredictable traffic.

- 5GB of storage
- Shared RAM
- Shared vCPU

View flex pricing