

NoSQL

SQL ve Daha Fazlası (Not Only SQL)

Hazırlayan: Ahmet Cevahir ÇINAR

APACHE
HBASE

 **Cassandra**



CouchDB
relax



riak



mongoDB

HYPERTABLE INC



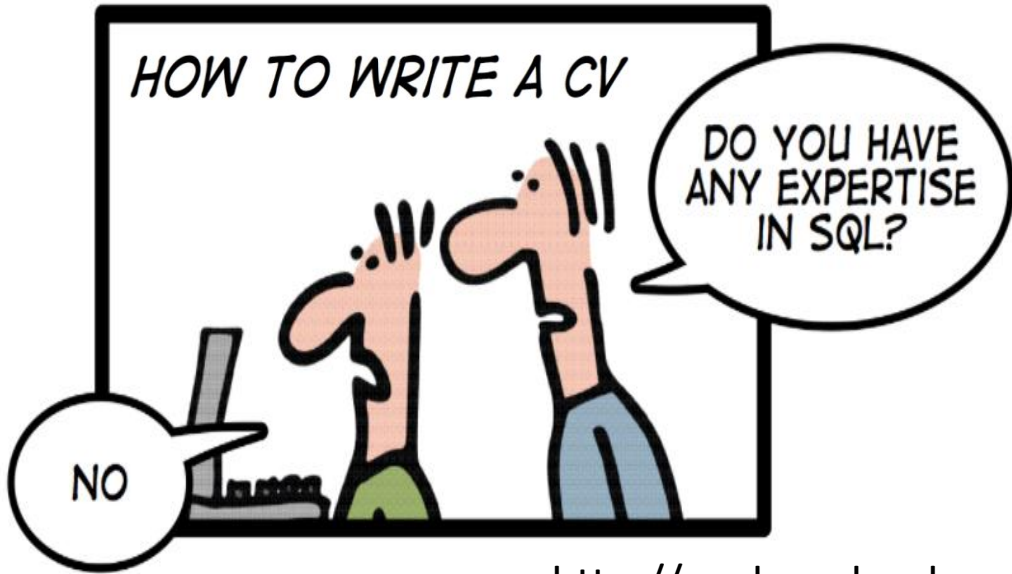
Neo4j



redis

Halkımızın Bilgisine:

Bu sunum, en sondaki kaynaklar sayfasında belirtilen içerikler öncülüğünde, tek tek kaynakları belirtilememiş bir çok görselden meydana gelmiştir. İçeriğin hazırlanması sırasında çok titiz bir referans gösterme çabasına girilmemiştir.



<http://geek-and-poke.com/>



NoSQL



Gaming



Social



IoT



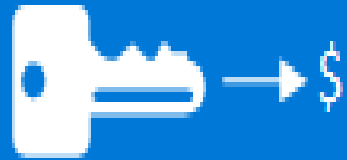
Web



Mobile



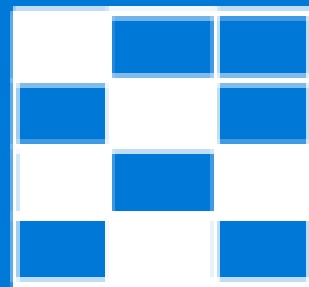
Enterprise



Key/value store



Document
database



Column family store

SQL



Web



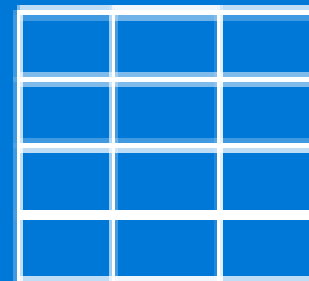
Mobile



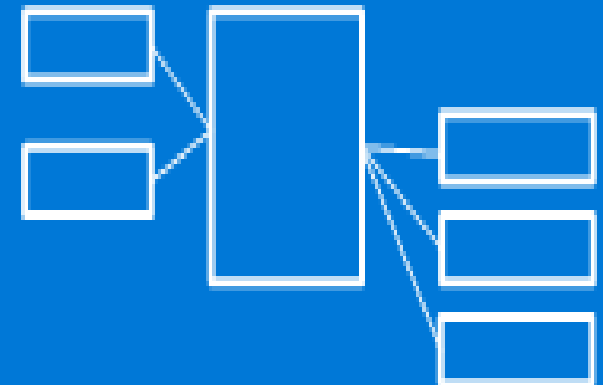
Enterprise



Data mart



Relational table storage



Relationships use joins

Temel Kavramlar ve Kısaltmaları

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

Dünyadaki insanların yaklaşık yarısı internet kullanmaktadır.

Bu insanlar ya mobil cihazlar üzerinden ya da online web siteleri üzerinde anlık veri girmekte ve internet üzerinde bir saniyelik zaman diliminde terabyte'lar düzeyinde büyük veriyi oluşturmaktadırlar.

Oluşan bu veri üzerinde gizli bilginin mantıksal bir zaman diliminde açığa çıkartılabilmesi için akıllı analiz algoritmalarının kullanılması gerekmektedir.

Verinin nasıl tutulacağı kısmıyla Veritabanı yaklaşımları ilgilenir.

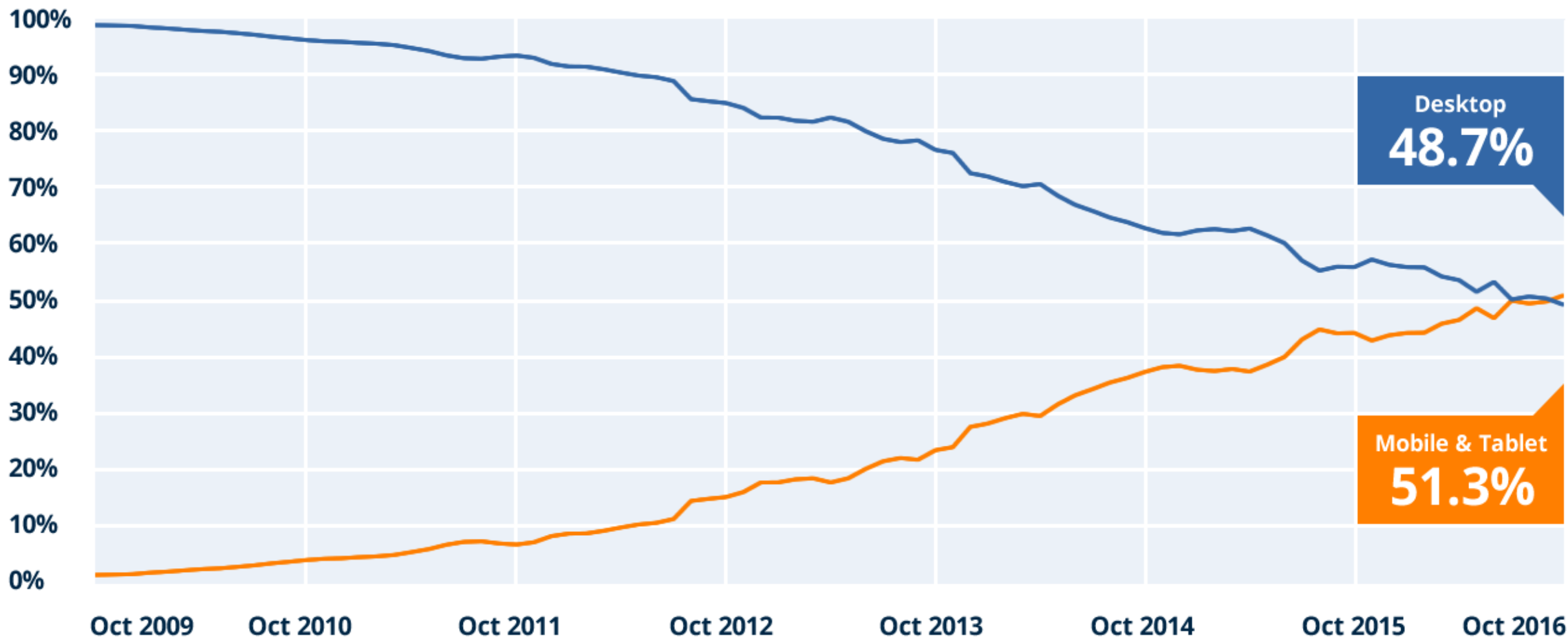
**WORLD INTERNET USAGE AND POPULATION STATISTICS
JUNE 30, 2016 - Update**

World Regions	Population (2016 Est.)	Population % of World	Internet Users 30 June 2016	Penetration Rate (% Pop.)	Growth 2000-2016	Table % Users
Asia	4,052,652,889	55.2 %	1,846,212,654	45.6 %	1,515.2%	50.2 %
Europe	832,073,224	11.3 %	614,979,903	73.9 %	485.2%	16.7 %
Latin America / Caribbean	626,119,788	8.5 %	384,751,302	61.5 %	2,029.4%	10.5 %
Africa	1,185,529,578	16.2 %	340,783,342	28.7 %	7,448.8%	9.3 %
North America	359,492,293	4.9 %	320,067,193	89.0 %	196.1%	8.7 %
Middle East	246,700,900	3.4 %	141,489,765	57.4 %	4,207.4%	3.8 %
Oceania / Australia	37,590,820	0.5 %	27,540,654	73.3 %	261.4%	0.8 %
WORLD TOTAL	7,340,159,492	100.0 %	3,675,824,813	50.1 %	918.3%	100.0 %

Internet Usage Worldwide

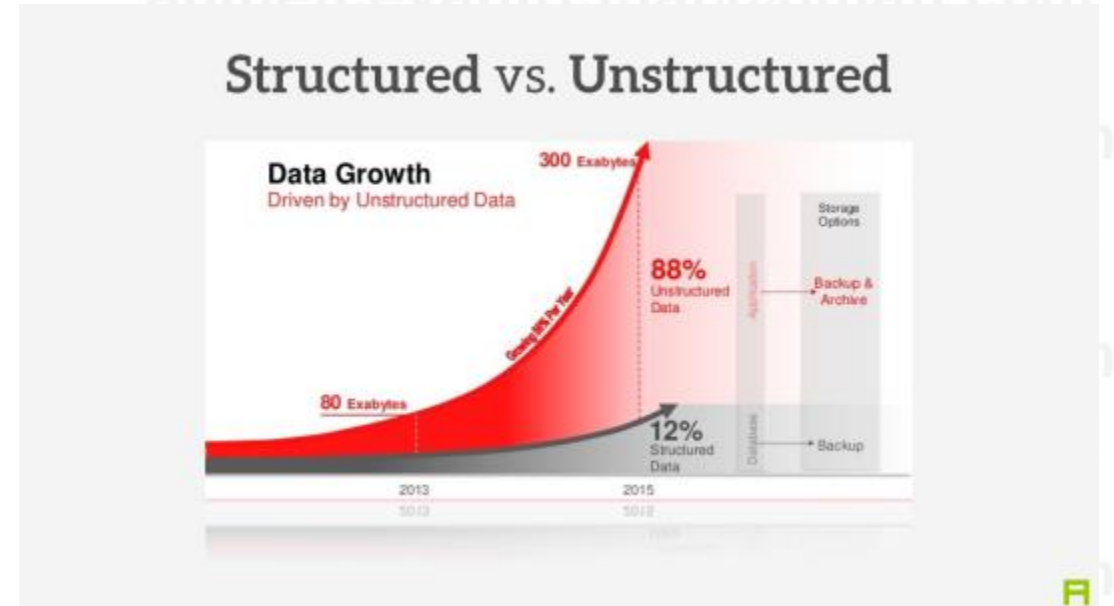
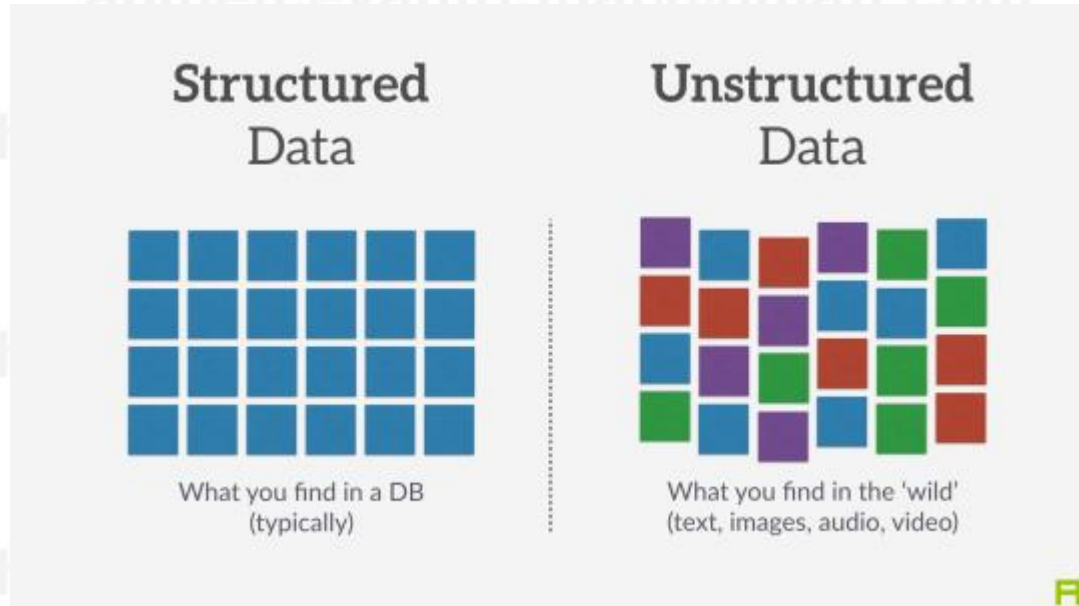
October 2009 – October 2016

■ Desktop ■ Mobile & Tablet



Yapılandırılmış Veri ... Yapılandırılmamış Veri

- Büyük veri, yapılandırılmış veya yapılandırılmamış veri şeklinde olabilir.
- Yapılandırılmış verilerde her satırdaki sütun sayısı sabitken yapılandırılmamış verilerde her bir satırda farklı sayıda sütun olabilir.

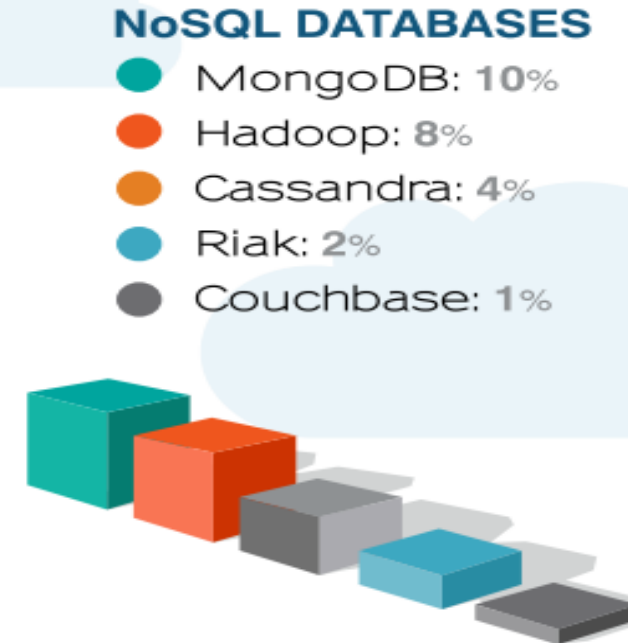
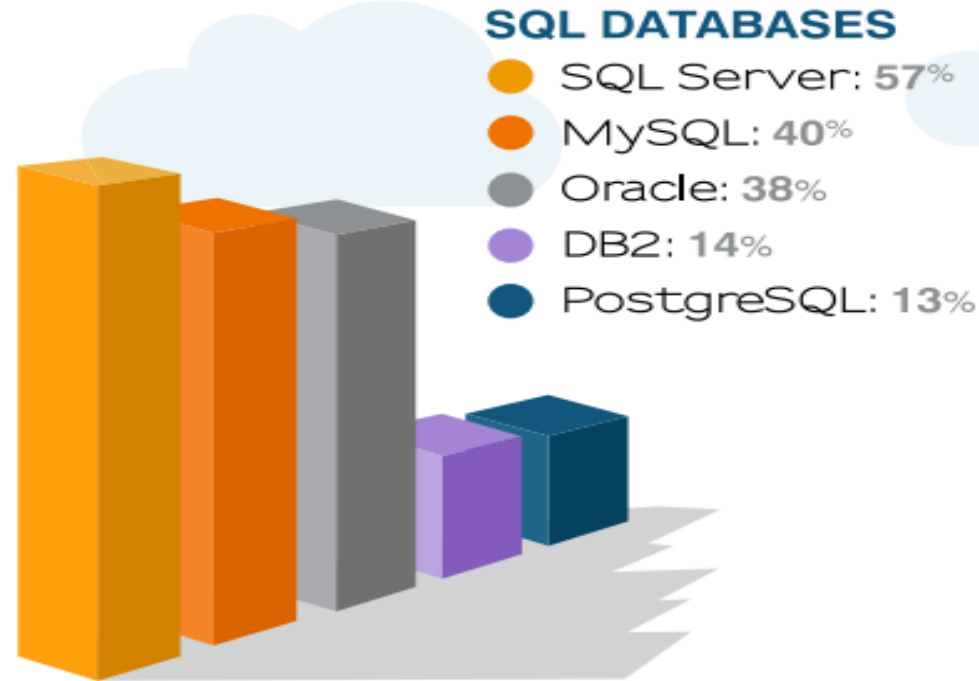


NoSQL ilişkisel (RDBMS) (Relational Database Management System) veri tabanlarına alternatif olarak çıkan, nesneyi dikkate alan SQL'deki gibi belirli kolonlara ihtiyaç duymayan veri depolama yaklaşımıdır.

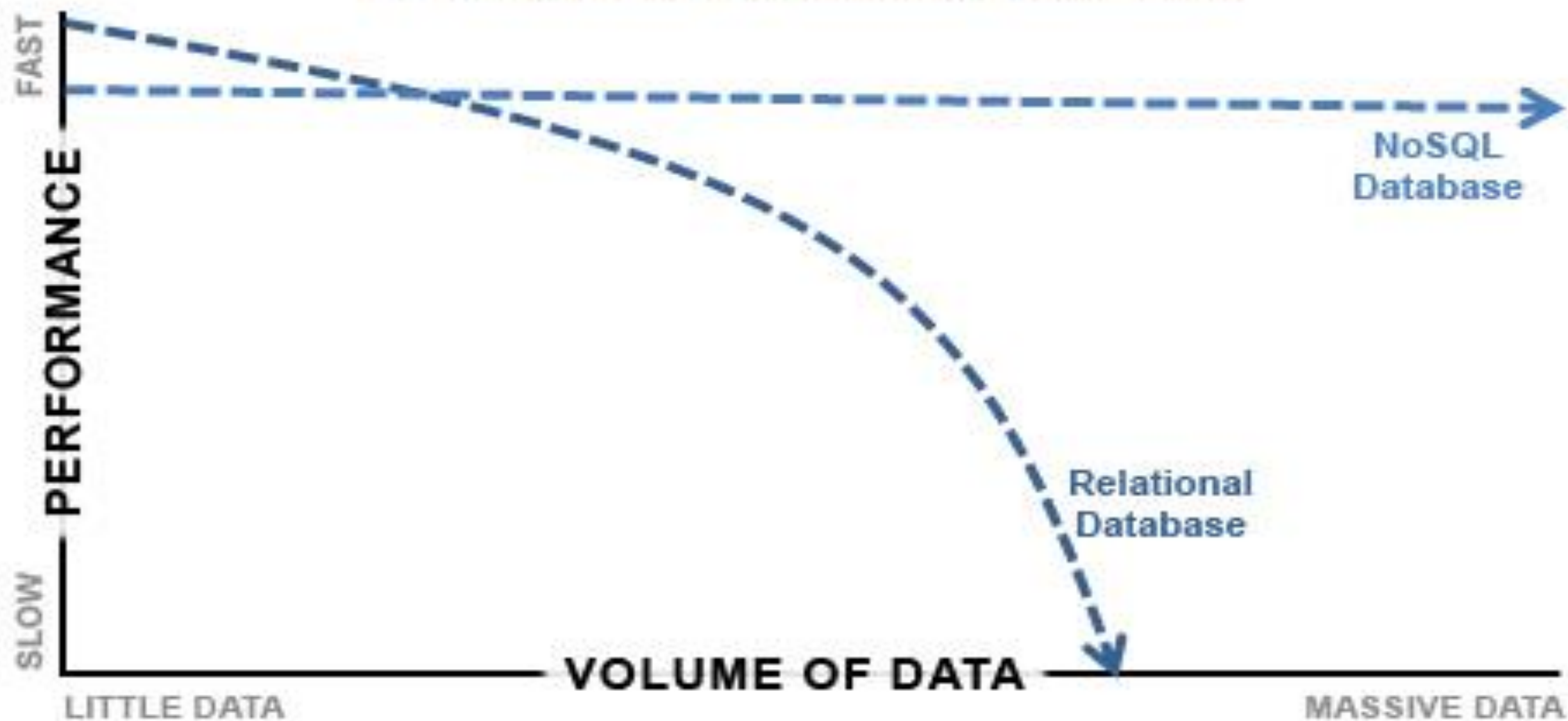
NoSQL dendiğinde bilinen Hadoop, MongoDB, ElasticSearch vb. gibi birçok marka bulunmaktadır.

Bu ders kapsamında, yaygın kullanımı nedeniyle NoSQL veri tabanı olarak MongoDB tercih edilmiştir.

Şirketiniz Hangi Veri Tabanlarını Kullanıyor?



Scalability of NoSQL Database vs Traditional Relational Database



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:

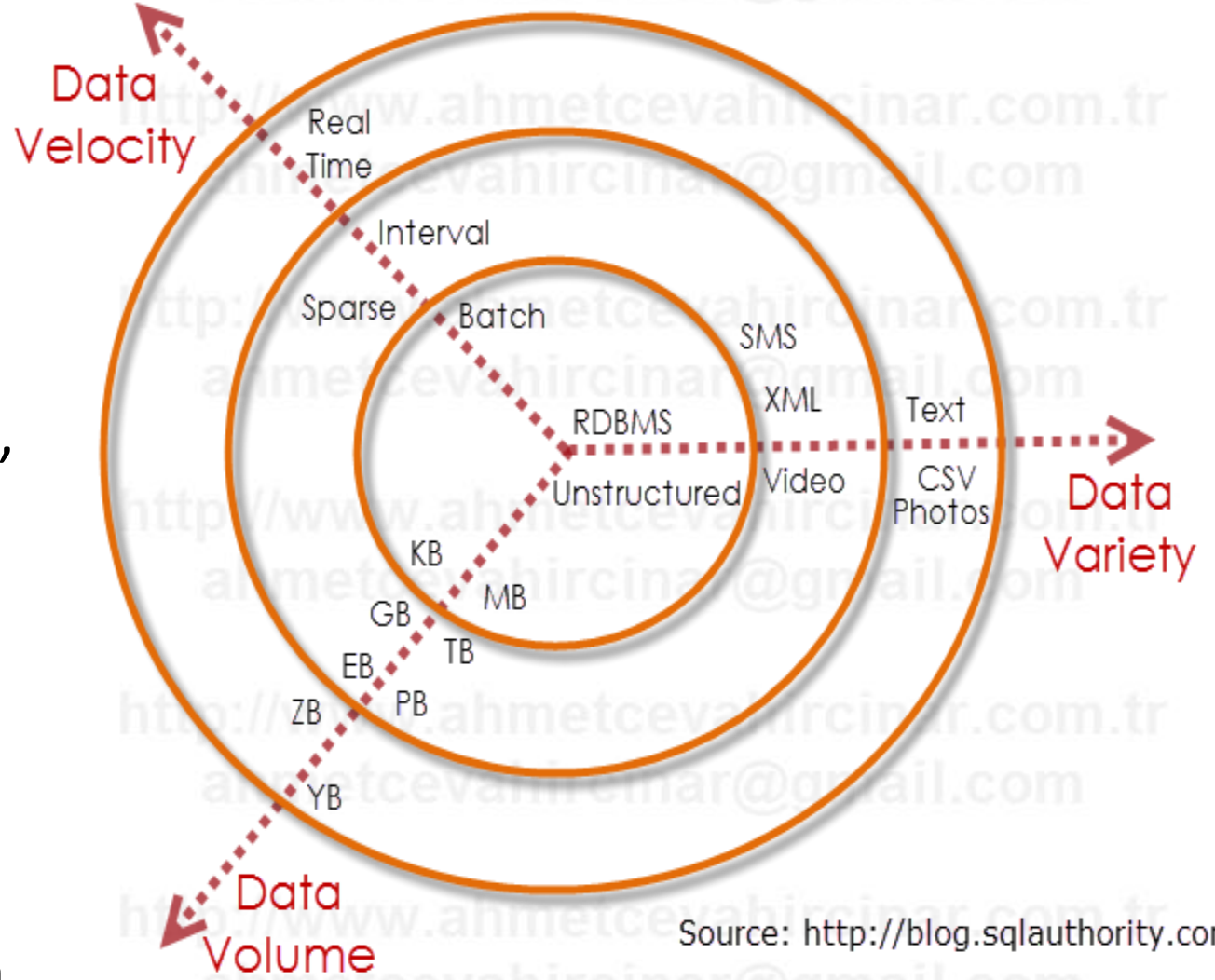
Volume: Verinin **hacim**sel büyüklüğü

Velocity: Veri akışının **hızı** (sensör verileri, finansal işlemler, uygulama kullanım bilgileri -log kayıtları- gibi)

Variety: Verinin farklı **format**larda gelebilmesi

NoSQL veritabanları, bu üç özelliğe sahip verilerin işlenmesini sağlayan dağıtık, ölçeklendirilebilir ve yüksek erişime (high availability) sahip veritabanlarıdır.

3Vs of Big Data

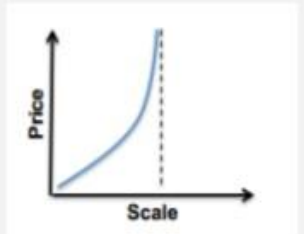


NoSQL yapılar ilişkisel veri tabanlarının performans sorunlarının yanında lisans ve işletme maliyetlerinin de karşısına çıkmıştır.

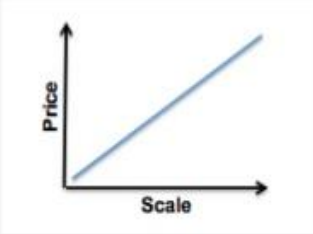
Scale-out vs Scale-up



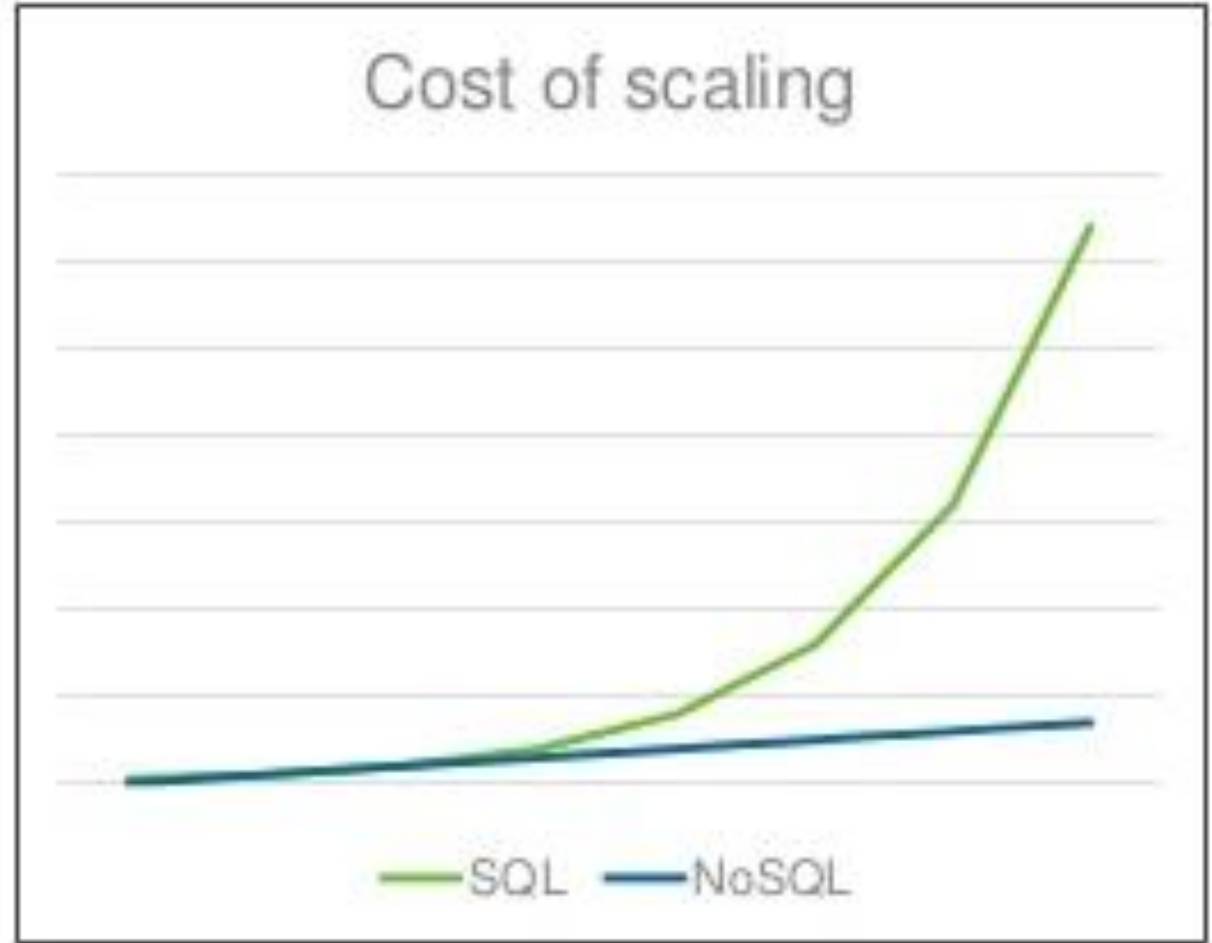
Then: Scale Up



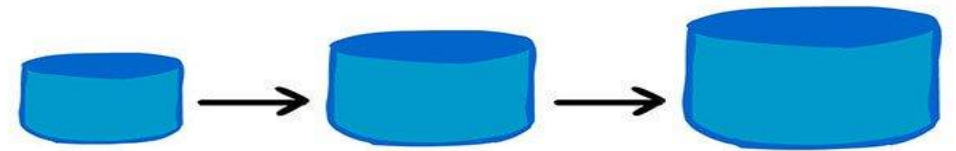
Now: Scale Out



Cost of scaling



Scale-up



Scale-out





<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

<http://www.ahmetcevahircinar.com.tr>

<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

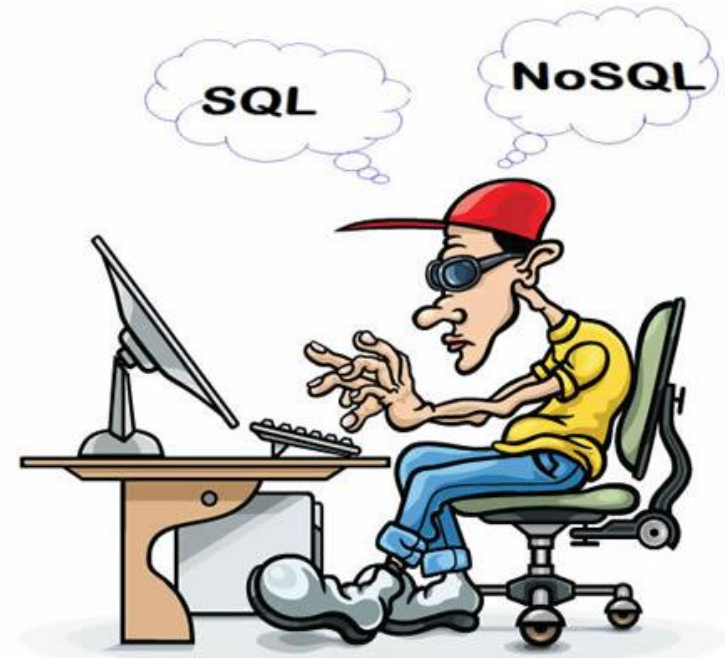
<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

<http://www.ahmetcevahircinar.com.tr>
ahmetcevahircinar@gmail.com

h1

h1

h1



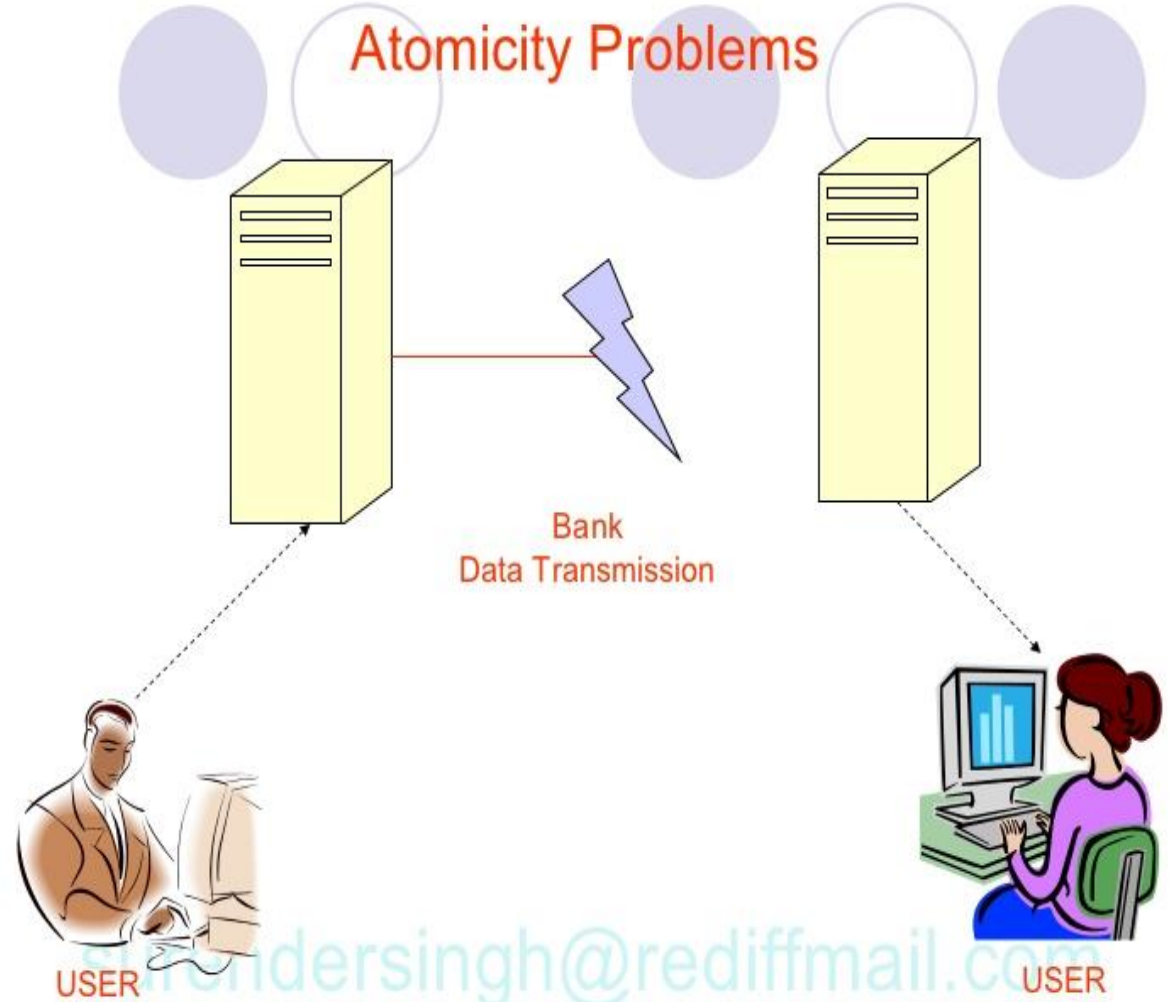
- NoSQL ACID garantisi vermeyen dağıtık veri depolarını dikkate alarak yola çıkmıştır.
- Bir transaction oluşabilmesi için ACID ilkelerine uyması gerekir.
- ACID, Atomicity –Consistency – Isolation – Durability
- (bölünmezlik– tutarlılık - izolasyon - dayanıklılık)
anlamlarına gelen kelimelerin baş harfleridir
- ACID, IBM DB2, MySQL, Microsoft SQL Server, PostgreSQL, Oracle İVTYS, Informix gibi klasik ilişkisel veritabanı sistemlerinde sağlanan temel özelliklerdendir.

Atomicity (Bölünmezlik)

Transaction'ı oluşturan tüm işlem parçaları hataya düşmemelidir ve her bir işlemin başarılı olarak sonuçlanması gerekmektedir. Aksi halde başlatılan işlem geri alınır. "Ya hep Ya hiç" mantığına dayanır.

Atomicity fiziksel olaylar dahil olmak üzere elektrik kesintisi, çökme, hata benzeri durumlarda işlem gerçekleşmesini garanti eder.

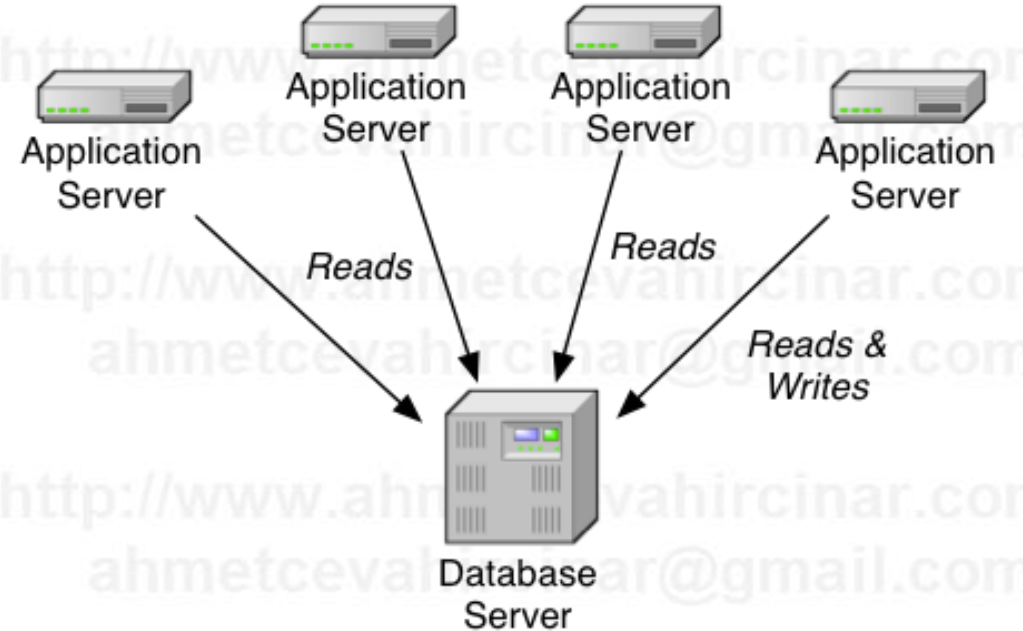
Dışardan bakıldığında (atom) bölünmez olmak ilkesi göze çarpar.



Consistency (Tutarlılık)

Her kullanıcı verilerin tutarlı halini görürler. Buna kullanıcının kendi yaptığı değişiklikler ve diğer kullanıcıların yaptığı değişiklikler de dahildir.

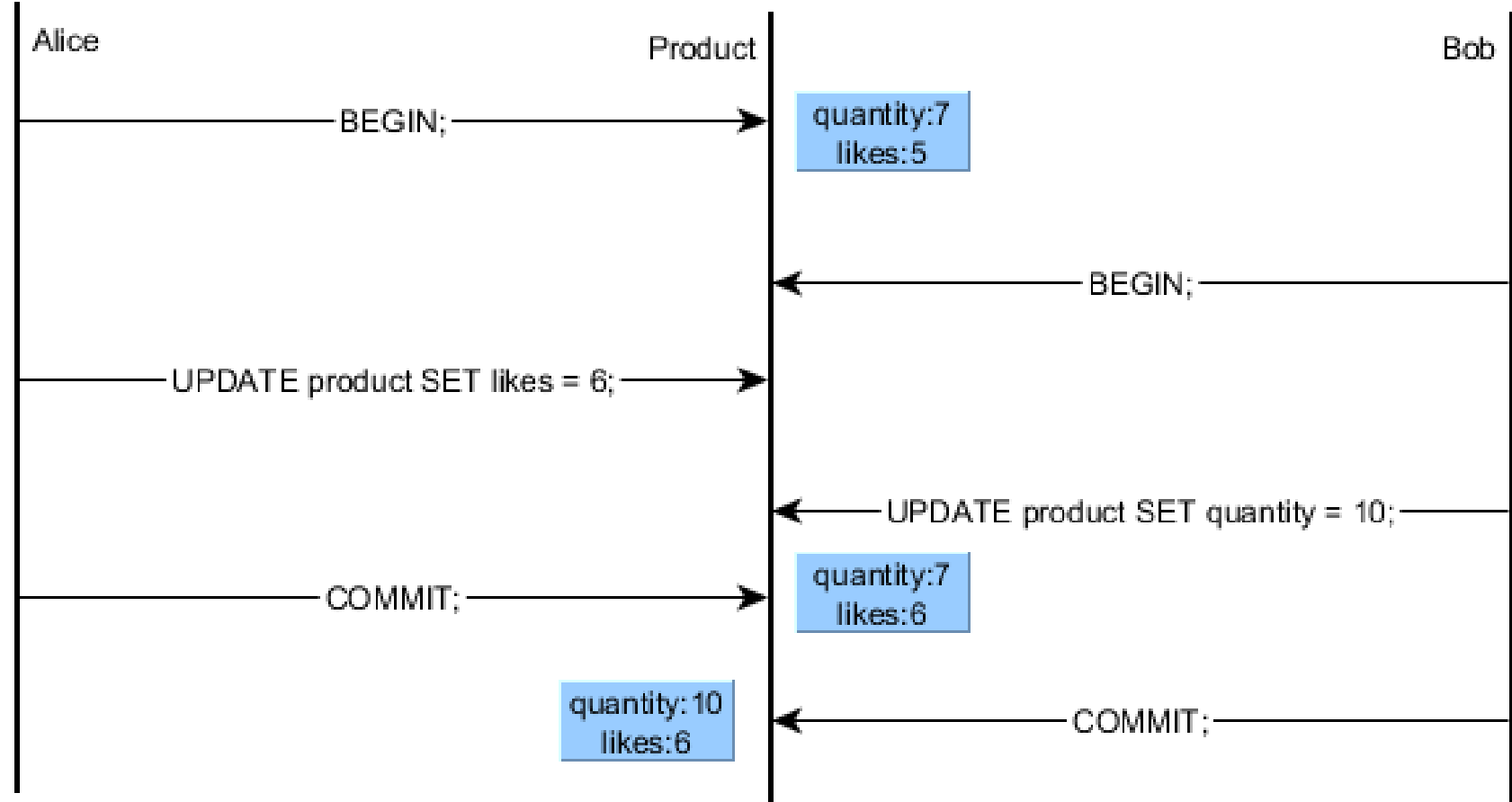
Bir kullanıcı bir tablodaki veriye eriştiği zaman ve o veriyi "update" ederken, diğer hiçbir kullanıcı o veriyi update veya delete edemez. Çünkü ilk kullanıcı tablonun update etmeye çalıştığı kolonu üzerine kilitlemiştir.



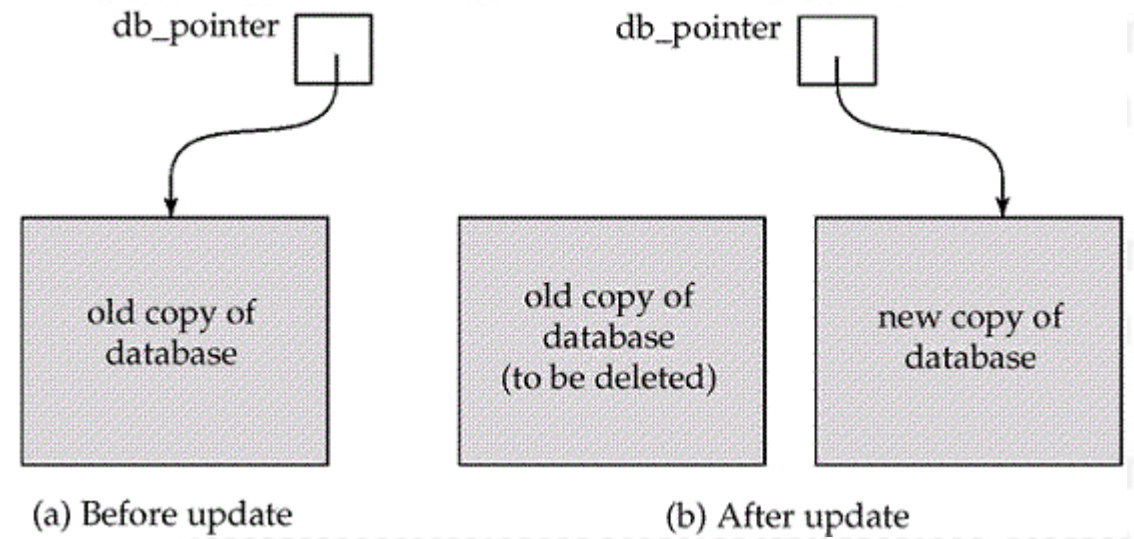
Isolation (İzolasyon)

Veritabanında oluşan transactionlar için eş zamanlı çalışma sürecinde birbirlerini etkilemeden çalışma esasıdır.

Ek olarak kuyruk yönetimi ile bu çalışma süreci kontrol altına alınmaktadır.



Durability(dayanıklılık)



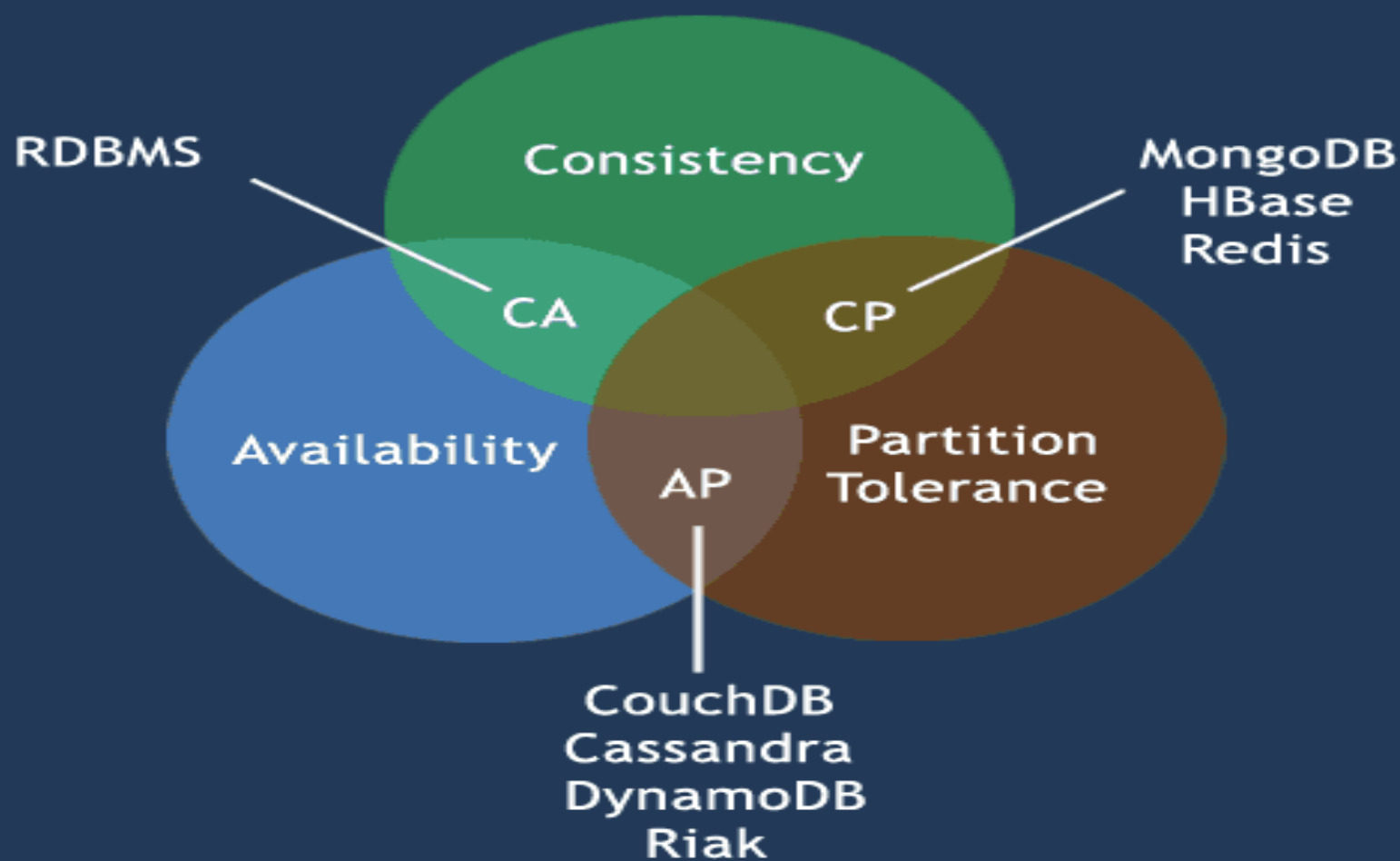
- **Durability(Rollback)** : Atomcity içerisinde bahsettiğimiz transaction işlemlerini konu almaktadır.
- Elektrik kesintisi, çökme ve ya hata gibi durumlarda oluşabilecek durumlar veri tabanındaki tüm verileri kaybetmemize sebep olabilir.
- SQL Transaction işlemlerinin hızlı olabilmesi için Caching modeli kullanılarak RAM 'den faydalanılmaktadır.
- Elektrik kesintisinde ise RAM'de bulunan işlemlere erişme imkânı kalmayacağı gibi hata durumlarında RAM'e erişim yönetimi imkânsız olacaktır.
- Durability, hata durumlarına karşın RAM yönetimi yerine temp file ya da log yönetimi esas alınır.

CAP (Consistency, Availability, Partition Tolerance)

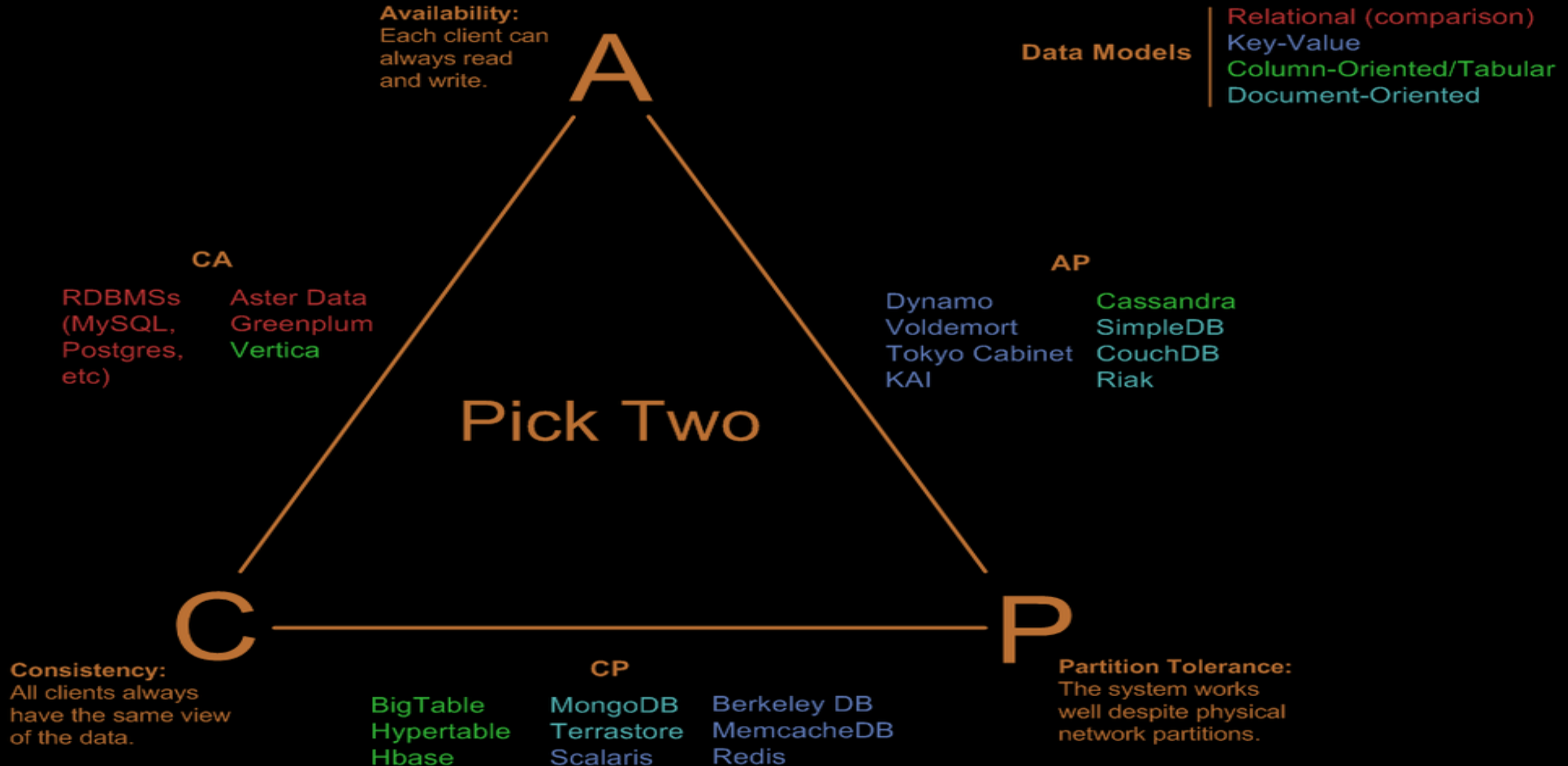
Tutarlılık, Müsaitlik ve Parçalanma Payı

- CAP teorisine göre herhangi dağıtık bir sistem aynı anda Tutarlılık (Consistency), Müsaitlik (Availability) ve Parçalanma payı (Partition tolerance) 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 bütününe kaybetmeme özelliklerine aynı anda sahip olamaz. Bunlardan mutlaka birinde problem çıkacaktır. NoSQL sistemlerde ise bu kavramlar esnetilerek yatayda büyüme ile performans kazancı amaçlanmıştır. Örneğin veriyi bölerek, belirli sayıdaki kopyalarını da dağıtık sistemin farklı 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 Theorem



Visual Guide to NoSQL Systems

















• Remembrance Inc! — Asla unutmayın!

- Not defterinizdeki müşteri sayfasına bilgi kaydedilir.
- **Sorun:** Müşteriler çoğalır, telefonda **bekleme** süresi 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 **tutarlı** değil!
- **Çözüm:** Yeni bilgi girildikçe karşılıklı güncelleme yapılacak.
- **Sorun:** Birisi işe gelemese?
- **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 **uygunluk** açısından çalışıyor olsa da **bölünebilme toleransı** bakımından zayıf kalıyor.

- **Tutarlılık:** Kayıt ekleyen müşterilen, ne kadar çabuk geri ararlarsa arasınlar, her zaman her koşulda en güncel kayda ulaşacaklardır.
- **Uygunluk:** Remembrance Inc! siz ve/veya eşiniz işte olduğu sürece her zaman çağrı kabul etmeye müsait olacaktır.
- **Bölünebilme/parçalanma toleransı:** Remembrance Inc! sizin ve eşinizin arasında iletişim kopukluğu olsa bile çalışmaya devam edecektir.
- Yerel kayıtlar değişince arka planda çalışan bir işlem tarafından diğerleri de güncellenir. (**eventual consistency**) Tek sorun arada sırada tutarlılığı kaybediyor olmanızdır. Örneğin müşteri önce eşinize kayıt yaptırıp sizin kayıtlarınız güncellenmeden evvel tekrar aradığında size ulaşırsa bilgileri bulunamaz.

NoSQL

anti-İVTYS değildir,
anahtar-değer depolarının,
belge veritabanlarının
Graph veritabanlarının
kullanımının altını çizmesine
rağmen ilişkisel olmayandır.

Document Database	Graph Databases
  	 
Wide Column Stores	Key-Value Databases
   	    

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

- NoSQL sistemlerini genel olarak 4 grupta toplayabiliriz:
- **Döküman (Document) tabanlı**: Bu sistemlerde bir **kayıt döküman** olarak isimlendirilir. Dökümanlar genelde **JSON** formatında tutulur. Bu dökümanların içerisinde sınırsız alan oluşturulabilir. Farklı formatlarda büyük verilerin saklandığı uygulamalar için uygundur. MongoDB, CouchDB, HBase, Cassandra ve Amazon SimpleDB bunlara örnektir.
- **Anahtar / Değer (Key / Value) tabanlı**: 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**: 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**: **Çok büyük verilerin tutulduğu dağıtık sistemler** için uygundur. Veriler anahtar & değer ikilisi şeklinde saklanır fakat anahtar birden fazla kolona işaret etmektedir. Örnek sistemler: Cassandra, HBase

Sorgu Modeline Göre NoSQL Veritabanları

- İlişkisel veritabanları karmaşık sorgu ihtiyaçlarına NoSQL sistemlerden daha iyi cevap vermektedir.
- **Doküman tabanlı** veritabanları verideki **tüm alanlar üzerinden sorgulama** imkânı sunarlar. Ayrıca farklı indeks seçenekleriyle (text, sparse, TTL, ...) sorgu performansını artırmak da mümkündür.
- **Çizge tabanlı** veritabanları veriler arasındaki ilişkiler ve bu ilişkilerin özellikleri üzerinden sorgulama imkânı sunar. Diğer sorgu modellerinde yüksek performans sunamadıkları için genel kullanımda tercih edilmezler.
- **Anahtar – değer** tabanlı veri tabanları sadece **anahtar** üzerinden erişim sunarlar. Bazı ürünlerin ikincil indeks desteği olsa da bu destek sınırlıdır.
- **Kolon** tabanlı veritabanları, anahtar – değer tabanlı veritabanları gibi veriye anahtar üzerinden erişim sağlarlar. Doküman tabanlı sistemler gibi **verinin tüm alanlarından sorgulama yapmak mümkün değildir**. Bu grupta yer alan Cassandra veritabanı geniş ikincil indeks ve özel veri tipleri (set, map, list, tuple, ...) desteği ile çok farklı sorgu modellerine destek verebilmektedir.

Performans için ne yapmalıyız?

➤ Sorgu iyileştirmeleri, yeni indeksler oluşturmak...

➤ Sunucu güçlendirme (dikey büyüme)

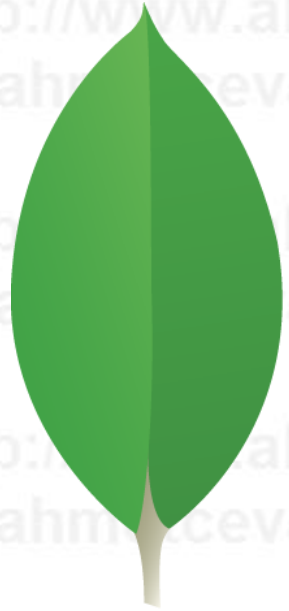
➤ Yatay bölünmesi (Sharding)

❖ Bağlantı (join) ve Birleştirme (aggregation) yapılamaz

❖ Verilerin taşınması

❖ Şema değişiklikleri

Yatay bölünmenin sorunlu olması nedeniyle NoSQL sistemler ön plana çıkmıştır.



mongoDB®

MongoDB=Hu**MONGO**us **DB**

- **10gen** firmasınca 2007 yılında başlanan ve 2009 yılında **AGPL** lisansıya 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.
- MongoDB dokümanları JSON nesneleri benzemektedir(**BSON**), alanların değerleri diğer dokümanları ya da dizileri içerebilir.
- 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.

BSON

- Binary JSON = BSON
- {"hello":"world"}
- Bson:

\x16\x00\x00\x00

// total document size

\x02

// 0x02 = type String

hello\x00

// field name

\x06\x00\x00\x00world\x00

// field value (size of value, value, null terminator)

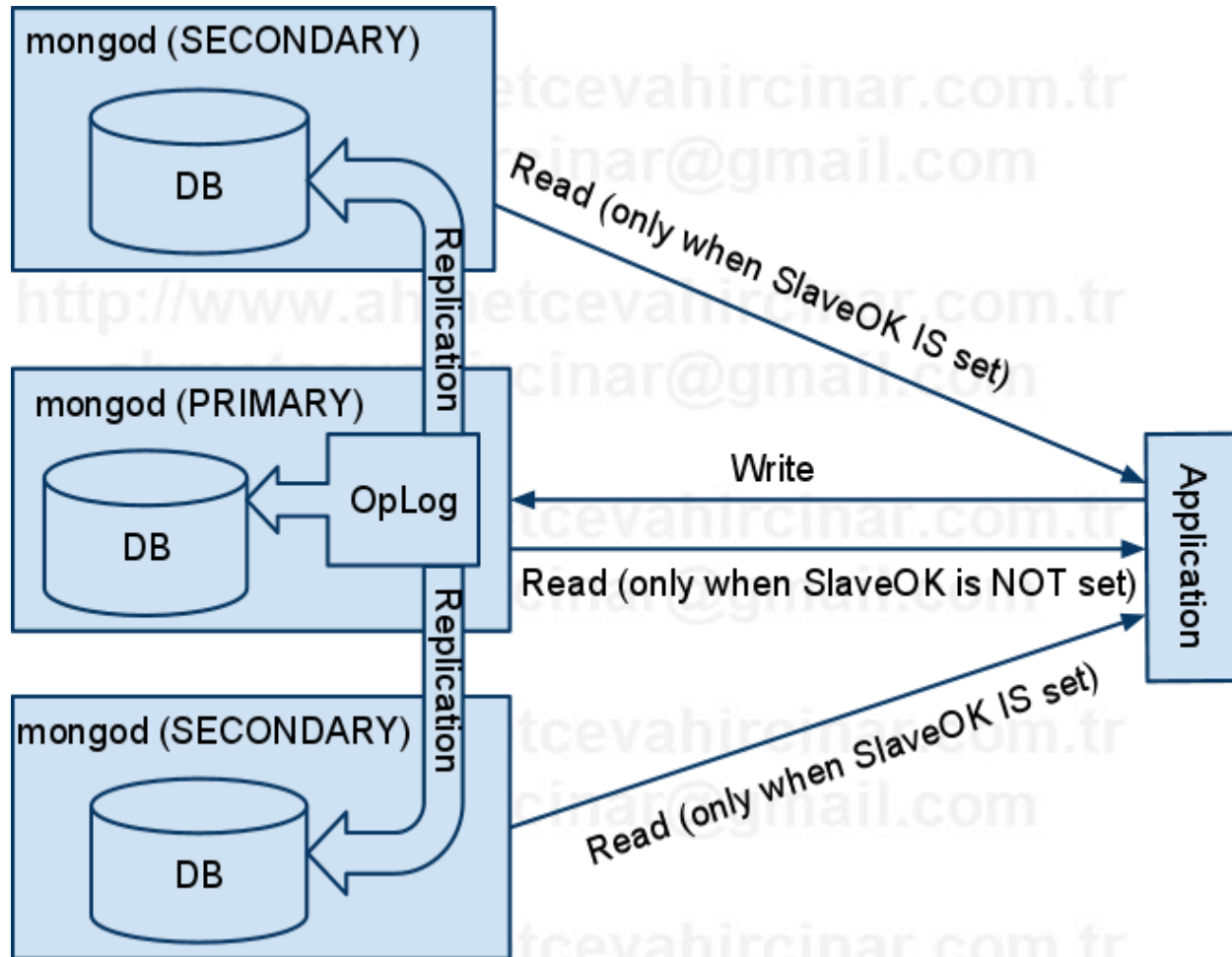
\x00

// 0x00 = type EOO ('end of object')

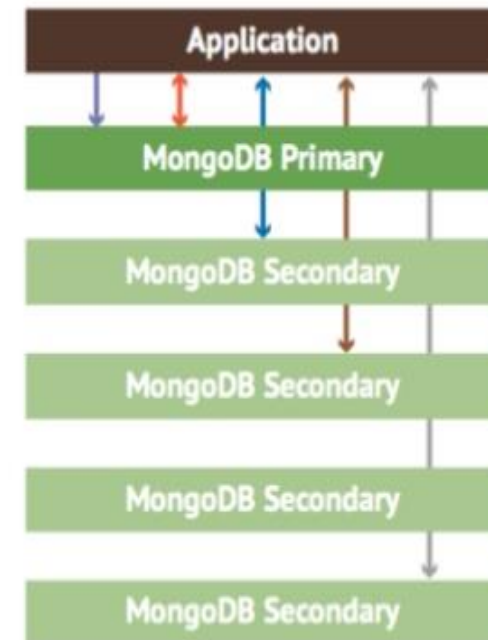
BSON, veri kümesinin boyutunu da sakladığından, çeşitli algoritmalar ile arama süresi azaltılabilmektedir. Dolayısıyla gezinme çok daha kolay gerçekleşmektedir.

x16 = 22 = 0001 0110
xFF = 255 = 1111 1111

Master-Slave Replication destekleri ile Master sunucu yazma işlemi yaparken Slave sunucu okuma işlemi yapmaktadır. Master sunucu herhangi bir sebepler cevap veremez durumda olursa Slave'lerden biri belirlenen koşullarda Master görevini üstlenmektedir.



Data Durability – Write Concerns



- Configurable per operation
- Default is ACK by primary

- Unacknowledged
- ↔ Acknowledged by Primary
- ↔ Acknowledged by Primary and 1 Secondary
- ↔ Acknowledged by Replica Set Majority
- ↔ Acknowledged by Replica Set Members

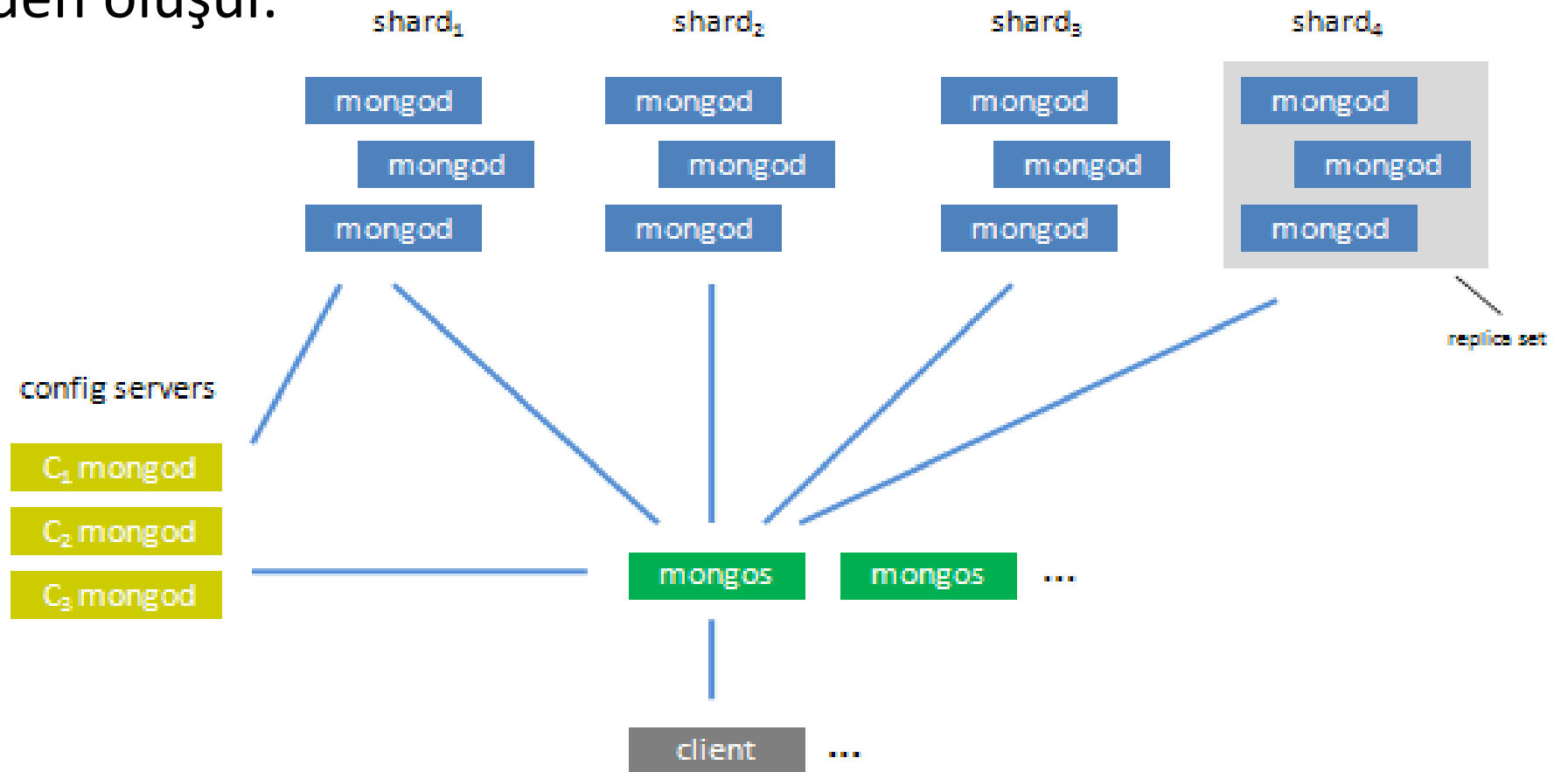
MongoDB'in en önemli özelliklerinden biri ise **Sharding**'dir. Replication yapısı ile beraber kullanılan bir sistemdir. Büyük veri ve yoğun şekilde veri yazma işlemi kullanılacaksa ve tek Mongod işleme yetiştirmiyorsa, sistem ölçek olarak yetmiyorsa “ram ve ya disk gibi” devreye Shard Mimarisi girer.

Shard üç ana bölümden oluşur.

Shards

Mongos

Config Server



Shards

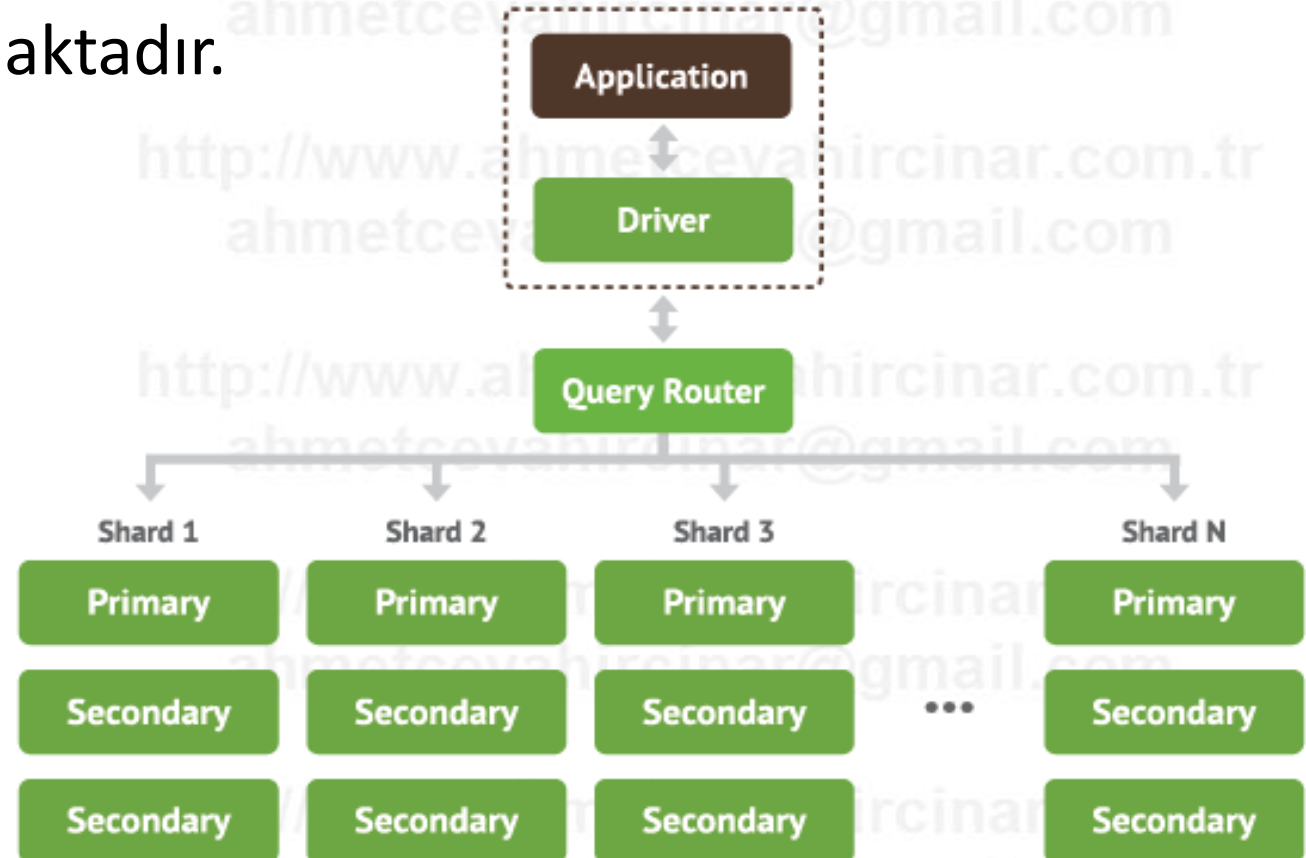
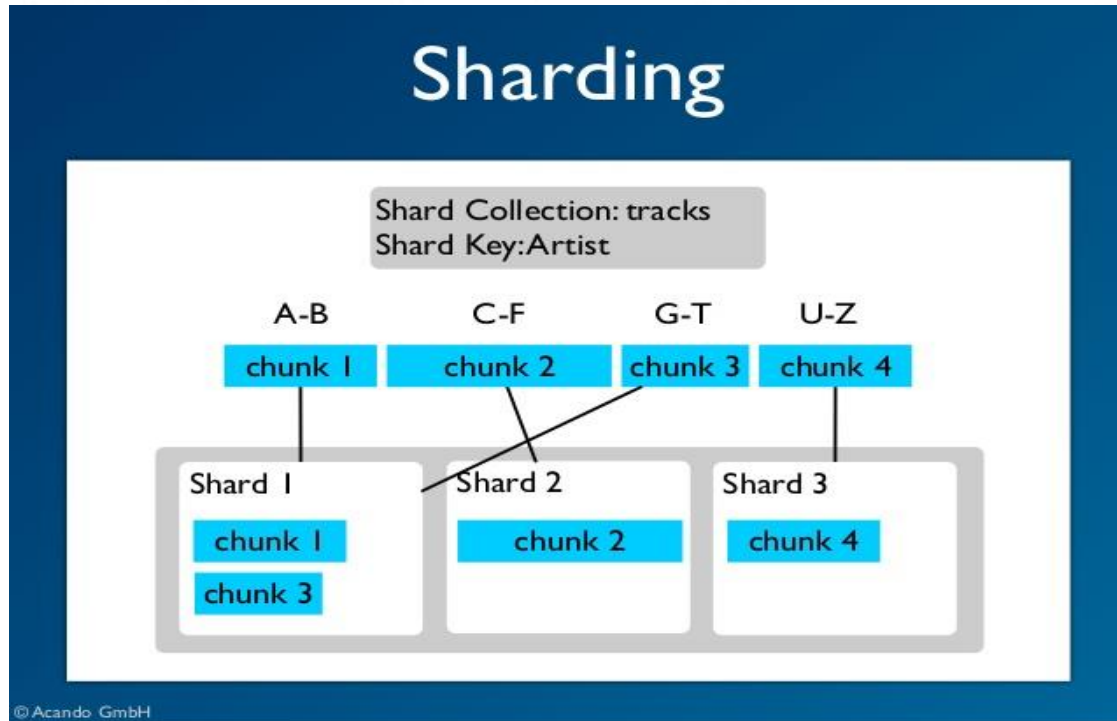
Verinin Mongod işlemlerine bölündüğü alandır.

Ayrı ayrı Mongod'ların yönetilmesini Mongos sağlar.

Her bir veri belirli Chunk'lara bölünür.

Aynı verinin bölünmüş Chunk'ları farklı Shard'larda barınır.

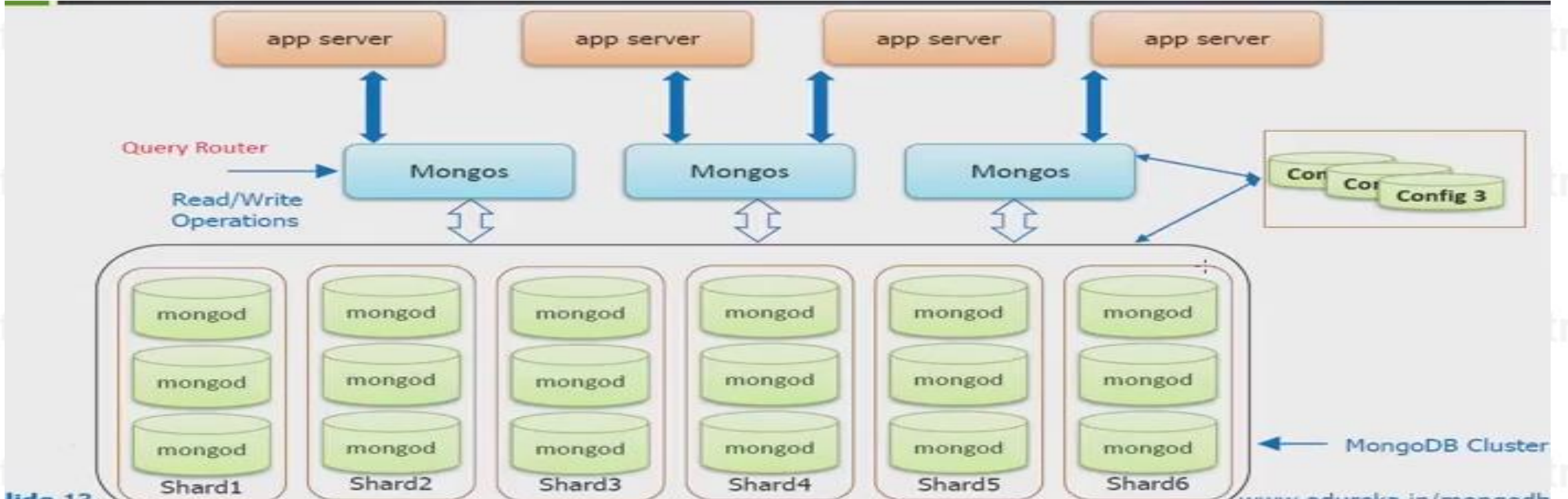
Her bir Shard Replica Set olarak çalışmaktadır.



Mongos “Query Router”

İstekler direk olarak Shard’larla bağ kuramazlar. Çünkü hangi veri hangi Shard’da bilemezler. Bu işi üzerine Mongos alır. Bu sebepler Query Router denmektedir.

İstek oluştuğunda Mongos devreye girerek ilgili Shard’lara görev yönlendirmesi yapar ve dönen sonuçları birleştirerek oluşan isteğin cevabını iletir. Mongos veri tutmaz sadece işlem yönetimini üstlenir. İşleyişle ilgili süreci takip edebilmesi için işlem süreç verilerine ihtiyaç tutar.



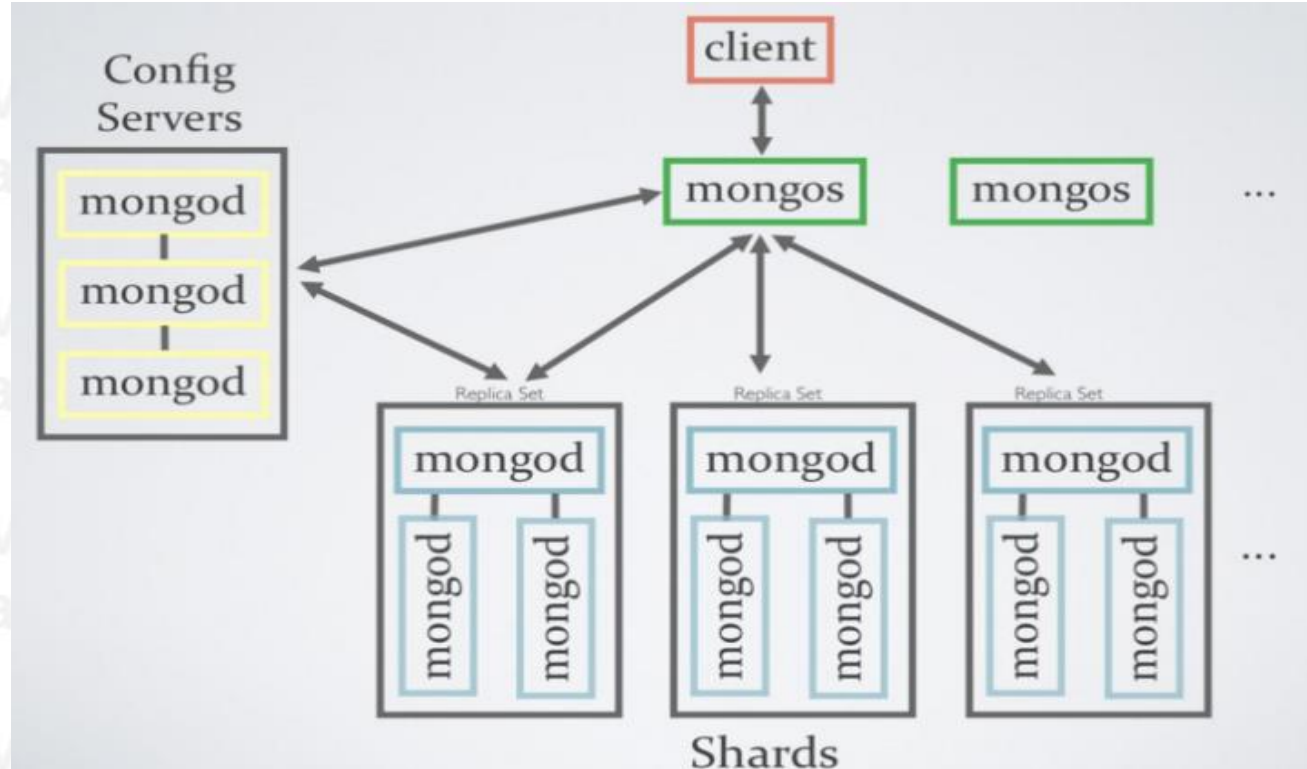
Config Server

Mongos'un görev sürecindeki işleme durumlarının veri olarak tutulduğu Mongod yapılandırma örneğidir.

Kısaca Cluster'ın tüm bilgisinin barındıran veri alanıdır.

Her 200MB 'lık veri için 1KB'lık Config Server alanı ayrılması önerilir.

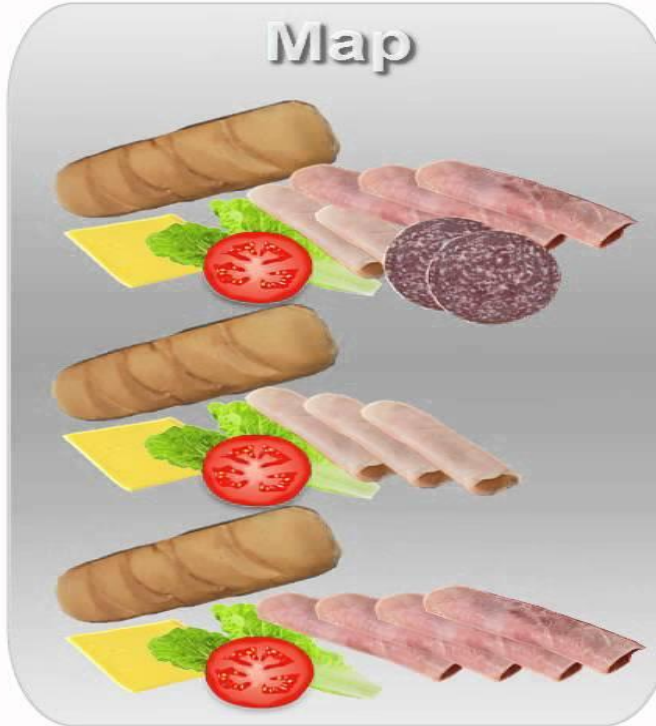
Mongos'a gelen isteklerle ilgili hangi Shard 'da hangi Chunk barınır bilgisini bununla beraber serverların durum bilgisini barındırırlar.



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



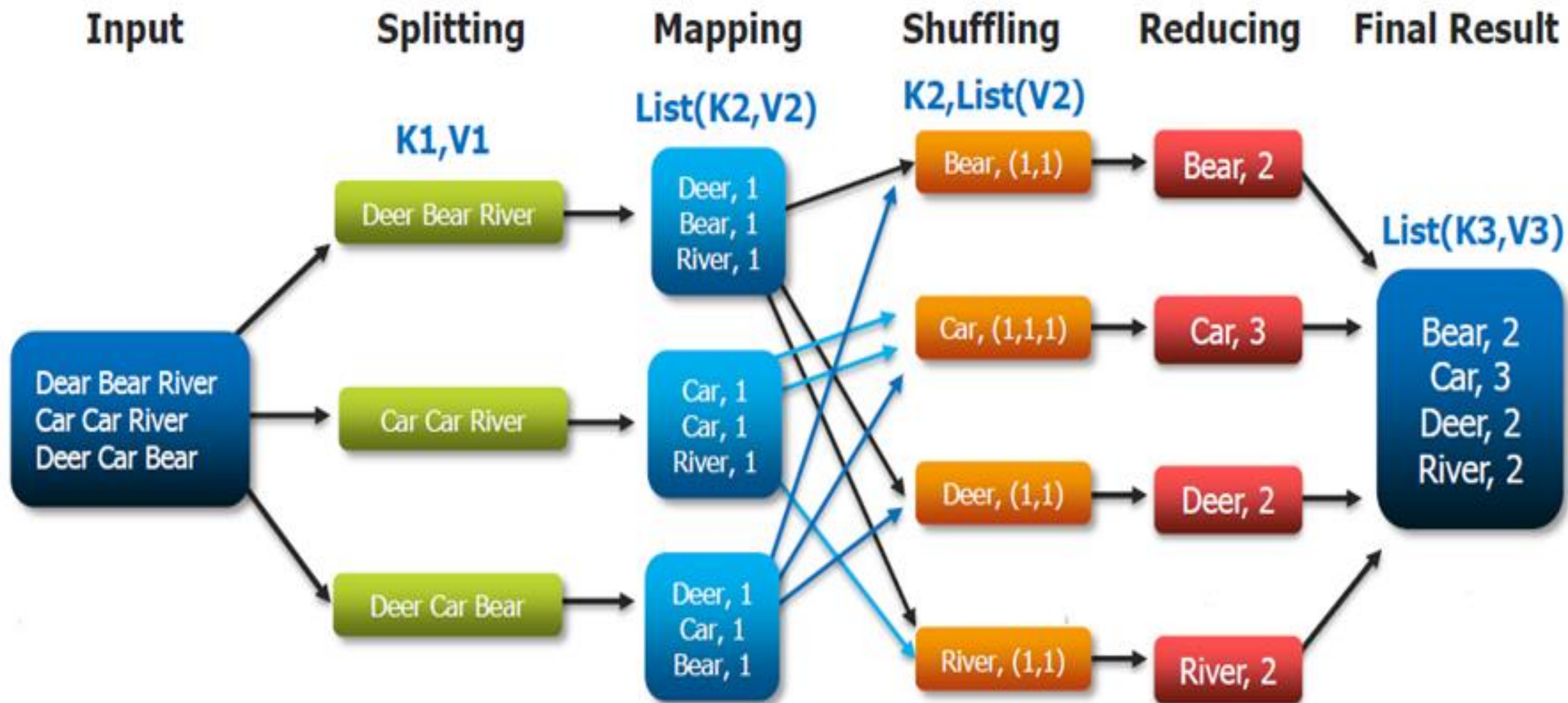
Map



Reduce



The Overall MapReduce Word Count Process

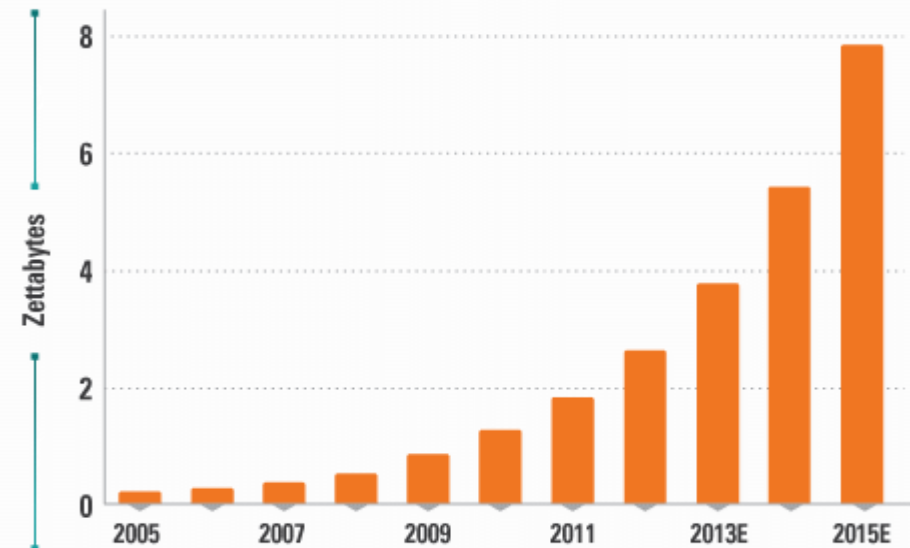


- Klasik veritabanı sistemleri ile Petabyte mertebesindeki verilerin işlenebilmesi ancak milyon dolar seviyesindeki donanım ve yazılım ile mümkün iken, MapReduce bu soruna çok ciddi bir alternatif durumundadır.

Bayt Birimleri						
Yaygın örnek				Binari örnek		
Ad	Sembol	Ondalık	İkilik	Ad	Sembol	İkilik
kilobayt	KB	10^3	2^{10}	kibibayt	KiB	2^{10}
megabayt	MB	10^6	2^{20}	mebibayt	MiB	2^{20}
gigabayt	GB	10^9	2^{30}	gibibyte	GiB	2^{30}
terabayt	TB	10^{12}	2^{40}	tebibayt	TiB	2^{40}
petabayt	PB	10^{15}	2^{50}	pebibayt	PiB	2^{50}
eksabayt	EB	10^{18}	2^{60}	eksbibayt	EiB	2^{60}
zettabayt	ZB	10^{21}	2^{70}	zebibayt	ZiB	2^{70}
yottabayt	YB	10^{24}	2^{80}	yobibayt	YiB	2^{80}

A Digital Data Explosion

Global digital information created and shared



Source: KPCB, IDC

techandinnovationdaily.com

RDBMS vs MongoDB

RDBMS vs MongoDB		
	RDBMS	MongoDB
Veri Yapısı	Sabit	Düzensiz
Join	Dahil	İç içe yapı
Transaction	Dahil	Atomic operasyonlar
Ölçeklenebilme	Dikey	Yatay
SQL Dili	Dahil	API 'ler ile
Primary Key	Var	Var
Foreign Key Constraint	Var	Referans

MongoDB verilerini **Collection** içerisinde Document olarak tutar. Veri yapılarını dikkate alırsak SQL veri tabanları ile karşılaştıırırsak Collection tabloya, Document ise satırlara karşılık gelir.

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

RDBMS'lerde veriler tablolarda, tanımlı sütunlarda satır satır bulunurken, NoSQL sistemler sabit tablo tanımlarına bağımlı değildirler. Yani daha sonradan özel bir işlem yapmadan yeni kolonlar eklenebildiği gibi, kayıtlar arasında kolon farklılıkları olabilir.

Örneğin 1. satırda a ve b kolonları varken 2. satırda bambaşka kolonlar bulunabilir.

Bu esnek yapı bize **denormalizasyon** kolaylığı sağlar.

RDBMS'lerde veriler ilişkilerine göre ayrı tablolara düzenli bir şekilde yerleştirilerek normalize edilirken, verilere erişim için birleştirme (**join**) kullanılır.

Dağıtık sistemlerde birleştirme operasyonu sıkıntılara sebep olur. Çünkü birleştirilmek istenen veriler farklı parçalarda olabilir. Bu yüzden dağıtık sistemlerde birleştirme işlemi elle yapılmalıdır. Yani önce a verisi çekilir, sonra bununla ilişkili b verisi çekilip programatik olarak birleştirilme yapılır. Normal sistemlerde bu işlemler performans kaybına sebep olurken, dağıtık sistemlerde bu performans kaybı sistemin genel performansı yanında önemsizdir. İşte bu sebeple dağıtık sistemlerde birleştirme operasyonu yapmak yerine veri denormalize tutularak verilere erişim kolaylaştırılır.

Denormalize edilmiş veriler içerisinde veriler kendini tekrar edecektir ancak bu da dağıtık sistemlerin çalışma performansını arttırdığı için önemli değildir. NoSQL sistemler de bu sebeple sabit tablo tanımı içermez, verilerin denormalize tutulmasını kolaylaştırır.

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

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.

```
{
  "_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"
}
```


ObjectId – 12 byte hexadecimal bir değerdir

507f191e-810c19-729d-e860ea

a 4-byte value representing the seconds since the Unix epoch,

a 3-byte machine identifier,

a 2-byte process id, and

a 3-byte counter, starting with a random value.

12 byte = 96 bit = 2^{96} = 79,228,162,514,264,337,593,543,950,336 adet
farklı kayıt numarası tutulabiliyor.

- NoSQL sistemler temel olarak verilere **tekil anahtarlar** üzerinden erişir. Her NoSQL sisteminde **ikincil indeks (secondary index)** yeteneği de bulunmayabilir. Bu yüzden verilere erişim SQL sorguları ile yapıldığı gibi kolay yapılamaz.
- Uygulamanın özelliğine göre birincil anahtarlar belirli bir mantığa göre verilir ve bu sayede veriye erişmeden önce zaten bu anahtar biliniyor ya da oluşturulabiliyor olur.
- Örneğin **zamana göre artan ön ek (prefix), kaydın anahtarı ile birleştirilerek** tek seferde ilgili kaydın belirli bir zaman aralığındaki kayıtlarına erişmek mümkündür.
- Bu sayede herhangi bir sorguya gerek olmaksızın tamamen **anahtarlar üzerinden verilere hızlıca erişim sağlanmış** olur.

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

```
SELECT *  
FROM people
```

MongoDB

```
db.people.find()
```


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 } })

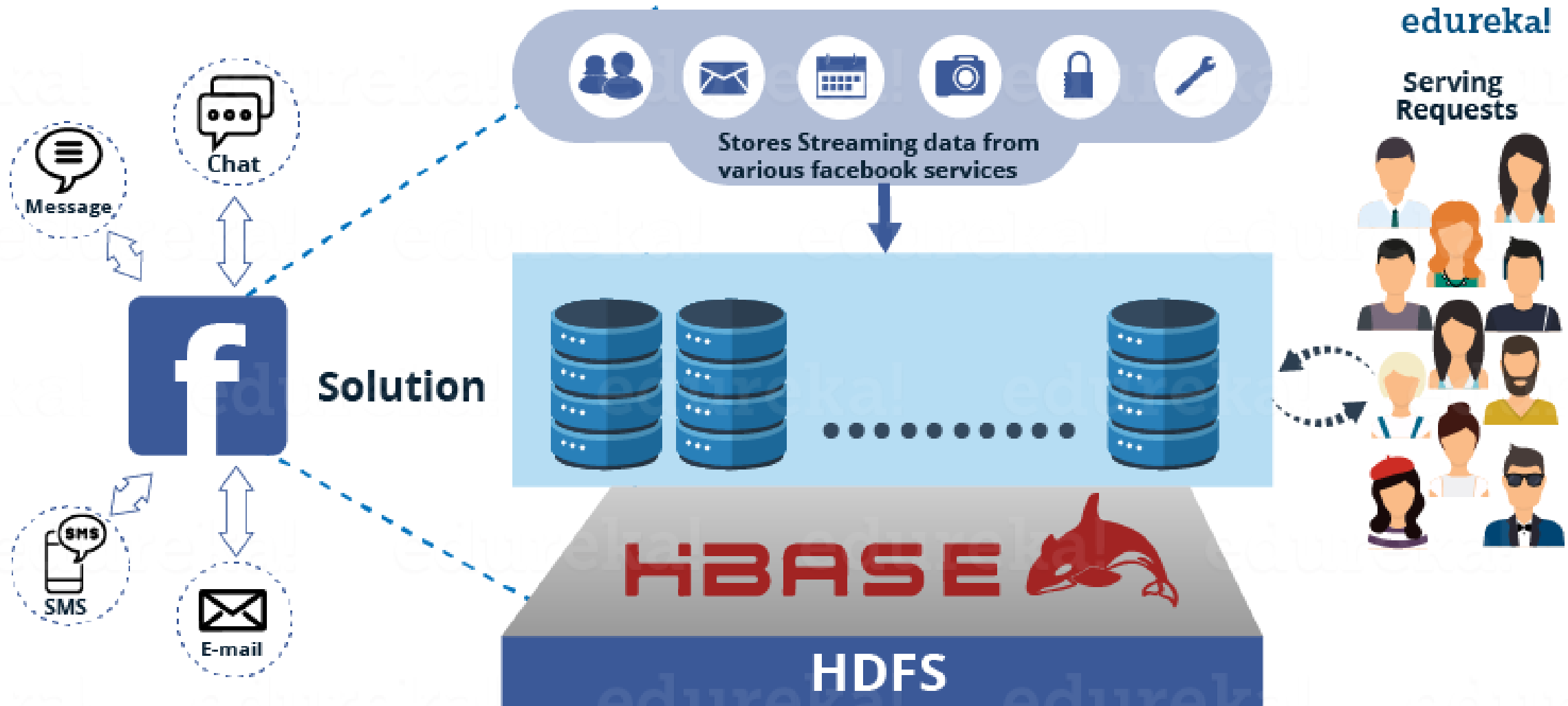
Silme nasıl yapılır?

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

Genel Tekrar

	SQL	NoSQL
Type	Relational	Non-Relational
Data	Structured Data stored in Tables	Un-structured stored in JSON files but the graph database does supports relationship
Schema	Static	Dynamic
Scalability	Vertical	Horizontal
Language	Structured Query Language	Un-structured Query Language
Joins	Helpful to design complex queries	No joins, Don't have the powerful interface to prepare complex query
OLTP	Recommended and best suited for OLTP systems	Less likely to be considered for OLTP system
Support	Great support	community depedent, they are expanding the support model
Integrated Caching	Supports In-line memory(SQL2014 and SQL 2016)	Supports integrated caching
flexible	rigid schema bound to relationship	Non-rigid schema and flexible
Transaction	ACID	CAP theorem
Auto elasticity	Requires downtime in most cases	Automatic, No outage required

Facebook mesajlaşma altyapısında **HBase** kullanıyor, bu sayede milyarlarca mesaj, chat mesajını saklayıp bize sorunsuz bir hizmet sunabiliyor.

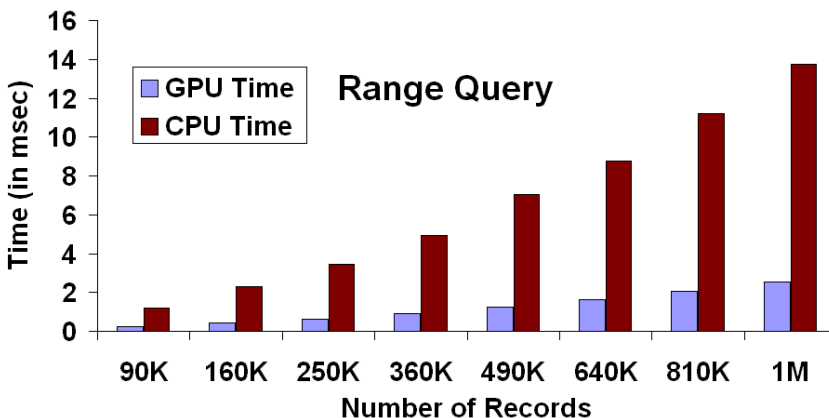
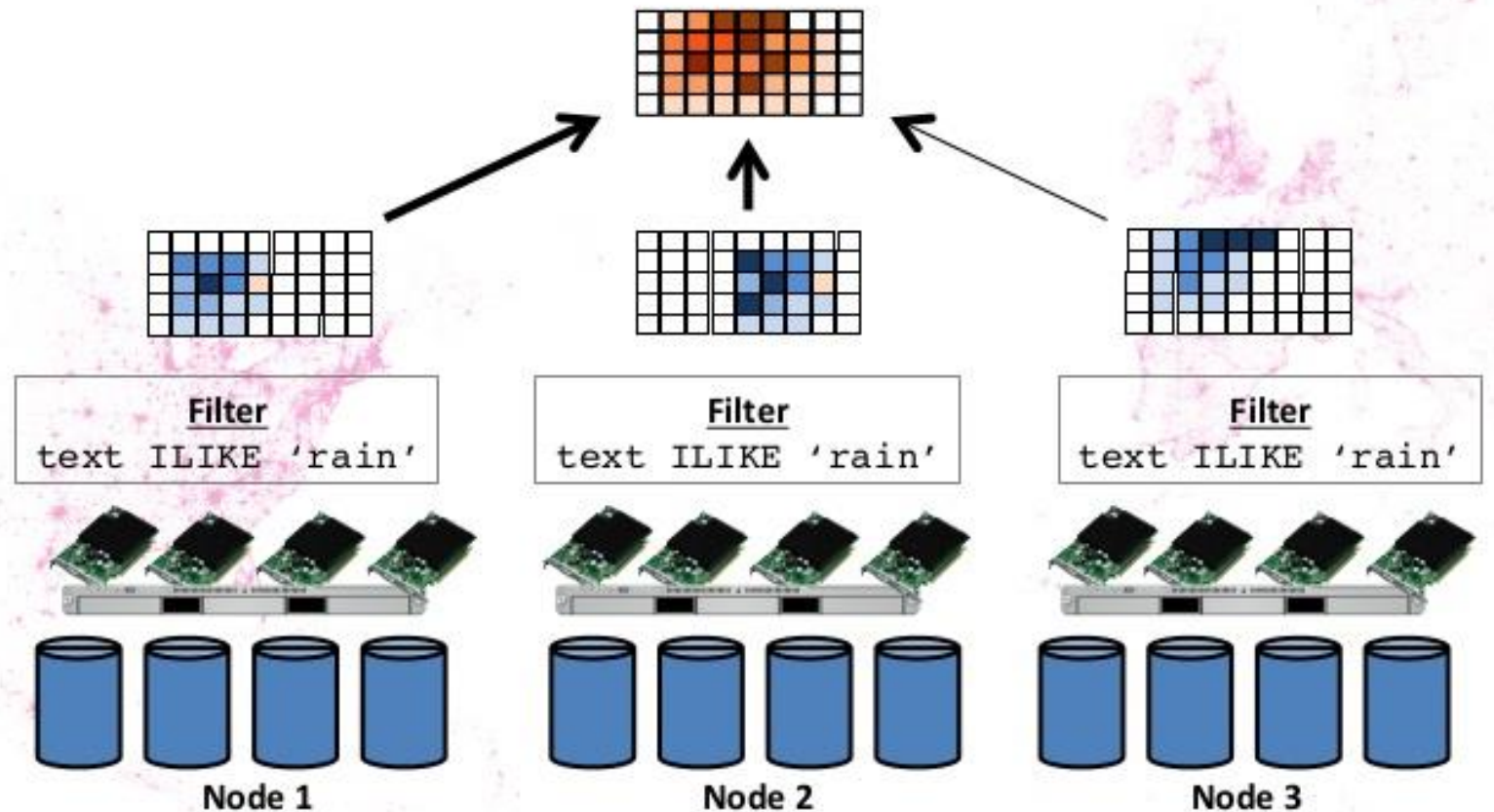


Ek Bilgi: GPU üzerinde çalışan veri tabanları

- MapD
- Kinetica
- BlazingDB
- Blazegraph
- PG-Strom

Shared Nothing Processing

Multiple GPUs, with data partitioned between them



Kaynaklar

- «No-SQL VERİ TABANLARI ÜZERİNDE BİR METİN MADENCİLİĞİ UYGULAMASI» Gürkan AYDIN'ın Yüksek Lisans Tezi
- <http://devveri.com/nosql-nedir>
- <https://medium.com/turkce/cap-teoremi-nedir-49a23e6d2e10>
- <http://ksat.me/a-plain-english-introduction-to-cap-theorem/>
- <https://kodcu.com/2013/04/nosql-kavrami-ve-mongodb/>
- <https://kodcu.com/2016/03/nosql-veritabanlari/>
- <https://kodcu.com/2016/03/hangi-nosql-veritabani/>