**T.C.**

**TRAKYA ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

# EN ÇOK KULLANILAN İLİŞKİSEL VE NOSQL VERİTABANI YÖNETİM SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

**BERNA DUMANLI**

**YÜKSEK LİSANS TEZİ**

**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**Tez Danışmanı: Dr. Öğr. Üyesi Altan MESUT**

**EDİRNE-2019**

# KABUL VE ONAY SAYFASI

Berna DUMANLI'nın hazırladığı "En Çok Kullanılan İlişkisel ve NoSQL Veritabanı Yönetim Sistemlerinin Performans Karşılaştırması" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Yüksek Lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad): İmza

Dr. Öğr. Üyesi Deniz TAŞKIN .....

Dr. Öğr. Üyesi Bora ASLAN .....

Dr. Öğr. Üyesi Altan MESUT .....

Tez Savunma Tarihi: 25/01/2019

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

Dr. Öğr. Üyesi Altan MESUT İmza

Tez Danışmanı .....

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

(Ünvan, Ad, Soyad)

Prof. Dr. Murat YURTCAN

Fen Bilimleri Enstitüsü Müdürü

.....

**T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI**

# DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

07/02/2019

Berna DUMANLI

imza

Yüksek Lisans Tezi

En çok kullanılan İlişkisel ve NoSQL veritabanı yönetim sistemlerinin performans karşılaştırması

T.Ü Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

# ÖZET


Veritabanı yönetim sistemleri, sağladıkları yararlar sebebiyle uzun yıllardır kullanılmaktadırlar. Büyük miktarda verinin, hızlı bir şekilde depolanmasında son yıllarda NoSQL veritabanları tercih edilmektedir. Bu sistemler ölçeklenebilme kolaylığı ve hızlı depolama yönünden avantajlı olmasına rağmen, ACID modelini tam olarak desteklememeleri ve bazı sorgularda yavaş kalmaları sebebiyle kullanım alanı kısıtlı olmaktadır. Bu dezavantajlara rağmen yeni sürümleriyle her geçen gün yeni özellikler kazanmakta ve NoSQL veritabanlarının kullanımı artmaktadır.

Bu tezde, en çok kullanılan NoSQL veritabanı sistemlerinden üç tane ve en çok kullanılan ilişkisel veritabanlarından üç tane seçilerek sistemlerin yeteneklerini ve farklı işlemlerde nasıl tepkiler verdiklerini ortaya çıkarmak için testler yapılmıştır. Bu amaçla, birçok iş yükü yüklenmiş ve iki adet test ortamı hazırlanmıştır. Bu çalışmanın sonuçları kullanılan her bir sistemin zayıf ve güçlü yanlarını ortaya koymaktadır. Test edilen her bir veritabanı farklı mimari ve özelliklere sahip olduğu için farklı sonuçlar ortaya çıkarmıştır.

Karşılaştırmada, ilk olarak Yahoo tarafından geliştirilen Yahoo Cloud Serving Benchmark (YCSB) kullanılmıştır. Elde edilen sonuçlara göre, NoSQL sistemler içinde MongoDB en iyi performansı gösterirken, Couchbase de ona yakın bir performans göstermiştir. Çok düğüme sahip kümelerde performansı yüksek olan Cassandra ise tek düğüm üzerinde beklenen performansı gösterememiştir. Test edilen ilişkisel veritabanları MySQL, SQL Server ve Oracle ise okuma performansı olarak NoSQL sistemler ile yakın olsa da, ACID özelliğine sahip olmaları nedeniyle yazma ve güncellemede düşük performans göstermişlerdir.

İkinci test olarak, veritabanlarına farklı büyüklüklerde veriler yüklenip, farklı türden sorguları ne kadar sürede tamamladıkları ölçülmüştür. En hızlı yanıtları ilişkisel veritabanlarından SQL Server ve Oracle vermiştir. MongoDB testimizdeki diğer ilişkisel veritabanı olan MySQL'e yakın sonuçlar verebilmiştir. Cassandra ve Couchbase ise bu testte iyi bir performans gösterememiştir.



Yıl : 2019

Sayfa Sayısı : 64

Anahtar Kelimeler : İVTYS, NoSQL, MongoDB, Cassandra, Couchbase, SQL Server, Oracle, MySQL

Master's Thesis

Performance comparison of most used Relational and NoSQL database management systems

Trakya University Institute of Natural Sciences

Computer Engineering Department

# ABSTRACT

Database management systems have been used for long time due to benefits they provide. NoSQL databases are preferred for fast storage of big amounts of data. These systems are advantageous in terms of ease of scalability and fast storage. However, it does not support the ACID model and suggests that it should be slow in some queries. Despite these disadvantages, new versions get new features and the use of NoSQL databases increase every passing day.

In this thesis, the most commonly three NoSQL database systems and relational databases are used to test the capabilities of the systems and react in different processes. For this purpose, many workloads are loaded and two test platform are prepared. The results of this study reveal the weaknesses and strengths of each used system. Different results have occurrenced because each tested database has different acrhitecture and features.

In comparison, Firstly Yahoo Cloud Serving Benchmark (YCSB), Yahoo who developed, used. According to the results, MongoDB performed the best in NoSQL systems also Couchbase performed well. Cassandra showed high performance in multi-node clusters but has not been able to achieve desired performance on a single node. Relational databases that tested for this thesis(MySQL, SQL Server, Oracle) have reading performance close to nosql databases but because they have ACID , they performed poorly on writing and updating.

As the second test, data of different sizes uploaded to databases and queries of different types used and their completion times measurement. SQL server and Oracle have given the fastest responses as Relational databases. MongoDB was able to give results close to MySQL, which is the other relational database in our test. Cassandra and Couchbase did not perform well in this test.

Year : 2019

Number of Pages : 64

Keywords : RDBMS, NoSQL, MongoDB, Cassandra, Couchbase, SQL Server, Oracle, MySQL

# TEŞEKKÜR

Bu tezde en çok kullanılan ilişkisel ve NoSQL veritabanlarının performans karşılaştırması amaçlanmış, bu yönde çalışmalar yapılmıştır.

Tez çalışmamın konusunun belirlemesi ve hazırlanma sürecinin her aşamasında bilgi ve tecrübelerini benden esirgemeyen, kendisine danıştığımda değerli vaktini ayırarak her fırsatta çalışmamla ilgili yol gösteren ve çözümler sunan değerli danışman hocam Dr. Öğr. Üyesi Altan MESUT'a sonsuz saygı ve teşekkürlerimi sunarım.

Tez çalışmam boyunca bilgi ve yardımlarını benden esirgemeyen Arş. Gör. Dr. Emir ÖZTÜRK'e teşekkür ederim.

Son olarak her zaman yanımda olan, bugünlere gelmemde büyük katkıları olan, eğitim hayatım boyunca maddi ve manevi desteklerini üzerimden esirgemeyen sevgili annem Gülay, babam Necdet ve kardeşim Bensu DUMANLI'ya sonsuz sevgilerimle.

# İÇİNDEKİLER

# SİMGE VE KISALTMALAR DİZİNİ

| | | |
|---|---|---|
| µs | : | mikro saniye |
| op | : | operasyon |
| sn | : | saniye |

| | | |
|---|---|---|
| ACID | : | Atomicity, Consistency, Isolation, Durability |
| BASE | : | Basically Available, Soft State, Eventualy Consistency |
| BSON | : | Binary JSON |
| CAP | : | Consistency, Availability, Partition Tolerance |
| CAS | : | Check and Set |
| CQL | : | Cassandra Query Language |
| DCL | : | Data Control Language |
| DDL | : | Data Definition Language |
| DML | : | Data Manipulation Language |
| FK | : | Foreign Key |
| FTS | : | Full-Text Search |
| JSON | : | Javascript Object Notation |
| RAM | : | Random Access Memory |
| RDBMS | : | Relational Database Management System |
| SDK | : | Software Development Kit |
| SEDA | : | Staged Event-Driven Architecture |
| SQL | : | Structured Query Language |
| YCSB | : | Yahoo Cloud Serving Benchmark |

# ÇİZELGELER DİZİNİ

# ŞEKİLLER DİZİNİ

# BÖLÜM 1

# GİRİŞ

Günümüzde teknolojinin gelişmesi, akıllı cihaz ve evlerin artması, uygulamaların artmasıyla beraber ortaya pek çok veri çıkmaktadır. Dünyadaki insanların %55,1'i internet kullanmaktadır(https://www.internetworldstats.com/stats.htm). Bunun sonucunda ortaya çıkan verilerin boyutları da sürekli artmaktadır. Internet ortamındaki veriler genellikle yapısal olmayan türdedir. Yapısal türde verileri saklamak için en yaygın kullanılan veritabanı yönetim sistemleri İlişkisel Model (Codd, 1970) kullanırken, yapısal olmayan veriler için ilişkisel modeli kullanmayan NoSQL veritabanları tercih edilmektedir. NoSQL veritabanı sistemleri de verileri depolamak için kullandıkları modeller ve sahip oldukları özellikler ile birbirlerinden ayrılmaktadır.

İlişkisel veritabanlarında veriler satırlar ve sütunlardan oluşan tablo yapısında saklanmaktadır. Verilerin tutarlılığını sağlamak için ACID özelliği bulunduğundan bankacılık gibi hataya yer olmayan kritik işlemlerde kullanımı vazgeçilmezdir. Fakat ölçeklenebilme maliyetinin yüksek olması ve veri yazma hızlarının yavaş olması nedeniyle, büyük verileri işleme ihtiyacı olan Google, eBay ve Amazon gibi şirketler tarafından son yıllarda NoSQL veritabanları tercih edilmiştir (Gökşen & Aşan, 2015). Ölçeklenebilme kolaylığı ve hızlı depolama avantajları tercih sebebi olurken, ACID prensibini desteklememeleri ve bazı kapsamlı sorguları yapamamaları dezavantaj olarak gösterilmektedir (Otey, 2010).

NoSQL veritabanları kendi içerisinde sınıflandırılmaktadır. Bu veritabanlarının bazı yönleri güçlü bazı yönleri de zayıftır. Farklı özelliklere ve teknik üstünlüklere sahip bu geniş aileden uygun olanı seçmek çok önemlidir. NoSQL veritabanı türlerinin veri yapısı, sorgu modeli ve veri tutarlılığı açısından incelenerek, hangi uygulamalar için uygun olduklarının belirlenmesi amaçlanmaktadır.

NoSQL veritabanı yönetim sistemlerinin performans karşılaştırması üzerine son yıllarda yapılmış birçok tez çalışması bulunmaktadır. Bu tezlerden birinde en çok

kullanılan NoSQL veritabanları çeşitli özellikleri yönünden incelenmiş ve ilişkisel veritabanları ile farkları ortaya konmuştur (Aladily, 2015). YCSB (Yahoo Cloud Serving Benchmark) üzerinden CouchDB, MongoDB, Hbase ve Cassandra üzerinde performans karşılaştırmaları yapılmıştır ve iş yüklerinin çoğunda Cassandra üstün performans göstermiştir.

YCSB ile testlerin yapıldığı başka bir tez çalışmasında ise MongoDB, Cassandra ve Hbase üzerinde performans karşılaştırmaları yapılmıştır (Hammood, 2016). Bu çalışma MongoDB'nin düşük yükler ile iyi performans gösterdiğini, Cassandra ve Hbase'in ise ağır yükler altında iyi performans gösterdiğini ortaya koymuştur.

NoSQL veritabanlarının yapısal farklarının anlatıldığı ve benzer şekilde yine YCSB kullanılarak MongoDB, Cassandra, Hbase ve Riak veritabanlarının performans testlerinin yapıldığı bir başka tez çalışmasında ise, MongoDB'nin okuma işlemlerinde başarılı olmasına rağmen yazma ve güncelleme ağırlıklı iş yüklerinde geride kaldığı görülmektedir (Singh, 2015). Ağır iş yükleri altında yine Cassandra ve Hbase iyi performans göstermiştir.

NoSQL veritabanlarının en önemli özelliklerinden biri olan ölçeklenebilme özelliği de bir başka tez çalışmasında test edilmiştir (Swaminathan, 2015). MongoDB, Cassandra ve Hbase yine YCSB üzerinde bu defa daha çok düğüm ile performans testine tabi tutulmuştur. İş yüklerinin çoğunda Cassandra daha iyi performans göstermiştir.

İlişkisel veritabanı MySQL ile ilişkisel olmayan MongoDB ve Cassandra arasında performans karşılaştırması yapan tez çalışmasında ise, MongoDB yine okuma ağırlıklı iş yüklerinde başarılı olurken, yazma ve güncelleme ağırlıklı iş yüklerinde Cassandra'nın gerisinde kalmıştır (Taha, 2017). MySQL ise hem okuma hem de yazma işlemlerinde daha kötü performans göstermiştir.

Haziran 2018'de yayınlanmış olan bir tezde, ilk aşamasında MongoDB ve Cassandra YCSB 0.12.0 sürümü kullanılarak test edilmiş, sonuçlar grafik ile ortaya konulmuştur. İkinci aşamasında güvenlik ele alınmıştır (Shwaysh, 2018).

Daha yeni bir NoSQL VTYS olan Couchbase'in diğer VTYS'ler ile kıyaslandığı herhangi bir tez çalışması yapılmamıştır. Eylül 2014'te Altoros şirketinin yayınladığı bir test raporunda Couchbase, Cassandra ve MongoDB hem mimari, yönetim, yayılma, geliştirme ve ölçeklenebilirlik başlıkları altında bazı teknik özelliklerine göre

kıyaslanmış, hem de YCSB ile performans testlerine tabi tutulmuştur (Altaros, 2014). Bu raporda Couchbase diğerlerine göre daha üstün bulunmuştur.

Daha önce sadece bir tez çalışmasında MySQL ile NoSQL VTYS'ler kıyaslanmışken, bu tezde diğer tezlerden farklı olarak en çok kullanılan ilişkisel veritabanları olan MS SQL Server ve Oracle gibi daha güçlü VTYS'ler de test edilmiştir. Ayrıca, en sık karşılaştırılan iki NoSQL veritabanı olan MongoDB ve Cassandra'nın yanı sıra resmî sitesinde dünyanın en güçlü NoSQL veritabanı olduğunu iddia eden Couchbase de teste dâhil edilmiştir. Tezin amacı hem ilişkisel ve NoSQL veritabanlarının zayıf ve güçlü yanlarını analiz etmek ve performans farklılıklarını ortaya koymak, hem de kendi türlerindeki rakipleri arasında kıyaslamaktır. Diğer tez çalışmalarından farklı olarak bu çalışmada sadece YCSB testleri ile yetinilmemiş, farklı türde sorgular üzerindeki performansları da ortaya konmuştur.

Tezin ikinci bölümünde, detaylı bir literatür çalışması sonucunda ölçeklendirme, ACID ve BASE modelleri ele alınmıştır.

Üçüncü bölüm ilişkisel veritabanları yönetim sistemlerinin açıklandığı bölümdür. Bu bölümde ilişkisel veritabanlarının yapısından, kavramlarından, avantaj ve dezavantajlarından bahsedilmiş ve test ortamında kullandığımız MySQL, MS SQL Server ve Oracle ile ilgili bilgiler verilmiştir.

Dördüncü bölüm NoSQL veritabanı yönetim sistemleri bölümüdür. NoSQL VTYS tanımı yapılmış, avantajları ve dezavantajları, türleri açıklanmıştır. Bu bölümde test etmek için kullanacağımız veritabanları olan MongoDB, Cassandra ve Couchbase, veri modelleri, mimarileri, avantaj ve dezavantajları ve sağlayabildikleri ACID özellikleri ile ele alınmıştır.

Beşinci bölüm seçilen veritabanlarının karşılaştırmasının yapıldığı bölümdür. Karşılaştırmanın yapıldığı test ortamının özellikleri ve teste başlamadan yapılan konfigürasyon işlemleri verildikten sonra, biri YCSB kullanılarak yapılan, diğeri kendi yarattığımız veriler üzerinde çeşitli sorgular yürüterek yapılan iki farklı performans değerlendirmesinin sonuçları bu bölümde verilmiştir.

Son bölümde ise yapılan testlerin sonuçları analiz edilmiş ve sonuç değerlendirmesi yapılmıştır.

# BÖLÜM 2

# KAVRAMLAR

## 2.1. Ölçeklendirme

Ölçeklenebilirlik, bir sistemin artan miktarda işle başa çıkmak, işlem kabiliyeti ve büyümeye uyum sağlaması amacıyla genişletme potansiyeli olarak tanımlanmaktadır (Bondi, 2000). Kaynaklar eklendiğinde artan bir yük altında toplam çıktı artmakta ise sistem ölçeklenebilir olarak kabul edilmektedir. Ölçeklenebilirlik 2 şekilde yapılabilmektedir:

Dikey Ölçeklendirme (Vertical Scaling – Scale Up): Tek bir cihaza donanım yatırımı yapmak anlamına gelmektedir. Örneğin CPU, hafıza ve disk artırımı yapmak bu kapsamdadır. Sanallaştırma ile bu yöntem daha pratik hale gelmiş olsa da donanım yatırımı yapmak pahalı ve riskli olabilmektedir (Swaminathan, 2015).

Yatay Ölçeklendirme (Horizontal Scaling – Scale Out): Nispeten ucuz cihazları birleştirerek güçlü bir sistem oluşturma esasına dayanır. Böyle bir sistemi büyütüp küçültmek daha az maliyetli ve daha az risklidir (Swaminathan, 2015).

## 2.2. ACID ve BASE Modelleri

### 2.2.1. ACID Modeli

İlişkisel veritabanlarında veride değişiklik yapan işlemler hareket (transaction) olarak bilinir. Her hareket diğer hareketlerden bağımsız olarak ve onlardan etkilenmeden yürütülmelidir. ACID modelinde tutarlılık ön plandadır ve veri kaybı yaşanması istenmemektedir. Hızdan daha çok veri kaybı olmaması istenmektedir. ACID baş harflerini aldığı şu prensipleri garanti etmektedir:

- **Atomicity (Atomiklik):** Hareket işlemini bir bütün olarak görür. İşlem sırasında birden fazla veritabanı/tablodaki verinin güncellenmesi gerçekleşiyor ise tüm bunların hepsi birden başarılı olacaktır veya başarısız olacaktır. Veritabanları

erişilemez olabilir, network problemi olabilir ya da herhangi bir hata oluşabilmektedir. Böyle durumlarda işlem geçersiz sayılacaktır.

- **Consistency (Tutarlılık):** Hareket işlemi sonucunda veritabanındaki verinin geçerli durumunun, bir sonraki geçerli duruma geçmesidir. Hareket tam anlamı ile gerçekleşinceye kadar işlemden etkilenen veriler bir önceki geçerli değerlerini vermelidir. Veritabanına yazılan tüm veriler, kısıtlamalar ve tetikleyiciler tarafından tanımlanmış tüm kurallara göre geçerli olmalıdır.

- **Isolation (Yalıtım):** Aynı anda aynı veri üzerinde birden fazla hareket değişiklik yapmak isteyebilir. Hareketler birlikte işletildiğinde, henüz tamamlanmamış (ve belki de tamamlanmayarak geriye alınacak) bir hareket tarafından gerçekleştirilen değişiklik işlemleriyle oluşturulan veri değerlerinin diğer hareketler tarafından görülmemesi yalıtım özelliğidir.

- **Durability (Dayanıklılık):** Hareket sırasında fiziksel veya işlemsel bir hata olması durumunda sistemin kendisini bir önceki geçerli veri durumuna döndürebilme kabiliyetidir.

### 2.2.2.BASE Modeli

İlişkisel olmayan veritabanları, yani NoSQL'de bulunan bir modeldir. ACID'in tam tersi özellikler göstermektedir. Tutarlılık ön planda değildir. BASE baş harflerini aldığı şu kavramları temsil etmektedir (Gökşen & Aşan, 2015):

- **Basically Available (Basit Kullanılabilirlik):** Sistem CAP [1] teoremine uygun olarak sürekli çalışır. Her bir talebe cevap vermektedir. Fakat bu zorunluluk bir hata durumunda bile geçerli olduğu için veri tutarlılığını garanti etmez ve tüm veriye erişimi mümkün kılmamaktadır. Kısacası verinin bir kısmından feragat ederek daha basit bir erişilebilirlik sağlanmaktadır.

[1] *Eric Brewer tarafından 1990'lı yıllarda ortaya atılan CAP teoremi; Consistency, Availability, Partition Tolerance (Tutarlılık, Kullanılabilirlik, Bölüm Toleransı) üçlüsünden en fazla iki tanesinin aynı anda garanti edilebileceğini ifade eder. Buradaki tutarlılık, ACID'te yer alan kavrama göre biraz farklıdır (her okuma isteği, en son yazılan en güncel veriyi veya bir hata almalıdır).*

- **Soft State (Esnek Durum):** Veriler yazılırken tutarlı olmayabilmektedir. Bu geliştiricinin görevi olarak görünmektedir. Ayrıca verilerin tüm cihazlarda aynı şekilde görünmesi garanti edilmemektedir.
- **Eventualy Consistency (Nihai Tutarlılık):** İşlemlerin etkileri sistemin durumuna bağlı olarak ancak bir süre sonra diğer cihazlara yansımaktadır. Neredeyse tutarlı bir sistemdir.

# BÖLÜM 3

# İLİŞKİSEL VERİTABANI YÖNETİM SİSTEMLERİ

## 3.1. Tarihçesi

1970 yılında Edgar Frank Codd tarafından öne sürülen ilişkisel veri modeline (Codd, 1970) dayanan bir veritabanlarına İlişkisel Veritabanı (Relational Database) denir. İlişkisel veritabanlarını yönetmek için kullanılan çeşitli yazılım sistemleri de İlişkisel Veritabanı Yönetim Sistemi (RDBMS: Relational DataBase Management System (Codd, 1970) olarak bilinmektedir. İlişkisel veritabanları ortaya çıkmadan önce *hiyerarşik veri modeli* ve *ağ veri modeli* gibi farklı modellere dayalı veritabanı sistemleri kullanılmaktaydı. Bunların dışında, veri bütünlüğünün programcı tarafından sağlanmasını gerektiren klasik dosya sistemlerinde verileri saklamak ta başka bir alternatifti. İlişkisel veritabanı yönetim sistemlerinin 80'li yıllarda yaygınlaşmasıyla diğer alternatiflerin kullanımı oldukça azalmıştır. 90'lı yıllarda ilişkisel modele nesne, sınıf ve miras gibi kavramların dâhil edilmesiyle nesne-ilişkisel veritabanı yönetim sistemleri (ORDBMS: Object-RDBMS) de ortaya çıkmıştır.

## 3.2. İlişkisel Veri Modeline Ait Temel Kavramlar

**Tablo (Table):** İlişkisel veri modelinde veriler, satır ve sütunlar hâlinde tablolarda saklanır. Tablonun her satırı bir kayıt, her sütunu ise o kayda ait bir nitelik verisini saklar. Örneğin öğrenci kayıtlarını saklamak için Öğrenciler adında bir tablomuz olduğunu ve bu tablonun her öğrenciye ait Öğrenci No, Adı, Soyadı, Doğum Tarihi ve Doğum Yeri verilerini (nitelikleri) saklayacağını düşünelim. Tablonun nitelik şeması olarak gösteriminde tablo adından sonra parantez içinde aralarına virgül konarak nitelik isimleri verilebilir: *Öğrenciler (ÖğrenciNo, Adı, Soyadı, DoğumTarihi, DoğumYeri)*. Niteliklerin gösterildiği sütunlara alan ismi verilir. Çizelge 3.1'de içinde 4 farklı öğrenciye ait kaydın saklandığı Öğrenciler tablosu verilmiştir.

**Çizelge 3.1.** Öğrenciler Tablosu

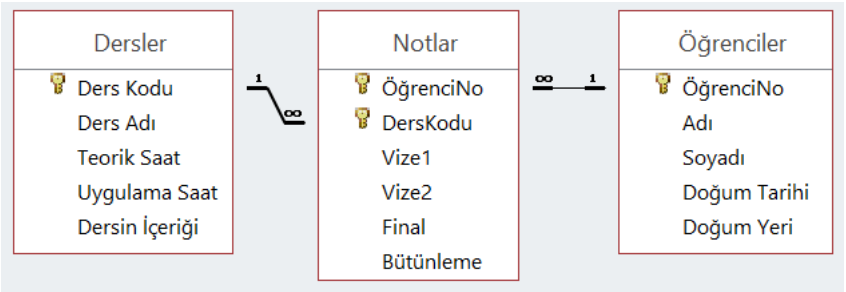
| ÖğrenciNo | Adı | Soyadı | Doğum Tarihi | Doğum Yeri |
|---|---|---|---|---|
| 1010302032 | Gonca | Şahin | 3.6.1994 | Samsun |
| 1010302045 | Kemal | Sarıca | 24.5.1995 | İstanbul |
| 1010302092 | Ali | Yılmaz | 20.3.1994 | Tekirdağ |
| 1010302095 | Selim | Çetin | 27.4.1994 | Edirne |

**Anahtar (Key):** Birincil Anahtar (PK: Primary Key), Yabancı Anahtar (FK: Foreign Key) ve Tekil Anahtar (UK: Unique Key) olmak üzere 3 çeşit anahtar vardır. Birincil anahtar bir kaydı başka bir kayıttan ayırmamızı sağlayan, yani her kayıtta diğerinden farklı bir değere sahip olması gereken niteliktir (örneğimizde ÖğrenciNo). Dış anahtar ise, aynı veya başka bir tablodaki birincil anahtar ile ilişkide olan ve o birincil anahtar sütununda yer alan değerler dışında bir değere sahip olamayan niteliktir. Birincil anahtardan farklı olarak dış anahtarda aynı değer başka kayıtlarda tekrar edebilir. Tekil anahtar olarak belirlenen bir alanda bir değer sadece bir kayıtta yer alabilir. Fakat birincil anahtardan farklı olarak bazı kayıtlarda o alan boş bırakılabilir (NULL değer kabul edebilir) ve bir tabloda birden çok tekil anahtar olabilir. Örneğin Öğrenciler tablosunda öğrencinin telefon numarasının ve e-posta adresinin saklanacağı alanlar bulunsaydı, bu alanlar UK olarak tanımlanabilirdi. Bu sayede öğrenci beyan etmezse telefon numarası veya e-posta adresi girilmeyebilir, fakat girilen her öğrenci için de başka bir öğrenci ile aynı değere sahip olmaması sağlanırdı.

**Sorgu (Query):** Veritabanının bir veya daha çok tablosunda saklanan kayıtlar içinden istenen niteliklere ait değerlerinin gösterilmesi için verilen tanımdır. İlişkisel veritabanı yönetim sistemlerinde standart sorgulama dili olarak SQL (Structured Query Language) kullanılır. Adı Yapısal Sorgulama Dili olsa da veritabanındaki tüm diğer işlemler için de gerekli komutlara sahiptir. Veri ekleme, silme ve güncelleme komutları DML (Data Manipulation Language), veritabanı nesnelerini ekleme, nesnelerin tanımlarını değiştirme ve nesneleri yok etme komutları DDL (Data Definition Language), kullanıcılara yetki verme ve verilen yetkileri geri alma komutları DCL (Data Control Language) olarak bilinir.

**Nesne (Object):** İlişkisel veritabanlarında tablolar dışında birçok mantıksal nesne vardır. Aranan kayda daha hızlı erişim ve verilerin sıralı bir şekilde gösterilmesi gibi işlemler için kullanılan İndeks (Index) en önemlilerinden biridir. Genellikle sorgularda kayıtlar anahtar alanlar üzerinden arandığı için PK olarak belirlenen alanlar için otomatik olarak indeks oluşturulur. Bir sorgu tanımının saklandığı Görüntü (View), veritabanında saklanan programlama yapıları olan Saklı Yordamlar (Stored Procedures) ve Fonksiyonlar (Functions), bir olay gerçekleştiğinde otomatik olarak yürütülen bir saklı yordam olan Tetikleyiciler (Triggers) veritabanı nesnelerinden bazılarıdır. Saklı yordam, fonksiyon ve tetikleyici tanımları tüm RDBMS'lerde aynı şekilde yapılmaz. Örneğin MS SQL Server'da bu nesneler T-SQL diline göre Oracle'da ise PL/SQL diline göre yapılır.

**İlişki (Relation):** Tablolar arasında veri tutarlılığını sağlamak için kurulan bağlantıdır. Genellikle bir tablodaki birincil anahtar bir alanın başka bir tablodaki dış anahtar bir alana bağlanması ile oluşur. Şekil 3.1'de görüldüğü gibi Dersler tablosu Notlar tablosuna DersKodu alanı üzerinden 1-∞ türünde bir ilişki ile bağlanmıştır. Notlar tablosunda aynı DersKodu değeri ∞ defa yer alabilir. Fakat bu ilişki sayesinde Notlar tablosunun DersKodu alanına girilecek bir değerin Dersler tablosunun DersKodu alanında da olması gerekliliği veritabanı tarafından sağlanır (uygulama programcısı bunun denetimi ile uğraşmaz). Benzer bir veri tutarlılığı denetimi Notlar tablosu ile Öğrenciler tablosu arasında verilen ilişki için de sağlanır. Bir ders ile veya bir öğrenci ile ilgili kaydın silinmesi durumunda, Notlar tablosunda silinen ders veya öğrenci ile ilgili kayıtların bulunması veri tutarsızlığına yol açacaktır. Bunun önüne geçmek için veritabanı tanımlanan kurala göre; ya Öğrenciler ve Dersler tablolarından kayıtların silinmesine engel olur, ya silinen kayıtlar ile ilgili Notlar tablosundaki kayıtları da siler, ya da Notlar tablosundaki ilgili kayıtların FK alanlarına NULL değeri atar.




**Şekil 3.1.** Üç Tablonun İki Faklı İlişki ile Bağlanması

### 3.3. Avantajları

- **Yönetilebilirlik:** İlişkisel veritabanlarının yönetimi ve kullanımı kolaydır. Tablo biçiminde saklanan veriler kullanıcı tarafından kolayca anlaşılmaktadır. Standartlaşmış bir sorgulama dili vardır.
- **Güvenlik:** Kullanıcı yönetimi birçok RDBMS'de mevcut olduğu için belirli kullanıcıların belirli verilere erişimi sınırlandırılabilmektedir. Kullanıcılar belirli rollere üye yapılarak, izinlerin ve kısıtların o role üye olan tüm kullanıcılar için kolayca yapılması sağlanabilmektedir.
- **Güvenilirlik:** Elektrik kesintisi gibi sistem arızalarında son yapılan değişikliklerin kaydının tutulduğu log dosyalarının yardımı ile sistem tekrar başlatıldığında kurtarma (recovery) yapılabilir. Disk hataları nedeniyle veri kaybı yaşanmaması için ise; belirli zaman aralıkları ile veri tabanının tamamının veya belirli bir kısmının otomatik olarak yedeklenmesi sağlanabilir.
- **Veri Tutarlılığı:** İlişkisel veritabanında veri tutarlılığının sağlanması için ACID prensiplerine uygun bir hareket idaresi vardır. Aynı anda aynı veri üzerinde değişiklik istekleri veri kilitleme mekanizması ile engellenebilir. PK-FK ilişkileri sayesinde veri bütünlüğü yani verinin doğru ve tutarlı olması veritabanı tarafından otomatik olarak sağlanır.
- **Esneklik:** Verileri güncellemek gerektiğinde, bunu yalnızca bir tabloda yapmak yeterlidir. Yapılan değişikliğin diğer tablolardaki ilgili verileri de değiştirmesi otomatik bütünlük veya tetikleyiciler yardımıyla sağlanabilir.
- **Hızlı Sorgulama:** Veriler yapısal olarak tablolarda saklandığı için sadece anahtar alana göre değil, istenen alana göre kolaylıkla indekslenebilir ve bu indekslerin yardımıyla sorgu sonuçları hızlıca elde edilebilir.

### 3.4. Dezavantajları

- **Ölçeklenebilirlik:** İlişkisel veritabanları genellikle tek ve donanımsal olarak güçlü bir sunucu üzerinde çalıştırılmaktadır. Kapasite yeterli gelmemeye başladığında dikey ölçeklendirme tercih edilmektedir. Bu da genellikle yatay ölçeklendirmeye göre daha pahalı bir donanım satın alınması gerektiği anlamına gelmektedir. Bazı

gelişmiş RDBMS'ler dağıtık sistemlerde de çalıştırabilir, fakat kurulumu ve yatay ölçeklendirmesi genellikle zahmetlidir.

- **Büyük veride performans:** Verilerin hacim, hız, çeşitlilik ve karmaşıklıkta hızlı büyümesi daha da karmaşık ilişkiler yaratmakta ve bu durum performansı yavaşlatabilmektedir. Ekleme, silme ve güncelleme işlemlerinde ise ACID modeline bağlı bir yapıda olması nedeniyle performans olumsuz yönde etkilenmektedir.
- **Şemaya bağımlılık:** Yapısal olan, yani tablo gibi önceden belirli veri tiplerine uygun olarak hazırlanan bir yapıda saklanabilecek veriler için uygun olsa da, yapısal olmayan, yani çok farklı veri tiplerinin aynı anda saklanmasını gerektiren uygulamalar için uygun bir yapı sunmamaktadır. Özellikle hızlı bir şekilde saklanması gereken ve yapısal bir şemaya uydurulması zor olan sosyal medya verileri için elverişli değildir.

## 3.5. Tezde Karşılaştırılan İlişkisel Veritabanları

### 3.5.1. Oracle Veritabanı

1979 yılında ilk sürümü piyasaya sürülen Oracle, SQL dilini kullanan ilk RDBMS olmuştur. Önceleri *Relational Software, Inc.* olan şirket adı, 1982 yılında ürünü ile aynı ismi alıp *Oracle Systems Corporation* olmuştur. 1986 yılında istemci-sunucu modeline geçilmiş ve 1988'de yayınlanan 6 sürümü ile daha alt seviyede çalışmak için prosedürel bir dil olan PL/SQL ortaya çıkmıştır. Fakat saklı yordam ve tetikleyici yapıları 1992'de yayınlanan 7 sürümü ile birlikte kullanılabilir olmuştur. Aynı sürümde nesne-ilişkisel veritabanı kavramı da yer almıştır. 1998'de yayınlanan 8i sürümü ile Java desteği ve 2001'de yayınlanan 9i sürümü ile kümeleme (RAC: Real Application Clusters) desteği gelmiştir.

Oracle firması 2000'li yıllarda ilişkisel veritabanı dışında farklı yazılım ürünleri de üretmeye ve şirketler satın alarak büyümeye odaklanmıştır. 2010 yılında en büyük donanım üreticilerinden Sun Microsystems firmasını satın alarak kendi veritabanını çalıştırmak için güçlü sunucular (Exadata) üretebilmeye başlamıştır (bu satın alma ile Java ve MySQL'in de sahibi olmuştur).

Büyük boyutta verilerini kesintisiz çalışan, güvenilir ve hızlı işleme kapasitesi olan bir veri tabanında saklama ihtiyacı olan kurumlar (bankalar gibi) tarafından günümüzde en çok tercih edilen veritabanıdır. Birçok farklı işletim sistemi ile çalışabilme yani platform bağımsız olma özelliği kullanım oranını arttırmıştır.

### 3.5.2. MS SQL Server

İlk sürüm olan SQL Server 1.0, OS/2 işletim sistemi için 1989 yılında Sybase ve Ashton-Tate ile birlikte Microsoft tarafından piyasaya sürülmüştür. Daha sonra Sybase tarafından Microsoft'a satılmıştır. Microsoft'un Jet veritabanı moturunu kullanan dosya-paylaşımı temelli Access'ten farklı olarak, SQL Server istemci/sunucu tabanlı bir ilişkisel veritabanı yönetim sistemidir.

Önceleri sadece Microsoft işletim sistemlerinde kullanılabilirken, artık Linux işletim sistemlerine de kurulabilmektedir. Fakat Windows altında kullanılabilen "SQL Server Management Studio" adındaki görsel arayüz Linux üzerinde henüz yoktur.

SQL Server 2012 sürümüyle "Sütun depolama indeksi" özelliğine sahip olmuştur. Bu özellik OLAP işlemleri için geliştirilmiş bir indeks yapısıdır. Satır indeksinden farkı, indeksleri sütun bazlı tutuyor olmasıdır. Bu da bir sütundaki sayıların toplanması, ortalamasının alınması gibi sorgularda, sonuçların daha hızlı alınmasını sağlamaktadır.

### 3.5.3. MySQL

MySQL, ücretsiz dağıtımı yapılan, aynı zamanda ticari kullanım için ücretli sürümleri de bulunan bir RDBMS'dir. 1994 yılında geliştirilmeye başlanan MySQL'in ilk sürümü 1995'te yayınlanmıştır. 2004 yılında yayınlanan 4.1 sürümü ile alt sorgu desteği eklenmiş, 2005 yılında yayınlanan 5.0 sürümü ile de saklı yordamlar ve tetikleyiciler gibi yeteneklere kavuşmuştur.

2008 yılında MySQL Sun Microsystems şirketi tarafından satın alınmış, 2010 Ocak ayında da Oracle'ın Sun Microsystems'i satın almasıyla bu şirketin bünyesine geçmiştir. 2010 Aralık ayında yayınlanan 5.5 sürümünde hareket (transaction) idaresi ve FK bütünlük kısıtlaması desteği sağlayan InnoDB saklama motorunu kullanmaya başlamıştır (daha önceki sürümler MyISAM motorunu kullanmaktaydı). 2010 öncesinde

veri tutarlılığını sağlama görevi programcıya bırakılmışken, hareket idaresi sayesinde bu tarihten sonra ACID özelliğini sağlamaya başlamıştır.

Küçük çaplı web uygulamaları için en çok tercih edilen veritabanıdır. Birçok farklı üçüncü parti görsel yönetim aracı ile kullanılabilse de resmi olarak MySQL firması tarafından geliştirilen “MySQL Workbench” adında bir grafik kullanıcı arayüzü vardır.

# BÖLÜM 4

# NoSQL VERİTABANI YÖNETİM SİSTEMLERİ

## 4.1. Tarihçesi

Carlo Strozzi 1998 yılında ilişkisel bir veritabanı tasarlamış, fakat sorgulamada SQL kullanmadığı için ismini NoSQL koymuştur. 2009 yılının başlarında, Rackspace'in bir çalışanı olan Eric Evans tarafından, açık kaynak dağıtık veritabanları sistemlerinin görüşüleceği bir toplantı düzenlenmek istendiği dönemlerde NoSQL terimi tekrar kullanılmaya başlanmıştır (Lith & Mattsson, 2010). Eric Evans aslında bu isimle sürekli olarak ortaya çıkmaya başlayan, ilişkisel olmayan dağıtık veri depolama sistemlerine bir isim vermeyi amaçlamıştır (Evans & Oskarsson, 2009). Diğer adlarından bazıları NonRel, NoRDBMS'dir (ilişkisel değil anlamında). NoSQL için "Not only SQL (Sadece SQL değil)" tabiri de kullanılmaktadır. Bunun nedeni SQL sorgularına alışmış geliştiriciler olarak aynı sorgu mantığının adapte edilmeye çalışılmasında yatmaktadır. Genel bir tanım vermek gerekirse; NoSQL, ilişkisel veritabanı sistemlerine alternatif bir çözüm olarak ortaya çıkan, değişken şema yapısına sahip verilerin saklanabilmesini ve büyük verilerin yüksek performansla sorgulanabilmesini sağlayan, yatay ölçeklendirilmesi kolay olan bir veri depolama sistemidir.

Bir bankacılık uygulamasında tutarlılık her zaman öncelikli olmaktadır. Hesabınızdan para çektiğiniz zaman eğer işlem başarılı ise ilgili tabloların hepsi aynı hareket (transaction) içerisinde güncellenmeli ve commit veya rollback edilmelidir. Twitter gibi çok fazla sayıda kullanıcının kullandığı uygulamalarda performans tutarlılıktan daha önemlidir. Bir Twitter uygulamasında veriler tutarsızdır denilmemektedir, uygulamaya gönderdiğiniz bir istek farklı sunucularda farklı zamanlarda ele alınabilir. Bankacılık uygulamalarında tutarlılıktan bahsedilmektedir, NoSQL mimarisinde ise tutarlılık ikinci plana atılarak veriler dağıtık bir yapıda tutulmaktadır.

## 4.2. Avantajları ve Dezavantajları

**Avantajları:**

İlişkisel veritabanı sistemlerine göre yüksek erişilebilirlik imkânı sunmaktadırlar. Esnek yapısından dolayı programlama ve bakım anlamında kolaylık sağlarlar. Yazma performansları genel olarak ilişkisel veritabanı sistemlerine göre daha üstünken, okuma işlemlerinde ise anahtar değer üzerinden arama yapıldığında daha hızlı olabilirler.

Farklı veritabanı türleri veriyi farklı şekillerde tutmakta ve sunmaktadır. Bu durum bazı uygulamalar, programlama dilleri ve geliştiriciler için kolaylık ve anlaşılırlık sağlamaktadır.

NoSQL sistemleri yatay olarak ölçeklenebilir. Binlerce sunucu bir arada bir küme (cluster) olarak çalışarak çok büyük veri üzerinde işlem yapılabilmektedir. Bir sunucu havuzuna sunucu ekleyip çıkarmak genelde bir satırı değiştirmek ya da bir komut girmek kadar kolaydır. Aynı nedenle sunuculara bakım yapmak istenirse bakım moduna alınması gerekmemektedir (Yishan & Manoharan, 27-29 Aug. 2013).

NoSQL veritabanlarında birçok farklı seçenek arasından seçim yapma şansı bulunmaktadır. Çoğu NoSQL veritabanı açık kaynak kodlu projelere ve bulut bilişim teknolojilerine uygun olduğu için maliyet olarak RDBMS'lere göre daha avantajlıdır.

**Dezavantajları:**

RDBMS kullanan uygulamaların NoSQL sistemlere taşınması zorluklar yaşanmasına yol açabilmektedir. Birleştirme (join) kullanılan sorgu var ise taşınırken kodların değiştirilmesi gerekmektedir. Sorgu tabanlı veri erişimi yerine anahtar tabanlı veri erişimi kullanılmaktadır. Buna göre bir yapılandırmaya gidilmesi zaman alabilmektedir.

İlişkisel veritabanı yönetim sistemlerindeki ACID özelliği (Bak: Bölüm 2.5.1), NoSQL veritabanı sistemlerinde bulunmadığı için veri kaybı söz konusu olabilmektedir.

NoSQL veritabanı sistemleri veri güvenliği konusunda ilişkisel veritabanı yönetim sistemleri kadar gelişmiş özelliklere sahip olmamaktadır.

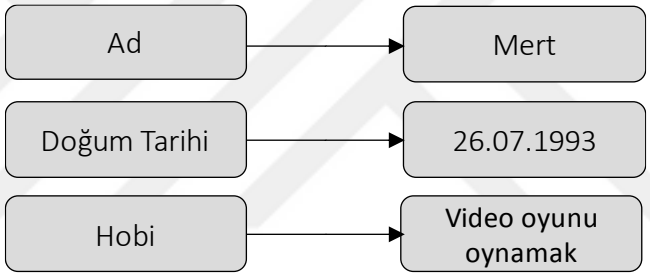
Bazı NoSQL veritabanı projelerinin dokümantasyon ve profesyonel destek konusunda eksikleri bulunmaktadır.

### 4.3. NoSQL Veritabanı Depolama Türleri

NoSQL veritabanları depolama biçimlerine göre genellikle anahtar-değer, doküman, sütun ve çizge tabanlı gibi farklı türlere ayrılmaktadır.

#### 4.3.1. Anahtar-Değer Tabanlı Depolama Yapan Veritabanı

Veriler bloklar halinde anahtar değer ilişkisi kurularak kaydedilmektedir. Sorgulama esnasında anahtar verilip, değer sorgulanmaktadır. Saklanan değerin içeriğinin ne olduğu veritabanı tarafında önemli değildir. Bu tür veritabanlarına örnek olarak; Redis, DynamoDB, FoundationDB, Voldemort, Riak, MemcacheDB, Aerospike, LevelDB, GenieDB ve Oracle NoSQL DB verilebilir. Redis ve MemcacheDB, performans amacıyla verileri RAM'de saklarlar. Şekil 4.1'de anahtar-değer tabanlı veritabanı örneği gösterilmektedir.




**Şekil 4.1.** Anahtar-Değer tabanlı depolama yapan veritabanı

Anahtar-değer tabanlı depolama yapan veri tabanlarının avantajları ve dezavantajları:

Küçük ve çok sayıda okuma yazma işleminin yapıldığı uygulamalar için uygundur. Bir anahtara karşılık gelen veri genellikle integer gibi basit türden bir veridir.

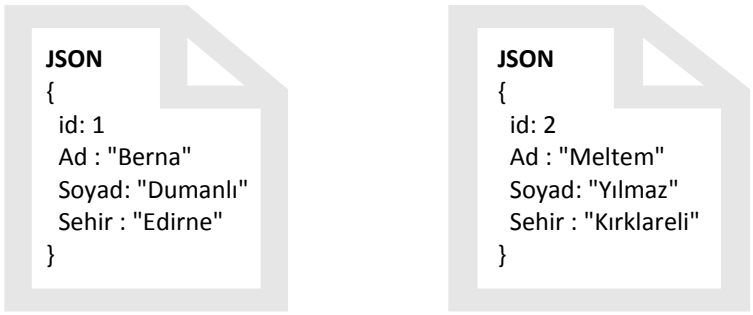
Genelde önbellekleme yapılmak istendiğinde kullanılmaktadır. Önbelleğe atmak istediğiniz veriyi tekrar alabilmek için hızlı bir şekilde tekrar size geri döndürür veya kullanıcının oturumunu korumak için NoSQL kullanmak daha avantajlı ve hızlı olmaktadır.

Tüm işlemler tek bir anahtar üzerinden yapıldığı için tutarlılık ve bütünlük sağlaması kolaydır; uygulama geliştirici anahtar verisini kullanarak kolaylıkla ekleme, silme ve güncelleme işlemlerini üç yöntem (put, remove, get) ile gerçekleştirebilir (Aydemir, 2016, s. 25).

Değer sütununda tutulan verinin şema bilgisi kullanıcı tarafından belirlenmektedir; kullanıcı veriyi aldığında anlamını kendisi çıkarmalıdır. Veritabanı saklanan verinin türünü ve içeriğini bilmemektedir. Bunun sonucunda da verinin içeriği ile ilgili herhangi bir sorgulama ve indeksleme yapılamamaktadır; kopyalama (replication) ve çoğaltma (duplication) ise kolaylaştırılmıştır (Gortan & Klein, 2014).

### 4.3.2. Doküman Tabanlı Depolama Yapan Veritabanı

Doküman tabanlı veritabanları Lotus Notes'dan esinlenmiştir. Bir anahtara karşılık gelen veriler doküman adı verilen nesnelerde saklanırlar. Verileri depolamanın en popüler yollarından biri, her bir kaydın ve ilgili verilerin bir "doküman" olarak düşünüldüğü bir modeldir. Veriler genellikle JSON (Javascript Object Notation) biçiminde tutulmaktadır. Doküman tipinde kaydedilen veriler, diğer sistemleri desteklemek amacıyla benzersiz kimlik ile kullanmaktadırlar. Bu tür veritabanlarına örnek olarak en çok kullanılan NoSQL veritabanı olan MongoDB başta olmak üzere, CouchbaseDB, CouchDB, RavenDB, Terrastore, SisoDB, RethinkDB ve RaptorDB verilebilir. Şekil 4.2'de doküman tabanlı veritabanının saklama yapısı olan JSON biçimi gösterilmektir.




**Şekil 4.2.** Doküman tabanlı depolama yapan veritabanı

Doküman tabanlı depolama yapan veri tabanlarının avantajları ve dezavantajları:

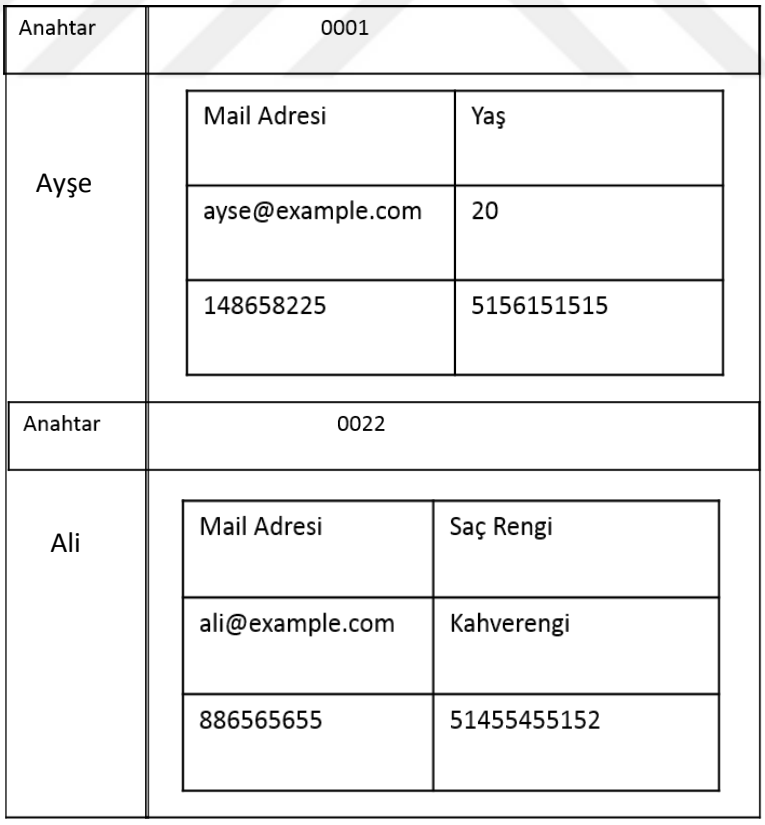
Uygulama geliştiren kaydedeceği veri tipini belirleyebilmektedir. Veri tipleri değiştirilebilmekte ve pek çok programlama dillerinin kullandığı veri tipiyle eşleşebilmektedir. Bu uyumu büyük esneklik sağlamaktadır.

Veritabanındaki verinin içeriği hakkında bilgi sahibidir ve onun üzerinden işlem yapabilmektedir. Bu sebeple dokümanın herhangi bir alanına göre indeksleme ve sorgulama yapılabilmektedir. Eğer doküman boyutu büyük ise veri ikili biçime (BSON: Binary JSON) dönüştürülerek kaydedilebilmektedir (Gortan & Klein, 2014).

İndeksleme sisteme ek olarak yük getirdiğinden maliyeti arttırabilmektedir.

### 4.3.3. Sütun Tabanlı Depolama Yapan Veritabanı

Yüksek okuma yazma performansı ve yüksek erişilebilirlik (high availibity) için tasarlanmıştır. Birden çok sunucu üzerinde dağıtık olarak çalışırlar ve bu sayede tek bir sunucuda tutulamayacak kadar büyük verileri saklayabilirler. Nesne yapısı, ilişkisel veritabanı mimarisi ile benzerlik taşımaktadır. İlişkisel veritabanı ile Anahtar-değer veritabanı modellerinin mantıksal olarak birleşmesinden oluşmuştur (Aydemir, 2016, s. 26). Örnek olarak en çok tercih edilen Cassandra ve Hbase başta olmak üzere Amazon SimpleDB, HPCC, Cloudata, Stratosphere, Hybertable, Accumulo, Druid veritabanları sayılabilir. Şekil 4.3'te sütun tabanlı depolama yapan veritabanı gösterilmektedir.



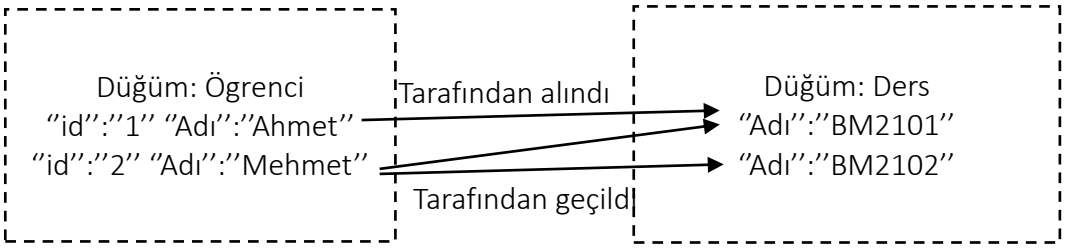

**Şekil 4.3.** Sütun tabanlı depolama yapan veritabanı

Sütun tabanlı depolama yapan veri tabanlarının avantajları ve dezavantajları:

Büyük veriler için hızlı ölçeklenebilirlik ve hızlı veri yükleme yapılabilmektedir. İşlemler atomik olarak yapıldığından dolayı veri kopyalama, güncelleme, silme ve ekleme işlemleri hızlı gerçekleşmektedir. Atomik olması nedeniyle tutarlılık ve bütünlük problemi de yoktur; kopyalama ve çoğaltma kolaylıkla yapılabilmektedir (Gortan & Klein, 2014). Veritabanının, verinin içeriği hakkında bilgisi bulunmamaktadır.

### 4.3.4. Çizge Tabanlı Depolama Yapan Veritabanı

Veritabanlarında veriler düğümler (node), ilişkiler (edge) ve özellikler (properties) şeklinde tutulmaktadır. Veriler düğüm olacak şekilde ilişkileri ile kaydedilmektedir; bu veriler ilişkileri ile en uygun hale getirilmektedir. Bu veritabanlarının en sık kullanımı depolanan bilgileri birbirleriyle bağlı öğeler ile temsil edebildiği durumdur ve örnek olarak sosyal ağlar, yol haritaları, taşıma yönlendiricileri verilebilmektedir (Abramova, Bernardino, & Furtado, 2014). En çok tercih edilen çizge tabanlı veritabanları Neo4J, HyperGraphDB, Infinite Graph, VertexDB, InfoGrid, FlockDB olarak verilebilir. Şekil 4.4'te çizge tabanlı depolama yapan veritabanı gösterilmektedir.




**Şekil 4.4.** Çizge tabanlı depolama yapan veritabanı

Çizge tabanlı depolama yapan veri tabanlarının avantajları ve dezavantajları:

Diğer NoSQL türlerine göre ilk kayıt oluşturma ve güncelleme de yavaş kalmaktadır. Veri yapısı güncellenebilir bir yapıya sahiptir, bu geliştiriciler için bir avantajdır. Kullanım alanı doküman, sütun ve anahtar-değer depolama yapan kadar geniş değildir (Gortan & Klein, 2014).

## 4.4. Tezde Karşılaştırılan NoSQL Veritabanları

### 4.4.1. MongoDB

MongoDB, doküman tabanlı, C++ ile geliştirilmiş, açık kaynak olarak dağıtılan ve platform bağımsız bir NoSQL veritabanı uygulamasıdır. 2007'de *10gen* firması tarafından geliştirilmeye başlanmıştır. Firma daha sonra *Mongodb Inc.* ismini almıştır. MongoDB, sorgulama, indeksleme, yatay ölçeklenebilirlik ve esnekliğe sahip bir doküman tabanlı veritabanıdır (Bhamra, 2017). Windows, MacOS, Solaris ve Linux dağıtımlarında kullanılabilmektedir.

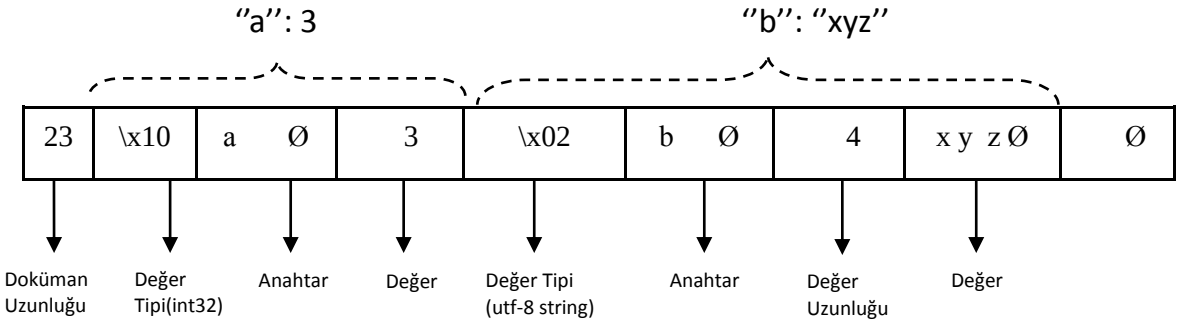
#### 4.4.1.1. Veri Modeli

MongoDB'de veriler, Şekil 4.5'te görüldüğü gibi, BSON (Binary JSON) adı verilen JSON yapısının ikili karşılığı diyebileceğimiz bir biçimde saklamaktadır. Veriler yani dokümanlar koleksiyon içinde saklanır ve "_id" tanımıyla her biri benzersiz bir anahtara sahiptir.

JSON:

```
{
   ''a'': 3,
   ''b'': ''xyz''
}
```

BSON:




**Şekil 4.5.** MongoDB JSON- BSON Gösterimi

MongoDB'de kullanılan mantıksal veritabanı yapıları ile ilişkisel veritabanlarındaki yapıların karşılaştırması Çizelge 4.1'de verilmiştir.

**Çizelge 4.1.** MongoDB ve RDBMS Veritabanı Yapıları

| **MongoDB** | **RDBMS** |
|---|---|
| Veritabanı | Veritabanı |
| Koleksiyon | Tablo |
| Doküman | Satır |
| Alan | Sütun |
| Gömülü Dokümanlar | Tablo İlişkisi |

- **Veritabanı:** Koleksiyonların fiziksel olarak barındırıldığı yerdir.
- **Koleksiyon:** MongoDB dokümanlarını içeren yapıdır. İlişkisel veritabanlarındaki karşılığı tablodur. İçeriğinde şema yapısı barındırmamaktadır. Yani içerdiği her doküman eşit sayıda ve aynı veri türlerinde alanlara sahip olmak zorunda değildir.
- **Doküman:** Geleneksel veritabanlarındaki satır kavramıyla aynıdır. Her döküman, koleksiyonlar içerisinde dinamik olarak BSON yapısında saklanmaktadır. Dokümana ait _id değeri, her doküman için benzersizdir ve 12 byte büyüklüğünde, hexadecimal değerden oluşmalıdır. Doküman oluştururken _id değerini kullanıcı yazmaz ise MongoDB kendi oluşturmaktadır (Bhamra, 2017).

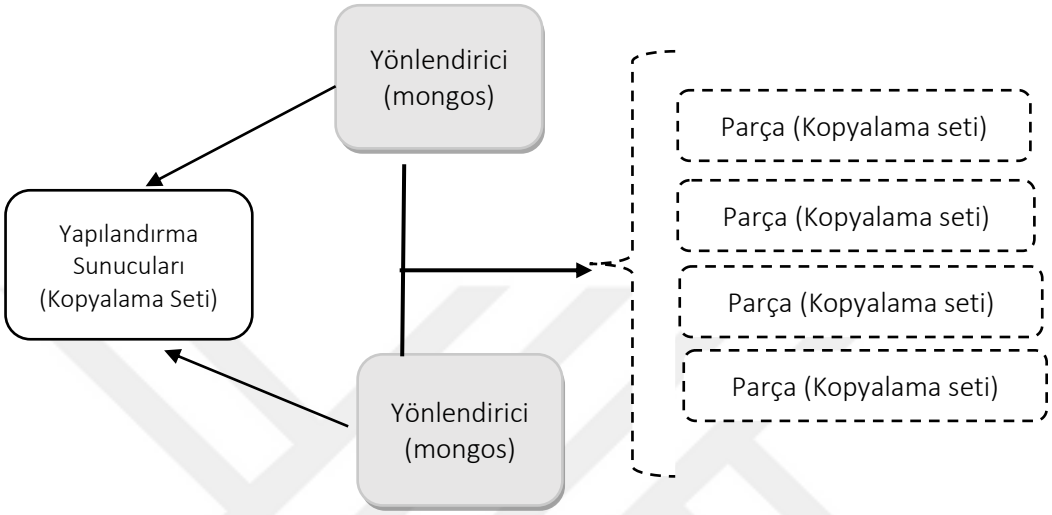
MongoDB FK bütünlük kısıtlaması sağlamamaktadır, veriler arasındaki ilişkiler ek sorgularla sağlanabilmektedir. SQL'den farklı bir sorgulama biçimi vardır.

#### 4.4.1.2. Mimarisi ve Kullanım Alanları

MongoDB, istemci-sunucu mimarisine sahiptir ve sunucuyu başlatmak için mongod.exe kullanılmaktadır. JSON dokümanlarına hızlı erişim ve bellekte kalıcı hale getirmek için bellekte BSON olarak diskte saklamaktadır. BSON, JSON formatındaki dokümanları bilgisayar ağı üzerinden etkin aktarımı için tasarlanmış bir formattır.

MongoDB mimarisinde 3 temel yapı vardır: **Shard'**lar, veriyi depolamak için

kullanılmaktadır. Doğru veriye ulaşmak için **mongos** birimi yönlendirme yapmaktadır. Kümenin durumunu takip etmek için ise **configdb** (yapılandırma sunucusu) kullanılmaktadır. İstemciden gelen isteğin tipine göre mongos'lar isteği shard'lara gönderir ve onlardan gelen cevabı istemciye tekrar döndürür.




**Şekil 4.6.** MongoDB Sharding

MongoDB'nin kullanım alanları; Coğrafi bilgi saklamak, büyük veri projeleri, günlük verisi saklamak, zaman içinde yapısı değişecek uygulamalar, çok sunucu gerektirebilecek dağıtık uygulamalarıdır (Bhamra, 2017).

#### 4.4.1.3. Avantajları

- Esnek veri modeline sahiptir, tablo gibi belirli bir şema oluşturulmamaktadır. Anahtara bağlı olarak tüm değer setlerini barındırabilmektedir.
- Yazma işleminde yüksek performans göstermektedir.
- Dokümanın her niteliğine ve anahtarına göre indeksleme yapabilmektedir. İndeks oluşturduktan sonra sorgularda hızlanma olurken, aggregate fonksiyonlarında indeksin bir etkisi görülmemektedir.
- Sharding özelliği sayesinde büyük verilerin sunucular arasında paylaştırılması, yani yatay ölçeklendirme yapılabilmektedir. Ölçeklenmeyi sağlayabilmek için *Master-Slave Replikasyon* desteği sunmaktadır.

- *MapReduce* paralel programlama özelliği sayesinde hızlı analiz etme ve veri işlemeyi sağlamaktadır. *Map* aşamasında analiz edilecek veri içerisinden gerekli veriler çekilmekte, *Reduce* aşamasında ise çekilen veri üzerinde istenen analiz gerçekleştirilmektedir.
- Metin indeksleri yaratma ve bunlar üzerinden tam-metin arama (FTS: Full-Text Search) desteği bulunmaktadır.
- GridFS yapısı ile dosyalar veritabanı içerisinde saklanabilmektedir.

#### 4.4.1.4. Dezavantajları

- MongoDB'de tetikleyici gibi bir yapı bulunmamaktadır.
- MongoDB disk alanını otomatik olarak temizleyememektedir. Bu yüzden dokümanların güncellenmesi veya silinmesi durumunda yeni disk alanı açılmamaktadır, bu işlemin manuel olarak yapılması gerekmektedir.
- İki dokümanın birleştirilmesi (join) sorgularda kullanılmamaktadır. Bu nedenle çok sayıda sorgulama yapılmaktadır, bu da kodun dağınık gözükmesine sebep olmaktadır.
- MongoDB her bir değer çifti için anahtar isimleri saklar (ilişkisel veritabanı olarak düşünürsek her kayıtta alan isimlerinin tekrar etmesi gibi). Ayrıca, birleşme fonksiyonelliğinin olmaması nedeniyle veri fazlalığı da vardır. Bunlar gereksiz bellek ve disk kullanımının artmasına neden olur.
- Eğer indeksleme kötü ve birçok parçadan oluşmuş ve yanlış bir düzende yapılmış ise MongoDB çok yavaş olmaktadır.
- Bir dokümanın büyüklüğü en fazla 16MB olabilmektedir.

### 4.4.2. Cassandra

Cassandra, sütun tabanlı, açık kaynak kodlu, Java ile geliştirilmiş bir NoSQL veritabanıdır. Veri modelinde Big Table'dan ve Dynamo'nun dağıtım modeli esasından etkilenmiştir. Facebook tarafından, gelen kutusu aranması amacıyla geliştirilmiştir ve 2008 de piyasaya çıkmıştır. Mart 2009'da Apache inhibitörlüğünü almıştır ve bir Apache projesi olmuştur.

Ölçeklendirilebilir olması, çok yüksek boyutlardaki veriyi saklayabilmesi, çok yüksek okuma ve yazma hızı, %80 e varan veri sıkıştırma özelliği, çoklu veri merkezini destekleyen dağıtık yapısı gibi teknik özellikleri nedeniyle yaygın olarak kullanılmaktadır (https://academy.datastax.com/).

#### 4.4.2.1. Veri Modeli

Cassandra veritabanında en dış yapı birlikte çalışan birçok düğümden oluşan kümedir. Hata durumlarında veri kaybı olmaması için, her düğüm bir kopya (replica) içerir. Cassandra düğümleri küme içinde bir halka biçiminde düzenler ve bunlara veri atar. Cassandra'da kullanılan yapılar ile ilişkisel veritabanlarında bu yapılara karşılık gelen ifadeler Çizelge 4.2'de verilmiştir.

**Çizelge 4.2.** Cassandra ve RDBMS Veritabanı Yapıları

| **Cassandra** | **RDBMS** |
|---|---|
| Anahtar alanı (Keyspace) | Veritabanı |
| Tablo (Sütun ailesi) | Tablo |
| Satır Anahtarı | Birincil Anahtar |
| Sütun adı/anahtarı | Sütun adı |
| Sütun değeri | Sütun değeri |

**Anahtar alanı (Keyspace):** Cassandra veri modeli en üst düzeyde anahtar alanlardan oluşmaktadır. Anahtar alanlar, ilişkisel veritabanındaki şemaya veya veritabanına benzer yapıdadır. Bir anahtar alanı bir veya daha çok tablo içerebilmektedir ve bir tablo sadece bir anahtar alana aittir. Anahtar alanı yaratılırken belirtilen kopyalama faktörü (replication factor), her verinin kaç farklı düğüm üzerinde saklanacağını belirler. Sınıf (class) olarak verilen değer ise, halka şeklinde kurulan düğüm serisinde kopyaları yerleştirme stratejisidir. *SimpleStrategy,* sadece tek bir veri merkezi(datacenter) ve bir rack için kullanılmaktadır. İlk kopyayı bölümleyici tarafından belirlenen bir düğüme yerleştirmektedir. *NetworkTopologyStrategy*, kümeyi birden fazla veri merkezine dağıtmak için kullanılmaktadır. Bu strateji, her bir veri merkezinde kaç tane kopya istenildiğini belirtmektedir. Verilen kod parçacığında; *SimpleStrategy* yerine

*NetworkTopologyStrategy* kullanılırsa, her veri merkezinin ismi verilip o ismin yanında kopyalama faktörü değeri verilmesi gerekmektedir.

```
CREATE KEYSPACE testDB
WITH replication = {'class': 'SimpleStrategy',
                    'replication_factor' : '4'};
```

**Tablo:** Cassandra'nın eski sürümlerinde tablo yerine sütun ailesi terimi kullanılmakta idi. Anahtar alanı içerisinde tanımlanan tablolar birincil anahtar ve bir dizi sütundan oluşmaktadır. Yani RDBMS'de tablo bir dizinin dizisi (satır × sütun) iken, Cassandra'da ise 'iç içe anahtar-değer çiftleri'nin bir listesidir (satır × sütun anahtarı × sütun değeri).

**Sütun:** Sütun Cassandra'da tek bir veri parçasını temsil eder ve tanımlanmış bir türe sahiptir. Bir sütun, üç değere sahiptir; sütun adı (name), değer (value) ve zaman damgası (timestamp). Ayrıca, birçok alt-sütunun tek isim altında birleşmesi ile oluşan süper-sütun adında bir yapı da vardır.

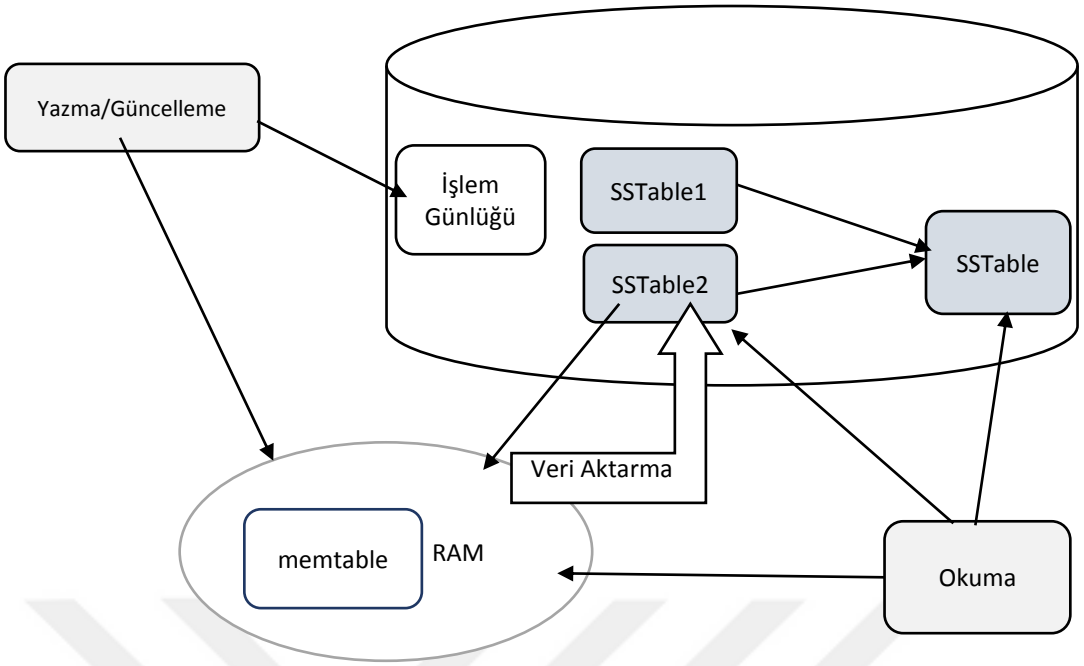
**Satır:** Satır Cassandra'da bir kopyalama birimidir. Sütunlardan oluşmaktadır ve her sütun bir varlığı temsil etmektedir. Sütunlar her bir satır için tanımlandığından, bu satırların adı, sütun numarası ve veri türleri farklı olabilmektedir.

(https://www.tutorialspoint.com/cassandra/cassandra_data_model.htm)

#### 4.4.2.2. Mimarisi

Cassandra'nın mimarisi, birçok farklı teoriden oluşmuş karmaşık bir yapıdır. Meta verileri saklamak için kullanılan *system* adında bir anahtar alan mevcuttur (MS SQL Server'da *master* veritabanı ve Oracle'da *system* şeması gibi).

Cassandra'da bir yazma işlemi gerçekleştirdiğinizde, bu işlem hemen işlem günlüğüne (commit log) yazılır. İşlem günlüğü, Cassandra'nın dayanıklılık hedeflerini destekleyen bir hata kurtarma mekanizmasıdır. İşlem günlüğüne yazıldıktan sonra, değer *memtable* adı verilen hafızda tutulan bir veri yapısına yazılır. Burada depolanan nesne sayısı bir eşik değere ulaştığında, *memtable* içeriği *SSTable* adlı diskte saklanan bir dosyaya aktarılır.




**Şekil 4.7.** Cassandra Mimarisi

Cassandra'da bir silme işlemi gerçekleştirdiğinizde, gerçekte hemen silme yapılmaz. Aslında "yumuşak silme (soft delete)" olarak ifade edebileceğimiz bir güncelleme işlemi yapılır. Yani silinen kayda mezar taşı (tombstone) adı verilen bir ifade eklenir. Sonradan bir sıkıştırma (compaction) işlemi yürütüldüğünde bu kayıt gerçekten silinir.

Ölçeklendirme için önemli olan eşler arası (peer to peer) teorisinde eşler arasında dağılım eşittir, her düğümde eşit büyüklükte veri bulunmaktadır. Her bir düğüm eş olduğu için yeni bir düğüm eklemek te kolaydır.

Dedikodu protokolleri (Gossip protocol), hangi düğümün çalıştığını diğer düğümler arasındaki durum bilgisini tutmaktadır. Genellikle ağın hatalı olduğunu varsaymaktadır. Arıza tespitinde, iki temel fikir bulunmaktadır. İlk fikir başarısızlık tespitidir (failure detection); uygulamadan ayrıştırılarak esnek olmalıdır. İkincisi ise daha geleneksel olan kalp atışı (heartbeat) ile uygulanan karar verme hata detektörüdür. Bu fikre göre bir kalp atışının atıp atmamasına göre bir düğümün yaşadığına ya da öldüğüne karar vermektedir. Ancak gerçekleşen başarısızlık tespiti bu yaklaşımın naif olduğuna karar vermektedir ve ölüler ile canlıların arasında bir yer bulmaktadır.

Anti-entropi (Anti-Entropy), farklı düğümlerdeki verilerin en yeni sürüme güncellenmesini sağlamak için Cassandra'daki kopya eşleme mekanizmasıdır. Anti-entropi manuel bir işlemdir; dedikodu protokolleri tarafından tetiklenmez.

Cassandra'da eşzamanlılık (concurrency) modeli SEDA (Staged Event-Driven Architecture)'ya dayanmaktadır (Welsh, Culler and Brewer, 2001). SEDA'da tek bir işlem bir iş parçacığı (thread) ile başlayabilir sonra başka bir iş parçacığına gönderilebilir. Güçlü bir mimaridir, çünkü eşzamanlılık minimum bağlantı ile yapılabilmektedir.

(Hewitt, 2010, s. 87-98)

#### 4.4.2.3. Avantajları

- Doğrusal ölçeklenebilir bir yapısı vardır bu da geliştirilen bir sistemin kademeli bir şekilde takip edilmesini sağlamaktadır. Cassandra'nın doğrusal ölçeklenebilme özelliği, veriyi tutarlı bir anahtarlama fonksiyonuyla eş düzey düğümlere dağıtma özelliğinden gelmektedir (Lelek, 2017).
- Master-slave yapısı yerine peer-to-peer yapısını kullanmaktadır. Master sistem kullanan yapılarda fazla sayıda istek geldiğinde sunucu bunu karşılayamazsa sistemde sıkıntılar yaşanmaktadır. Fakat Cassandra ile böyle bir problem yaşanmaz, Veri merkezi içerisine istenen sayıda küme eklenebilmektedir (Lelek, 2017).
- Cassandra'nın eşdüzey yapısı sayesinde tek noktadan aksaklık durumlarıyla karşılaşılmaz. Kayıtların kopyalama faktörü parametresiyle ayarlanan sayıda düğüme kopyalanması da erişilebilirliğe olumlu katkı sağlayan avantajlarından biridir.
- Cassandra yazma tutarlılığı parametresini sorgu bazında ayarlamaya izin vermektedir. Özellikle yazma işleminde büyük verim sağlamaktadır. Bu sayede, tekil bazlı ve lot (çoklu) bazlı izlemede farklı yazma tutarlılıkları kullanmak mümkün olabilmektedir. Lot bazlı izlemede, istemciler arası çekişme durumu ortaya çıkabilir. Dolayısıyla, lot bazlı takipte yazma tutarlılığı parametresi daha kısıtlayıcı bir değere ayarlanabilir. Yazma tutarlılığını duruma göre daha az kısıtlayıcı ayarlayabilmek, özellikle performans açısından önemli avantajlar sağlamaktadır.

- Veriler birden fazla veri merkezine kopyalanabilmektedir. En yaygın kullanımı, uygulamaların en yakındaki veri merkezine yazması ve diğer veri merkezlerine kopyalanması şeklindedir. Çoklu veri merkezi kullanılması durumunda kopya sayısı, veritabanı ve veri merkezi bazında ayarlanabilmektedir.
- Cassandra veritabanında sorgulamalar Cassandra Query Language (CQL) adı verilen özel bir dil ile yapılmaktadır. SQL diline çok benzer yapıda olması ilişkisel veritabanlarından geçişi kolaylaştırmaktadır.
- Sütun tabanlı bir veritabanı olması nedeniyle denormalize bir yapıdadır, bu sebeple bir satırda milyonlarca sütun oluşturulabilmektedir.

### 4.4.3. Couchbase Server

Couchbase Server, açık kaynaklı, *Couchbase Inc.* tarafından geliştirilen doküman tabanlı bir NoSQL veritabanıdır. *Couchbase Inc.* şirketi 8 Şubat 2011'de *Membase* ile *CouchOne* (başka bir NoSQL VT olan CouchDB'in üreticisi) şirketlerinin birleşmesi ile kurulmuş ve Couchbase Server ürününü geliştirmeye odaklanmıştır.

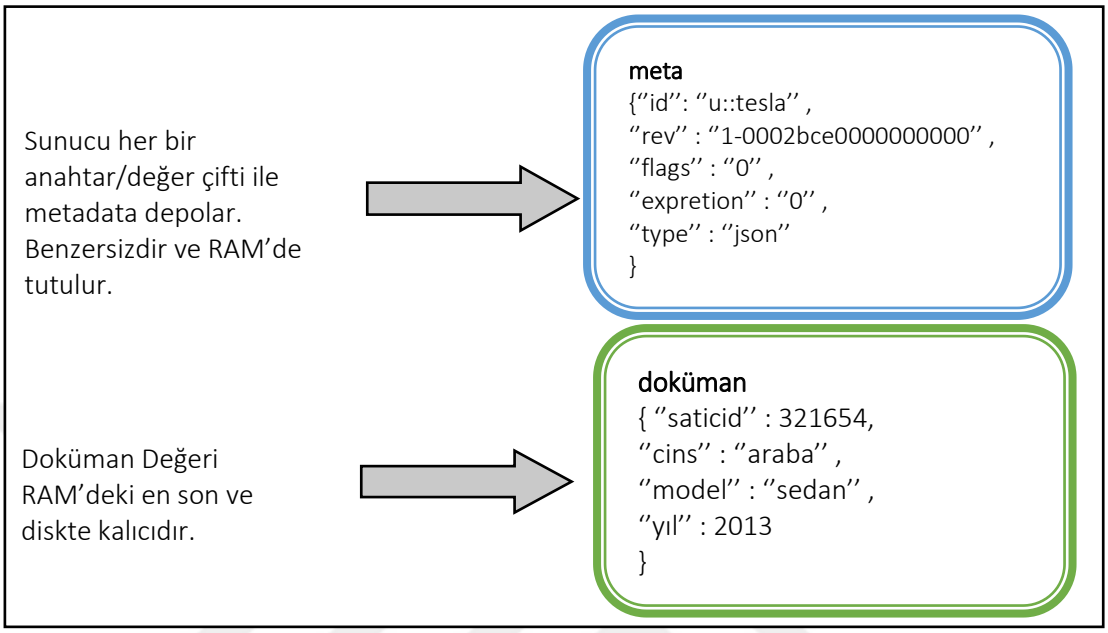
Her Couchbase düğümü bir veri servisi, dizin servisi, sorgu servisi ve küme yöneticisi bileşeninden oluşur. 4.0 sürümünden başlayarak, gerektiğinde üç hizmet kümenin ayrı düğümlerinde çalışmak üzere dağıtılabilir. Eric Brewer'in CAP teoremine göre, Couchbase normalde tutarlılık ve bölüm toleransı sağlayan bir CP (Consistency, Partition Tolerance) sistemi veya birden fazla kümeye kurulabilen bir AP (Availability, Partition Tolerance) sistemdir.

#### 4.4.3.1. Veri Modeli

Bir Couchbase kümesinde dokümanlar JSON biçiminde, kova (bucket) adı verilen bir yapı içinde saklanır. Bu yapı ilişkisel veritabanlarındaki tablo veya MongoDB'deki koleksiyon gibi düşünülebilir. Fakat MongoDB'de küme ile koleksiyon arasında veritabanı gibi bir ara katman varken, Couchbase'de kovalar direkt olarak kümenin altında yer alır. Bu anlamda veritabanına denk gibi de düşünülebilir.

Kova içindeki dokümanlar bir doküman kimliği (ilişkisel veritabanındaki PK gibi) ile eşlenmektedir. Bu eşleşmeye ait meta-veri RAM'de tutulur. Bu meta-veri içinde; anahtar-değer çifti, CAS değeri (Check and Set - eşzamanlılık için gerekli bir veri), TTL

değeri (Time to Leave - saniye cinsinden bir dokümanın son kullanma tarihi) ve saklama biçimi ile ilgili ek değerler yer alır. Dokümanın değeri ise, son kullanılanlar hariç genellikle diskte yer alır. Şekil 4.8'de Couchbase veri modeli gösterilmiştir.




**Şekil 4.8.** Couchbase Veri Modeli

vBucket, bir Couchbase kümesinin anahtar alanının bir alt kümesinin sahibi olarak tanımlanmaktadır. Hem kümedeki verileri etkili bir şekilde dağıtmak hem de birden fazla düğümdeki kopyaları desteklemek için kullanılmaktadır. Her doküman kimliği bir vBucket'e aittir ve hangi vBucket'e ait olduğunu hesaplamak için eşleme fonksiyonu (doküman kimliği alıp vBucket kimliği döndüren bir hash fonksiyonu) kullanılmaktadır.

Couchbase Server'da kullanılan yapılar ile ilişkisel veritabanlarında bu yapılara karşılık gelen ifadeler Çizelge 4.3'te verilmiştir.

**Çizelge 4.3.** Couchbase Server ve RDBMS Veritabanı Yapıları

| **Couchbase Server** | **RDBMS** |
|---|---|
| Kova (Bucket) | Veritabanı |
| | Tablo |
| JSON Doküman | Satır |
| Doküman Kimliği | Birincil Anahtar |

Couchbase Server, MongoDB'den farklı olarak anahtar-değer türü veri modelini de desteklemektedir. Aralarındaki temel farklılıklar Çizelge 4.4'te verilmiştir.

**Çizelge 4.4.** MongoDB ile Couchbase Server Arasındaki Temel Farklar (https://blog.couchbase.com/moving-from-mongodb-to-couchbase-server)

| | **MongoDB** | **Couchbase Server** |
|---|---|---|
| **Veri Modeli** | Doküman (BSON) | Doküman (JSON) ve anahtar-değer |
| **Sorgulama** | Kendine özgü sorgu dili, map/reduce aggregation | N1QL (SQL for JSON), map/reduce görüntüleri, anahtar-değer |
| **Eşzamanlılık** | WiredTiger ile İyimser ve kötümser kilitleme | İyimser ve kötümser kilitleme (bak: 4.4.3) |
| **Ölçekleme Modeli** | Kopya setleri (replica sets) ile Master-Slave | Dağıtık Master-Master |

#### 4.4.3.2. Mimarisi

Yatay ölçeklemeyi kolaylaştırmak amacıyla, Couchbase'de verilerin tüm düğümlere eşit şekilde dağılmasını sağlayan karma paylaşım (hash sharding) kullanır. Sistem 1024 bölüm tanımlar ve bir dokümanın anahtarı belirli bir bölüme eklendiğinde, dokümanın bulunduğu yer bu bölüm olacaktır. Her bölüm kümedeki belirli bir düğüme atanır. Düğümler eklenir veya çıkarılırsa, sistem bölümleri bir düğümden diğerine geçirerek kendini yeniden dengeler. Bir çökme olması durumunda, manuel veya otomatik olarak, sadece kopyalanan kadar düğüm kurtarabilmektedir (Objelean ).

Bir kümedeki her düğüm bir veri yöneticisi ve bir küme yöneticisi olmak üzere iki temel işlemi yürütür. Veri yöneticisi, uygulamalardan veri işlemlerine yanıt olarak dokümanları depolar ve alır. Küme yöneticisi ise, bir kümedeki tüm sunucuların yapılandırmasını ve davranışını denetler.

Veritabanları temelde üç farklı iş yükünü idare eder: temel veri işlemleri, indeksleme ve sorgu işleme. Couchbase Server, bu iş yüklerinin bir kümede bağımsız dağıtımını sağlamak için Veri, Dizin ve Sorgu Servisleri içerir. Her düğüm bu servislerin hepsini veya bir kısmını çalıştırabilir.

Her vBucket, dosya sisteminde ayrı bir dosya olarak temsil edilir. Dokümanlara anahtarları üzerinden hızlıca erişmek için bir B+ ağaç yapısı kullanır. Yazma işlemlerinin daha hızlı ve güvenli olması için, her dosya için sadece sona ekleme türünde bir yazma modeli kullanır. Silinen bir dokümandan kalan boşluk ise sıkıştırma (compaction) yapılarak yeni bir dokümanla doldurulabilmektedir. Arka planda çalışan sıkıştırma işlemi manuel, belirli zamana ayarlanarak veya otomatik olarak yürütülebilir ve sunucunun performansına fazla yük getirmeyecek şekilde tasarlanmıştır.

#### 4.4.3.3. Avantajları

- Master-slave kopyalama desteği ile peer-to-peer desteği bulunmaktadır.
- Yapılandırılabilir kopyalama sayısı vardır.
- Kopyalanmış veriler hızlı bir şekilde kurtarılmaktadır.
- Sorgu dili olan N1QL, SQL diline çok benzediği için ilişkisel veritabanı bilgisi olan kullanıcılar için kolaylık sağlar.
- Dokümanlar, anahtar-doküman şeklinde saklandığı için sorgulama esnasında kolaylık sağlar.

#### 4.4.3.4. Dezavantajları

- Çok fazla RAM kullanmaktadır.
- Sorgulamalar ya tam yapılamamakta ya da yarıda kesilebilmektedir.
- Hem veritabanı hem de tablo niyetine tek bir yapının (kova) kullanılması, ilişkisel veritabanlarından veri aktardığınızda, birden fazla tablodaki kayıtları tek bir kovada saklamanızı gerektirir.

### 4.4.4. NoSQL Veritabanlarında ACID

Her ne kadar NoSQL veritabanları BASE modelini ön plana alsalar da son yıllarda geliştirdikleri yeni sürümlerinde ACID özelliklerini sağlayacak çözümler de sunmaya başlamışlardır. Tezde incelediğimiz 3 farklı NoSQL veritabanının ACID özelliklerine değinmeden önce, bir örnek üzerinden bu özelliklerin ilişkisel veritabanlarında olduğu kadar hayati bir önemde olmadığı gösterilmeye çalışılacaktır.

İlişkisel veritabanı modelinde tablolar arasındaki ilişkiler birbirleri ile aynı verileri saklayan sütunlar üzerinden yapıldığı için, özellikle bu sütunlardan biri üzerinde yapılan bir değişiklik diğer tablonun da tutarlılık açısından kontrol edilmesini gerektirir. Örneğin bir satın alma işlemi ilişkisel veritabanında SİPARİŞ ( SiparişID, KullanıcıID, Tarih ) tablosuna 1 adet, SEPET ( SiparişID, ÜrünID, Fiyat, Miktar ) tablosuna ise o siparişteki ürün adedi kadar kayıt eklenmesini gerektirirken, doküman tabanlı bir modelde tek bir doküman içinde tüm bu veriler saklanabilir (Şekil 4.9).

```
key siparis::001
{
    "musteri": "Berna DUMANLI",
    "tarih": "2018-12-22T13:57:31.2311892-04:00",
    "urunler": [
        { "urunad": "şapka", "fiyat": 39.99, "miktar": 2 },
        { "urunad": "eldiven", "fiyat": 19.99, "miktar": 1 },
        { "urunad": "ayakkabı", "fiyat": 89.99, "miktar": 5 }
    ]
}
```

**Şekil 4.9.** Sipariş Verisini Saklayan Doküman Örneği

Örnekte verilen SiparişID bilgisinin ilişkisel modelde iki farklı tabloda tutulması bu verilerin her daim tutarlılık açısından kontrolünü gerektirirken, doküman tabanlı yaklaşımda tek bir dokümanda saklanması nedeniyle tutarlılık kontrolüne ihtiyaç olmamaktadır.

Tezde incelenen NoSQL veritabanlarının ACID prensiplerini nasıl karşıladığı gösterilecektir.

#### 4.4.4.1. Cassandra

- **Atomicity:** Atomiklik, satır veya bölüm düzeyinde desteklenmektedir; satırdaki sütunları eklemek ya da güncellemek yazma işlemi olarak kabul edilir. Ancak yüksek kullanılabilirlik yapılandırmaları ve hızlı yazma durumlarında atomiklik göz ardı edilmektedir.
- **Consistency:** Nihai tutarlılık göstermektedir. Bölünme toleransına odaklanır.
- **Isolation:** Tam satır düzeyinde yalıtımı desteklemektedir. Yazmayı yapan ilk istemciye kadar satırı tümüyle saklamaktadır.

- **Durability:** Dayanıklılık Cassandra'da en iyi şekilde temsil edilmektedir. Verilerin kopyaları oluşturulduğu gibi, log üzerine kayıt etme de kullanılmaktadır.

  (https://docs.datastax.com/en/cassandra/)

#### 4.4.4.2. MongoDB

- **Atomicity:** MongoDB eski sürümlerinde atomikliği tek-doküman düzeyinde sağlamakta iken, 4.0 sürümü ile birlikte çoklu-doküman düzeyinde atomiklik sağlamaya başlamıştır (şu anda ReplicaSet düzeyinde ama 4.2 sürümü ile birlikte paylaşımlı kümenin tamamı üzerinde olması planlanmaktadır). İlişkisel veri modelinin tersine aynı dokümanın birçok yerde farklı kopyaları bulunmadığı için birçok uygulama aslında tek-doküman düzeyi ile yetinebilmekteydi.
- **Consistency:** Sadece birincil seviyedeki sunucuda yazma işlemleri olduğu için tutarlılık bu seviyede güçlüdür. Varsayılan yapılandırmada ikinci seviyeden okuma yapılmadığı için, bu seviyedeki verinin güncel olmaması pek önemli değildir.
- **Isolation:** Bütünüyle yalıtım sağlamaktadır; doküman güncellendiğinde ve hiçbir hata işlemin geri çekilmesine neden olmazsa, sunucu dokümanın tutarlı bir görünümünü almaktadır.
- **Durability:** Dayanıklılık birinci hedef ise MongoDB bunu sağlayabilmektedir fakat birden fazla kopya olduğunda zorlanabilmektedir. Daha fazla konfigürasyon gerektirmektedir.

#### 4.4.4.3. Couchbase

- **Atomicity:** Tek bir doküman için atomiklik sağlamaktadır. Doküman almak için bir işlem ya başarılı ya da başarısız olmaktadır.
- **Consistency:** Atomik işlem sonucunda başarısız olunursa verinin eski haliyle gösterilmesini gerekli kılmaktadır. Couchbase dokümandan dokümana farklı bir JSON şeması kullanmaya engel değildir. Her bir dokümanın benzersiz bir anahtarı olmalıdır. Diğer veritabanlarına performans olarak fark atmak için verileri eş zamansız olarak güncellemektedir.

- **Isolation:** Couchbase, varsayılan olarak tek bir doküman düzeyinde okumaya yönelik yalıtım sağlamaktadır. Daha iyi bir yalıtım elde etmek için kötümser (pessimistic) kilitleme ve iyimser (optimistic) kilitleme olmak üzere iki çeşit kilit bulunmaktadır. İyimser kilitleme, CAS (Check and Set: kontrol et ve değiştir) olarak adlandırılan bir değere dayanmaktadır. Daha önce belirttiğimiz gibi her dokümanın meta-verisinde CAS değeri vardır. Doküman her değiştiğinde, yeni bir CAS değeri almaktadır. Bir dokümanı güncellemeye çalıştığınızda, işlemin bir parçası olarak CAS değerine iletilmektedir. CAS değerleri eşleşirse, Couchbase çalışmaya izin vermektedir, değilse; işleme izin verilmez ve Couchbase bunun yerine bir hata döndürmektedir. Bir kilidi gerçekten ayarlamak için kötümser kilitleme kullanılmaktadır. Birden çok dokümanı değiştirmek için bir doküman grafiğini kilitlemek istenirse yararlıdır.

  Couchbase'de "GetAndLock" adı verilen bir atomik operasyon bulunmaktadır. Bu işlem dokümanı ve bir CAS değerini döndürmektedir. Bu noktada, doküman "kilitli" olarak kabul edilmektedir. Diğer işlemlerle artık başka kilitler yapılamamakta ve yalnızca CAS değeri dokümanın kilidini açabilmektedir.

- **Durability:** Bilgisayar ağının disk erişiminden daha hızlı olmasından dolayı hatalara karşı dayanıklılık için diğer düğümlere kopyalama Couchbase'de tercih edilen mekanizmadır. Couchbase "memory-first" mimarisine sahiptir. Bu, yazma işlemlerinin sonuçlarının belleğe alındığında onaylanacağı ve daha sonra diske eşzamansız olarak yazılacağı veya kısa bir süre sonra başka bir düğüme kopyalanacağı bir sıraya konduğu anlamına gelmektedir. Yani, bir işlem belleğe yazıldıktan hemen sonra sistem kapanırsa veri kaybı yaşanabilir. Ancak, yüksek performans adına kullanılan bu varsayılan yapılandırma yerine, düşük performansa razı olunarak dayanıklılık gereksinimleri daha güçlü bir dayanıklılık düzeyi belirlenebilir (https://blog.couchbase.com/acid-properties-couchbase-part-1/).

# BÖLÜM 5

# SEÇİLEN VERİTABANLARININ KARŞILAŞTIRILMASI

Bu bölümde, çalışmamızda kullanılan test ortamı açıklanacak ve test sonuçları verilecektir. Literatürdeki birçok karşılaştırmada NoSQL veritabanlarından MongoDB, Cassandra, CouchDB ve Hbase kullanılmıştır. Bu çalışmada ise 4. bölümde bahsedilen MongoDB, Cassandra ve Couchbase veritabanları tercih edilmiştir. İlişkisel veritabanı olarak ise (Taha, 2017) çalışmasından farklı olarak sadece MySQL için değil, aynı zamanda MS SQL Server ve Oracle için de sonuçlar alınmıştır.

## 5.1. Test Ortamı

Karşılaştırma için Oracle Linux Server 7.6 üzerine kurulan altı tane veritabanının sürümleri şu şekildedir: MySQL Community Server 8.0.13, Oracle 12cR2 (12.2.0.1.0), SQL Server 2017 (17.2.0000.1), MongoDB 4.0.4 Enterprise, Couchbase Enterprise Edition 5.1.1 build 5723, Apache Cassandra 3.11. Test bilgisayarının özellikleri ise şu şekildedir: Intel® Core™ i7-3770 CPU, 8GB RAM, 1 TB 7200rpm HDD. Testimizde kullanılan YCSB sürümü ise 0.14.0'tür.

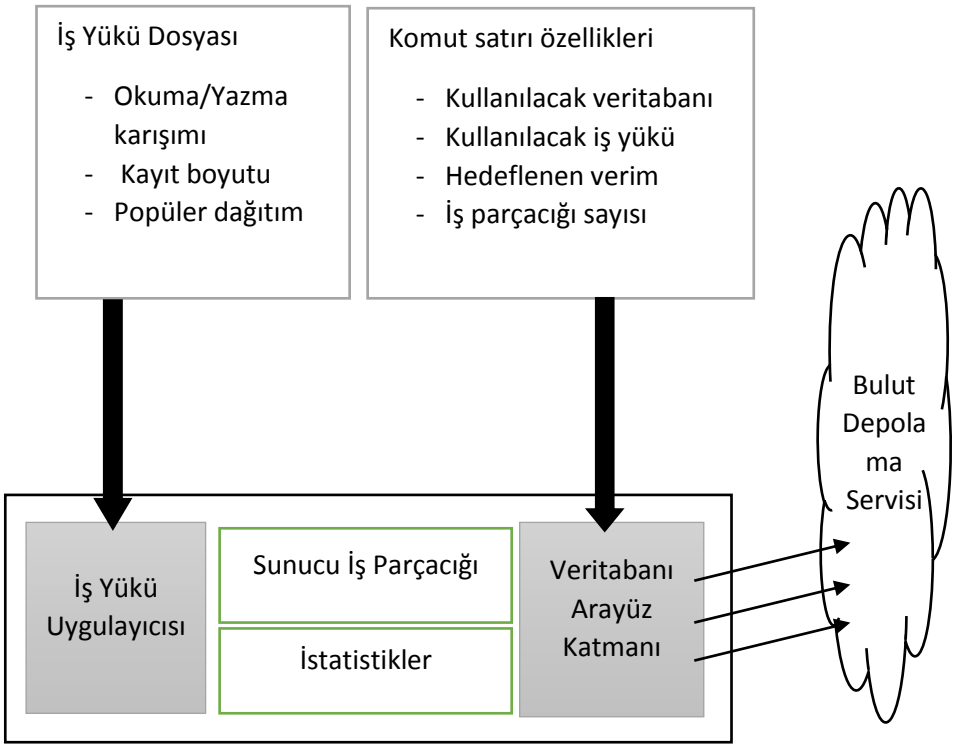
Veritabanları hem YCSB hem de kendi oluşturduğumuz veriler üzerinde sorgulamalar yapılarak test edilmiştir. Sadece YCSB ile yetinmemizin sebebi ise; YCSB anahtar üzerinden değere ulaşılan sorgular ve veri değişikliği yapan (ekleme ve güncelleme gibi) komutlar ile test yapmaktadır. NoSQL veritabanlarının anahtar üzerinden değere ulaşma işlemlerinde hızlı olduğu, fakat değer üzerinden yapılan sorgularda yavaş çalıştıkları bilinmektedir. Testler için oluşturulan sorgular ağırlıklı olarak değer üzerinden kayıtlara erişim ve istatistiksel bilgi elde etme üzerindir. Böylelikle, bu tip sorgularda NoSQL veritabanlarının RDBMS veritabanlarına göre ne derece başarı sağlayabildikleri analiz edilmiştir.

Hem YCSB hem de sorgu ile performans denemelerimizde veritabanlarının her birinden 3'er defa sonuç alınmış ve bunların ortalaması grafik olarak sunulmuştur. Her denemenin sonucunu ayrı ayrı görmek için EKLER kısmındaki tablolara bakılabilir.

## 5.2. YCSB (Yahoo! Cloud Serving Bencmark)

YCSB, veritabanlarına bulut ortamından veri yükleyip, belirli iş yükleri üzerinden objektif performans değerleri elde etmemizi sağlayan bir platformdur. NoSQL veritabanı yönetim sistemlerinin performanslarını karşılaştırmak için sıklıkla kullanılır. Yahoo'nun araştırma bölümünde çalışanlar tarafından 2010 yılında geliştirilmeye başlanmıştır (Cooper, Silberstein, Tam, Ramakrishnan, & Sears, 2010).

Açık kaynak kodlu olduğu için kullanıcının geliştirmesi ve genişletmesi için olanak sağlar. Ekleme, silme, güncelleme ve okuma işlemleri yapmaktadır. Çekirdek pakette okuma, yazma, güncelleme ve eklemeyi farklı senaryolaştırarak birleştiren 6 iş yükü bulunur. Ayrıca kullanıcılar tarafından farklı iş yükleri de oluşturulabilir. Bu iş yükleri hedeflenen saniyedeki işlem sayısına göre (throughput: op/sn) test edilebilir ve ortalama gecikmeler elde edilebilir. YCSB İstemci Mimarisi Şekil 5.1'de verilmiştir.




**Şekil 5.1.** YCSB İstemci Mimarisi

YCSB'nin 6 iş yükü aşağıda verilmiştir (https://github.com/brianfrankcooper/YCSB/):

- **Workload A:** Güncelleme ağırlıklı iş yükü (%50 okuma, %50 güncelleme)
- **Workload B:** Okuma ağırlıklı iş yükü (%95 okuma, %5 güncelleme)
- **Workload C:** Sadece okuma iş yükü (%100 okuma)
- **Workload D:** En son eklenen verilerin okunduğu iş yükü (%95 okuma, %5 ekleme)
- **Workload E:** Ayrı bölgelerdeki kayıtların tek tek değil, kısa aralıklardaki kayıtların topluca okunduğu (tarama) iş yükü (%95 tarama, %5 ekleme)
- **Workload F:** Okunan kayıtların değiştirilerek tekrar yazıldığı iş yükü (Read-modify-write)

Bu iş yüklerinin hepsi aynı veri setini kullandıkları için her iş yükü öncesi tabloya veri yükleme (load) yapmaya gerek yoktur. Fakat D ve E iş yükleri %5 oranında veri ekleme de yaptıkları için aynı iş yükü tekrar edilecekse veya sonrasında başka bir iş yükü yürütülecekse, mevcut veri silinerek tekrar yükleme işlemi yapılmalıdır. Testin gerçekleştirme adımları şu şekildedir:

1. İlk olarak veritabanı sisteme kurulmalıdır.
2. Kullanılacak iş yükü seçilmelidir.
3. Çalışma için uygun parametreler hazırlanmalıdır.
4. Veri yüklenmelidir.
5. İş yükü çalıştırılmalıdır.

### 5.2.1. MongoDB Konfigürasyon Adımları

1. MongoDB servisini açma: `sudo service mongod start`
2. Yükleme yapmak için kullanılan komut satırı (Workload A için):

   ```
   bin/ycsb load mongodb-async -s -P workloads/workloada
   ```

3. Çalıştırmak için kullanılan komut satırı (Workload A için):

   ```
   bin/ycsb run mongodb-async -s -P workloads/workloada
   ```

### 5.2.2. Cassandra Konfigürasyon Adımları

1. Cassandra servisini açma: `sudo service cassandra start`
2. CQLSH kullanılarak "ycsb" adında bir anahtar alanı tanımlanmalıdır:

```
Create keyspace ycsb WITH REPLICATION =
  {'class': 'SimpleStrategy', 'replication factor': 4};
```

3. CQLSH ile "ycsb" anahtar alanı üzerinde testlerde kullanılacak olan "usertable" adında bir tablo yaratılmalıdır:

```
USE ycsb;
create table usertable ( ycsb_id varchar primary key,
                         field0 varchar, field1 varchar,
                         field2 varchar, field3 varchar,
                         field4 varchar, field5 varchar,
                         field6 varchar, field7 varchar,
                         field8 varchar, field9 varchar);
```

4. Yükleme yapmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb load cassandra-cql -P workloads/workloada -p
hosts=localhost
```

5. Çalıştırmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb run cassandra-cql -P workloads/workloada -p
hosts=localhost
```

### 5.2.3. Couchbase Konfigürasyon Adımları

1. Couchbase servisini açma: `sudo service couchbase-server start`
2. Couchbase üzerinde **ycsb** isminde bir kullanıcı ve **bucket** yaratılır.
3. Zorunlu olmasa da komut satırında daha az parametre vermek için **host, bucket, password** gibi bilgilerin yer aldığı **couchbase2.properties** yapılandırma dosyası hazırlanabilir (https://github.com/brianfrankcooper/YCSB/tree/master/couchbase2).

4. Yükleme yapmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb load couchbase2 -P workloads/workloada -P
couchbase2.properties
```

5. Çalıştırmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb run couchbase2 -P workloads/workloada -P
couchbase2.properties
```

### 5.2.4. MySQL Konfigürasyon Adımları

1. MySQL servisini açma: `sudo service mysqld start`
2. MySQL istemcisine bağlanıp (`mysql -u root -p` ile) bir veritabanı yaratıldıktan sonra, bu veritabanı altında testlerde kullanılacak olan "usertable" adında bir tablo yaratılmalıdır:

```
create database TestDB;
use TestDB;
create table usertable ( ycsb_key varchar(255) primary key,
                         field0 text, field1 text,
                         field2 text, field3 text,
                         field4 text, field5 text,
                         field6 text, field7 text,
                         field8 text, field9 text);
```

3. **mysql.properties** yapılandırma dosyası hazırlanmalıdır:

```
db.driver = com.mysql.jdbc.Driver
db.url = jdbc:/mysql://127.0.0.1:3306/ycsb
db.user = admin
db.password = …
```

4. Yükleme yapmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb load jdbc -P workloads/workloada -P mysql.properties
-cp mysql-connector-java-8.0.11.jar
```

5. Çalıştırmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb run jdbc -P workloads/workloada -P mysql.properties
-cp mysql-connector-java-8.0.11.jar
```

YCSB üzerinde RDBMS testleri yapabilmek için o ilişkisel veritabanına ait JDBC sürücüsünün indirilmesi gerekmektedir. Bu sürücü veritabanı işlemlerini yapmamızı sağlayan standart Java API'sidir. MySQL için indirdiğimiz sürücünün adı "mysql-connector-java-8.0.11.jar" olduğu için komut satırında –cp parametresine bu dosya ismi yazıldı. Sonraki bölümlerde MS SQL Server için "mssql-jdbc-6.4.0.jre8.jar", Oracle için ise "ojdbc8.jar" JDBC sürücülerinin dosya isimleridir.

### 5.2.5. MS SQL Server Konfigürasyon Adımları

1. MS SQL Server servisini açma: `sudo service mssql-server start`
2. SQLCMD ile bir veritabanı yaratıldıktan sonra, bu veritabanı altında testlerde kullanılacak olan "usertable" adında bir tablo yaratılmalıdır (MySQL ile aynı).
3. **mssql.properties** yapılandırma dosyası hazırlanmalıdır.
4. Yükleme yapmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb load jdbc -P workloads/workloada -P mssql.properties
-cp mssql-jdbc-6.4.0.jre8.jar
```

5. Çalıştırmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb run jdbc -P workloads/workloada -P mssql.properties
-cp mssql-jdbc-6.4.0.jre8.jar
```

### 5.2.6. Oracle Konfigürasyon Adımları

1. Oracle servisini açmak için SQLPLUS'a SYS kullanıcısı ile bağlanarak (`sqlplus sys/[şifre] as sysdba`) STARTUP yazılmalıdır.
2. Diğer RDBMS'lerde olduğu gibi yine bir veritabanı yaratılıp, altında testlerde kullanılacak olan "usertable" adında bir tablo yaratılmalıdır.

3. **oracle.properties** yapılandırma dosyası hazırlanmalıdır.
4. Yükleme yapmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb load jdbc -P workloads/workloada -P
oracle.properties -cp ojdbc8.jar
```

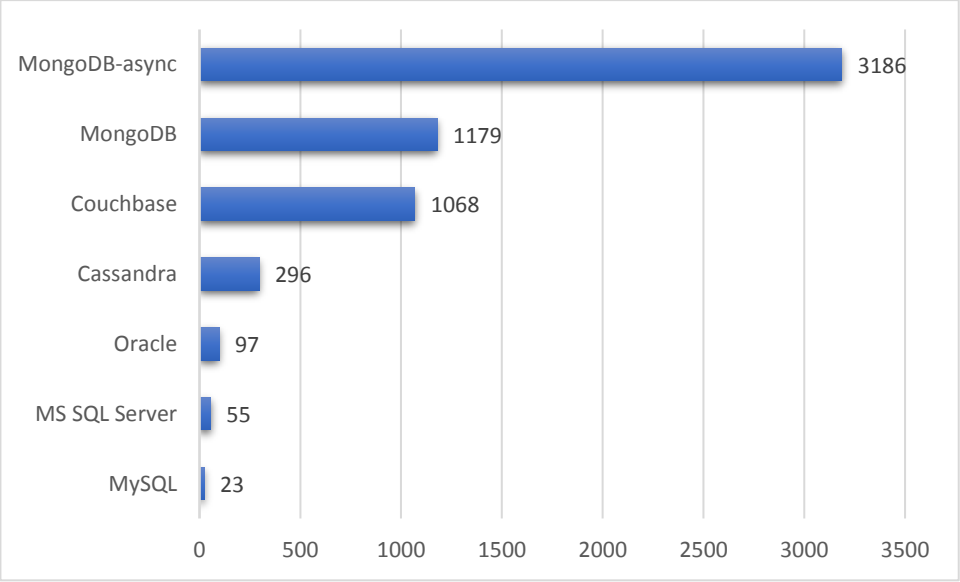
5. Çalıştırmak için kullanılan komut satırı (Workload A için):

```
bin/ycsb run jdbc -P workloads/workloada -P
oracle.properties -cp ojdbc8.jar
```

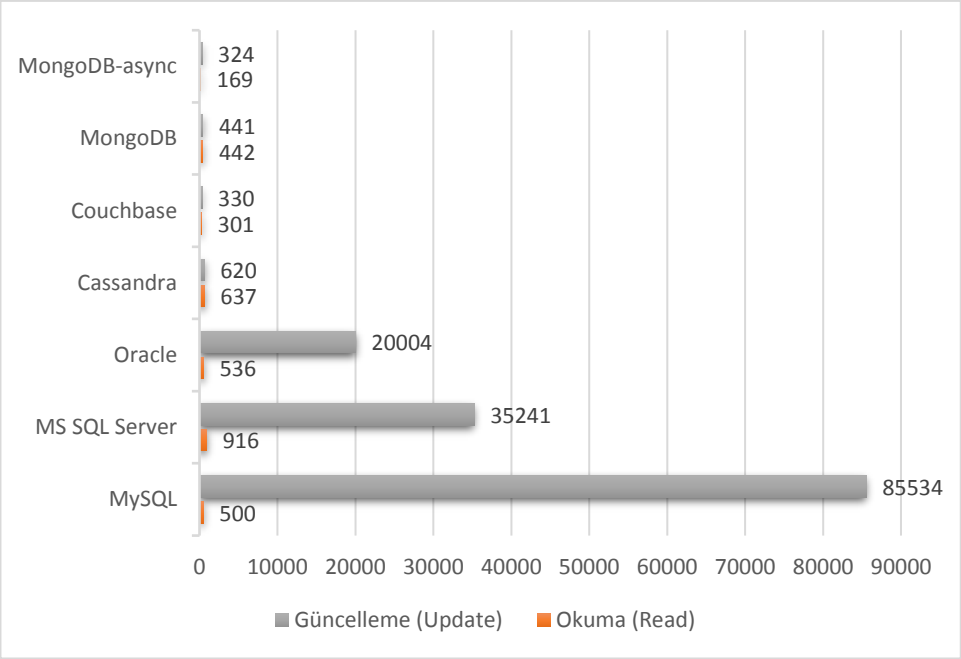
### 5.3. YCSB Testinin Sonuçları

Standart 6 iş yükü üzerinde yapılan test sonuçları bu kısımda verilmiş, elde edilen sonuçlar üzerine yorumlar ise son bölüme bırakılmıştır. Kullandığımız YCSB sürümünde MongoDB için hem asenkron hem de senkron sürücüler mevcut olduğundan her ikisi ile de test yapılmıştır. 5.2.1 bölümündeki "load" ve "run" komut satırlarında `mongodb-async` yazılmıştı, senkron sürücü için ise sadece `mongodb` yazılır.

**Workload A:** Güncelleme ağırlıklı iş yükü olarak adlandırılan bu iş yükü %50 okuma ve %50 güncelleme işlemi içerir. Son eylemleri kaydeden oturum örnek olarak verilebilir. Şekil 5.2'de bu iş yükü için elde edilen Throughput değerleri, Şekil 5.3'te ise okuma ve güncelleme için ortalama gecikmeler verilmiştir.

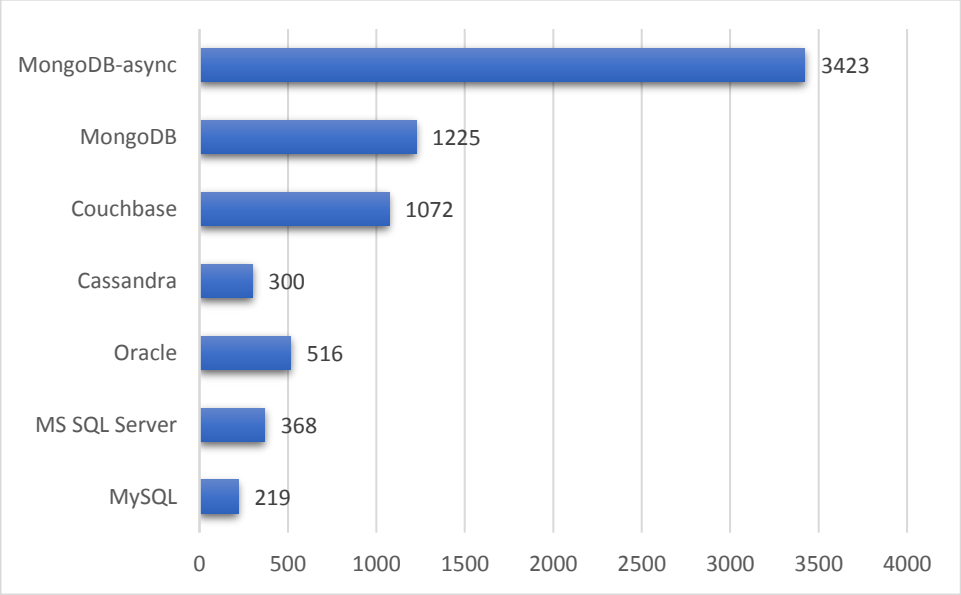



**Şekil 5.2.** Workload A - Throughput (op/sn)

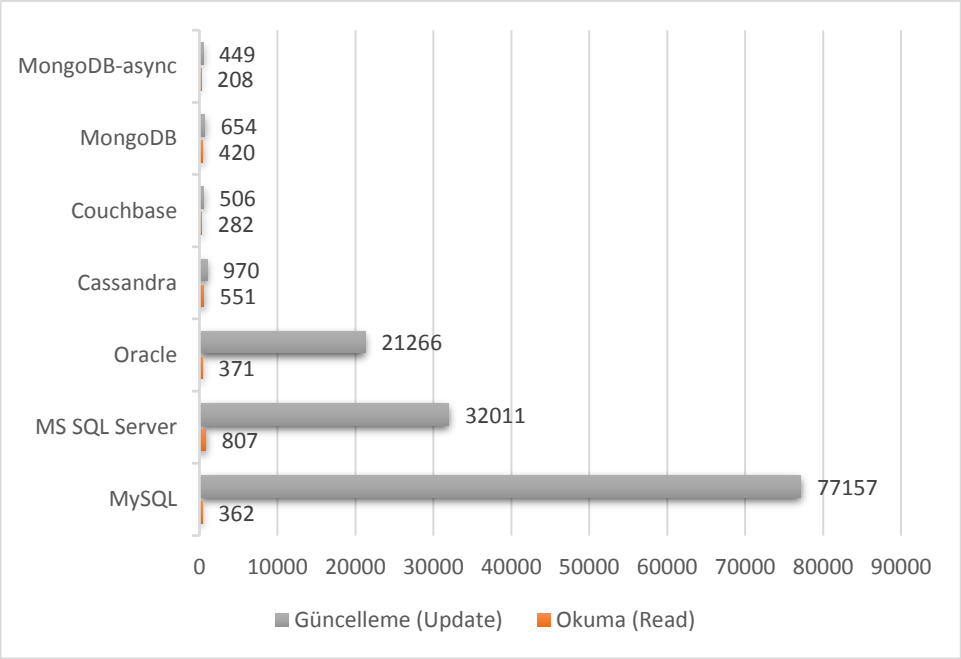



**Şekil 5.3.** Workload A - Ortalama Gecikmeler (µs)

**Workload B:** Okuma ağırlıklı iş yüküdür. %95 okuma, %5 güncelleme içermektedir. Fotoğraf etiketlemek büyük anlamda okuma ama aynı zamanda güncelleme işlemidir. Şekil 5.4'te bu iş yükü için elde edilen Throughput değerleri, Şekil 5.5'te ise okuma ve güncelleme için ortalama gecikmeler verilmiştir.

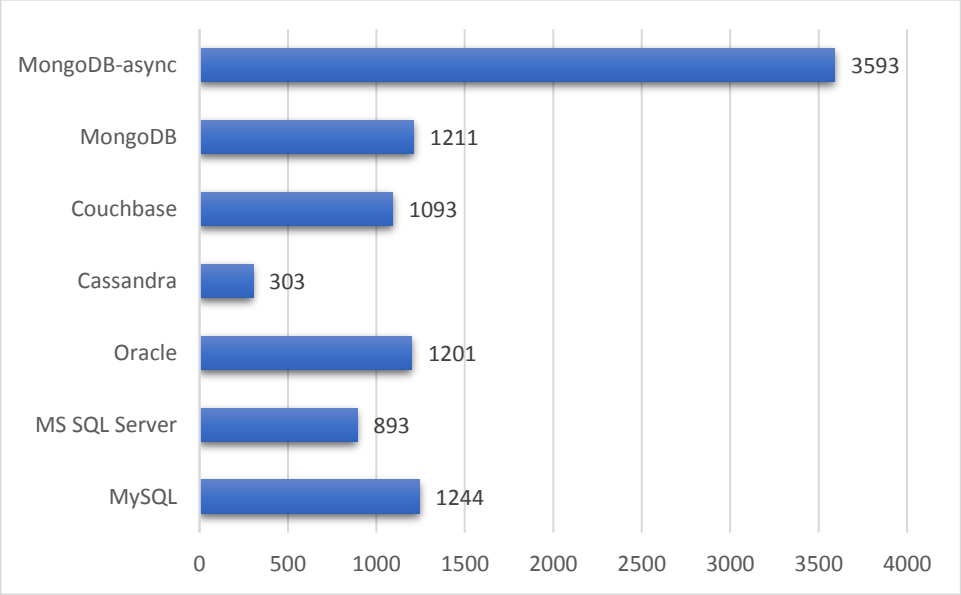



**Şekil 5.4.** Workload B - Throughput (op/sn)

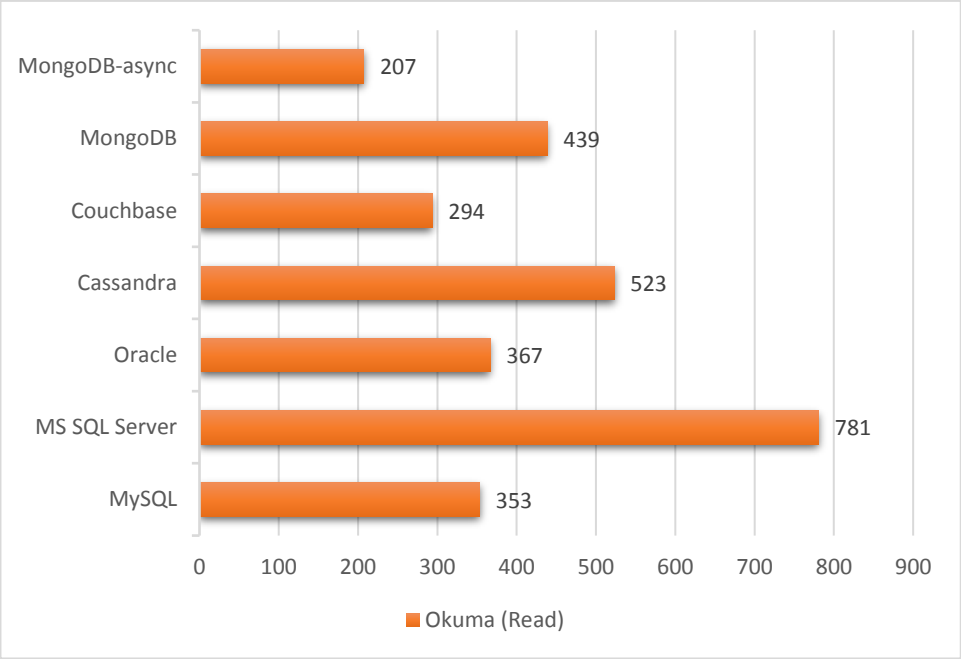



**Şekil 5.5.** Workload B - Ortalama Gecikmeler (µs)

**Workload C:** %100 okuma içeren sadece okuma iş yüküdür. Profillerin başka yerlerde oluşturulduğu kullanıcı profil önbelleği örnek olarak verilebilmektedir. Şekil 5.6'da bu iş yükü için elde edilen Throughput değerleri, Şekil 5.7'de ise ortalama gecikmeler verilmiştir.

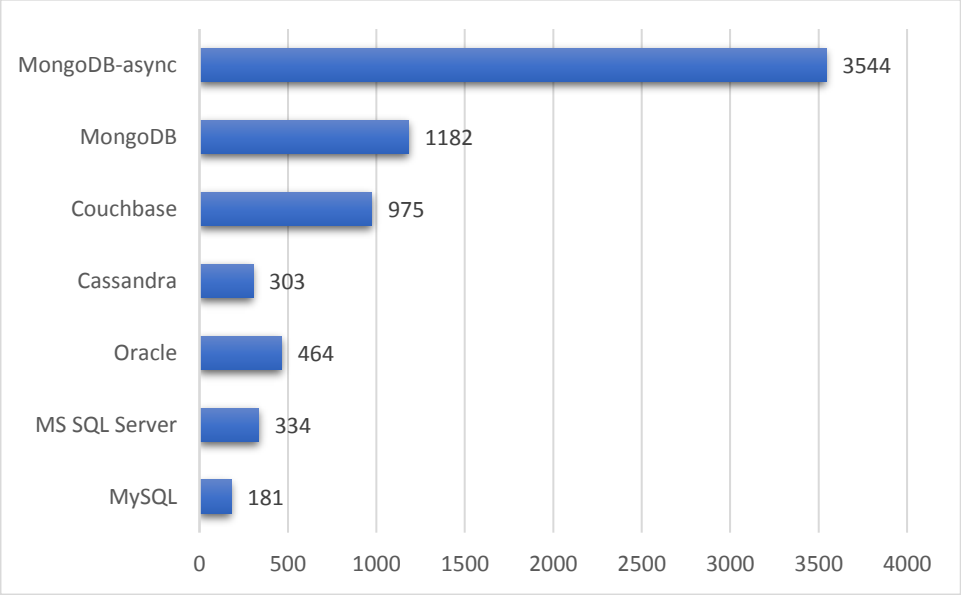



**Şekil 5.6.** Workload C - Throughput (op/sn)

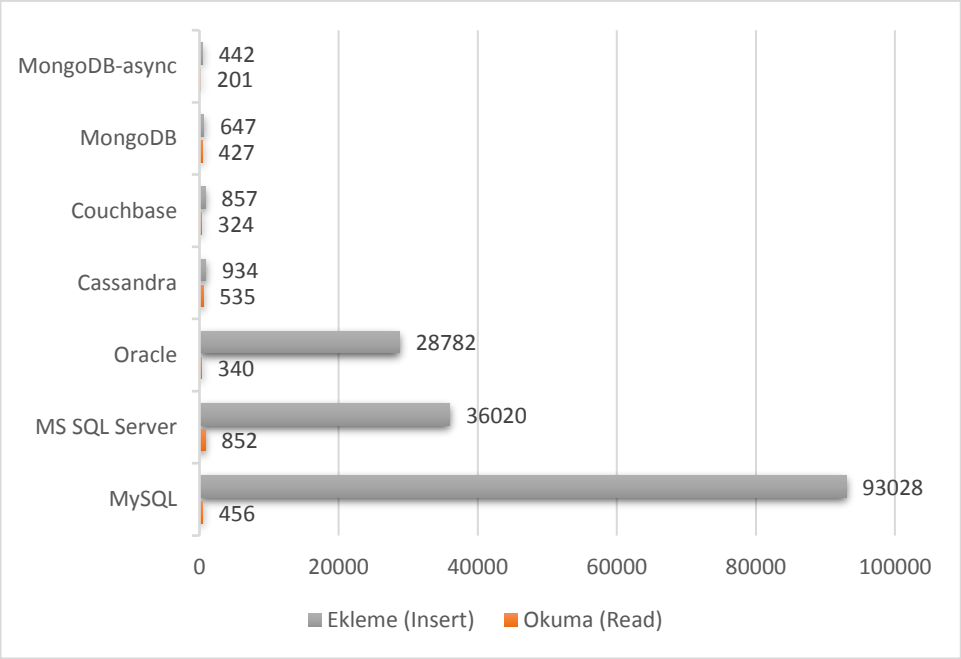



**Şekil 5.7.** Workload C - Ortalama Gecikmeler (µs)

**Workload D:** Yeni kayıtlar eklenir, en son eklenen en popülerdir. %95 okuma, %5 ekleme içermektedir. Kullanıcı durum güncellemeleri örnek olarak verilebilir. En son durumları insanlar merak eder. Şekil 5.8'de bu iş yükü için elde edilen Throughput değerleri, Şekil 5.9'da ise okuma ve ekleme için ortalama gecikmeler verilmiştir.

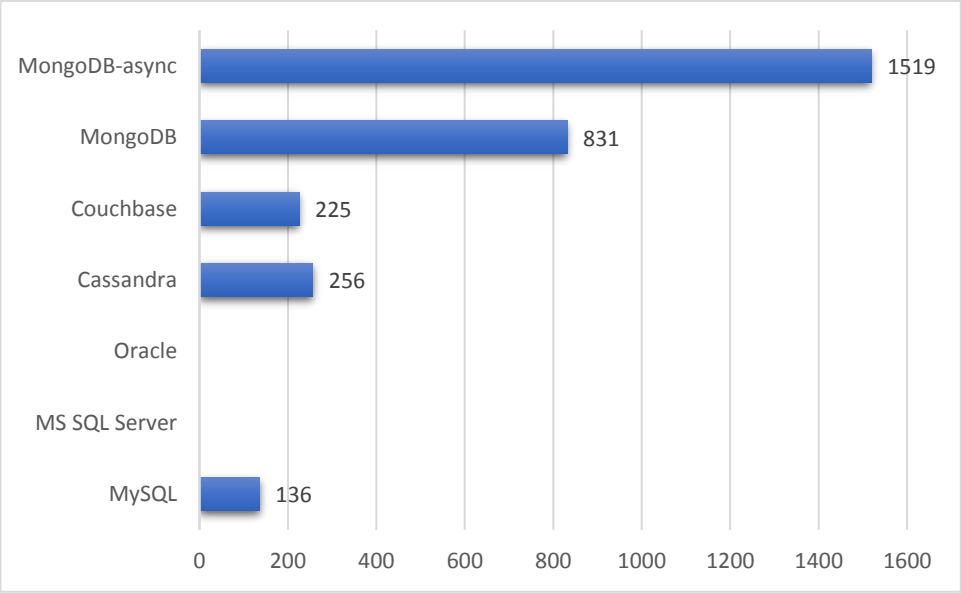



**Şekil 5.8.** Workload D - Throughput (op/sn)

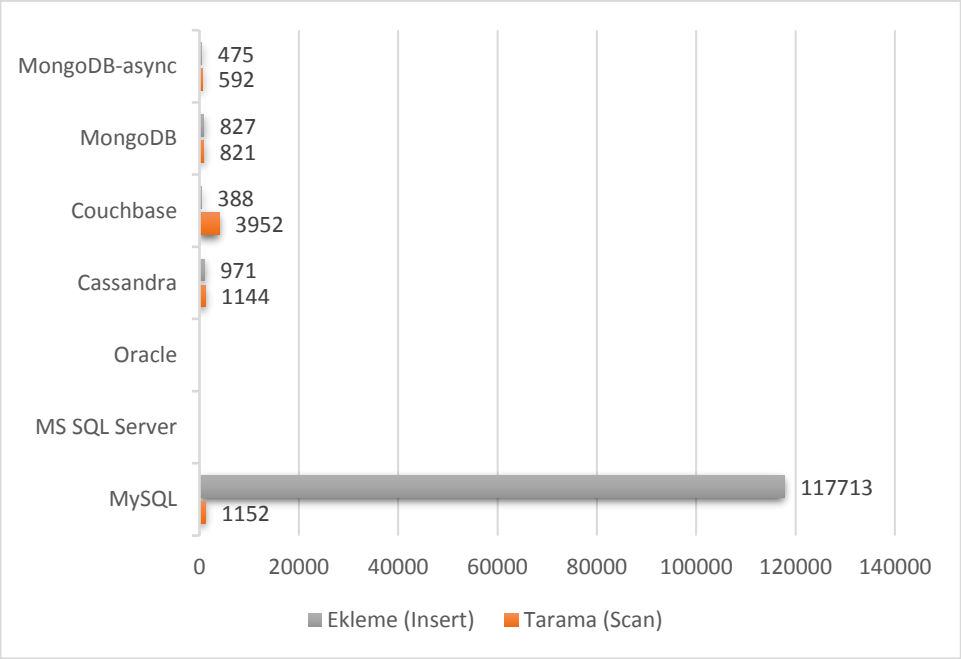



**Şekil 5.9.** Workload D - Ortalama Gecikmeler (µs)

**Workload E:** Tekil kayıtlar yerine, kısa aralıktaki kayıtlar sorgulanmaktadır. %95 tarama, %5 ekleme içermektedir. Şekil 5.10'da bu iş yükü için elde edilen Throughput değerleri, Şekil 5.11'de ise okuma ve tarama için ortalama gecikmeler verilmiştir. Oracle ve MS SQL Server tarama işlemi için uygun komutu alamadıkları için bu iş yükünde çalışmamışlardır.

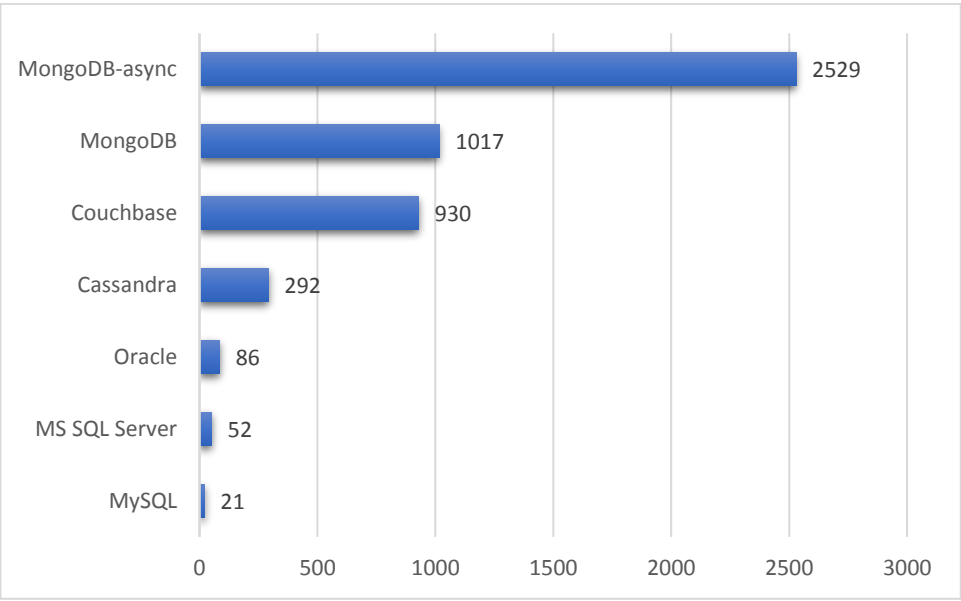



**Şekil 5.10.** Workload E - Throughput (op/sn)

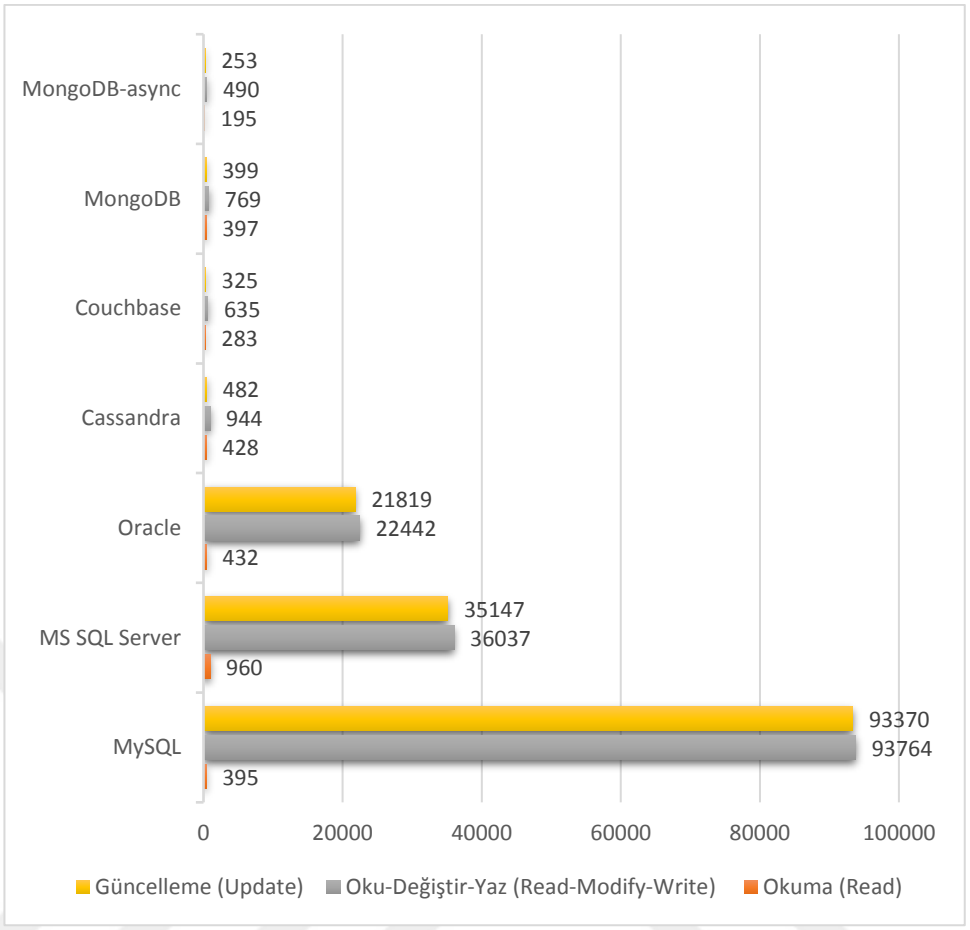



**Şekil 5.11.** Workload E - Ortalama Gecikmeler (µs)

**Workload F:** İstemci bir kayıt okuyacak, değiştirecek ve değişiklikleri yazacaktır. Kullanıcı kayıtlarının kullanıcı tarafından okunup değiştirildiği veya kullanıcı etkinliğini kaydettiği kullanıcı veritabanı örnek olarak verilebilmektedir. Şekil 5.12'de bu iş yükü için elde edilen Throughput değerleri, Şekil 5.13'te ise okuma, oku-değiştir-yaz ve güncelleme için ortalama gecikmeler verilmiştir.




**Şekil 5.12.** Workload F - Throughput (op/sn)




**Şekil 5.13.** Workload F- Ortalama Gecikmeler (µs)

## 5.4. Sorgu Testinin Sonuçları

Sorgularda kullanılan "kelimeler" ve "satirlar" tablolarını oluşturmak için "Pizza & Chili Corpus" üzerinden elde edilen "English.200MB" dosyası kullanılmıştır (http://pizzachili.dcc.uchile.cl/texts/nlang/). Bu dosyadaki noktalama işaretleri temizlendikten sonra, boşluk karakterlerinden bölerek bulunan 36.832.584 adet kelimenin her biri "words" adındaki sütuna ayrı bir kayıt olarak yazılarak "kelimeler" tablosu oluşturulmuştur (MongoDB'de koleksiyon adı olarak "words", içindeki dokümanlarda value olarak "word" kullanılmıştır). Benzer şekilde "satirlar" tablosunu oluşturmak için de satır sonu (\n) karakterinden bölerek bulunan 3.470.639 adet satır kullanılmıştır. Ortalama hesabında kullanılan "sayilar" tablosu için ise rastgele üretilen 10.000.000 adet 32-bit tamsayı kullanılmıştır. Veriler önce CSV türündeki dosyalara yazılmış ve bu dosyalardan veritabanlarına aktarılmıştır.

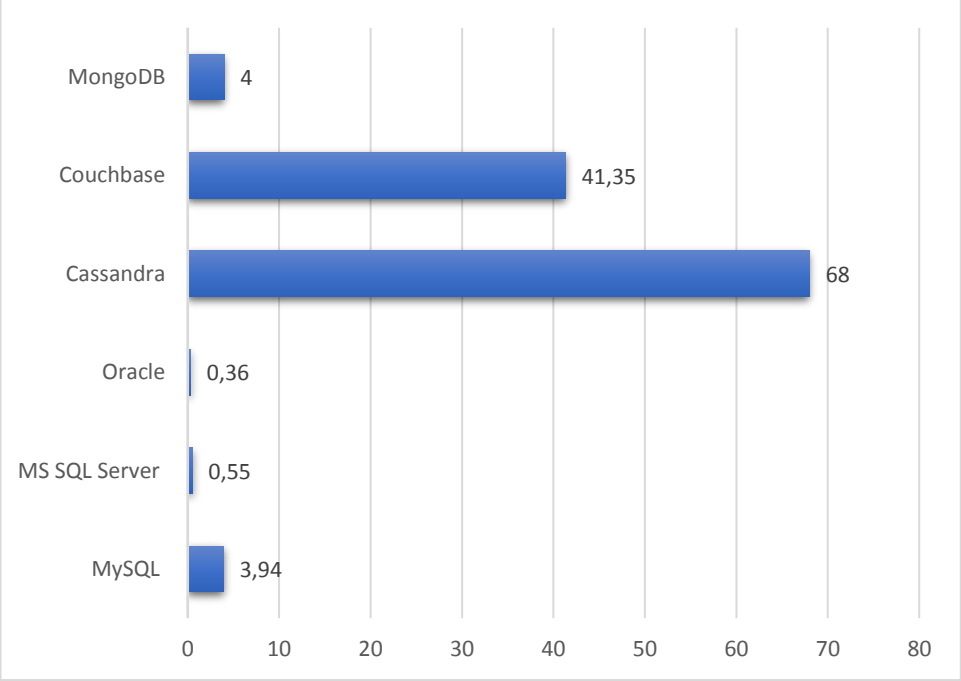
**Sorgu 1:** Birinci sorgumuzda 10 milyon tam sayının ortalaması alınmıştır (AVG). MongoDB haricindeki diğer beş veritabanının sorgu yazım şekli aynıdır. MySQL, MS SQL Server, Oracle, Couchbase ve Cassandra için yazılan sorgu satırı:

```
select avg(sayi) from sayilar;
```

MongoDB için yazılan sorgu satırı:

```
db.sayilar.aggregate({
    $group: {
       _id: null,
       ortalama: { $avg: "$sayi" }
    }
})
```

Şekil 5.14'te test sonuçları gösterilmektedir. Ortalama hesaplama işini en hızlı yapan Oracle olmuştur. MS SQL Server da Oracle'a yakın bir sonuç verirken, MySQL ise Oracle'dan 11 kat, MS SQL Server'dan 7 kat daha yavaş hesaplayabilmiştir. MongoDB *aggregation pipeline* mimarisinin avantajı ile MySQL'e yakın bir sonuç verebilmiştir. Couchbase 41,35 sn'lik sonucu "sayi" alanı üzerinde indeks oluşturulunca elde etmiştir.




**Şekil 5.14.** 10 milyon sayının ortalamasını bulma süresi (sn)

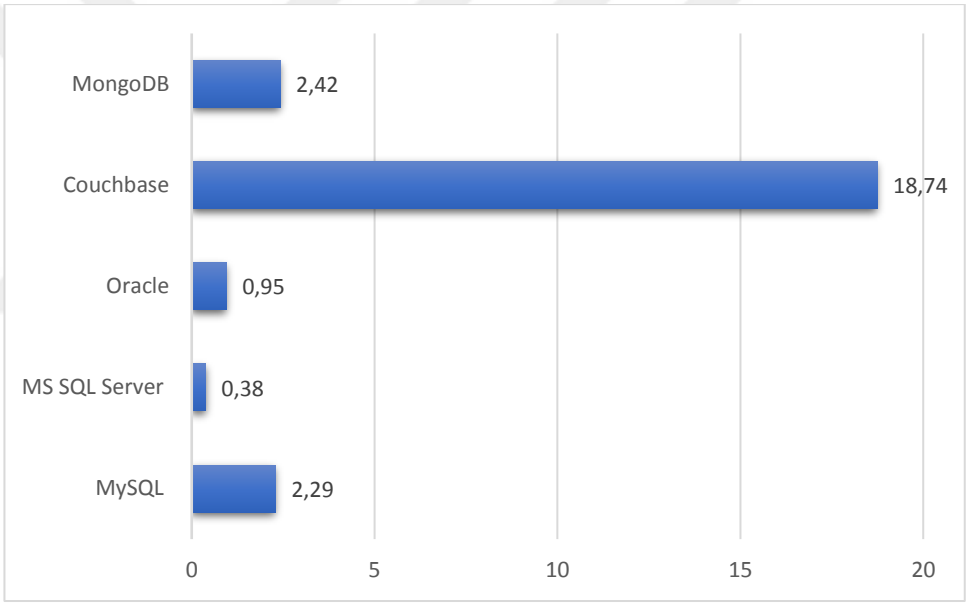
**Sorgu 2:** İkinci sorgumuzda "satirlar" tablosunda yer alan string'ler içinde satırların kaç tanesinde "the" kelimesi geçtiği bulunmuştur. MongoDB haricindeki veritabanlarında kullanılan sorgu satırı:

```
select count(*) from satirlar where line like '%the%';
```

MongoDB için yazılan sorgu satırı:

```
db.satirlar.find({line:/the/});
```

Veritabanlarının bu sorguyu tamamlama süreleri Şekil 5.15'te gösterilmiştir. Bu sorguda en hızlı sonuç üreten veritabanı MS SQL Server olmuştur. Oracle'dan 2,5 kat, MySQL'den de 6 kat daha hızlı sürede sorguyu tamamlamıştır. MongoDB yine MySQL'e yakın bir sürede tamamlayarak NoSQL veritabanlarının en iyisi olmuştur.




**Şekil 5.15.** Satırların kaç tanesinde 'the' kelimesi geçtiğini bulma süresi (sn)

Couchbase'in bu sorguyu yürütebilmesi için de "line" alanı üzerinde indeks oluşturmak gerekmiştir. Cassandra'da string içinde kelime arayabilmek için aşağıdaki kod ile SASI türünde bir indeks oluşturulmuş, ama sorgu sonucu 0 çıkmıştır.

```
CREATE CUSTOM INDEX fn_contains ON satirlar (line)
USING 'org.apache.cassandra.index.sasi.SASIIndex'
WITH OPTIONS = { 'mode': 'CONTAINS',
'analyzer_class': 'org.apache.cassandra.index.sasi.analyzer.NonTokenizingAnalyzer',
'case_sensitive': 'false'};
```

**Sorgu 3:** Üçüncü sorgumuzda 36.8 milyon kayıt içeren "kelimeler" tablosu kullanılmıştır. Sorguda COUNT kullanılarak "the" kelimesinin kaç kayıtta yer aldığını bulma süresi hesaplanmıştır. Sorgulama hem indeks kullanarak hem de kullanmadan yapılmıştır. MongoDB haricindeki veritabanlarında kullanılan sorgu satırı:

```
select count(*) from kelimeler where words = 'the';
```

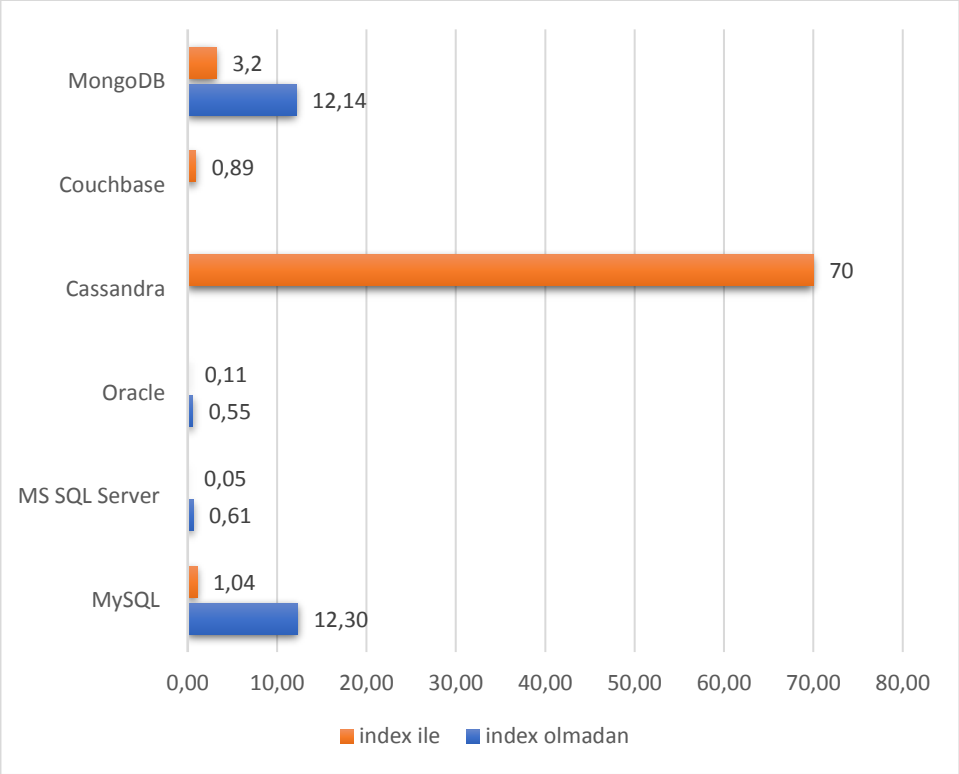
MongoDB için kullanılan sorgu satırı:

```
db.words.find({word:"the"}).count();
```

İndeks oluşturulmak için kullanılan komut satırı:

```
create index index1 on kelimeler(words);
```

Üçüncü sorgunun sonuçları Şekil 5.16'da verilmiştir. En iyi sonucu veren MS SQL Server indeks ile Oracle'dan 2 kat hızlı sonuç üretmiş, indeks olmadığında ise Oracle'ın gerisinde kalmıştır. Couchbase ve Cassandra indeks olmadan sonuç üretememiştir. MongoDB indeks olmadan MySQL'i geçebilmiş, ama indeks kullanıldığında 3 kat yavaş kalmıştır. NoSQL veritabanlarının birincisi Couchbase olmuştur.




**Şekil 5.16.** 'the' kelimesinin kaç kayıtta olduğunu bulma süresi (sn)

**Sorgu 4:** Dördüncü sorgumuzda "kelimeler" tablosu üzerinde kelimelerin tekrar sayılarına göre azalan sırada sıralanması gerçekleştirilmiştir. Diğer sorgulara göre daha karmaşık olan bu sorguda COUNT ile birlikte ORDER BY ve GROUP BY kullanılmıştır. Couchbase ve Cassandra üzerinde bu karmaşık sorgu çalıştırılamamıştır. Sorgu 3 gibi, bu sorgu da hem indeks kullanarak hem de kullanmadan yürütülmüştür.

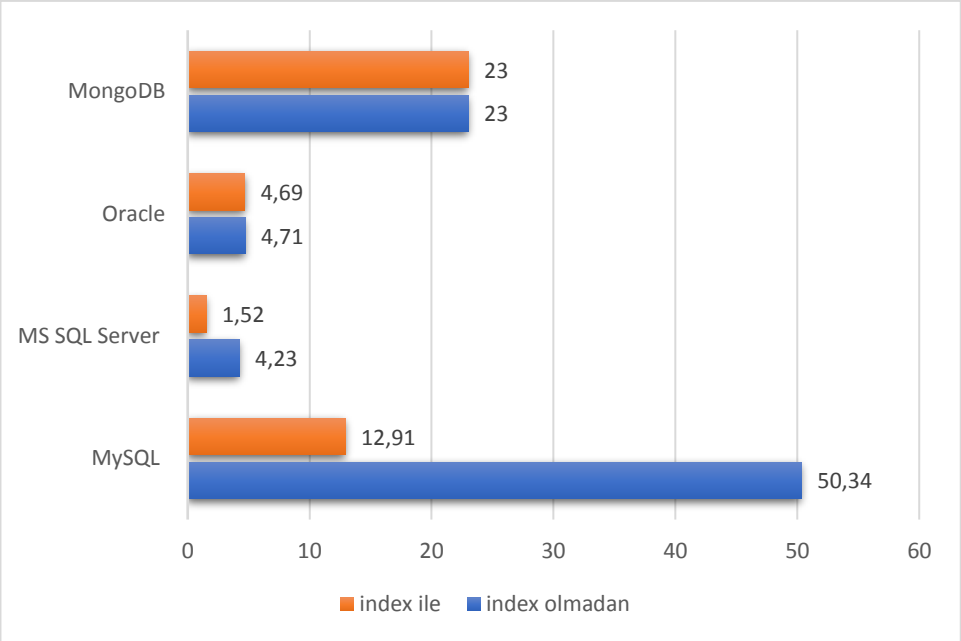
MySQL, MS SQL Server, Oracle için kullanılan sorgu satırı:

```
select words, count(*) from kelimeler
group by words
order by count(*) desc;
```

MongoDB için kullanılan sorgu satırı:

```
db.words.aggregate(
   [
      {
        $group: {
           _id: "$word",
           tekrar: { $sum: 1 }
        }
      },
      { $sort: { tekrar: -1 } }
   ],{ allowDiskUse: true }
)
```

Sorgunun süre sonuçları Şekil 5.17'de verilmiştir. MS SQL Server bu sorguda da en iyi sonuçları vermiştir. Şekilden görülebileceği gibi Oracle ve MongoDB bu sorgu için yaratılan indeksten faydalanmamıştır. İndeks kullanıldığında MS SQL Server yaklaşık 3 kat, MySQL ise yaklaşık 4 kat daha hızlı sonuç üretmiştir.




**Şekil 5.17.** Kelimeleri tekrar sayılarına göre sıralama süresi (sn)

# BÖLÜM 6

# SONUÇLAR

Internet kullanımının artmasıyla birlikte son yıllarda web siteleri üzerinden veritabanlarına yüklenen veri boyutları devasa hale gelmiştir. Bu veriler yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış olarak üç gruba ayrılabilir. RDBMS'lerde ölçekleme işleminin genellikle dikey olarak yapılmasından ve bu işlemin hem zor hem de yüksek maliyetli olmasından dolayı alternatif olarak NoSQL veritabanları doğmuştur. Yatay ölçeklendirmenin kolay ve düşük maliyetle yapılabilmesi dışında, yarı yapılandırılmış veya yapılandırılmamış türden verileri saklamanın da kolay olmasından dolayı NoSQL veritabanları avantajlıdır. Veritabanları, kullanıcıların ihtiyaçlarına göre tercih edilmektedir, kötü veya iyi veritabanı diye adlandırmak bu yüzden yanlıştır. Her veritabanının kendini üstün hale getirdiği bir kullanım alanına sahiptir.

Bu tez çalışmamızda 3 NoSQL veritabanının (MongoDB, Couchbase, Cassandra) ve 3 ilişkisel veritabanının (MySQL, MS SQL Server, Oracle) karşılaştırma sonuçları sunulmuştur. Bu veritabanları hakkında Bölüm 3 ve Bölüm 4'te temel bilgiler verilmiştir. YCSB ve Sorgu Sonuçları olarak iki ayrı kısımda incelenebilecek test sonuçları ise Bölüm 5'te verilmiştir. Test ettiğimiz veritabanları farklı mimarileri nedeniyle farklı iş yükleri ve sorgular üzerinde farklı sonuçlar üretmiştir.

YCSB platformu üzerinde gerçekleştirilen testlerde, literatürde yer alan bazı çalışmalara göre farklı sonuçlar elde edilmiştir. (Klein, 2015) ve (Veronika & Bernardino, 2013) çalışmalarının aksine, MongoDB'nin performansı Cassandra'ya oranla daha iyi çıkmıştır. Couchbase de MongoDB'ye yakın performans göstermiştir. Cassandra'nın performansının düşük çıkmasının nedeni, muhtemelen testlerin tek bir düğüm üzerinde gerçekleştirilmiş olmasıdır. İlişkisel veritabanlarını kişisel bilgisayarlardan oluşan bir küme üzerine kurmanın zorluğundan ve elimizde daha yüksek kapasiteli bir sunucu olmamasından dolayı tek düğüm ile yetinilmiştir. Eğer çok

düğüme sahip büyük bir küme üzerinde çalışılsaydı, farklı sonuçlar elde edilebileceği düşünülmektedir.

Ortalama gecikme sürelerine bakıldığında, Couchbase genellikle MongoDB'nin asenkron sürücüsüne göre geride kalmasına rağmen senkron sürücüsünü geçmiştir. Fakat throughput olarak her iki sürücüsünün de gerisinde kalmıştır. İlişkisel veritabanları okuma gecikme sürelerinde NoSQL veritabanları ile başa baş sonuçlar vermelerine rağmen, ACID garantisi nedeniyle ekleme ve güncelleme gecikmelerinde çok geride kalmışlardır. Okuma gecikme sürelerinde en başarılı RDBMS MySQL olmuştur. %100 okuma işleminin yapıldığı Workload C'de elde ettiği throughput değeri ile sadece Oracle ve MS SQL Server'ı değil, NoSQL veritabanlarını da geride bırakmıştır (MongoDB-async hariç). Fakat ekleme ve güncelleme işlemlerinde çok yavaş kalması nedeniyle, diğer iş yüklerinde MySQL en kötü throughput değerlerini elde etmiştir. Oracle tüm iş yüklerinde MS SQL Server'ı geride bırakmıştır.

Sorgu testinde, tüm veritabanlarına aynı veriler eklenerek, bu veriler üzerinde farklı türde sorgular ile veritabanları test edilmiştir. İlişkisel veritabanlarının tamamında ve MongoDB'de yürütülebilen bu sorguların bazıları Cassandra ve Couchbase üzerinde çalıştırılamamıştır. Test edilen 4 sorgudan 2 tanesinde indeksli sonuçlar da elde edildiği için elde edilen 6 sonuca göre; 4 tanesinde en iyi çıkan MS SQL Server ile 2 tanesinde en iyi çıkan Oracle'ın sorgu testlerinin en iyileri olduğu söylenebilir. MongoDB, MySQL'e yakın performans göstererek sorgu testlerinde de NoSQL veritabanlarının en iyisi olmuştur. Sadece *Sorgu 3*'te Couchbase MongoDB ve MySQL'i geçmiştir.

İndeks kullanıldığında *Sorgu 3* için MS SQL Server ve MySQL 12 kat daha hızlı sonuç üretirken, Oracle'da bu oran 5 kat olabilmiştir. *Sorgu 4* için ise MS SQL Server 3 kat ve MySQL 4 kat daha hızlı sonuç üretirken, Oracle indeksten faydalanmamıştır. Couchbase ve Cassandra Sorgu 3'ü indeks kullanılmadığında tamamlayamamışlardır. Cassandra ayrıca Sorgu 2 için de sonuç üretememiştir.

Bu tez çalışmasına daha önceki tez çalışmalarından farklı olarak 3 farklı ilişkisel veritabanı ve daha önce herhangi bir tezde kullanılmayan Couchbase dâhil edilmiştir. Ayrıca sadece YCSB ile yetinilmeyip, sorgu performansları açısından da değerlendirme yapılmıştır. RDBMS'lerin sorgu performansının özellikle tek düğüm kullanıldığında çok daha iyi olduğu ispatlanmıştır.

# KAYNAKLAR


Abramova, V., Bernardino, J., & Furtado, P. (2014). Which NoSQL Database?A Performance Overview. *Open Journal of Databases (OJDB)*, 19.

Aladily, A. (2015). *The performance wise comparison of the most widely used noSQL databases* (Yayınlanmamış Yüksek Lisans Tezi). Kadir Has Üniversitesi/Fen Bilimleri Enstitüsü, İstanbul.

Altaros. ( 2014). *NoSQL Technical Comparison Report.* Araştırma Raporu.

Asiltürk, O. (n.d.). *Oracle Nedir?* Retrieved from Oracle Nedir?: http://www.fibiler.com/Divisions/Ehil/Mecmua/Magazines/Articles/txt/html/article_WhatIsOracle.html

Aydemir, F. (2016). *Sınır Güvenliği İçin Büyük Veri Teknik Ve Teknolojileri İle Boru Hattı Tasarımı* (Yayınlanmamış Yüksek Lisans Tezi). Gazi Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara

Bhamra, K. (2017). *A Comparative Analysis of MongoDB and Cassandra (Yüksek Lisans Tezi)*. University of Bergen/ Department of Informatics, Norway.

Bondi, André B. *(2000). Ölçeklenebilirliğin özellikleri ve performansa etkileri . Yazılım ve performans üzerine ikinci uluslararası çalıştayın bildirileri - WOSP '00. s. 195.* doi : 10,1145 / 350.391,350432

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 377-387.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010). YCSB. *Benchmarking Cloud Serving Systems with YCSB*. Yahoo Research.

Evans, E., & Oskarsson, J. (2009). NoSQL 2009. San Fransisco: Last.fm.

Gortan, I., & Klein, J. (2014). Big Data; Architectures and Technologies. *SEI Conference SATURN2014.* Portland.

Gökşen, Y., & Aşan, H. (2015). *Veri Büyüklüklerinin Veritabanı Yönetim Sistemlerinde Meydana Getirdiği Değişim; NoSQL*, 125-131.

Hammood, A. H. (2016). *NoSQL veritabanı sistemlerinin karşılaştırılması: MongoDB, apache HBase, and apache Cassandra üzerine bir çalışma (*Yayınlanmamış Yüksek Lisans Tezi*)*. Çankaya Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara.

Hewitt, E. (2010). *Cassanra: The Definitive Guide.* Sebastepol: O'Reilly Media.

Cassandra, https://academy.datastax.com/planet-cassandra/what-is-apache-cassandra/, Erişim Tarihi:01.09.2018

Couchbase Özellik, https://blog.couchbase.com/acid-properties-couchbase-part-1/, Erişim Tarihi: 02.09.2018

Couchbase Depolama Mimarisi, https://docs.couchbase.com/server/5.1/architecture/storage-architecture.html, Erişm Tarihi:01.07.2018

Couchbase- MongoDB Karşılaştırma, https://blog.couchbase.com/moving-from-mongodb-to-couchbase-server, Erişim Tarihi: 01.07.2018

Cassandra, https://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlTransactionsDiffer.html, Erişim Tarihi: 10.08.2018

YCSB Workloads, https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads, Erişim Tarihi: 20.05.2018

YCSB couchbase2, https://github.com/brianfrankcooper/YCSB/tree/master/couchbase2, Erişim Tarihi: 09.09.2018

Cassandra Data Model, https://www.tutorialspoint.com/cassandra/cassandra_data_model.htm, Erişim Tarihi: 02.09.2018

Klein, J. (2015). Performance evaluation of nosql databases: A case study. *1st Workshop on Performance Analysis of Big Data Systems.* ACM.

Lelek, T. (2017). *Learning Apache Cassandra.* Packt.

Lith, A., & Mattsson, J. (2010). Investigating storage solutions for large data (Yüksek Lisans Tezi). Chalmers University Of Technology/Department of Computer Science and Engineering, Sweden

Objelean , A. https://www.todaysoftmag.com/article/1506/introduction-to-couchbase-nosql-document-database. *Today Software Magazine*.

Otey, M. (2010). *NoSQL? No Way! No fear that SQL Server will be replaced by nonrelational databases.* SQL Server Magazine.

Shwaysh, M. M. (2018). *Security and performance comparison of NoSQL database systems* (Yayınlanmamış Yüksek Lisans Tezi). Çankaya Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara.

Singh, K. (2015). *Survey of NoSQL Database Engines for Big Data.*

Swaminathan, S. N. (2015). *QUANTITATIVE ANALYSIS OF SCALABLE NOSQL DATABASES.* (Yayınlanmamış Yüksek Lisans Tezi). The University Of Texas At Arlington, ABD.

Taha, İ. A. (2017). *A comprehensive comparison of NOSQL and relational database management systems* (Yayınlanmamış Yüksek Lisans Tezi) Çankaya Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara.

Veronika, A., & Bernardino, J. (2013). NoSQL databases: MongoDB vs Cassandra. *international computer science and software engineering Conf.* ACM.

Welsh, M., Culler, D., & Brewer, E. (2001). SEDA: an architecture for well-conditioned, scalable internet services. *SOSP '01 Proceedings of the eighteenth ACM symposium on Operating systems principles*, 230-243.

World Internet Users Statics and 2018 World Population Stats, https://www.internetworldstats.com/stats.htm, Erişim Tarihi: 20.12.2018

Yishan, L., & Manoharan, S. ( 27-29 Aug. 2013). A performance comparison of SQL and NoSQL databases. *EE Pacific Rim Conference on Communications*, (pp. 15-19). BC, Vistoria.

# EKLER

Bu bölümde testlerde elde ettiğimiz 3 farklı sonuç ve grafiklerde gösterilen ortalama değerleri verilmiştir.

Workload A Sonuçları

| | **Throughput** | **Read Latency** | **Update Latency** |
|---|---|---|---|
| MySQL 1 | 24,26 | 562,78 | 83.541,98 |
| MySQL 2 | 21,69 | 478,55 | 86.058,71 |
| MySQL 3 | 22,74 | 458,95 | 87.002,06 |
| **MySQL Ortalama** | **22,89** | **500,09** | **85.534,25** |
| MS SQL Server 1 | 56,60 | 1.012,84 | 34.047,72 |
| MS SQL Server 2 | 48,95 | 910,71 | 38.956,43 |
| MS SQL Server 3 | 59,55 | 825,08 | 32.719,74 |
| **MS SQL Server Ortalama** | **55,03** | **916,21** | **35.241,30** |
| Oracle 1 | 95,99 | 391,45 | 20.846,68 |
| Oracle 2 | 98,45 | 632,58 | 19.013,60 |
| Oracle 3 | 95,91 | 585,03 | 20.152,02 |
| **Oracle Veritabanı Ortalama** | **96,79** | **536,35** | **20.004,10** |
| Cassandra 1 | 293,17 | 588,84 | 645,70 |
| Cassandra 2 | 287,94 | 714,81 | 711,00 |
| Cassandra 3 | 305,62 | 606,56 | 503,73 |
| **Cassandra Ortalama** | **295,58** | **636,74** | **620,14** |
| Couchbase 1 | 1.061,57 | 294,55 | 340,20 |
| Couchbase 2 | 1.082,25 | 284,46 | 306,64 |
| Couchbase 3 | 1.060,45 | 323,10 | 341,77 |
| **Couchbase Server Ortalama** | **1.068,09** | **300,70** | **329,54** |
| MongoDB 1 | 1.144,16 | 430,93 | 471,12 |
| MongoDB 2 | 1.186,24 | 423,86 | 456,64 |
| MongoDB 3 | 1.207,73 | 470,01 | 396,45 |
| **MongoDB Ortalama** | **1.179,38** | **441,60** | **441,41** |
| MongoDB-async 1 | 3.125,00 | 164,93 | 330,65 |
| MongoDB-async 2 | 3.289,47 | 162,74 | 314,74 |
| MongoDB-async 3 | 3.144,65 | 179,37 | 326,34 |
| **MongoDB-async Ortalama** | **3.186,38** | **169,01** | **323,91** |

Workload B Sonuçları

| | Throughput | Read Latency | Update Latency |
|---|---|---|---|
| MySQL 1 | 206,57 | 370,27 | 78.012,62 |
| MySQL 2 | 230,04 | 359,79 | 74.432,67 |
| MySQL 3 | 219,59 | 356,17 | 79.026,00 |
| **MySQL Ortalama** | **218,73** | **362,08** | **77.157,09** |
| MS SQL Server 1 | 358,81 | 846,37 | 31.429,23 |
| MS SQL Server 2 | 360,36 | 770,15 | 32.424,60 |
| MS SQL Server 3 | 385,95 | 803,85 | 32.180,00 |
| **MS SQL Server Ortalama** | **368,37** | **806,79** | **32.011,28** |
| Oracle 1 | 507,87 | 385,39 | 21.891,61 |
| Oracle 2 | 553,71 | 373,09 | 20.995,29 |
| Oracle 3 | 485,91 | 353,72 | 20.911,35 |
| **Oracle Veritabanı Ortalama** | **515,83** | **370,74** | **21.266,08** |
| Cassandra 1 | 306,75 | 476,88 | 908,70 |
| Cassandra 2 | 298,78 | 557,16 | 1.066,71 |
| Cassandra 3 | 294,38 | 620,41 | 933,29 |
| **Cassandra Ortalama** | **299,97** | **551,48** | **969,56** |
| Couchbase 1 | 1.063,83 | 289,09 | 531,18 |
| Couchbase 2 | 1.097,69 | 272,64 | 532,32 |
| Couchbase 3 | 1.053,74 | 284,49 | 455,00 |
| **Couchbase Server Ortalama** | **1.071,76** | **282,07** | **506,16** |
| MongoDB 1 | 1.245,33 | 393,21 | 699,10 |
| MongoDB 2 | 1.231,53 | 419,91 | 661,95 |
| MongoDB 3 | 1.197,60 | 447,30 | 601,61 |
| **MongoDB Ortalama** | **1.224,82** | **420,14** | **654,22** |
| MongoDB-async 1 | 3.472,22 | 197,89 | 449,78 |
| MongoDB-async 2 | 3.496,50 | 210,98 | 420,00 |
| MongoDB-async 3 | 3.300,33 | 216,51 | 478,28 |
| **MongoDB-async Ortalama** | **3.423,02** | **208,46** | **449,35** |

Workload C Sonuçları

| | **Throughput** | **Read Latency** |
|---|---|---|
| MySQL 1 | 1.290,32 | 340,39 |
| MySQL 2 | 1.209,19 | 365,29 |
| MySQL 3 | 1.231,53 | 354,74 |
| **MySQL Ortalama** | **1.243,68** | **353,47** |
| MS SQL Server 1 | 928,51 | 741,90 |
| MS SQL Server 2 | 868,06 | 803,65 |
| MS SQL Server 3 | 881,06 | 796,83 |
| **MS SQL Server Ortalama** | **892,54** | **780,79** |
| Oracle 1 | 1.191,90 | 358,77 |
| Oracle 2 | 1.204,82 | 361,68 |
| Oracle 3 | 1.204,82 | 380,26 |
| **Oracle Veritabanı Ortalama** | **1.200,51** | **366,90** |
| Cassandra 1 | 311,24 | 447,21 |
| Cassandra 2 | 300,93 | 531,91 |
| Cassandra 3 | 296,38 | 590,44 |
| **Cassandra Ortalama** | **302,85** | **523,19** |
| Couchbase 1 | 1.096,49 | 299,07 |
| Couchbase 2 | 1.076,43 | 297,77 |
| Couchbase 3 | 1.104,97 | 284,57 |
| **Couchbase Server Ortalama** | **1.092,63** | **293,81** |
| MongoDB 1 | 1.200,48 | 431,13 |
| MongoDB 2 | 1.201,92 | 451,72 |
| MongoDB 3 | 1.230,01 | 433,30 |
| **MongoDB Ortalama** | **1.210,81** | **438,72** |
| MongoDB-async 1 | 3.663,00 | 205,88 |
| MongoDB-async 2 | 3.690,04 | 199,84 |
| MongoDB-async 3 | 3.424,66 | 216,59 |
| **MongoDB-async Ortalama** | **3.592,57** | **207,44** |

Workload D Sonuçları

| | **Throughput** | **Read Latency** | **Insert Latency** |
|---|---|---|---|
| MySQL 1 | 173,91 | 606,84 | 81.409,08 |
| MySQL 2 | 200,88 | 382,44 | 80.246,77 |
| MySQL 3 | 167,14 | 378,16 | 117.428,73 |
| **MySQL Ortalama** | **180,65** | **455,81** | **93.028,19** |
| MS SQL Server 1 | 348,43 | 854,97 | 31.481,19 |
| MS SQL Server 2 | 284,50 | 817,78 | 44.951,55 |
| MS SQL Server 3 | 369,55 | 882,00 | 31.627,67 |
| **MS SQL Server Ortalama** | **334,16** | **851,59** | **36.020,13** |
| Oracle 1 | 486,85 | 369,39 | 23.622,13 |
| Oracle 2 | 375,66 | 272,76 | 43.728,00 |
| Oracle 3 | 529,38 | 378,96 | 18.996,15 |
| **Oracle Veritabanı Ortalama** | **463,96** | **340,37** | **28.782,09** |
| Cassandra 1 | 296,65 | 597,60 | 1.077,76 |
| Cassandra 2 | 301,75 | 572,51 | 963,04 |
| Cassandra 3 | 309,79 | 435,50 | 760,52 |
| **Cassandra Ortalama** | **302,73** | **535,20** | **933,77** |
| Couchbase 1 | 1.021,45 | 310,70 | 924,33 |
| Couchbase 2 | 966,18 | 307,64 | 770,56 |
| Couchbase 3 | 937,21 | 354,99 | 876,35 |
| **Couchbase Server Ortalama** | **974,95** | **324,44** | **857,08** |
| MongoDB 1 | 1.168,22 | 410,76 | 710,04 |
| MongoDB 2 | 1.228,50 | 417,59 | 604,56 |
| MongoDB 3 | 1.149,43 | 454,09 | 625,32 |
| **MongoDB Ortalama** | **1.182,05** | **427,48** | **646,64** |
| MongoDB-async 1 | 3.484,32 | 204,26 | 439,44 |
| MongoDB-async 2 | 3.745,32 | 185,80 | 394,70 |
| MongoDB-async 3 | 3.401,36 | 213,73 | 491,36 |
| **MongoDB-async Ortalama** | **3.543,67** | **201,27** | **441,83** |

Workload E Sonuçları

| | **Throughput** | **Scan Latency** | **Insert Latency** |
|---|---|---|---|
| MySQL 1 | 158,91 | 1.073,32 | 85.258,25 |
| MySQL 2 | 128,85 | 1.243,43 | 142.020,47 |
| MySQL 3 | 119,13 | 1.138,31 | 125.859,20 |
| **MySQL Ortalama** | **135,63** | **1.151,69** | **117.712,64** |
| MS SQL Server 1 | SCAN işlemi hata veriyor (LIMIT yüzünden) | | |
| MS SQL Server 2 | | | |
| MS SQL Server 3 | | | |
| **MS SQL Server Ortalama** | | | |
| Oracle 1 | SCAN işlemi hata veriyor (LIMIT yüzünden) | | |
| Oracle 2 | | | |
| Oracle 3 | | | |
| **Oracle Veritabanı Ortalama** | | | |
| Cassandra 1 | 257,67 | 1.132,34 | 1.011,28 |
| Cassandra 2 | 259,47 | 1.078,20 | 950,00 |
| Cassandra 3 | 251,00 | 1.222,69 | 953,15 |
| **Cassandra Ortalama** | **256,05** | **1.144,41** | **971,48** |
| Couchbase 1 | 222,72 | 4.002,83 | 323,69 |
| Couchbase 2 | 232,40 | 3.768,27 | 550,64 |
| Couchbase 3 | 220,99 | 4.085,56 | 290,87 |
| **Couchbase Server Ortalama** | **225,37** | **3.952,22** | **388,40** |
| MongoDB 1 | 823,72 | 842,50 | 661,02 |
| MongoDB 2 | 833,33 | 789,65 | 1.228,53 |
| MongoDB 3 | 836,12 | 829,73 | 592,04 |
| **MongoDB Ortalama** | **831,06** | **820,63** | **827,20** |
| MongoDB-async 1 | 1.531,39 | 589,14 | 461,04 |
| MongoDB-async 2 | 1.562,50 | 568,74 | 471,25 |
| MongoDB-async 3 | 1.461,99 | 619,59 | 492,91 |
| **MongoDB-async Ortalama** | **1.518,63** | **592,49** | **475,07** |

Workload F Sonuçları

| | Throughput | Read Latency | Read-Modify-Write Latency | Update Latency |
|---|---|---|---|---|
| MySQL 1 | 20,10 | 398,96 | 96.890,38 | 96.486,75 |
| MySQL 2 | 22,81 | 392,44 | 84.920,20 | 84.536,77 |
| MySQL 3 | 19,86 | 395,05 | 99.482,90 | 99.085,76 |
| **MySQL Ortalama** | **20,92** | **395,48** | **93.764,49** | **93.369,76** |
| MS SQL Server 1 | 51,52 | 904,14 | 34.739,99 | 33.883,27 |
| MS SQL Server 2 | 54,21 | 1.027,36 | 35.435,85 | 34.487,21 |
| MS SQL Server 3 | 51,21 | 948,43 | 37.936,07 | 37.070,37 |
| **MS SQL Server Ortalama** | **52,31** | **959,98** | **36.037,30** | **35.146,95** |
| Oracle 1 | 91,03 | 444,61 | 21.167,93 | 20.520,47 |
| Oracle 2 | 87,17 | 409,43 | 21.358,77 | 20.777,95 |
| Oracle 3 | 79,10 | 441,12 | 24.800,51 | 24.157,32 |
| **Oracle Veritabanı Ortalama** | **85,77** | **431,72** | **22.442,40** | **21.818,58** |
| Cassandra 1 | 290,61 | 423,33 | 930,88 | 494,46 |
| Cassandra 2 | 291,04 | 477,09 | 989,01 | 484,95 |
| Cassandra 3 | 294,99 | 382,08 | 911,80 | 465,52 |
| **Cassandra Ortalama** | **292,21** | **427,50** | **943,90** | **481,64** |
| Couchbase 1 | 955,11 | 277,71 | 602,05 | 310,29 |
| Couchbase 2 | 923,36 | 273,78 | 607,88 | 333,38 |
| Couchbase 3 | 910,75 | 296,88 | 695,26 | 332,03 |
| **Couchbase Server Ortalama** | **929,74** | **282,79** | **635,06** | **325,23** |
| MongoDB 1 | 1.020,41 | 396,24 | 770,94 | 405,94 |
| MongoDB 2 | 1.058,20 | 383,57 | 723,99 | 374,66 |
| MongoDB 3 | 972,76 | 411,13 | 812,07 | 417,43 |
| **MongoDB Ortalama** | **1.017,12** | **396,98** | **769,00** | **399,34** |
| MongoDB-async 1 | 2.538,07 | 193,66 | 527,30 | 253,75 |
| MongoDB-async 2 | 2.610,97 | 187,43 | 536,55 | 256,45 |
| MongoDB-async 3 | 2.439,02 | 204,18 | 406,82 | 248,54 |
| **MongoDB-async Ortalama** | **2.529,35** | **195,09** | **490,22** | **252,91** |

# ÖZGEÇMİŞ

BERNA DUMANLI

**Doğum Tarihi:** 08.11.1994

## EĞİTİM

**2001-2008:** Cumhuriyet İlkokulu (Babaeski)

**2008-2012:** Babaeski Anadolu Lisesi (Babaeski)

**2012-2016:** Trakya Üniversitesi Mühendislik Fakültesi, Bilgisayar Mühendisliği (Edirne)

**2016-……:** Trakya Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı (Edirne)