DÜZCE ÜNİVERSİTESİ | LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

İLİŞKİSEL VE NOSQL VERİTABANLARININ İŞLETİM SİSTEMİ, ÇALIŞMA ORTAMI VE WEB TEKNOLOJİLERİNE GÖRE PERFORMANSLARININ ANALİZİ

MEHMET ARSLAN

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK VE BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DANIŞMAN
DR. ÖĞR. ÜYESİ FATİH KAYAALP

DÜZCE, 2023

T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

İLİŞKİSEL VE NOSQL VERİTABANLARININ İŞLETİM SİSTEMİ, ÇALIŞMA ORTAMI VE WEB TEKNOLOJİLERİNE GÖRE PERFORMANSLARININ ANALİZİ

Mehmet ARSLAN tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Dr. Öğr. Üyesi Fatih KAYAALP

Düzce Üniversitesi

Jüri Üyeleri

Dr. Öğr. Üyesi Fatih KAYAALP
Düzce Üniversitesi _____

Doç. Dr. Arafat ŞENTÜRK
Düzce Üniversitesi _____

Dr. Öğr. Üyesi Ümit ŞENTÜRK
Bolu Abant İzzet Baysal Üniversitesi _____

Tez Savunma Tarihi: 15/06/2023

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

15 Haziran 2023

(İmza)

Mehmet ARSLAN

TEŞEKKÜR

Yüksek lisans öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı değerli hocam Dr. Fatih KAYAALP'e en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

15 Haziran 2023 **Mehmet ARSLAN**

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

ACID	Atomicity Consistency Isolation Durability
AWS	Amazon Web Service
BASE	Basically Available, Soft State, Eventually Consistent
CAP	Consistency Availability Partition Tolerance
CGI	Common Gateway Interface
CPU	Central Processing Unit
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
DB	Database
Html	Hyper Text Markup Language
IaaS	Infrastructure as a Service
IMS	Information Management System
IOT	Inteernet of Things
Js	Java Script
JSON	Java Script Object Notation
NoSQL	Not Only SQL
PaaS	Platform as a Service
Php	Personal Home Page
REST	Representational State Transfer
SQL	Structured Query Language

SİMGELER

EB Exabyte
GB Gigabyte
MB Megabyte

ÖZET

İLİŞKİSEL VE NOSQL VERİTABANLARININ İŞLETİM SİSTEMİ, ÇALIŞMA ORTAMI VE WEB TEKNOLOJİLERİNE GÖRE PERFORMANSLARININ ANALİZİ

Mehmet ARSLAN
Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Elektrik Elektronik ve Bilgisayar Mühendisliği
Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Dr. Fatih KAYAALP
Haziran 2023, 77 sayfa

İnternet teknolojisinin kullanımının artmasıyla birlikte, web teknolojilerinin kullanımı da artmış bununla birlikte veri miktarı da önemli ölçüde artmıştır. Günümüzde neredeyse her şeyin internete bağlı hale gelmesi, büyük miktarda veri üretimine neden olmaktadır. İnternet kullanıcıları, sosyal medya platformlarına yükledikleri fotoğraflar, paylaştıkları mesajlar, video izleme ve indirme aktiviteleri gibi birçok etkinlikle veri oluştururlar. Artan veri miktarı geleneksel veritabanı sistemlerinin sorgulanmasına ve NoSQL veritabanı sistemlerinin kullanımının artmasına ve de internet tabanlı bilişim hizmetleri olan bulut teknolojilerinin kullanımının artmasına neden olmuştur. Veri tabanları, işletim sistemleri, web teknolojisi ve çalışma ortamı arasında karmaşık bir etkileşim bulunmaktadır. İşletim sistemleri, veri tabanlarının verimli bir şekilde çalışmasını sağlamak için önemli bir rol oynar. Aynı şekilde, web teknolojisi ve çalışma ortamı da veri tabanlarının performansını etkileyebilir. Bu çalışmada ilişkisel veritabanı olan MySQL ile NoSQL veri tabanları olan Cassandra, HBase ve MongoDB'nin performanslarını etkileyen parametreler incelenmeye çalışılmıştır. Yani bu çalışmanın amacı, farklı işletim sistemleri, programlama dilleri, veritabanı sistemleri ve çalışma ortamları arasında performans açısından karşılaştırmalar yaparak sonuçlar elde etmektir. Bununla beraber veri hacmi de önemli bir parametre olarak kullanılmıştır. Çalışmada veri hacmi 10'un katları şeklinde 100 bin veriye kadar arttırılmış olup farklı işletim sistemleri (Ubuntu, Windows) farklı web teknolojileri (Node.Js, Php) ve farklı çalışma ortamlarında (Bulut Sistemi, Kişisel Bilgisayar) testler yapılmıştır. Bu testlerde okuma, ekleme, silme ve güncelleme işlemleri yapılmış ve performans ölçütü olarak sorgu yanıtı gecikme süreleri esas alınmıştır. Yapılan testlerde okuma işlemi haricinde en iyi sonucu Cassandra veritabanı vermiştir. Çalışma ortamı bazında bakıldığında bulut sisteminin genelde daha iyi sonuçlar verdiği; işletim sistemi bazında bakıldığında Ubuntu işletim sisteminin daha iyi sonuçlar verdiği görülmüştür. Fakat işletim sistemlerinin birbirleriyle kıyaslanmasında iki sistemin arasında çok ciddi performans farklarının bulunmadığı gözlemlenmiştir. Web teknolojileri bazında bakıldığında da önemli performans farklarının bulunmadığı gözlemlenmiştir.

Anahtar sözcükler: Bulut, Gecikme süresi, Performans, Veritabanı, Web teknolojisi.

ABSTRACT

PERFORMANCE ANALYSIS OF RELATIONAL AND NOSQL DATABASES ACCORDING TO OPERATING SYSTEM, RUNNING ENVIRONMENT AND WEB TECHNOLOGIES

Mehmet ARSLAN
Düzce University
Graduate School of Education, Department of Electrical, Electronics, and Computer Engineering
Master's Thesis
Supervisor: Dr. Fatih KAYAALP
June 2023, 77 pages

With the increasing use of internet technology, the use of web technologies has also grown, leading to a significant increase in data volume. Nowadays, almost everything is connected to the internet, resulting in the generation of large amounts of data. Internet users create data through various activities such as uploading photos to social media platforms, sharing messages, watching and downloading videos. The growing data volume has led to increased querying of traditional database systems, an increase in the use of NoSQL database systems, and the rise of cloud technologies, which are internet-based computing services. There is a complex interaction between databases, operating systems, web technologies, and working environments. Operating systems play a crucial role in ensuring the efficient functioning of databases. Similarly, web technologies and working environments can also impact database performance. In this study, the parameters affecting the performance of relational database MySQL and non-relational databases Cassandra, HBase, and MongoDB were examined. The aim of this study is to obtain results through performance comparisons among different operating systems, programming languages, database systems, and working environments. Data volume was also considered as an important parameter. In the study, data volume was increased up to 100,000 records in multiples of 10, and tests were conducted with different operating systems (Ubuntu, Windows), web technologies (Node.js, PHP), and working environments (Cloud System, Personal Computer). Operations such as reading, insertion, deletion, and updating were performed in these tests and query response time latency is used as performance metric. Among these operations, Cassandra database yielded the best results except for the read operation. Generally, cloud systems showed better performance results when compared to personal computers in terms of working environment. Regarding operating systems, Ubuntu performed better than Windows. However, there were no significant performance differences observed between the two operating systems. When it comes to web technologies, it was observed that there were no significant performance differences.

Keywords: Cloud, Response Time, Performance, Database, Web technology.

1. GİRİŞ

2022 yılı sonu itibarıyla dünya üzerindeki internet kullanıcısı sayısı 7,9 milyara ulaşmış olup 2021 yılına göre yaklaşık yüzde 50 artış göstermiştir [1]. Kullanıcı sayısının artışını takiben üretilen veri miktarı da bu minvalde artmıştır. Veriler, bilimsel araştırmaların ham maddesini oluşturur ve veriler, doğal ortamlarında oluşan ve ilgili veri merkezinde toplanan bilgilerden oluştuğu gibi, deneysel çalışmalarla da kontrollü koşullarda elde edilebilmektedir. Veriler hacim, çeşitlilik ve oluşma hızı bakımından büyük veri boyutuna ulaşabilir ve bu veriler çok sayıda bağımlı ve bağımsız değişkenden oluşan karmaşık bir yapıda da olabilir [2].

Bunun yanında üretilen veriler sadece internet kaynaklı olmayıp sensörler gibi otomatik veri toplama araçları, birçok veri kaynağından bir tanesidir. Bunlar örneğin hava kalitesi ölçümleri, trafik yoğunluğu, su seviyesi, sıcaklık, nem gibi çeşitli parametreleri ölçerek veri üretirler. Bununla birlikte, veriler doğrudan insanlar tarafından da üretilebilir. Örneğin, bir anket veya bilgi formu doldurtularak da veriler elde edilebilir. Ayrıca bir web sitesi, bir blog, bir forum veya bir sosyal medya platformu gibi internet kaynaklarından da veri üretilebilir. Başka bir veri kaynağı da akıllı telefonlar, tabletler ve diğer cihazlar gibi kişisel elektronik cihazlardır. Bu cihazlar, kullanıcılarının etkileşimlerinden ve faaliyetlerinden veri toplayabilirler. Örneğin, bir kişinin nereye gittiği, ne zaman ve hangi uygulamaları kullandığı gibi veriler kaydedilebilir.

Bir başka veri toplama aracı olan araştırma, deney veya gözlem gibi özel yöntemler kullanılarak veriler elde edilebilir. Bu tür veri toplama yöntemleri örneğin, laboratuvar ortamında yapılan bir deneyde elde edilen veriler veya bir gözlemcinin bir olayın ayrıntılarını kaydettiği bir gözlem gibi olabilir.

Veri, ekonomik farklılaşmalar hakkında bilgi sağlar ve iş kararlarının alınmasında önemli bir rol oynar. Yatırımcılar, veri analizi kullanarak bir şirketin finansal durumunu, performansını ve büyüme potansiyelini ölçebilirler. Ayrıca, para birimleri arasındaki döviz kurları da veri analiziyle takip edilir ve yatırımcılar arasındaki para hareketleri ve ticaretin yönlendirilmesinde etkili olur. Günümüzde bilimsel araştırmaların önemli bir bölümünü bu büyük ve/veya karmaşık verilerden gizli bilgilerin çıkarılması

oluşturmaktadır [2]. Bu verilerin analizi ekonomik olarak da önemli bir gelir kalemi oluşturmaktadır. Günümüzde bu sebepten de ötürü para kazanma araçlarından bir tanesi veri analizidir.

Veri ayrıca tüketici harcamaları ve satın alma kararları hakkında da önemli bilgiler sağlar. Örneğin, tüketici trendleri ve satın alma alışkanlıkları hakkında veri analizi yaparak, şirketler reklam stratejilerini ve ürünleri geliştirmelerini yönlendirebilirler.

Veriye ulaşmak ise bilişim araçlarının yaygınlaşmasıyla beraber oldukça kolaylaşmıştır. İlk bilgisayarlar, yirminci yüzyılın başlarında geliştirilmiştir. Bu bilgisayarlar, daha önce manuel olarak yapılan işlemleri otomatik hale getirmiştir.

1960'larda, IBM ve diğer şirketler, işletmelerdeki büyük veri setlerini işlemek için kullanılabilecek, büyük veri merkezleri inşa etmeye başlamışlardır. Bu merkezler, verilerin depolanması, işlenmesi ve raporlanması için kullanılmıştır. 1970'lerde, veritabanı yönetim sistemleri geliştirilmiştir ve verilerin daha kolay depolanması, aranması ve yönetilmesi mümkün hale gelmiştir [3].

90'larda, internetin popülerleşmesi ve daha ucuz veri depolama maliyetleri, büyük veri kavramının ortaya çıkmasına neden olmuştur. Günümüzde, veri analizi, yapay zekâ ve makine öğrenmesi gibi teknolojilerin geliştirilmesi ile birlikte, büyük veri setlerinin daha verimli bir şekilde işlenmesi mümkün hale gelmiştir.

Günümüzde internet teknolojisi hiç şüphe yok ki altın çağını yaşamaktadır. Bu gelişim başta eğitim, iş ve medya dünyası olmak üzere hayatın her noktasında hissedilmektedir. İnternet teknolojisinin bu kadar gelişmesiyle beraber insanlar günlük faaliyetlerinde bile veri üretir duruma gelmişlerdir. Her gün 2,5 EB (1 EB=1 073 741 824 GB) hacminde verinin üretilmektedir [4]. Veri hacminin artması çeşitli araştırmalara ait bilgi işlem raporlarında da açıkça görülmektedir [5]. Bu hacim gün geçtikçe artış göstermektedir.

Bahsedilen veri miktarlarına ulaşılması analiz edilmesi kadar bu verilerin sağlıklı ve hızlı şekilde depolanması, istenilen durumda ulaşılması da oldukça önem arz etmektedir. Bu bağlamda veritabanları verilerin depolanmasında önemli bir bileşen olarak karşımıza çıkmaktadır.

Veritabanı, büyük miktarda veriyi depolama, erişme ve işleme konusunda yardımcı olan yapılandırılmış bir veri koleksiyonudur [6]. Başlangıçta bilgisayar sistemleri tek işlem yapmaktadır yani bir dizi içinde baştan sona tek bir işlem yapılmaktadır. Daha sonraları makinede birden fazla iş yapılmaya başlanmış ve bu da ona çoklu işlem yapma becerisi

kazandırmıştır. Hâlâ işler birbirinden bağımsızdır ve işlemlerin yapılması donanımsal açıdan beklemeyi gerektirmekte, bu durum sonucunda ise istemci-sunucu mimarisi olarak SQL(Structured Query Language) veri tabanları kullanılmaya başlanılmıştır [7]. Fakat son yıllarda veri hacminin üstel şekilde artması ve veri çeşitliliğinde artış olmasıyla beraber verilerin temin edinmesinden düzenlenmesi ve işlenmesine kadar birçok farklı metot geliştirilmiştir. Bu kapsamda NoSQL(Not Only Structured Query Language) veritabanı yönetim sistemleri birçok orta ve büyük ölçekli firma tarafından kullanılmaya başlanmıştır. NoSQL veritabanları, bugün büyük veri işleme için tercih edilen seçenek haline gelmiştir. Bunun sebebi, yüksek veri erişim hızlarına, esnek şemalara ve dağıtık veritabanlarına destek vermeleri ve kolay bir şekilde ölçeklenebilme yetenekleridir [8]. NoSQL sistemler ilişkisel veritabanlarının bazı sınırlamalarını aşmak için geliştirilen yeni nesil veritabanlarıdır ve büyük ölçekli verilerin bir sunucu kümesinde etkili bir şekilde depolanmasını ve alınmasını sağlayan bir mekanizma sunar [9]. MongoDB, Cassandra ve Firebase gibi birçok NoSQL veritabanı yönetim sistemi küçük, orta ve büyük çaplı projelerde kullanılmaktadır. NoSQL veri tabanlarının diğer avantajlarından bahsedecek olursak özellikle veri erişim sorunlarını ortadan kaldırmak için kopyaları kullanması ve paylaşılmış veya bölümlenmiş veriyi birçok sunucudan alması, NoSQL sistemlerin tutarsız ve süreksiz verilerin barınmasına da izin vermesi, tanımlanmayan bir zamanda tutarlılığın oluşacağı garanti edilmesi en önemli avantajlarındandır [10].

Verilerin aktarımı kadar bu verilerin ilgili kişilere sunulması da önem arz etmektedir. Bu noktada çeşitli web teknolojilerinden yararlanılmaktadır. Bu web teknolojileri içinde en önemlileri JavaScript, Python, Php ve Html/Css'dir [11].

Bütün bu sistemlerin uyumlu çalışabilmesi, ilgili veritabanı yönetim sisteminin nereden yönetileceği konusunu önemli hale getirmektedir. Bu bağlamda kişisel bir bilgisayar ya da sunucudan ziyade bulut bilişim sistemleri sıklıkla tercih edilmektedir. Bulut bilişim hizmetleri, hizmet olarak yazılım (SaaS), hizmet olarak platform (PaaS) ve hizmet olarak altyapı (IaaS) olmak üzere üç temel hizmet modeli ile hizmet sağlayıcılar tarafından sunulmaktadır. Bulut biliş im güvenliği "paylaşılan sorumluluk" anlayışına dayanmakta olup, farklı hizmet modelleri için hizmet alan ve hizmet sağlayıcıların sorumlulukları değişiklik göstermektedir [12]. Google Cloud, Amazon gibi birçok firma bulut bilişim hizmetini ücretli ve ücretsiz versiyonları ile kullanıcılarla buluşturmaktadır.

Yapılan çalışmada ilişkisel bir veritabanı olan MySQL ile NoSQLveritabanlarından Cassandra, HBase ve MongoDB'nin, çeşitli parametreler ışığında performans incelemesi yapılmıştır.

Php ve Node.js gibi farklı web teknolojileri alt yapısında hem Google Cloud hem de yerel bilgisayar sistemleri gibi çeşitli ortamlarda performans analizi gerçekleştirilmiştir. Bu analizlerde, kayıt sayısının giderek arttığı senaryolarda kayıt ekleme, kayıt silme, kayıt güncelleme ve kayıt okuma işlemleri üzerinde incelemeler yapılmıştır. Aşağıdaki maddeler çalışmanın amaçlarına ve içeriğine yönelik bilgiler vermektedir:

a. İşletim sistemlerinin veritabanı performansına etkisi incelenmeye çalışılmış olup deneylerde Windows ve Ubuntu işletim sistemleri kullanılmıştır.

b. Web programlama teknolojilerinin veritabanı performansına etkisi incelenmeye çalışılmış olup deneylerde Node.Js ve Php teknolojileri kullanılmıştır.

c. Çalışma ortamının veritabanı performansına etkileri incelenmeye çalışılmış olup deneylerde yerel bilgisayar ile Google Cloud yapısı kullanılmıştır.

d. Çalışmada üzerinde işlem yapılan kayıt sayısının veritabanı performansını etkisinin incelenmesi için kayıt sayısı 1'den itibaren 10'un katları şeklinde kademeli olarak (1,10,100,1000,10 000,100 000) arttırılarak deneyler yapılmıştır.

e. Çalışmada ilişkisel ve NoSQL veritabanları kullanılmış olup bu sistemler arsındaki performans analizi irdelenmeye çalışılmıştır.

Yapılan çalışmanın kapsamı bakımından bazı sınırlılıkları bulunmaktadır. Bu açıdan bakıldığında veri hacminin 100 000 veri ile sınırlandırılması özellikle NoSQL veritabanı performanslarının bir kısıtlılık yaratmış olabilir. Kullanılan sorguların çeşitliliği bakımdan incelendiğinde ise karmaşık sorguların kullanılmaması, bir sınırlılık olarak nitelendirilebilir. Kullanılan verinin sadece sayısal ve alfa sayısal verilerden oluşması, ses video gibi farklı veri örneklerinin çalışmanın dışında tutulması bir başka sınırlılık olarak görülmektedir. Çalışma esnasında kullanılan web tarayıcılarının da performansa etki edip etmemesi hesaba katılması gerekmektedir. Bu bağlamda farklı web tarayıcılarının kullanımı performans bazında farklı sonuçları ortaya koyabilir.

Bulut tabanlı sistemlerin yerel bilgisayar sistemlerine göre daha iyi sonuçlar verdiği gözlenmiştir. Bununla birlikte, bulut yapısının ticari bir ürün olması ve ücretlendirme politikalarının kullanıcıları etkilemesi önemli bir kısıtlama olarak ortaya çıkmıştır.

2. LİTERATÜR TARAMASI

Bu çalışma, ilişkisel ve NoSQL veritabanlarının, işletim sistemi, çalışma ortamları ve web teknolojileri alt yapılarına göre performanslarının incelenmesini amaçlamaktadır. Veritabanı performansı, modern uygulamaların etkili bir şekilde çalışabilmesi için önemli ve unutulmaması gereken bir faktördür. Bu sebeple, veritabanı seçiminde işletim sistemi, çalışma ortamı ve web teknolojilerinin de performans üzerindeki etkilerinin anlaşılması elzemdir.

Çalışma için yapılan literatür taramasında, yerli ve yabancı elektronik yayınlara ulaşmak için; Institute of Electrical and Electronics Engineers (IEEE), Science Direct, Scopus, DergiPark gibi veritabanlarından, ulusal ve uluslararası tez çalışmalarından ve konferans bildirilerinden ağırlıklı olarak son 6 yıldaki çalışmalardan tarama yapılmıştır. Araştırma kapsamında veritabanları, işletim sistemleri, bulut bilişim ve de web teknolojilerini kapsayan konularda yayınlanmış bilimsel makaleler, konferans bildirileri ve teknik dokümanlar incelenmiştir.

Literatür taraması sonucunda veritabanları performanslarını ve bu performansa etki edecek parametreleri barındıran çalışmalar gözden geçirilmiştir. Buna göre [4]'de NoSQL ile ilgili yaptığı çalışmada ilişkisel ve NoSQL veri tabanlarından MySQL ve MongoDB'nin artan veri hacmine göre temel veritabanı işlemleri olan okuma ekleme silme ve güncelleme işlemleri analiz edilmiş bu analizlerde veri miktarı arttıkça MongoDB veritabanının daha başarılı olduğu görülmüştür. Bununla beraber güvenlik, maliyet baz alındığında ilişkisel veri tabanlarının özellikle bankacılık gibi güvenliğin önem arz ettiği sistemlerde tercih edilemeye devam edileceğini öngörmüşlerdir

[8] tarafından yapılan çalışmada MySQL, Cassandra ve HBase'in yoğun yazma işlemleri için performansı bir web tabanlı REST (Representational State Transfer) uygulaması aracılığıyla değerlendirmektedir. Sonuç olarak Cassandra veritabanının, diğer iki veritabanı arasında en çok ölçeklendirme sağlayan ve çok hızlı yazma hızları sunan bir veritabanı olduğunu belirtilmiştir

[9] tarafından hazırlanan bu çalışmada, seçilen NoSQL veritabanlarının MongoDB, Cassandra ve HBase kullanılarak yapılan karşılaştırma sonuçları açıklanmıştır. Testler,

önceki çalışmalarda oluşturulan IoT (Internet of Things) çerçevelerinde gerçekleştirilmiştir. Elde edilen sonuçlar, MongoDB'de yapılandırılmış veri tiplerinin çalışma süresi performansı ve üretilen verimlilik değerlerinde üstün olduğunu göstermektedir. Cassandra için ise yapılandırılmamış veri tipinin çalışma süresi ve verimlilik metriklerinde daha iyi bir değeri vardır. Ancak kaynak kullanımı açısından MongoDB, yapılandırılmış ve yapılandırılmamış veri tiplerinde en iyi performansa sahip olduğu belirtilmiştir.

[13] 'de yapılan çalışmada ise MongoDB ve MySQL karşılaştırılmış özellikle sorguların karmaşıklaşması ve veri hacminin artmasının NoSQL veritabanı olan MongoDB üzerinde çeşitli avantajlar oluşturduğunun görüldüğü belirtilmiştir.

Çalışma ortamı üzerinden bakıldığında ise [14]'e ait tez çalışmasında bulut bilişim teknolojileri ve NoSQL veritabanları incelenmiş bulut bilişim teknolojilerinin artan veri hacminin de etkisi ile özellikle büyük hacimli çalışmalarda kullanılmasının maliyet yönünden "kullandığın kadar öde" prensibinden de hareketle daha uygun olduğu ve performans olarak da geleneksel sistemlere göre daha iyi sonuçlar verdiği görülmüştür [15]'in hazırladığı çalışmada ise çeşitli yazılım çerçeveleri (framework) CRUD işlemleri bazında karşılaştırılmış ve CodeIgniter, Laravel, Asp.Net gibi diğer yazılım çerçevelerinden daha iyi gecikme sürelerine ulaştığı sonucuna varılmıştır.

[16]'da yapılan çalışmada belirtilen performans testlerine göre, MongoDB hız açısından daha iyi performans göstermektedir ve MySQL'e göre yüzde 19,33'lük bir iyileşme sağlamaktadır. Bununla birlikte, çalışma ayrıca MySQL'in veritabanı erişim kontrolüyle daha güçlü bir güvenlik katmanı gerektirmeyen durumlarda daha iyi bir seçenek olabileceğini belirtmektedir.

[17]'de hazırlanan çalışmada NoSQL veritabanı olan MongoDB ve belge tabanlı MySQL'in, özellikle sorgu işlemlerinde CRUD işlemlerinin karmaşıklığı ve performansı açısından analizi yapılmıştır. Çalışmanın ana amacı, her bir özel veritabanının CRUD isteklerini gerçekleştirirken uygulama performansı üzerindeki etkisini karşılaştırmalı bir analiz yapmaktır. Sonuç olarak, her iki veritabanının da büyük veri uygulamaları için uygun olduğu söylenebilir. Çok büyük veri hacimleri ve çok karmaşık veritabanları ile çalışıldığında, sorguların karmaşıklığı ve veri miktarı ne olursa olsun çok kısa yanıt süreleri gözlemlendiği belirtilmiştir.

[18]'deki tez çalışmasında en çok kullanılan ilişkisel ve NoSQL veritabanı yönetim

sistemlerinin performans karşılaştırmasını yapmıştır. 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 koymuştur. Test edilen her bir veritabanı farklı mimari ve özelliklere sahip olduğu için farklı sonuçlar ortaya çıkardığını belirtmiştir.

[19]'da yapılan çalışmada ise ise NoSQL ve ilişkisel veritabanlarının kavramlarını, sınırlamalarını karşılaştırmakta ve NoSQL veritabanlarının avantajlarını belirtmişlerdir

[20]'de yapılan çalışmada, Google destekli arka uç (backend) hizmeti sunan bir bulut teknolojisi olan Firebase kullanılarak Java programlama dilinde bir mesajlaşma uygulaması geliştirilmiştir. Uygulama ile kişilerin, gönderilen ve alınan mesajların, kişilerin profil fotoğraflarının ve durum bilgilerinin, grup konuşmalarının bir Firebase bulut ortamında saklanması hedeflenmiştir. Çalışma gerçek zamanlı bulut üzerinde çalışan bir NoSQL veritabanı sistemi kullanarak ve veriyi JSON olarak saklayarak gerçekleştirilmiştir. Sonuç olarak Firebase veri tabanının veriyi senkron olarak çekmesi sayesinde verilere anlık olarak ulaşabildiği sonucu belirtilmiştir.

[21] 'de yapılan bulut bilişim hakkında hazırlanan çalışmada bulut depolama hizmeti sağlayıcıları arasında bir karşılaştırma sunmaktadır. İlk olarak, bulut hizmetlerinin tarihçesi kısaca tanıtılmakta ve ardından Amazon WS (Web Service) ve Google Cloud platformlarının tarihçesi üzerinde durulmaktadır. Ayrıca, iki bulut hizmetinin mevcut hizmetler, performans, maliyetler ve ağ bağlantısı açısından özellikleri incelenmekte ve tartışılmaktadır. Son olarak, her iki hizmetin önemli farkları, faydaları ve dezavantajları vurgulanarak yazarlar tarafından değerlendirme yapılmaktadır.

[22]'de yapılan çalışmadaki amaç farklı senaryolarda SQL ve NoSQL veritabanı yönetim sistemlerinin performansını karşılaştırmalı olarak analiz etmektir. Yazarlar, her iki seçeneğin performansını ölçmek için kapsamlı testler gerçekleştirmiş ve her birinin güçlü ve zayıf yönlerine ilişkin bilgiler sunmuştur. Buna göre analizin sonuçları, makalede başarılı işlemlerin sayısı, işlemci kullanımı, bellek kullanımı ve disk okuma/yazma hızı gibi metriklerle birlikte sunulmaktadır. Buna göre ilişkisel veritabanlarında Oracle, NoSQL veritabanlarında ise Redis'in daha iyi performans sonuçlarına ulaştığı belirtilmiştir.

Bu literatür taraması, ilişkisel ve NoSQL veritabanlarının işletim sistemi çalışma ortamı ve web teknolojileri alt yapılarına göre performanslarının incelenmesini amaçlamaktadır. Yapılan literatür taraması sonuçlarına göre ilişkisel veritabanlarının yapısal veriye dayalı uygulamalarda, NoSQL veritabanlarının ise büyük hacimli ve dağıtık veri yönetiminde etkili olduğu görülmüştür. İşletim sistemi seçimi, çalışma ortamının önemli olması, veri hacmi ve kullanılan web teknolojileri alt yapısı da performansı etkileyen önemli faktörlerdir. Literatür araştırmasında incelenen çalışmalar Çizelge 2.1'de özet olarak sunulmuştur.

Yukarıda verilen bilgiler ışığında bakıldığında veritabanları performans analizinin ve veritabanları performansını etkileyecek parametrelerin incelenmesi literatürde sıkça çalışılmış konuların başında gelmektedir. Özellikle web ortamında veri hacminin artışıyla beraber konuya olan ilgi artmıştır. Bu açıdan yapılan çalışmaya da ilham olan bu konu gelecekte de irdelenmeye devam edilecektir. İncelenen çalışmalarda genellikle bir parametre üzerinden incelemeler yapılmış olup veritabanlarının, veri hacmi parametresine bağlı olarak performans incelemesi yapılmıştır. Bu çalışmada ise veritabanı performans analizi yaparken literatürde kısmen az değinilmiş olan işletim sistemi, web teknolojisi, çalışma ortamı parametreleri ve veri hacmini baz aldığından daha geniş kapsamlı bir çalışma ortaya çıkmıştır. Bu açıdan bakıldığında çalışma, ilgili alanda bahsedilen faktörlerini tümünü incelemeye çalışarak daha fazla araştırmaya yol göstermektedir.

Çizelge 2.1. Literatür çalışma özeti.

Çalışma	Yıl	Çalışmada Kullanılan Araçlar	Yapılan İşlem	Sonuç
[4]	2015	MongoDB MySQL	Veri hacm arttırılarak CRUD işlemlerini gerçekleştirildiği belirtilmiştir.	MongoDB'nin bu testlerde MySQL veritabanına göre daha iyi performans gösterdiği belirtilmiştir.

Çizelge 2.1. Devam (Literatür çalışma özeti).

Çalışma	Yıl	Çalışmada Kullanılan Araçlar	Yapılan İşlem	Sonuç
[8]	2016	Cassandra HBase MySQL	MySQL, Cassandra ve HBase'in yoğun yazma işlemleri için web tabanlı bir REST uygulaması üzerinde performansı değerlendirmiştir.	Cassandra, diğer iki veritabanı arasında en çok ölçeklendirme sağlayan ve çok hızlı yazma hızları sunan bir veritabanıdır
[9]	2019	Cassandra HBase MySQL	IOT çerçevesinde kullanılan veritabanları performansları kıyaslandığı belirtilmiştir.	MongoDB yapılandırılmış veride, Cassandra ise yapılandırılmamış verilerde daha iyi sonuçlar verdiği belirtilmiştir.
[13]	2017	MySQL MongoDB Veritabanları	Veri hacmini arttırarak CRUD işlemlerinin gerçekleştiği belirtilmiştir.	MongoDB bu testlerde daha başarılı sonuçlar aldığı belirtilmiştir.
[14]	2017	Azure Cloud MySQL Neo4j	Kayıt okuma işlemleri ile veritabanlarının karşılaştırıldığı belirtilmiştir.	Neo4j bu testlerde daha başarılı sonuçla aldığı belirtilmiştir.
[15]	2019	Asp.Net CodeIgniter Laravel	Yazılım çerçevelerinin CRUD işlemlerine etkisin ölçülmeye çalışıldığı belirtilmiştir.	CodeIgniter ile yapılan bu testlerde daha iyi gecikme performanslarına ulaşıldığı belirtilmiştir.
[16]	2022	MongoDB MySQL	Farklı iş yüklerinde CRUD işlemlerinin gerçekleştiği belirtilmiştir.	MongoDB bu testlerde daha başarılı sonuçlar aldığı belirtilmiştir.
[17]	2022	MongoDB MySQL	Farklı iş yüklerinde ve karmaşık sorgularla CRUD işlemlerini gerçekleştirildiği belirtilmiştir.	Her iki veritabanının da büyük veri uygulamaları için uygun olduğu sonucu belirtilmiştir.

Çizelge 2.1. Devam (Literatür çalışma özeti).

Çalışma	Yıl	Çalışmada Kullanılan Araçlar	Yapılan İşlem	Sonuç
[18]	2019	Casssandra Couchbase MongoDB MySQL Oracle SQL Server Yahoo Cloud	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 hazırlandığı belirtilmiştir.	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 verdiği belirtilmiştir.
[19]	2020	MongoDB MySQL	Veri hacmini arttırarak CRUD işlemlerinin gerçekleştirildiği belirtilmiştir.	Deneylerde iki veri tabanının birbirine karşı üstünlük ve sınırlılıkları elde edilmeye çalışılmıştır.
[20]	2022	Firebase Java	Firebase ve Java kullanarak mesajlaşma uygulaması gerçekleştirildiği belirtilmiştir.	Veri JSON formatında saklanmış ve verilere anlık olarak ulaşıldığı belirtilmiştir.

Çizelge 2.1. Devam (Literatür çalışma özeti).

Çalışma	Yıl	Çalışmada Kullanılan Araçlar	Yapılan İşlem	Sonuç
[21]	2017	Amazon WS Google Cloud	Amazon ve Google Cloud, bulut depolama hizmeti sağlayıcıları arasında karşılaştırma yapıldığı belirtilmiştir.	Birbirlerine farklı konularda avantaj ve dezavantaj sağladığı ve işletmelerin beklentileri bu açıdan önem arz ettiği belirtilmiştir.
[22]	2022	MongoDB Oracle PostgreSQL Redis	Farklı senaryolarda SQL ile NoSQL veritabanı yönetim sistemlerinin performansı karşılaştırmalı olarak analiz edilmiştir.	İlişkisel veritabanı bazında Oracle, İlişkisel olmayan veritabanı bazında ise Redis performans anlamında daha iyi sonuçlar verdiği belirtilmiştir.

3. MATERYAL VE METOT

Bu bölümde çalışmada kullanılan parametreler hakkında bilgi verilmiştir. İlk olarak, veritabanı kavramı, modeli ve yapısı ile çeşitlerine değinilmiştir. Daha sonra, çalışmada kullanılan Node.Js ve Php web programlama teknolojileri anlatılmıştır. Son olarak, çalışma yapısı olarak tercih edilen bulut ortamı hakkında bilgi verilmiş ve Google Cloud sisteminin özellikleri aktarılmıştır.

3.1. VERİTABANI MODELİ

Veritabanından önce veri kavramından bahsetmek gerekmektedir. Veri kavramı bilgi kavramı ile de karıştırılabilmektedir. Veri ham gerçek olarak tanımlanır [23]. Ham ifadesi gerçek olarak tanımlaman kavramın henüz işlenmemiş olduğunu anlatır. Örneğin alışveriş merkezlerinin indirim için tüketicilere sunduğu indirim kartları her kullanıldığında tüketicinin yaptığı alışveriş bilgisi karta eklenir. Bu veri kavramına bir örnektir. Çünkü ortaya çıkan bu veri tek başına hiçbir anlam ifade etmez bir başka ifade ile veri işlenmemiştir. Fakat indirim kartını kullanan bütün tüketicilerin alışveriş bilgileri bir araya gelip bu veriler analiz edildiğinde ve analiz sonucunda anlamlı sonuçlar ortaya çıktığında ise ortaya bilgi kavramı ortaya çıkar. Çünkü bilgi kavramı verilerin işlenmiş halidir [23]. Verilerin birbirleriyle ilişkili olsun veya olmasın belli bir düzen altında tutuldukları yerlere ise veritabanı denir [24]. Veritabanları veritabanı yönetim sistemleri tarafından yönetilir.

Veritabanları farklı türlerde olabilir ve farklı amaçlar için kullanılırlar. Bu bölümde öncelikle veritabanı modellerinden daha sonra ise çalışmanın konusu olan ilişkisel ve NoSQL veri tabanları hakkında daha detaylı bilgiler verilecektir.

3.1.1. Hiyerarşik Veritabanları

Hiyerarşik veritabanı modeli, verileri ağaç yapısıyla temsil eder. Hiyerarşik modelde, bir düğümün yalnızca tek bir üst düğümü olabilir. Bu nedenle, birçok farklı düğümün birçok farklı düğüme bağlanması gereken M: N (Many to Many) ilişkileri doğrudan temsil etmek zordur [23].

Hiyerarşik modelin en tanınmış örneği IBM firmasının ortaya koyduğu IMS (Information Management System) veritabanıdır.

3.1.2. Ağ Veritabanları

Verileri düğümler ve kenarlar kullanarak temsil eder. Her düğüm birden fazla düğümle ilişkilendirilebilir ve veriler karmaşık bağlantılarla ilişkilendirilebilir. Veri erişimi, hiyerarşik ve dosya sistemi modellerine göre daha esnek olan veri tabanı modelidir [23].

3.1.3. İlişkisel Veritabanları

İlişkisel veritabanı yazılım endüstrisine uzun yıllardır yön veren bir veritabanı yönetim sistemi türüdür. Standart bilgisayar dili SQL kullanılarak karakterize edilir ve her bir ilişkisel veritabanı yönetim sistemi, SQL'e dayalı farklı diller kullanır. İlişkisel veritabanları, verileri satır ve sütunlardan oluşan bir tablo kümesinde yapılandırılmış bir şekilde düzenler. Sütunlar, kategorileri veya nitelikleri temsil eder ve satırlar, tanımlanmış her kategori için benzersiz bir veri örneği içerir [22].

İlişkisel veritabanı, ilişkisel modeli takip ettiği için bilgileri yapısal veya ilişkisel bir şekilde depolar. İlişkisel veritabanı, verilerin yavaş bir şekilde arttığı ve belirli bir yapıya uyduğu ortamlarda kullanılır. Ancak günümüzde veri hızla artmakta ve farklı kaynaklardan üretilmektedir [25].

3.1.4. NoSQL Veritabanları

Günümüzdeki uygulamalar giderek daha büyük veri kümelerini depolayıp işlemektedir. Özellikle çok sayıda istemcinin aynı anda işlettiği veya çok büyük veri kümelerinin olduğu durumlarda, ilişkisel veritabanı yönetim sistemlerinin sınırları genellikle aşılmaktadır. Performans, büyük veri uygulamaları için hangi veritabanının kullanılacağına karar verirken önemli bir faktördür [17]. Bu sorun, milyarlarca insanın veritabanına erişim sağladığı internet uygulamalarında ortaya çıkar. Ayrıca, hızla büyüyen büyük veri uygulamalarının veri analitiği için ölçeklenebilirlik ve kullanılabilirlik sağlayan veritabanı mimarilerine ihtiyacı vardır.

Bu gibi durumlarda, RDBMS sistemlerinin yerine daha basit ama daha verimli veri depolama sistemleri yaygın olarak kullanılmaktadır. Bugüne kadar birçok dağıtılmış veri deposu sistemi geliştirilmiş ve genellikle NoSQL olarak adlandırılmaktadır [26].

Bir başka ifade ile 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 [18].

3.1.5. Dağıtılmış Veritabanları

Dağıtılmış veritabanı, farklı lokasyonlarda bulunan iki veya daha fazla dosyadan oluşur. Veritabanı, farklı ağlara yayılan ya da aynı fiziksel konumda yer alan birden fazla bilgisayarda depolanabilir.

3.1.6. Nesne Odaklı Veritabanları

Nesne tabanlı bir veritabanı, verilerin nesnelerle temsil edildiği bir modele abone olan bir veritabanıdır [19]. Nesne odaklı bir veritabanındaki bilgiler, tıpkı nesne odaklı programlamada olduğu gibi nesneler biçiminde temsil edilir.

3.1.7. Graf Veritabanları

Sütun ve satır kullanım yerine daha esnek olan graf modelini temel alan bu veritabanları aynı zamanda birden çok bilgisayara ölçeklenebilmektedir. En az anlaşılan veritabanı türlerinden biri, graf veritabanıdır. Yüksek düzeyde bağlantılı verilerle çalışmak için tasarlanmış olan bir grafik veritabanı, bir ilişkisel veritabanından daha çok ilişkisel olarak tanımlanabilir [19].

3.2. İLİŞKİSEL VERİTABANI

Günümüzde en yaygın kullanılan veritabanı sistemlerinden biridir. Satır ve sütunların meydana getirdiği tablolardan oluşur. Bu tablolar birbiri ile ilişkileri olan tablolardır. Dolayısıyla bir veritabanında ilişkiden söz edebilmek için en az iki tablonun yer alması ve bu iki tablodaki verilerin birbiri ile herhangi bir şekilde ilişkilendiriliyor olması gerekir. Bu şekilde ilişkisel veri tabanları, veritabanı denilen büyük dosyalardan oluşur. Her bir tablo, belli yapıya uygun verileri saklamak üzere tasarlanır [27].

İlişkisel veritabanlarında, veritabanının yapısı veya şeması veritabanı kullanılmadan önce düzgün bir şekilde oluşturulmalıdır. Bu, verilerin organizasyonu ve bütünlüğünü yöneten tabloları, sütunları, veri tiplerini, ilişkileri ve kısıtlamaları belirlemeyi içerir. Şema bir kez tanımlandığında, yapıda yapılan herhangi bir değişiklik mevcut şemanın değiştirilmesini ve verinin taşınmasını gerektirebilir, bu da karmaşık ve maliyetli bir süreç olabilir [28].

İlişkisel veritabanlarında veri bütünlüğünün sorunsuz bir şekilde garanti altına alınması için bazı kurallar oluşturulmuştur. Bu kurallar, ACID olarak adlandırılan kuralları temsil eder. ACID kelimesini oluşturan her harf bir kurala karşılık gelmektedir. Buna göre kısaltmayı oluşturan harfler sırasıyla atomiklik (Atomicity), tutarlılık (Consistency), izolasyon, (Isolation) ve dayanıklılık (Durability) olarak bilinmektedir. İlişkisel veritabanı yönetim sistemleri veri bütünlüğü ve işlem tabanlı işlemlerin çalışması için ACID kurallarını garantilerler [29].

ACID prensibini oluşturan harflerden bahsedilecek olursa atomiklik, bütünlük anlamına gelir ve bir işlemi parça parça değil de bir bütün olarak ele alır. İşlemin gerçekleşmesi esnasında karşılaşılan herhangi bir problem işlemi sekteye uğrattıysa tüm işlem geçersiz olacaktır. Bu işlemlerin güvenle yapılması anlamında önem arz etmektedir.

Tutarlılık, işlemlerin tutarlı olması anlamına gelir. Veritabanında gerçekleşen tetikleyici (trigger) ve benzeri işlemlerde kural ihlali gerçekleşir ise bütün işlemin iptal edilmesi ve veritabanının eski durumunda dönmesi durumu olarak bilinmektedir.

İzolasyon temel mantığında veri üzerinde işlem gerçekleşirken aynı anda iki işlemin gerçekleşememesi durumudur. Bu kural yapılan işlemlerin birbirinden etkilenmesi engellemektedir.

Dayanıklılık ise işlem sırasında herhangi bir problem ile karşılaşıldığı durumda problem ortaya çıktığı zaman sistem verileri eski durumuna getirir. Örneğin basit bir banka işlemi yaparak para transferi sağlanmaya çalışıldığında işlem sırasında bir hesaptaki para azalırken diğer hesaptaki parada artış görünmelidir. İşlem tamamlanana dek hesap sahipleri artış ve azalışı göremez. Gerçekleşen bir problem para transferine engel olacaktır.

3.2.1. MySQL

MySQL açık kaynak olan ilişkisel olan veritabanı yönetim sistemidir. Adı, şirketin kurucu ortakları Michael Widenius'un kızı olan "My" ile "Structured Query Language" kelimesinin baş harflerinden oluşmuştur [30].

MySQL, yaygın olarak kullanılan popüler bir açık kaynaklı veritabanıdır. Genellikle Apache ve Php ile birlikte kullanılır, ancak diğer birçok programlama dili ve web sunucusu ile de kullanılabilir [19].

İlişkisel bir veritabanı verileri birbiriyle ilişkili olabilecek bir veya daha fazla veri

tablosunda düzenler, bu ilişkiler verilerin yapılandırılmasına yardımcı olur. SQL, programcıların ilişkisel veritabanından veri oluşturmak, değiştirmek ve ayıklamak ve ayrıca veritabanına kullanıcı erişimini kontrol etmek için kullandıkları bir dildir.

MySQL veritabanı yönetim sistemi, güvenlik, güvenilirlik, performans ve de kullanılabilirlik konularında avantajları ve sınırlamaları bulunmaktadır. Piyasada bulut sunucuları tarafından desteklenen birçok veritabanı yönetim sistemi mevcuttur ve bunlardan bazıları özel kuruluşlar için belirli amaçlarla kullanılabilmektedir [31].

3.3. NOSQL VERİTABANLARI

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 ve dağıtık veri tabanları sistemlerinin görüşüleceği bir toplantı düzenlenmek istendiği dönemlerde NoSQL terimi tekrar kullanılmaya başlanmıştır [32].

İlişkisel veri tabanları ACID protokolünü kullanarak verinin bütünlüğünü korumayı baz alır. NoSQL sistemler ise bu protokole harfiyen uymaz. Bunun yerine temeli CAP (Consistency Availability, Partition Tolerance) teoremine dayanan BASE (Basically Available, Soft state, Eventually consistent) protokollerini kullanır [29]. Bunların temelinde olansa kullanılan veri tabanlarının nasıl ölçeklendirildiğidir. Bu kapsamda bakıldığında ilişkisel veri tabanları genellikle dikey olarak ölçeklendirilir. Dikey ölçekleme sistemdeki bir bileşene ekstra donanım ekleyerek hesaplama kapasitesinin artırılmasıdır. Bu işlem daha fazla bellek, daha hızlı CPU veya daha büyük sabit disk ekleme yolu ile yapılır [29]. Örneğin e-ticaret yapan bir işletme veritabanının sorgu kapasitesini arttırmak adına sistem kaynaklarını yükseltebilir.

Yatay ölçekleme şey ise bir sistemin hesaplama kapasitesinin yeni bileşenler devreye sokularak artırılmasıdır. Yatay ölçeklemede her bir bileşenin birbiri ile haberleşmesi sonucunda meydana gelen network aşırı yükü, sistemin toplam hesaplama kapasitesini azaltmaktadır. Ama bu yük ölçekleme yapıldıktan sonra kazanılan performans artışı yanında göz ardı edilebilir [29].

Belirtilen kapsamda bakıldığında ACID protokolünü kullanan ilişkisel veri tabanları verinin zarar görmesini engellemek adına bütün işleyişi durdurma prensibine sahipken dağıtık veri tabanları olan NoSQL sistemlerinde ise BASE protokolü kullanıldığından

sistemin işleyişinde farklı sunucular kullandığı için sorunlu işlemin bulunduğu sunucu hariç çalışma devam eder. BASE protokolü daha önceden de söylenildiği gibi CAP teoremine (Consistency, Availability, Partition Tolerance) dayanır.

CAP teoremi, dağıtılmış bir veritabanında bir ağ arızası durumunda, tutarlılık veya kullanılabilirlik sağlamanın mümkün olduğunu, ancak her ikisini birden sağlamanın mümkün olmadığını ortaya koyan teoridir.

CAP teoremi üç adet özelliği bünyesinde barındırır. Bunlardan tutarlılık (Consistency), veritabanında gerçekleşen bir işlemin, tanımlanan kurallara uygun olarak veritabanının tutarlı bir durumda kalmasını sağlar. Bu, veritabanındaki verilerin her zaman güncel, geçerli ve tutarlı olması gerektiği anlamına gelir [33]. Kullanılabilirlik (Availability) veritabanına erişim, herhangi bir zaman aralığında sağlanabilir olmalıdır. Bu, kullanıcıların veritabanına istedikleri zaman erişebilmeleri ve veri işlemlerini gerçekleştirebilmeleri gerektiği anlamına gelir. Başka bir ifade ile kullanılabilirlik, sistem arızaları veya kesintileri sırasında bile erişilebilirlik sağlama yeteneğini içerir [33]. Bölüm Toleransı (Partition Tolerance) ise ağdaki iletişim kesintileri veya bölünmeler gibi parçalanmalara (Partitions) dayanıklı olmalıdır. Bölüm toleransı, ağdaki düğümlerin arasında iletişim sorunları olduğunda dahi veritabanının işlevselliğini korumasını sağlamamaktadır [33].

Normal işlemlerde, veri deponuz üç işlevi de sağlar. Ancak CAP teoremi, dağıtılmış bir veritabanı bir ağ hatasıyla karşılaştığında, tutarlılık veya kullanılabilirlik sağlanabileceğini ileri sürer [34].

NoSQL veri tabanları kendi içerisinde dört farklı türdedir.

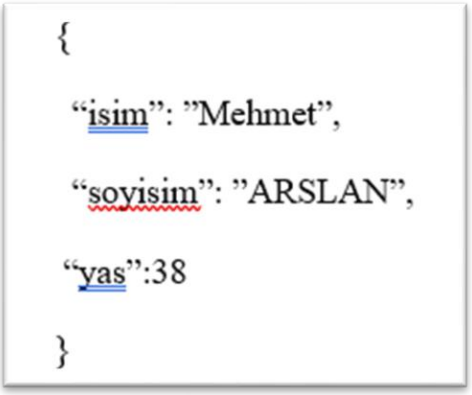
- Doküman Tabanlı (Document store)
- Anahtar Değer Tabanlı (Key Value)
- Kolon Tabanlı (Column oriented)
- Graf Tabanlı (Graph) sistemleridir.

3.3.1. Doküman Tabanlı

Dökuman tabanlı (Document store) veri modelinde, belgeleri tek bir belge altına taşıyabiliriz ve bunun dışında, sorguları daha hızlı çalıştırmak için herhangi bir belirli öğe dizine eklenebilir. Genellikle belgeler, birçok uygulamada kullanılan veri nesnelerine yakın olacak şekilde saklanır ve alınır, bu da verilerin uygulamalarda kullanılması için

çok daha az çevirinin gerekli olduğu anlamına gelir. JSON, genellikle verileri depolamak ve sorgulamak için kullanılan bir veri formatıdır [35].

Örnek bir JSON verisi Şekil 3.1 de verilmiştir.



Şekil 3.1. JSON veri tipi.

Doküman tabanlı sisteme örnek olarak Amazon DocumentDB, CosmosDB, ArangoDB ve bu çalışmada da kullanılan MongoDB örnek verilebilir.

MongoDB, kaynakta bulunan, platformlar arası, belge odaklı bir veritabanı programı olup. NoSQL veritabanı programı olarak sınıflandırılan MongoDB, isteğe bağlı şemalarla JSON benzeri belgeler kullanır. MongoDB, MongoDB Incorporated tarafından geliştirilmiştir ve çeşitli dağıtımlar tarafından ücretsiz olmadığı kabul edilen Sunucu Tarafı Kamu Lisansı (Genel kamu lisansı olarak da bilinir) kapsamında lisanslanmıştır [36].

3.3.2. Anahtar-Değer Tabanlı

Anahtar-Değer (Key -Value) veri modeli ya da veritabanı, Anahtar-Değer deposu olarak da adlandırılır. İlişkisel olmayan bir veritabanı türüdür. Burada, bir ilişkisel dizi, tek bir anahtarın bir koleksiyondaki yalnızca bir değerle bağlantılı olduğu temel bir veritabanı olarak kullanılır. Değerler için anahtarlar özel tanımlayıcılardır. Her türlü varlığa değer verilebilir. Ayrı kayıtlarda saklanan Anahtar-Değer çiftlerinin koleksiyonuna Anahtar-Değer veritabanları denir ve önceden tanımlanmış bir yapıya sahip değildirler [37]. Anahtar-Değer veri modeline örnek olarak AeroSpike, CouchBase ve bu çalışmada kullanılan Cassandra örnek olarak verilebilir. Anahtar-Değer mantığına örnek olarak Şekil 3.2'de belirtilen görsel incelenebilir.

Cassandra, Java ile geliştirilmiş, açık kaynak kodlu, dağıtık ve anahtar değer çifti olarak

verileri tutan bir NoSQL veritabanıdır. Facebook tarafından geliştirilmiştir. 2009 yılında Apache'ye devredilmiştir [38]. Şekil 3.2'de örnek Anahtar değer ikilisi gösterilmiştir.

Şekil 3.2. Anahtar değer ikilisi.

3.3.3. Kolon Tabanlı

Kolon tabanlı (Column oriented) veritabanları, verileri sütunlara göre depolar ve sütunların altında veri blokları oluşturur. Bu yapı, belirli bir sütundaki değerlere hızlı erişim sağlar ve sorgu performansını artırır. Sütun tabanlı veritabanları, analitik ve raporlama uygulamaları gibi okuma odaklı iş yükleri için özellikle etkilidir. Ayrıca, büyük veri kütlelerini yönetme, sık sık değişen veri şemalarıyla çalışma ve yatay ölçeklenebilirlik gerektiren uygulamalar için de uygundur [16].

Google BigTable, ScyllaDB ve bu çalışmada kullanılan Apache HBase bu sisteme örnektir. HBase, Google'ın Bigtable'ından sonra modellenen ve Java ile yazılmış, açık kaynaklı, NoSQL dağıtılmış bir veritabanıdır. Büyük hacimli verilere rastgele, gerçek zamanlı okuma/yazma erişimine ihtiyaç duyulduğunda kullanılır. Bu projenin amacı, donanım kümelerinin üzerinde çok büyük tablolar barındırmaktır. Apache HBase, Google Dosya Sistemi tarafından sağlanan dağıtılmış veri depolama alanından yararlanması gibi, çeşitli benzeri yetenekler sağlar [39]. Şekil 3.3'te kolon tabanlı veritabanı için örnek veriler gösterilmiştir. Kolon tabanlı sistemde veriler satırlarda değil sütunlarda toplanmıştır. Şekil 3.3 incelendiğinde aradaki fark görülmektedir.

ID	SOYAD	AD	NUMARA
51301	A	D	100
51302	B	E	101
51303	C	F	102

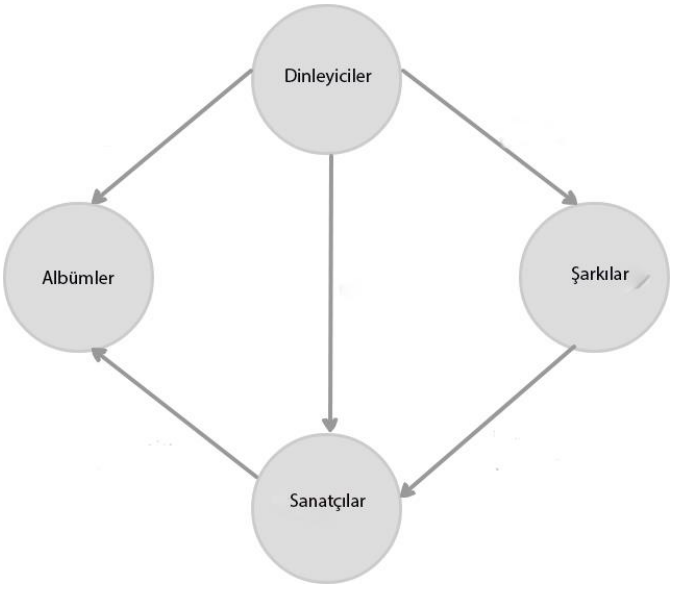
↓

51301	51302	51303
A	B	C
D	E	F
100	101	102

Şekil 3.3. Kolon tabanlı veritabanı örnek veri.

3.3.4. Graf Tabanlı

Graf (Graph) veritabanları, verileri graf teorisi tabanlı bir yapıda depolayan ve yöneten veritabanı sistemleridir. Graf veritabanları, düğümler ve kenarlar arasındaki ilişkileri kullanarak verileri temsil eder. Her düğüm, veri noktasını veya varlığı temsil ederken, kenarlar ise düğümler arasındaki ilişkileri gösterir. Graf veritabanları, büyük veri ağları içerisinde karmaşık ilişkileri yakalamak amacıyla öne çıkar [19]. Graf yapısına örnek olarak Şekil 3.4 incelendiğinde Bir kullanıcının favori şarkıları, sanatçıları ve albümleri ile ilgili bilgilerin depolandığı bir senaryo ortaya konulmaktadır. Kullanıcılar müzik platformunda farklı türlerde şarkılar dinler ve sevdikleri şarkıları favori olarak işaretler. Aynı şekilde, kullanıcılar farklı sanatçıları ve albümleri de favori olarak işaretleyebilir. Şarkılar, sanatçılar ve albümler varlıklar veya düğümler olarak temsil edilebilir.



Şekil 3.4. Graf yapısı örneği.

3.4. WEB TEKNOLOJİLERİ

Bu bölümde bu çalışmada kullanılmış web programlama teknolojileri olan Php ve Node.Js teknolojileri hakkında bilgiler verilecektir.

3.4.1. Node.Js

Node.Js "Ryan DAHL" tarafından 2009 yılında ortaya çıkartılmıştır. Eş zamansız, olaya dayalı bir JavaScript çalışma zamanı olan Node.js, ölçeklenebilir ağ uygulamaları oluşturmak için tasarlanmıştır [40].

Asenkron etkinlik odaklı bir JavaScript çalışma zamanı olan Node.Js, ölçeklenebilir ağ uygulamaları oluşturmak için tasarlanmıştır. Node.Js sisteminde birçok bağlantı eş zamanlı olarak işlenebilir. Her bağlantıda geri çağrı işlemi tetiklenir, ancak yapılacak iş yoksa Node.Js bekleme konumuna geçer.

Node.Js, Ruby'nin Event Machine ve Python'ın Twisted gibi sistemlerine benzer şekilde tasarlanmış ve bu sistemlerden etkilenmiştir. Node.Js olay modelini bir kütüphane yerine bir çalışma zamanı yapısı olarak sunar. Diğer sistemlerde, olay döngüsünü başlatmak için her zaman bloke eden bir çağrı vardır. Tipik olarak davranış, bir betiğin başlangıcında geri çağırmalar aracılığıyla tanımlanır ve sonunda bir sunucu, bloke eden çağrı kullanılarak (EventMachine::run()) başlatılır. Node.Js'de böyle bir olay-döngüsü başlatma çağrısı yoktur. Node.Js, girdi betiği yürütüldükten sonra sadece olay döngüsüne girer. Node.Js, daha fazla geri çağırma işlemi kalmadığında olay döngüsünden çıkar. Bu davranış, tarayıcı JavaScript'inde olduğu gibi olay döngüsü kullanıcıdan gizlenir [40].

3.4.2. Php

Php, (Personal Home Page) HTML içine gömülebilir açık kaynak kodlu, genel amaçlı, özellikle site geliştirmeye uygun bir betik dilidir. Dil yapısının önemli bir kısmını C, Java ve Perl gibi dillerden almış, kendisine has özelliklerle bu yapıyı pekiştirmiş, kolay öğrenilen bir dildir. Dilin ana amacı, site geliştiricilerinin devingen sayfalar oluşturmasını çabuklaştırmaksa da PHP ile çok daha fazlası yapılabilir[41].

Php'yi Javascript gibi kullanıcı tarafında çalışan dillerden ayıran, sunucu tarafında çalıştırılıyor olmasıdır. Php kodu kullanıcının kendi sunucusunda çalıştırılır ise, siteye bağlanan kullanıcılar kodu göremez. Sadece sonucu görebileceklerdir [41]

3.5. BULUT BİLİŞİM TEKNOLOJİLERİ

Çalışmanın başında da belirtildiği gibi veri üretimindeki artış bilgi teknolojilerinin de gelişmesiyle birlikte üstel olarak artmaktadır. Bu artışla beraber verilerin saklanması işlenmesi gerektiği yerde çağrılması ve benzeri işlemlerin yapılması güçlü alt yapıları zorunlu hale getirmiştir. Bu altyapıları oluşturmak da ciddi bir işgücü ve maliyet oluşturmaktadır.[42].

Bulut teknolojisi sistemleri yapılması istenen hesaplama, depolama, geliştirilen uygulamaların çalıştırılması, mevcut sistem güvenliğinin sağlanması ile bakım ve onarım faaliyetlerini gerçekleştirmektedir. Bu hizmetlerin kullanılması ise bulutun hangi özelliğinin kullanılacağına bağlı olarak değişmektedir. Kullandığın kadar ödeme, ölçeklenebilirlik ve kullanım kolaylığı sunduğu avantajlardan bazılarıdır [20].

Bulut bilgi işlem hizmet modelleri, isteğe bağlı bilgi işlem kaynaklarının, yazılımların ve bilgilerin internet üzerinden paylaşılması kavramına dayanmaktadır. Şirketler veya bireyler, hizmet sağlayıcıların sahip olduğu ve yönettiği uzak sunucularda yer alan bilgi işlem, depolama ve ağ hizmetleri dahil olmak üzere paylaşılan kaynakların sanal havuzuna erişmek için ödeme yapar.

Bulut teknolojilerinin sağladığı üç çeşit bulut bilişim hizmeti vardır. Platform hizmeti ile geliştiricilere kendi yazılımlarını, web uygulamalarını veya diğer programlama projelerini oluşturmaları için kullanıcılara kullanımı kolay bir platform sağlayan bir bulut bilişim çözümüdür. Yazılım hizmeti ile kullanıcıların yazılım uygulamalarına ve çeşitli bileşenlerine cihazlarında veya sabit disklerinde indirme, yükleme veya depolamaya gerek kalmadan erişebildiği bir bulut bilişim biçimidir. E-posta, takvim Microsoft Office 365 gibi uygulamalar örnek olarak verilebilir. Altyapı hizmeti ile kullanıcıların sunuculara, güvenlik duvarlarına, sanal makinelere, depolamaya ve diğer altyapılara erişimini sağlar. Kullanıcı geleneksel yöntemlerden farklı olarak firmaların sanallaştırılmış altyapısından hizmetleri satın alır [20].

Günümüzde bu sistemlere örnek olarak Amazon firmasının ortaya koyduğu “Amazon Web Service” ve bu çalışmada da yararlanılan Google firmasının ortaya çıkardığı “Google Cloud” verilebilir. İleriki zamanlarda farklı kurum ve kuruluşlar tarafından da bu hizmetler verilecektir.

3.5.1. Google Cloud

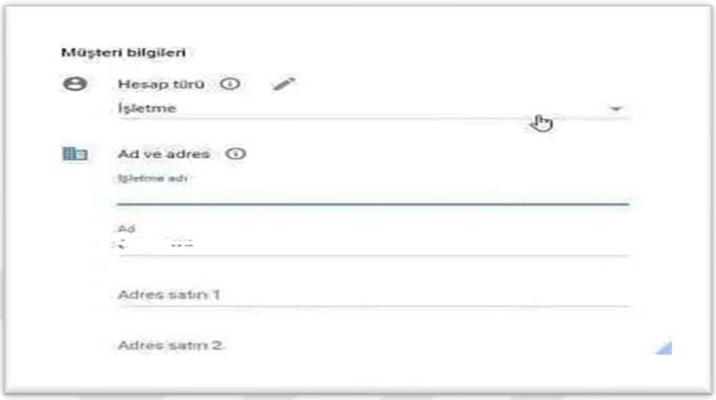
Google Cloud platformu, Google tarafından sağlanan bir bulut hizmetidir. Birçok özelliği sunmaktadır. Bu özelliklerden biri yönetilen sanal makinelerin kullanımıdır. Yönetilen sanal makineler kullanıcılara hem altyapı olarak hizmet hem de platform olarak hizmet kullanma imkânı sağlar çünkü Google'a göre bu ikisi arasında bir seçim yapılmamalıdır. Bu nedenle, kullanıcılara her ikisi de sunulmalıdır. Ayrıca, yönetilen bir VM kullanarak diğer sanal makinelerin doğru gereksinimlere sahip olduğu ve herhangi bir değişiklik veya güncelleme yapılabileceği şekilde kontrol altında tutulabilir. Sanal makineler kullanılarak kullanıcılar tercihlerine göre özel sanal makineler oluşturabilirler [21].

Google Cloud hizmetinden yararlanabilmek için bir Google hesabına sahip olmak gerekmektedir. Bu hesapla Google Cloud hizmetinden yararlanılabilir. Firma ilk olarak üç yüz dolarlık bir bakiyeyi kullanıcılara ücretsiz olarak vermektedir. Bu bakiye, hizmeti kullanırken seçilen donanımlardan ve kullanılan ek hizmetlerle beraber azalacaktır. Bakiye tamamen bittiğinde ise çalışmaların devam etmesi için ücret ödenmesi gerekmektedir. İlerleyen zamanlarda bulut hizmetlerinin yaygınlaşmasıyla ve farklı şirket bünyelerinde hizmetin verilmesiyle beraber bu ücretlerin fiyatlandırması makul düzeylere çekilecektir.[42].

Google Cloud hizmetinden faydalanmak için Google Cloud resmi sayfasından Şekil 3.5’de görüldüğü gibi oturum açarak işlemlere başlanır.

Şekil 3.5. Açılış ekranı.

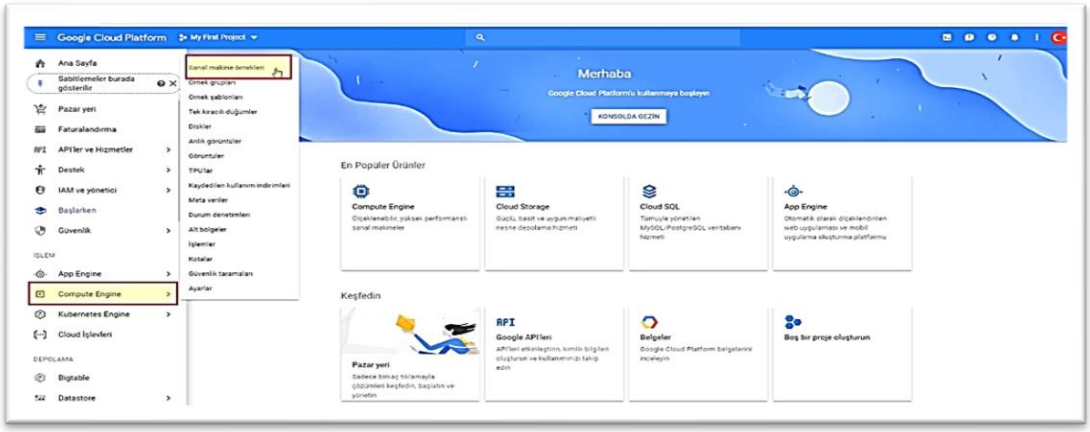
Google hesabı açıldıktan sonra Şekil 3.6'da görüldüğü gibi hesap türü ve adresini girilmesi gerekmektedir. Daha sonra ise ödeme işlemleri için Şekil 3.7'de görüldüğü gibi kart bilgilerinin girilmesi gerekmektedir. Bazı durumlarda Google kartın kişiye ait olup olmadığını doğrulamak için onay isteyebilir. Böyle bir istek sayfasına yönlendirilme söz konusu olduğunda bir fotoğraf göndermek gerekebilir. İşlemler tamamlandığında ise Cloud hesabı oluşturulmuş olacaktır.



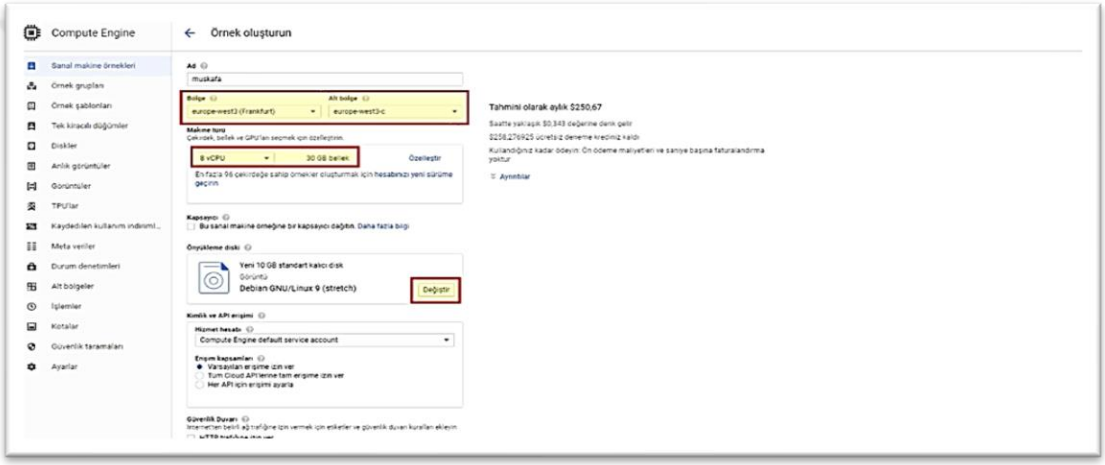
Şekil 3.6. Hesap türü ve adresi.

Şekil 3.7. Ödeme ekranı.

İşlemler sonlandırıldıktan sonra ise işlemlerin devam etmesi için bir sanal bilgisayar kurulması gerekmektedir. Şekil 3.8'de de görüldüğü üzere ilgili seçenekler seçilerek sanal makina kurulur. Sanal makine kurulurken ilgili ayarlar Şekil 3.9'da görülen pencerede yapılır. Bu pencerede istenile donanım özellikleri seçilir ve işlem tamamlanır.



Şekil 3.8. Bulut ortamında sanal bilgisayar oluşturma ortamı.



Şekil 3.9. Bulut ortamında kullanılan donanımın belirlenmesi.

Bu çalışmada farklı türde veri tabanları (MySQL, MongoDB, Cassandra, HBase) okuma, yazma, silme ve güncelleme işlemleri yapılmış olup seçilen farklı çalışma ortamları (Google Cloud ve Yerel bilgisayar) ve farklı işletim sistemlerini (Ubuntu ve Windows) ve de farklı web teknolojilerini (Php ve Node.Js) baz alarak işlem gecikme süreleri kriterine göre karşılaştırmalar yapılmıştır. Bu çalışma ile veri tabanlarının performansının farklı değişkenler bazında nasıl tepki verdiği ölçülmüştür.

4. DENEYLER

Deneyler Çizelge 4.1'de belirtilen özelliklere sahip bilgisayarda gerçekleştirilmiştir. Kullanılan web teknolojileri ve veritabanı sistemleri Çizelge 4.2 'de, Google Cloud için kullanılan özellikler Çizelge 4.3'te ve çalışmada kullanılan örnek veri şablonu Çizelge 4.4'de belirtilmiştir.

Çizelge 4.1. Deneyin yapıldığı yerel bilgisayarın özellikleri.

İşlemci	Xeon E5620
Çekirdek sayısı	4
İş Parçacığı Sayısı	8
Maks Turbo Frekansı	2,66 GHz
İşlemci Temel Frekansı	2,40 GHz
Önbellek	12 MB Intel® Smart Cache
Ram	24GB
İşletim Sistemi	Ubuntu 22.04.1 LTS

Çizelge 4.2. Kullanılan web ve veritabanı teknolojileri bilgileri.

Cassandra	3.11.14
MySQL	8.0
HBase	2.5.2
MongoDB	6.0
Node.Js	18.10.0
Php	8.0.25

Çizelge 4.3. Google cloud donanım özellikleri.

Çekirdek sayısı	4 vCpu
İş Parçacığı Sayısı	8.0.26
Saklama Alanı	24 Gb.
İşlemci Temel Frekansı	2,40 GHz
Ram	24GB
Bağlantı	Public IP
Bölge	us-central1(Iowa)

Çizelge 4.4. Kullanılan kayıt örneği.

Id	52
Name	"Ashkāsham"
State_Id	3901
State_Code	“BDS”
State_Name	"Badakhshan"
Country_Id	1
Country_Code	“AF”
Contry_Name	“Afghanistan”
Latitude	“36.68333000”
Longitude	“71.53333000”

WikiDataId	“Q4805192”

4.1. DENEYLERDE KULLANILAN SORGULAR

Yapılan çalışmada farklı veritabanı sistemlerinde yapılacak işlemleri için farklı veri tabanları ve farklı web teknolojilerinde kullanılmak üzere sorgular oluşturulmuştur. Daha önce de belirtildiği gibi bu sorgular temel veritabanı sorguları olan okuma, ekleme, silme ve güncelleştirme sorgularıdır. Kullanılan sorgular şablon haliyle aşağıda belirtilmiştir.

Sorgular çalıştırılırken zaman gecikmesini bulmak için Node.Js özelinde “SetInterval” metodundan yararlanılmış olup; Php özelinde ise “MicroTime” metodundan yararlanılmıştır. Sorgularda LIMIT komutundan faydalanılarak veri hacmi yükseltilmiştir.

4.1.1. Kayıt Okuma Sorguları

Her veritabanı için oluşturulan kayıt okuma sorguları Node.Js ve Php özelinde şablon olarak belirtilmiştir.

4.1.1.1. *MySQL*

Kayıt okuma işlemleri sırasında MySQL için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.1.1.1. Node.Js

```
connection.query('SELECT * FROM mytable ', (err, results, fields) => {
  if (err) {
    console.error('Veri listeleme işlemi başarısız oldu: ' + err.stack);
    return;
  }
```

4.1.1.1.2. Php

```
$query = "SELECT * FROM mytable ";
```

4.1.1.2. *MongoDB*

Kayıt okuma işlemleri sırasında MongoDB için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.1.2.1. Node.Js

```
db.collection('mycollection').find().toArray((err, result) => {
   if (err) throw err;
   console.log(result);
 });
```

4.1.1.2.2. Php

```
$result = $collection->find()->
foreach ($result as $doc) {
   print_r($doc);
}
```

4.1.1.3. *HBase*

Kayıt okuma işlemleri sırasında HBase için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.1.3.1. Node.Js

```
client.table('mytable').scan({
  maxVersions: 1,
}, (err, rows) => {
  if (err) throw err;
  rows.forEach(row => console.log(row));
```

4.1.1.3.2. Php

```
$rows = $client->scannerOpen('mytable', '', array());
$result = array();
while ($row = $client->scannerGet($rows)) {
```

```
    array_push($result, $row);
}
$client->scannerClose($rows);
```

4.1.1.4. *Cassandra*

Kayıt okuma işlemleri sırasında Cassandra için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.1.4.1. Node.Js

```
const query = 'SELECT * FROM mytable;
```

4.1.1.4.2. Php

```
const query = 'SELECT * FROM mytable;
```

4.1.2. Kayıt Ekleme Sorguları

Her veritabanı için oluşturulan kayıt ekleme sorguları Node.Js ve Php özelinde şablon olarak belirtilmiştir.

4.1.2.1. *MySQL*

Kayıt ekleme işlemleri sırasında MySQL için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.2.1.1. Node.Js

```
const query = 'INSERT INTO mytable (id,
state_id,state_code,state_name,country_id,country_code,country_name,latitude,longitud
e,wikiDataId) VALUES (?, ?, ?....)';
```

4.1.2.1.2. Php

```
const query = 'INSERT INTO mytable (id,
state_id,state_code,state_name,country_id,country_code,country_name,latitude,longitud
e,wikiDataId) VALUES (?, ?, ?)';
```

4.1.2.2. *MongoDB*

Kayıt ekleme işlemleri sırasında MongoDB için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.2.2.1.Node.Js

```
const myData = { id: value, state_id= value,state_code=value,state_name=
value,country_id= value,country_code= value,country_name= value, latitude=
value,longitude= value,wikiDataId= value };

 collection.insertOne(myData, function(err, res) {

   if (err) throw err;

   console.log(‘inserted');

   db.close();

 });
```

4.1.2.2.2.Php

```
$bulk->insert([id=> value, state_id=> value, state_code=>value, state_name=> value,
country_id=>value, country_code=> value, country_name=> value, latitude=> value,
longitude=>value, wikiDataId=> value]);
```

4.1.2.3. *Hbase*

Kayıt ekleme işlemleri sırasında HBase için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.2.3.1.Node.Js

```
const tableName = 'mytable';

const rowKey = 'myrow';

const data = { column1: 'value1', column2: 'value2'……. };

client.putRow(tableName, rowKey, data, function(err) {

 if (err) throw err;

 console.log ('Data inserted successfully');

});
```

4.1.2.3.2.Php

```
$tableName = "mytable";

$rowKey = "myrow";
```

```
$data = [
  "column1" => "value1",
  "column2" => "value2"];
$client = new HbaseClient(["host" => "localhost", "port" => 9090]);
$client->put ($tableName, $rowKey, $data);
```

4.1.2.4. *Cassandra*

Kayıt ekleme işlemleri sırasında Cassandra için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.2.4.1. Node.Js

```
const query = 'INSERT INTO mytable (id,
state_id,state_code,state_name,country_id,country_code,country_name,latitude,longitud
e,wikiDataId) VALUES (?, ?, ?....)';
```

4.1.2.4.2. Php

```
const query = 'INSERT INTO mytable (id,
state_id,state_code,state_name,country_id,country_code,country_name,latitude,longitud
e,wikiDataId) VALUES (?, ?, ?....)';
```

4.1.3. Kayıt Güncelleme Sorguları

Her veritabanı için oluşturulan kayıt güncelleme sorguları Node.Js ve Php özelinde şablon olarak belirtilmiştir.

4.1.3.1. *MySQL*

Kayıt güncelleme işlemleri sırasında MySQL için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.3.1.1. Node.Js

```
const query = 'UPDATE mytable SET column1 = ? WHERE state_name =?';
const params = ['new value1', default value];
```

4.1.3.1.2. Php

```
$sql = "UPDATE mytable SET column1='new value1' WHERE sate_name= default
value ";
```

4.1.3.2. *MongoDB*

Kayıt güncelleme işlemleri sırasında MongoDB için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.3.2.1. Node.Js

```
const db = client.db(dbName);
const collection = db.collection('mycollection');
const filter = { city: ‘value’ };
const update = { $set: { state_name: default value } };
collection.updateOne(filter, update, (err, result) => {
  if (err) throw err;
  console.log ('Data updated successfully');
  client.close();
});
});
```

4.1.3.2.2. Php

```
$updateResult=$db->updateOne(
      ['name'=>value],
      ['$set'=>["name"=>new value]],
  );
```

4.1.3.3. *HBase*

Kayıt güncelleme işlemleri sırasında HBase için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.3.3.1. Node.Js

```
const tableName = 'mytable';
const rowKey = 'myrow';
const data = { column1: 'new value1', column2: 'new value2' };
```

4.1.3.3.2. Php

```
$tableName = "mytable";

$rowKey = "myrow";

$data = [Column1" => "new value1"];

$client = new HbaseClient(["host" => "localhost", "port" => 9090]);

$client->put ($tableName, $rowKey, $data);
```

4.1.3.4. *Cassandra*

Kayıt güncelleme işlemleri sırasında Cassandra için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.3.4.1. Node.Js

```
const query = 'UPDATE mykeyspace.mytable SET column1 = WHERE state_name = default value';

const params = ['new value1',’value’];
```

4.1.3.4.2. Php

```
$sql = "UPDATE mytable SET column1='new value1' WHERE state_name=default value ";
```

4.1.4. Kayıt Silme Sorguları

Her veritabanı için oluşturulan kayıt silme sorguları Node.Js ve Php özelinde şablon olarak belirtilmiştir.

4.1.4.1. *MySQL*

Kayıt silme işlemleri sırasında MySQL için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.4.1.1. Node.Js

```
const query = 'DELETE FROM mytable WHERE state_name = default_value ';
```

4.1.4.1.2. Php

```
const query = 'DELETE FROM mytable WHERE state_name = default_value ';
```

4.1.4.2. *MongoDB*

Kayıt silme işlemleri sırasında MongoDB için kullanılan sorgu şablonları aşağıda

belirtilmiştir.

4.1.4.2.1.Node.Js

```
collection.deleteOne(filter, (err, result) => {
   if (err) throw err;
   console.log ('Data deleted successfully');
   client.close();
 });
```

4.1.4.2.2.Php

```
$bulkWrite->delete($filter, $options);
```

4.1.4.3. *HBase*

Kayıt silme işlemleri sırasında HBase için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.4.3.1.Node.Js

```
table.delete(rowKey)
     .then(() => {
       console.log ('Data deleted successfully');
       connection.close();
     })
```

4.1.4.3.2.Php

```
$rowKey = 'row1';
$client->deleteAllRow ($tableName, $rowKey, null);
```

4.1.4.4. *Cassandra*

Kayıt silme işlemleri sırasında Cassandra için kullanılan sorgu şablonları aşağıda belirtilmiştir.

4.1.4.4.1.Node.Js

```
const query = 'DELETE FROM mytable WHERE state_name = default value ';
```

4.1.4.4.2.Php

const query = 'DELETE FROM mytable WHERE state_name = default value ';

4.2. DENEY SONUÇLARI

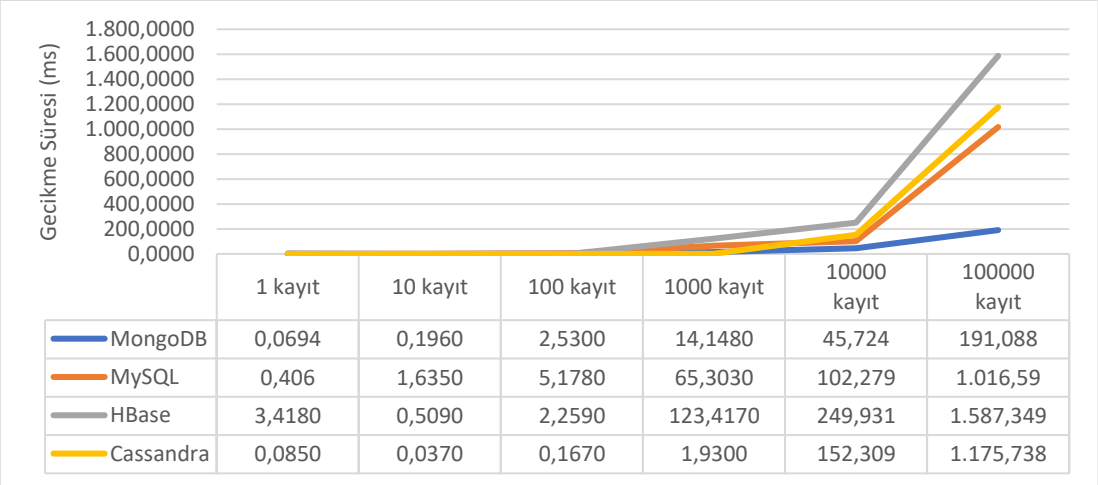
Deneylerde temel veritabanı işlemlerinden olan ve kısaca CRUD olarak adlandırılan ekleme, okuma, güncelleştirme ve silme işlemleri yapılmıştır. Bu işlemlerde kayıt sayıları 1'den başlayarak 10'un katları olarak kademeli şekilde artarak devam etmiş ve 100 bin kayıtta sonlandırılmıştır. CRUD işlemlerinin her bir adımı farklı işletim sistemlerinde farklı web teknolojilerinde, farklı çalışma ortamlarında ve farklı veritabanlarında yapılmıştır.

4.2.1. Kayıt Okuma İşlemleri

Kayıt okuma işlemlerine ait sonuçların bulunduğu grafiklere bu bölümde yer verilmiştir.

4.2.1.1. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.1. incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Cassandra ve MySQL ise birbirine yakın sonuçlar ortaya koymuştur.

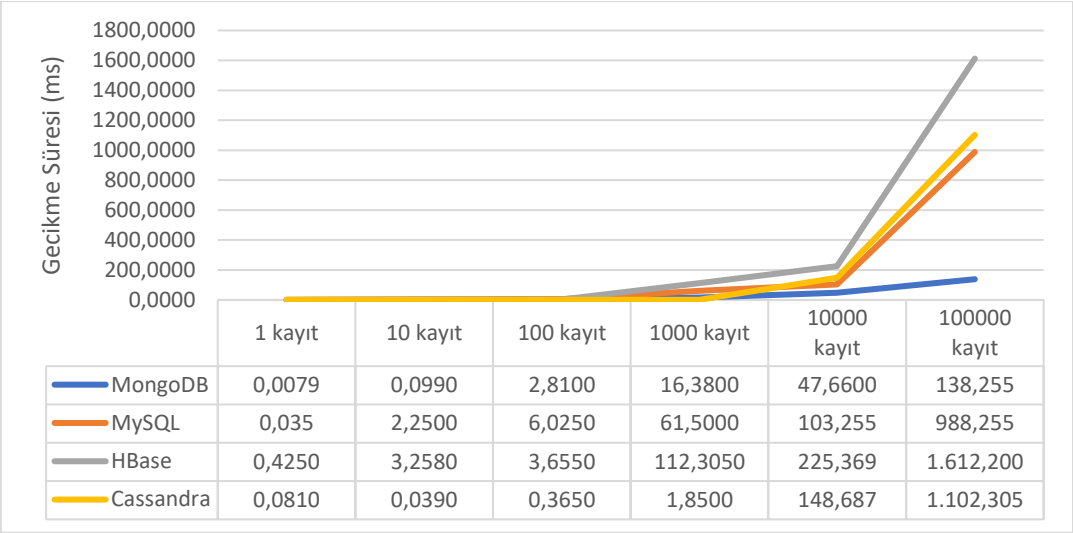


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0694	0,1960	2,5300	14,1480	45,724	191,088
MySQL	0,406	1,6350	5,1780	65,3030	102,279	1.016,59
HBase	3,4180	0,5090	2,2590	123,4170	249,931	1.587,349
Cassandra	0,0850	0,0370	0,1670	1,9300	152,309	1.175,738

Şekil 4.1. Yerel bilgisayarda Ubuntu-Node.Js 'de kayıt okuma gecikme süreleri.

4.2.1.2. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.2. incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Cassandra ve MySQL ise birbirine yakın sonuçlar ortaya koymuştur.

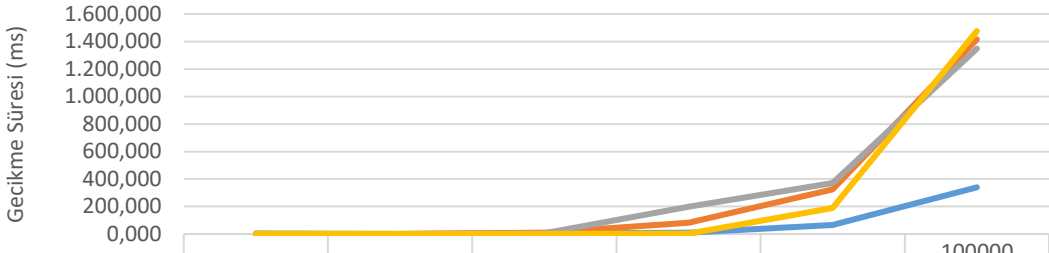


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0079	0,0990	2,8100	16,3800	47,6600	138,255
MySQL	0,035	2,2500	6,0250	61,5000	103,255	988,255
HBase	0,4250	3,2580	3,6550	112,3050	225,369	1.612,200
Cassandra	0,0810	0,0390	0,3650	1,8500	148,687	1.102,305

Şekil 4.2. Yerel bilgisayarda Ubuntu-Php ‘de kayıt okuma gecikme süreleri.

4.2.1.3. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.3. incelendiğinde en iyi sürenin MongoDB, en kötü sürenin ise Cassandra veritabanına ait olduğu görülmektedir. Hbase ve MySQL ise birbirine yakın sonuçlar ortaya koymuştur.

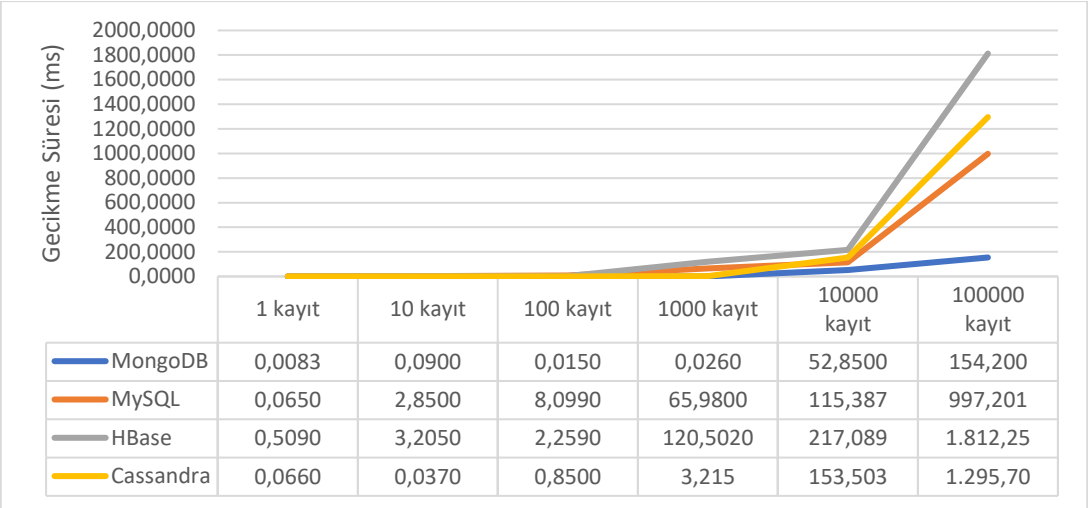


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,063	0,116	2,429	9,152	64,764	339,787
MySQL	0,435	1,695	11,036	82,057	323,749	1.415,623
HBase	5,658	0,599	5,801	198,417	369,991	1.348,205
Cassandra	0,105	0,049	0,255	2,135	188,255	1.476,232

Şekil 4.3. Yerel bilgisayarda Windows-Node.Js ‘de kayıt okuma gecikme süreleri.

4.2.1.4. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.4. incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Cassandra ve MySQL ise birbirine yakın sonuçlar ortaya koymuştur.

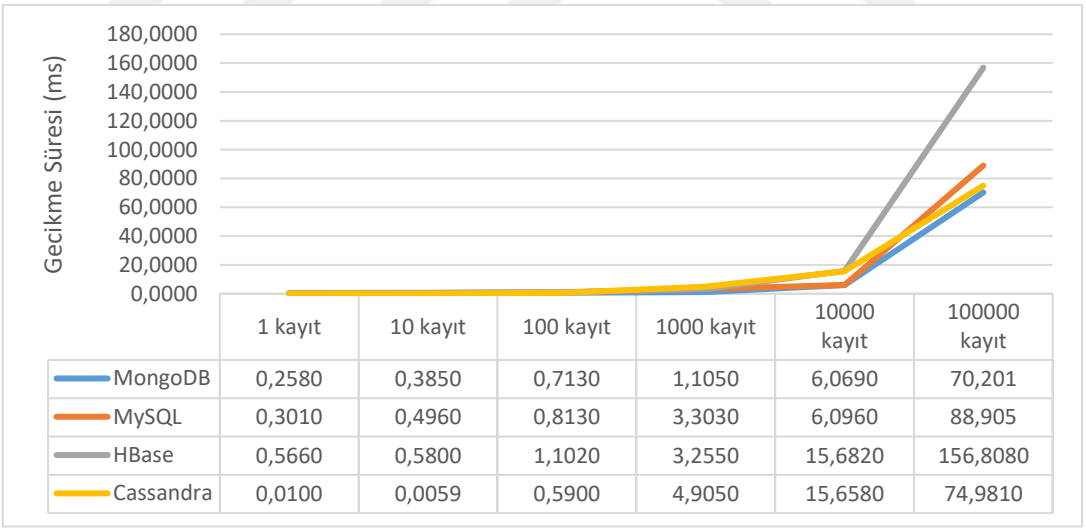


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0083	0,0900	0,0150	0,0260	52,8500	154,200
MySQL	0,0650	2,8500	8,0990	65,9800	115,387	997,201
HBase	0,5090	3,2050	2,2590	120,5020	217,089	1.812,25
Cassandra	0,0660	0,0370	0,8500	3,215	153,503	1.295,70

Şekil 4.4. Yerel bilgisayarda Windows-Php ‘de kayıt okuma gecikme süreleri.

4.2.1.5. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.5. incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Dört veritabanının da birbirine çok uzak olmayan sonuçlar verdiği görülmektedir.

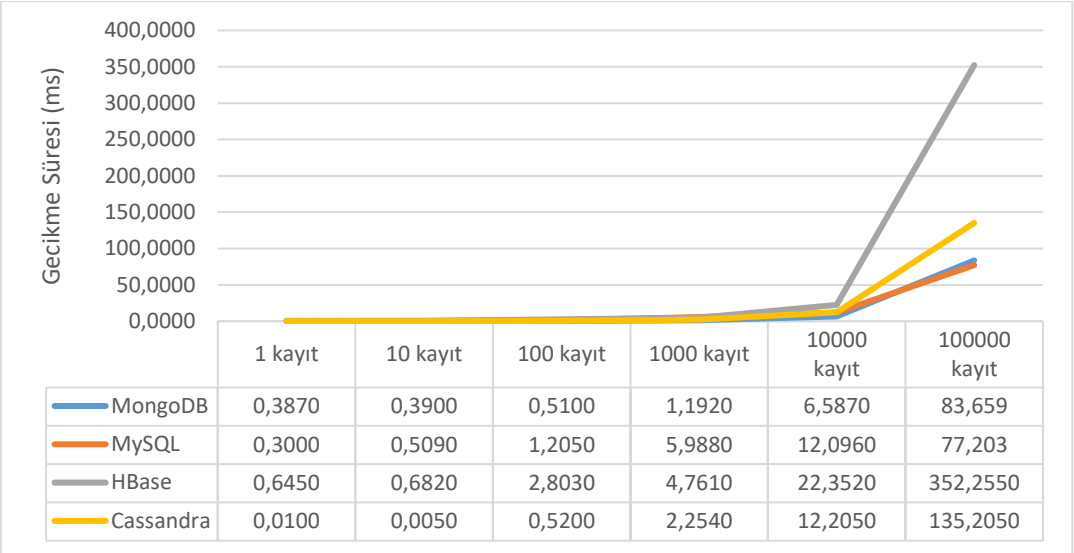


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,2580	0,3850	0,7130	1,1050	6,0690	70,201
MySQL	0,3010	0,4960	0,8130	3,3030	6,0960	88,905
HBase	0,5660	0,5800	1,1020	3,2550	15,6820	156,8080
Cassandra	0,0100	0,0059	0,5900	4,9050	15,6580	74,9810

Şekil 4.5. Bulut ortamında Ubuntu-Node.Js’de kayıt okuma gecikme süreleri.

4.2.1.6. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.6. incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Dört veritabanının da birbirine çok uzak olmayan sonuçlar verdiği görülmektedir.

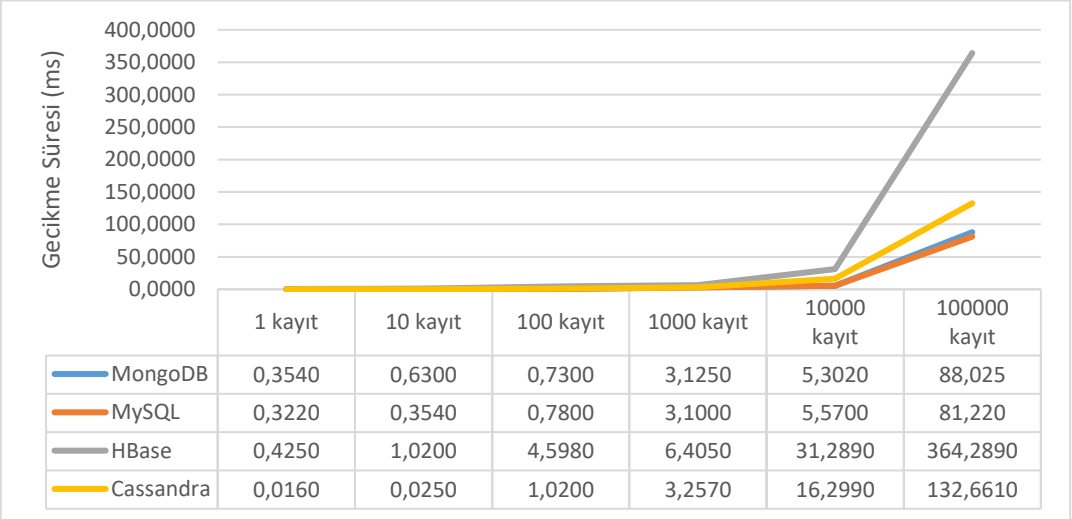


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,3870	0,3900	0,5100	1,1920	6,5870	83,659
MySQL	0,3000	0,5090	1,2050	5,9880	12,0960	77,203
HBase	0,6450	0,6820	2,8030	4,7610	22,3520	352,2550
Cassandra	0,0100	0,0050	0,5200	2,2540	12,2050	135,2050

Şekil 4.6. Bulut ortamında Ubuntu-Php'de kayıt okuma gecikme süreleri.

4.2.1.7. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.7 incelendiğinde sonuçların birbirine yakın olduğu fakat HBase veritabanının bu deneyde en yüksek gecikme süresine ulaştığı görülmüştür.

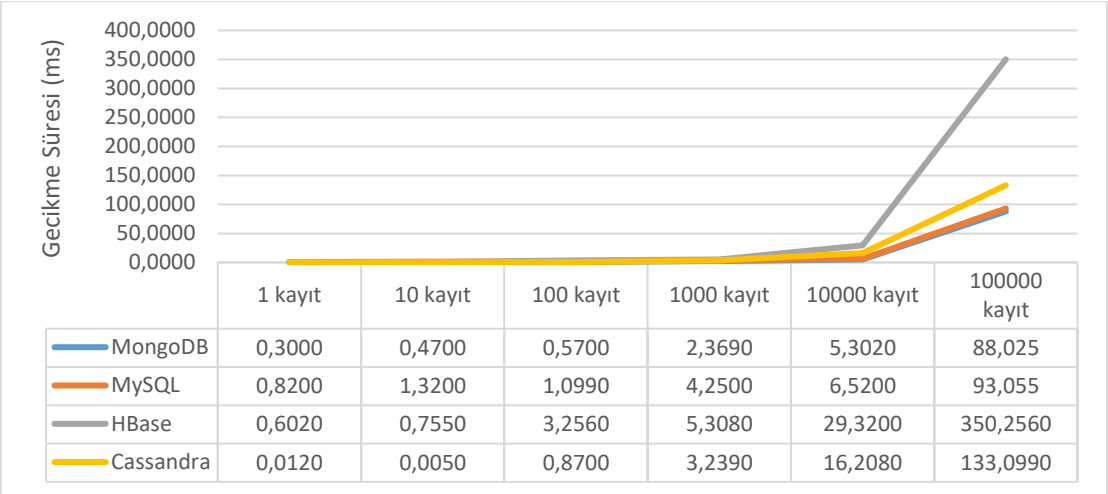


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,3540	0,6300	0,7300	3,1250	5,3020	88,025
MySQL	0,3220	0,3540	0,7800	3,1000	5,5700	81,220
HBase	0,4250	1,0200	4,5980	6,4050	31,2890	364,2890
Cassandra	0,0160	0,0250	1,0200	3,2570	16,2990	132,6610

Şekil 4.7. Bulut ortamında Windows-Node.Js'de kayıt okuma gecikme süreleri.

4.2.1.8. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.8 incelendiğinde en iyi gecikme süresinin MongoDB, en kötü gecikme süresinin ise HBase veritabanına ait olduğu görülmektedir. Dört veritabanının da birbirine çok uzak olmayan sonuçlar verdiği görülmektedir.



	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,3000	0,4700	0,5700	2,3690	5,3020	88,025
MySQL	0,8200	1,3200	1,0990	4,2500	6,5200	93,055
HBase	0,6020	0,7550	3,2560	5,3080	29,3200	350,2560
Cassandra	0,0120	0,0050	0,8700	3,2390	16,2080	133,0990

Şekil 4.8. Bulut ortamında Windows-Php'de kayıt okuma gecikme süreleri.

Kayıt okuma işlemleri için yapılan sorgulara ait sonuçlar aşağıda bulunan tablolarda belirtilmiştir. Tabloları baz alarak sonuçları işletim sistemleri değişkenine bakarak yorumladığımızda Ubuntu işletim sisteminin Windows işletim sisteminden daha iyi sonuçlar verdiği görülmektedir.

Tabloları baz alarak web teknolojileri bazında değerlendirildiğinde ise Php ile Node.Js arasında belirgin farklar bulunmamakla beraber Php teknolojisinde daha iyi sonuçların alındığı görülmüştür. Tabloları baz alarak bir diğer değişken olan bilgisayar sistemleri incelendiğinde Cloud yapısında yapılan testlerin daha iyi sonuçlar verdiği görülmüştür. Yapılan deneylerde ilk 10 bin kayıta kadar ciddi farkların olmadığı da görülmüştür.

Bu bağlamda bakıldığında genel olarak MongoDB ile yapılan ölçümlerin diğer veritabanı sistemlerine göre daha hızlı olduğu görülmüştür. Yapılan testlerde diğer veri tabanlarına göre daha yavaş olan sistem ise HBase olarak görülmüştür. Bu ölçümlerde her ne kadar HBase veritabanının en yavaş olduğu ölçülse de özellikle bulut özelinde yapılan çalışmalarda sonuçların birbirine yakın olmasıyla birlikte HBase haricinde diğer veritabanlarının daha yakın sonuçlar verdiği görülmüştür. Çizelge 4.5'de kayıt okuma senaryosunun genel değerlendirmesi yer almaktadır.

Çizelge 4.5. Kayıt okuma işleminin genel değerlendirmesi.

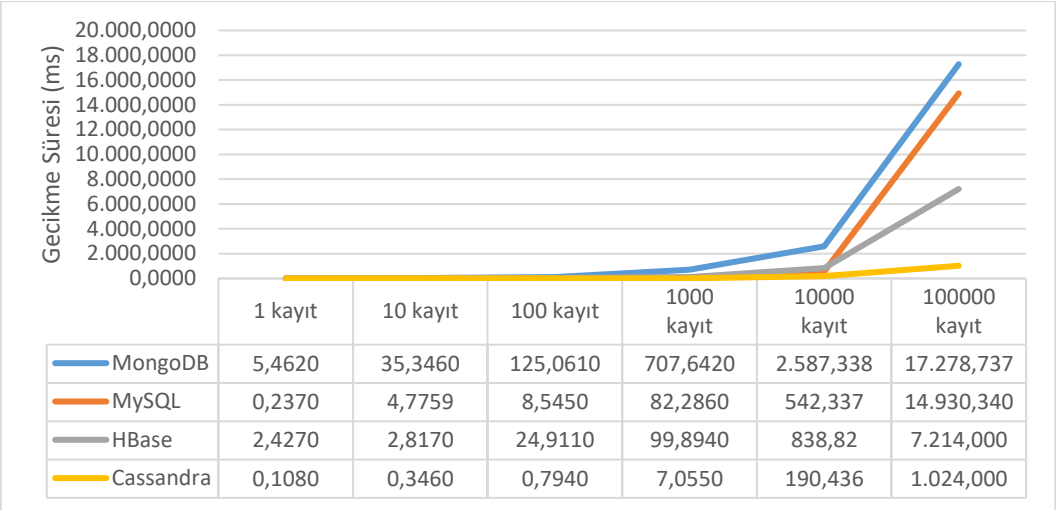
İşletim sistemi	Web Teknolojileri	Çalışma Ortamı	En iyi gecikme süresine sahip veritabanı	En kötü gecikme süresine sahip veritabanı
Ubuntu	Node.Js	Yerel Bilgisayar	MongoDB	HBase
Ubuntu	Php	Yerel Bilgisayar	MongoDB	HBase
Windows	Node.Js	Yerel Bilgisayar	MongoDB	Cassandra
Windows	Php	Yerel Bilgisayar	MongoDB	HBase
Ubuntu	Node.Js	Bulut Sistemi	MongoDB	HBase
Ubuntu	Php	Bulut Sistemi	MySQL	HBase
Windows	Node.Js	Bulut Sistemi	MySQL	HBase
Windows	Php	Bulut Sistemi	MongoDB	HBase

4.2.2. Kayıt Ekleme İşlemleri

Kayıt okuma işlemlerine ait sonuçların bulunduğu grafiklere bu bölümde yer verilmiştir.

4.2.2.1. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.9.'daki grafiğe göre Cassandra veritabanının en iyi gecikme sürelerine sahip olduğu, MongoDB veritabanının ise bu testlerde geride kaldığı görülmüştür.

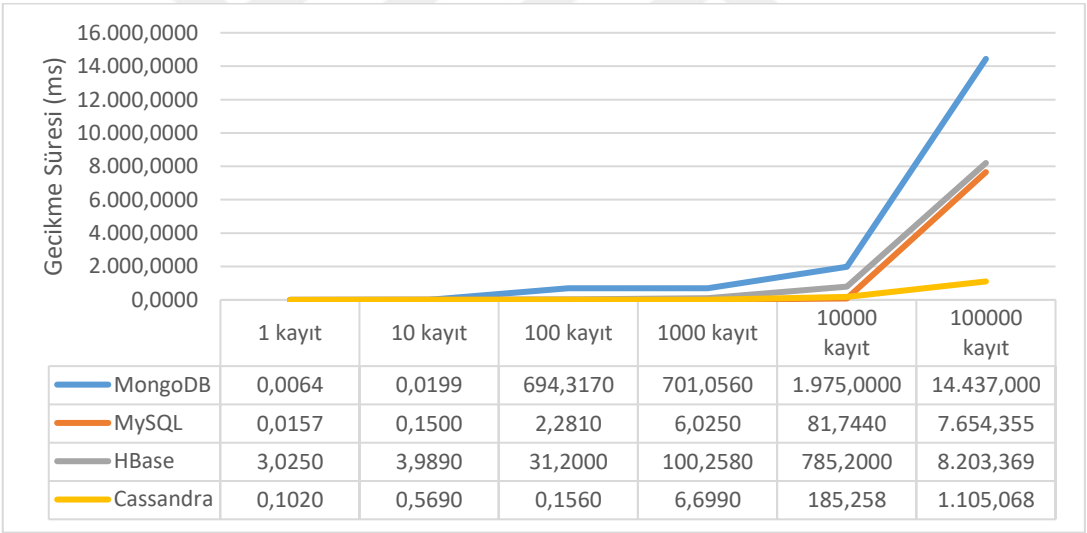


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	5,4620	35,3460	125,0610	707,6420	2.587,338	17.278,737
MySQL	0,2370	4,7759	8,5450	82,2860	542,337	14.930,340
HBase	2,4270	2,8170	24,9110	99,8940	838,82	7.214,000
Cassandra	0,1080	0,3460	0,7940	7,0550	190,436	1.024,000

Şekil 4.9. Yerel bilgisayarda Ubuntu-Node.Js'de kayıt ekleme gecikme süreleri.

4.2.2.2. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.10.'daki grafiğe göre Cassandra veritabanının en iyi gecikme sürelerine sahip olduğu, MongoDB'nin ise bu testlerde geride kaldığı görülmüştür.

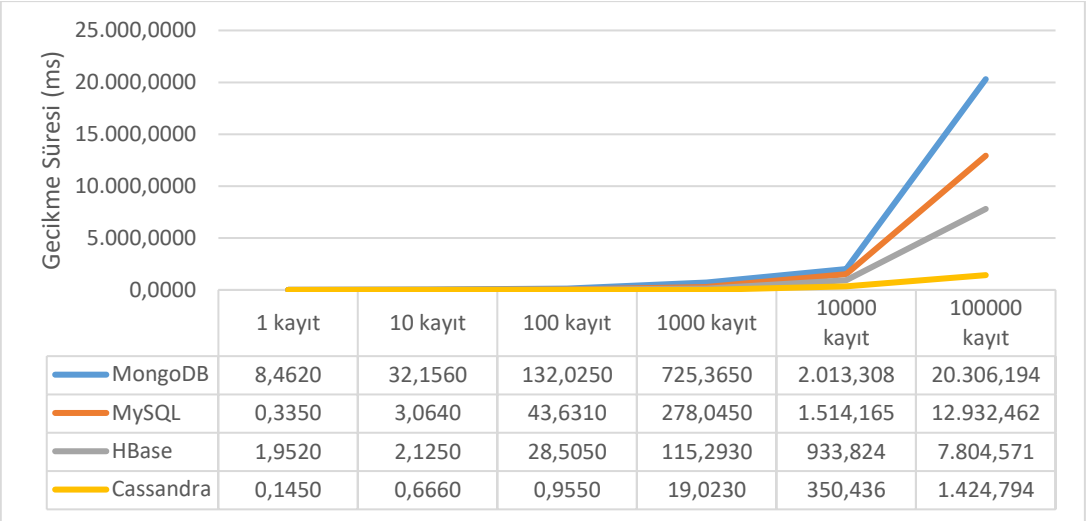


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0064	0,0199	694,3170	701,0560	1.975,0000	14.437,000
MySQL	0,0157	0,1500	2,2810	6,0250	81,7440	7.654,355
HBase	3,0250	3,9890	31,2000	100,2580	785,2000	8.203,369
Cassandra	0,1020	0,5690	0,1560	6,6990	185,258	1.105,068

Şekil 4.10. Yerel bilgisayarda Ubuntu-Php'de kayıt ekleme gecikme süreleri.

4.2.2.3. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.11'daki grafiğe göre Cassandra veritabanının en iyi gecikme sürelerine sahip olduğu, MongoDB'nin ise bu testlerde geride kaldığı görülmüştür. Cassandra veritabanı da MySQL veritabanına yakın sonuçlar vermiştir.

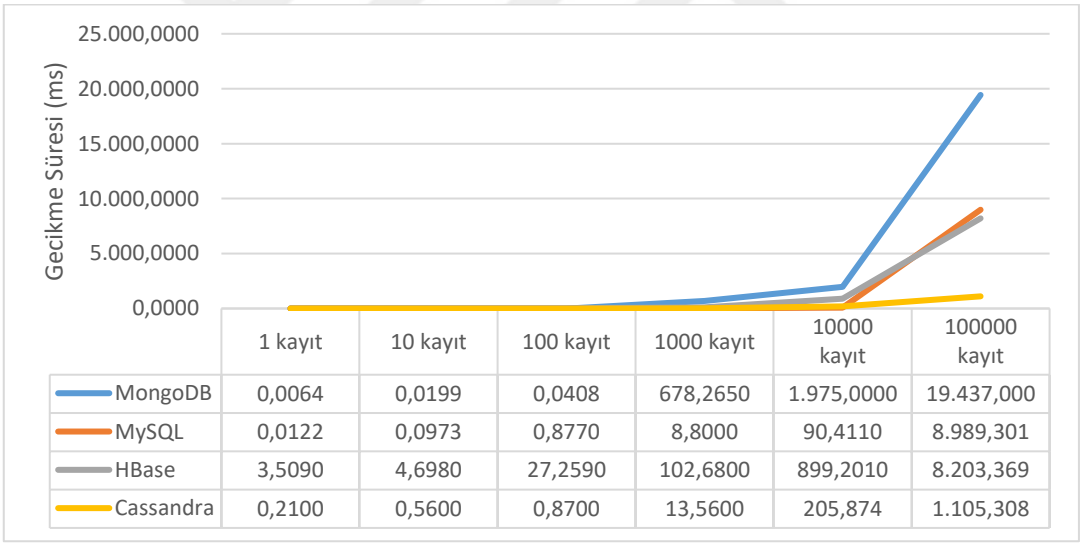


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	8,4620	32,1560	132,0250	725,3650	2.013,308	20.306,194
MySQL	0,3350	3,0640	43,6310	278,0450	1.514,165	12.932,462
HBase	1,9520	2,1250	28,5050	115,2930	933,824	7.804,571
Cassandra	0,1450	0,6660	0,9550	19,0230	350,436	1.424,794

Şekil 4.11. Yerel bilgisayarda Windows-Node.Js'de kayıt ekleme gecikme süreleri

4.2.2.4. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.12.'deki grafiğe göre Cassandra veritabanının en iyi gecikme sürelerine sahip olduğu, MongoDB'nin ise bu testlerde geride kaldığı görülmüştür.

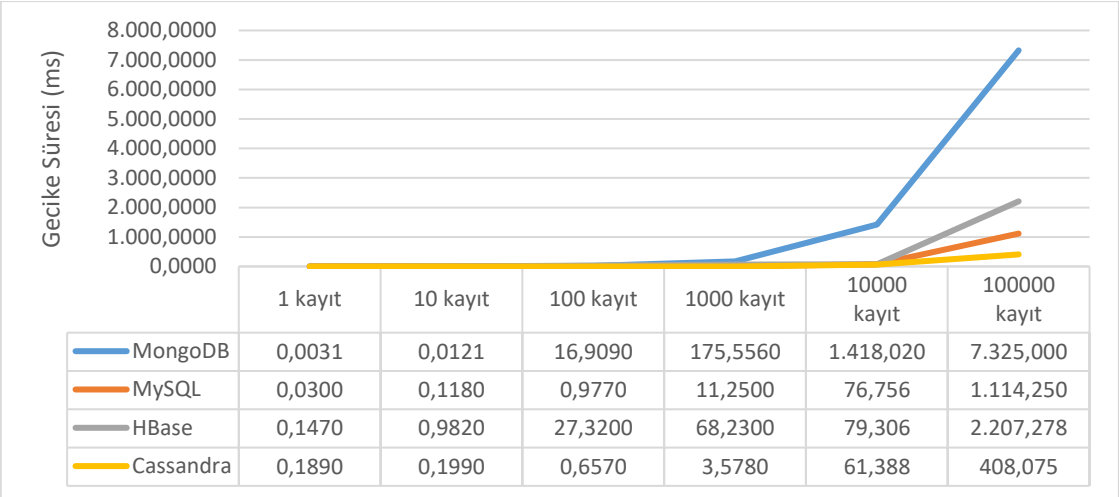


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0064	0,0199	0,0408	678,2650	1.975,0000	19.437,000
MySQL	0,0122	0,0973	0,8770	8,8000	90,4110	8.989,301
HBase	3,5090	4,6980	27,2590	102,6800	899,2010	8.203,369
Cassandra	0,2100	0,5600	0,8700	13,5600	205,874	1.105,308

Şekil 4.12. Yerel bilgisayarda Ubuntu-Php'de kayıt ekleme gecikme süreleri

4.2.2.5. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar- Bulut*

Şekil 4.13'deki grafik incelendiğinde bulut çalışma ortamının da etkisiyle gecikme sürelerinin düştüğü, fakat MongoDB veritabanının bu çalışma sürelerini baz aldığımızda diğer veritabanlarına oranla kötü sonuçlar ortaya koyduğu görülmüştür.

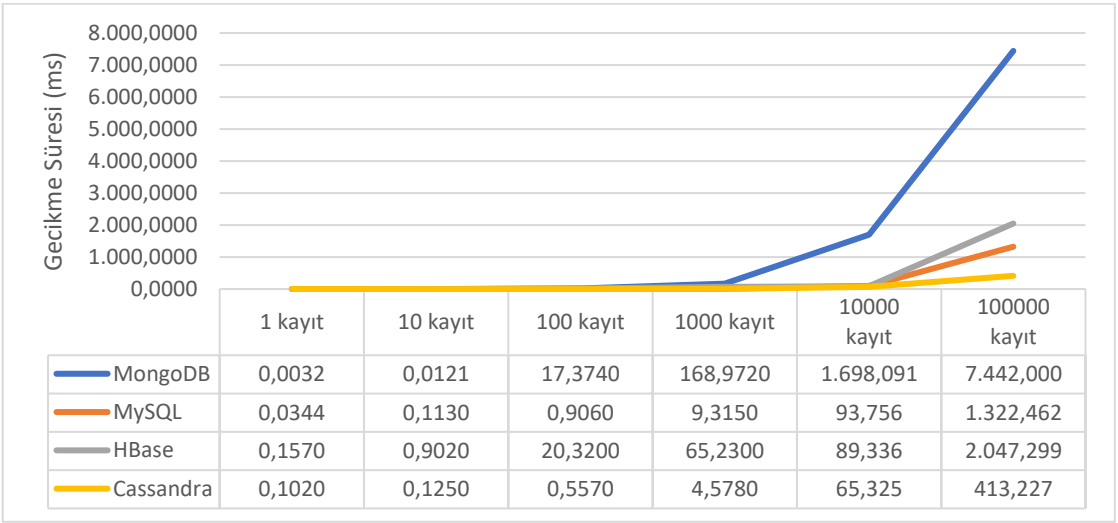


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0031	0,0121	16,9090	175,5560	1.418,020	7.325,000
MySQL	0,0300	0,1180	0,9770	11,2500	76,756	1.114,250
HBase	0,1470	0,9820	27,3200	68,2300	79,306	2.207,278
Cassandra	0,1890	0,1990	0,6570	3,5780	61,388	408,075

Şekil 4.13. Bulut ortamında Ubuntu-Node.Js'de kayıt ekleme gecikme süreleri.

4.2.2.6. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar- Bulut*

Şekil 4.14'deki grafik incelendiğinde bulut çalışma ortamının da etkisiyle gecikme sürelerinin ciddi şekilde düştüğü, fakat MongoDB veritabanının bu çalışma sürelerini baz aldığımızda diğer veritabanlarına oranla kötü sonuçlar ortaya koyduğu görülmüştür.

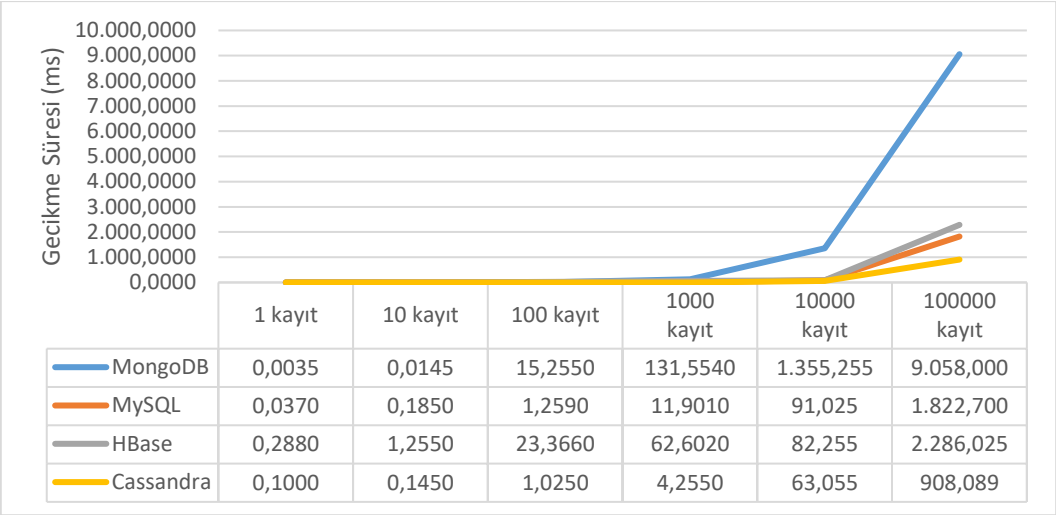


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0032	0,0121	17,3740	168,9720	1.698,091	7.442,000
MySQL	0,0344	0,1130	0,9060	9,3150	93,756	1.322,462
HBase	0,1570	0,9020	20,3200	65,2300	89,336	2.047,299
Cassandra	0,1020	0,1250	0,5570	4,5780	65,325	413,227

Şekil 4.14. Bulut Ortamında Windows-Php'de kayıt ekleme gecikme süreleri.

4.2.2.7. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar- Bulut*

Şekil 4.15'deki grafik incelendiğinde bulut çalışma ortamının da etkisiyle gecikme sürelerinin ciddi şekilde düştüğü, fakat MongoDB veritabanının bu çalışma sürelerini baz aldığımızda diğer veritabanlarına oranla kötü sonuçlar ortaya koyduğu görülmüştür.

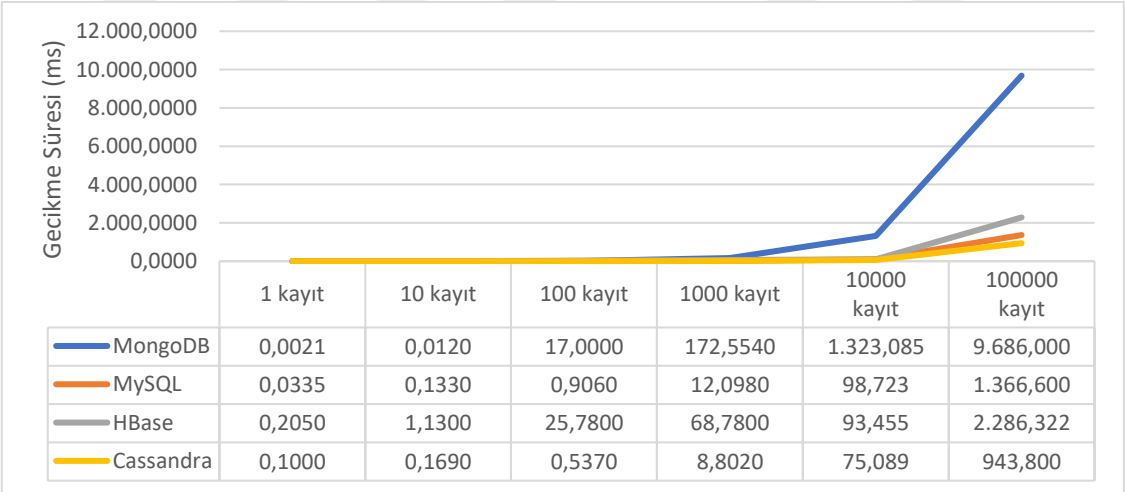


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0035	0,0145	15,2550	131,5540	1.355,255	9.058,000
MySQL	0,0370	0,1850	1,2590	11,9010	91,025	1.822,700
HBase	0,2880	1,2550	23,3660	62,6020	82,255	2.286,025
Cassandra	0,1000	0,1450	1,0250	4,2550	63,055	908,089

Şekil 4.15. Bulut ortamında Windows-Node.Js'de kayıt ekleme gecikme süreleri.

4.2.2.8. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar- Bulut*

Şekil 4.16'daki grafik incelendiğinde bulut çalışma ortamının da etkisiyle gecikme sürelerinin ciddi şekilde düştüğü, fakat MongoDB veritabanının bu çalışma sürelerini baz aldığımızda diğer veritabanlarına oranla kötü sonuçlar ortaya koyduğu görülmüştür.



	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0021	0,0120	17,0000	172,5540	1.323,085	9.686,000
MySQL	0,0335	0,1330	0,9060	12,0980	98,723	1.366,600
HBase	0,2050	1,1300	25,7800	68,7800	93,455	2.286,322
Cassandra	0,1000	0,1690	0,5370	8,8020	75,089	943,800

Şekil 4.16. Bulut ortamında Windows-Php'de kayıt ekleme gecikme süreleri.

Kayıt ekleme işlemleri için yapılan sorgulara ait sonuçlar yukarıda bulunan grafiklerde belirtilmiştir. Tabloları baz alıp sonuçları işletim sistemleri değişkenine bakarak yorumladığımızda Ubuntu işletim sisteminin Windows işletim sisteminden daha iyi sonuçlar verdiği görülmektedir.

Tabloları baz alarak web teknolojileri bazında değerlendirme yapıldığında ise Php ile

Node.Js arasında belirgin farklar bulunmamakla beraber Php teknolojisinde daha iyi sonuçların alındığı görülmüştür. Tabloları baz alarak bir diğer değişken olan bilgisayar sistemleri incelendiğinde Bulut yapısında yapılan testlerin çok daha iyi sonuçlar verdiği görülmüştür. Bu bağlamda bakıldığında Cassandra ile yapılan ölçümlerin diğer veritabanı sistemlerine göre daha hızlı olduğu görülmüştür. Yapılan testlerde diğer veri tabanlarına göre daha yavaş olan sistem ise MongoDB olarak görülmüştür. Özellikle Bulut altyapısında gecikme sürelerinin oldukça düştüğü fakat buna rağmen NoSQL veritabanı olan MongoDB'nin MySQL veritabanının gerisinde kaldığı görülmüştür. HBase ise MySQL veritabanından sonra gelmiştir. Yapılan deneylerde ilk 10 bin kayıta kadar ciddi farkların olmadığı da görülmüştür. Çizelge 4.6'da kayıt güncelleme senaryosunun genel değerlendirmesi yer almaktadır.

Çizelge 4.6. Kayıt ekleme işleminin genel değerlendirmesi.

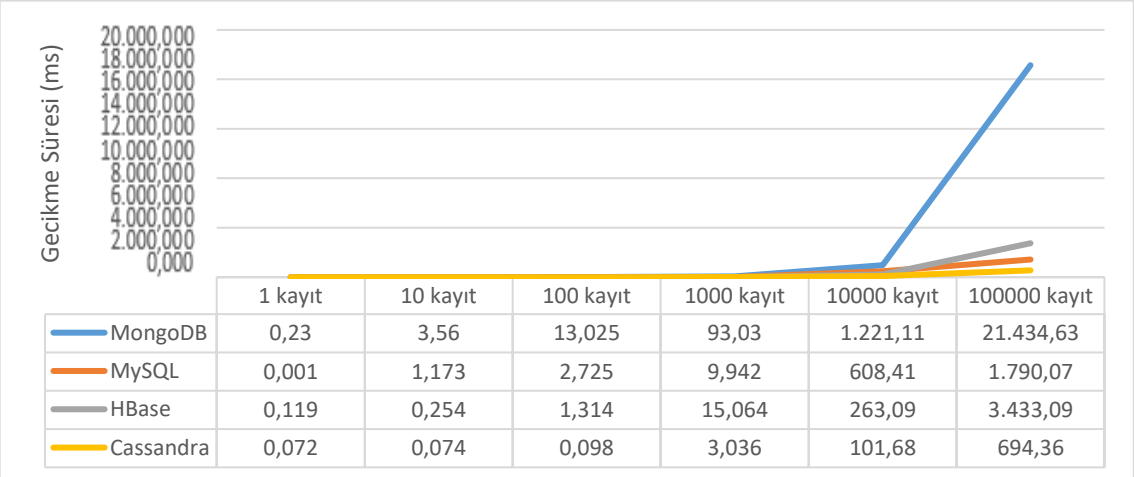
İşletim sistemi	Web Teknolojileri	Çalışma Ortamı	En iyi gecikme süresine sahip veritabanı	En kötü gecikme süresine sahip veritabanı
Ubuntu	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Ubuntu	Php	Yerel Bilgisayar	Cassandra	MongoDB
Windows	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Windows	Php	Yerel Bilgisayar	Cassandra	MongoDB
Ubuntu	Node.Js	Bulut Sistemi	Cassandra	MongoDB
Ubuntu	Php	Bulut Sistemi	Cassandra	MongoDB
Windows	Node.Js	Bulut Sistemi	Cassandra	MongoDB
Windows	Php	Bulut Sistemi	Cassandra	MongoDB

4.2.3. Kayıt Güncelleme İşlemleri

Kayıt güncelleme işlemlerine ait sonuçların bulunduğu grafiklere bu bölümde yer verilmiştir.

4.2.3.1. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.17'deki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

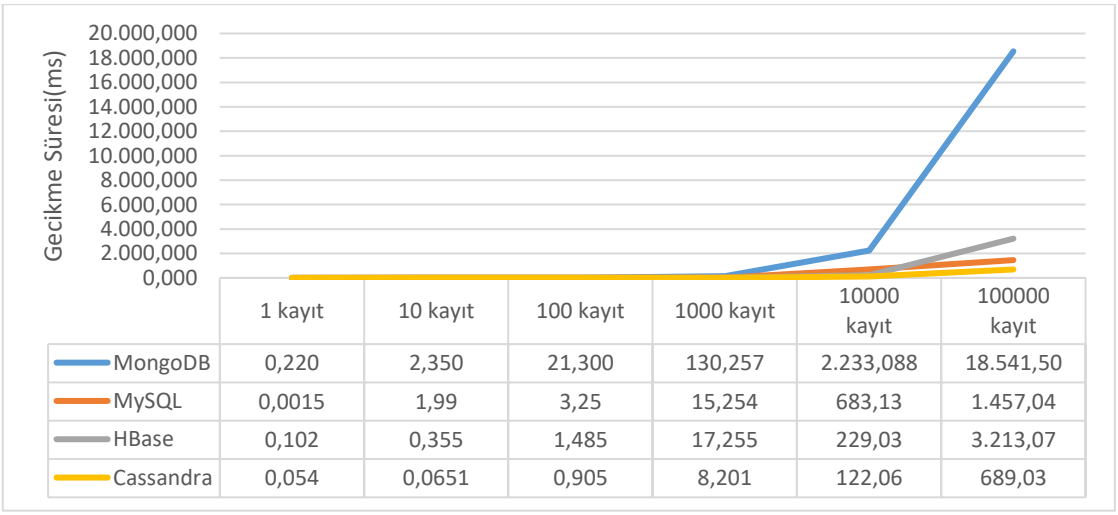


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,23	3,56	13,025	93,03	1.221,11	21.434,63
MySQL	0,001	1,173	2,725	9,942	608,41	1.790,07
HBase	0,119	0,254	1,314	15,064	263,09	3.433,09
Cassandra	0,072	0,074	0,098	3,036	101,68	694,36

Şekil 4.17. Yerel bilgisayarda Ubuntu-Node.Js'de kayıt güncelleme gecikme süreleri.

4.2.3.2. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.18'deki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

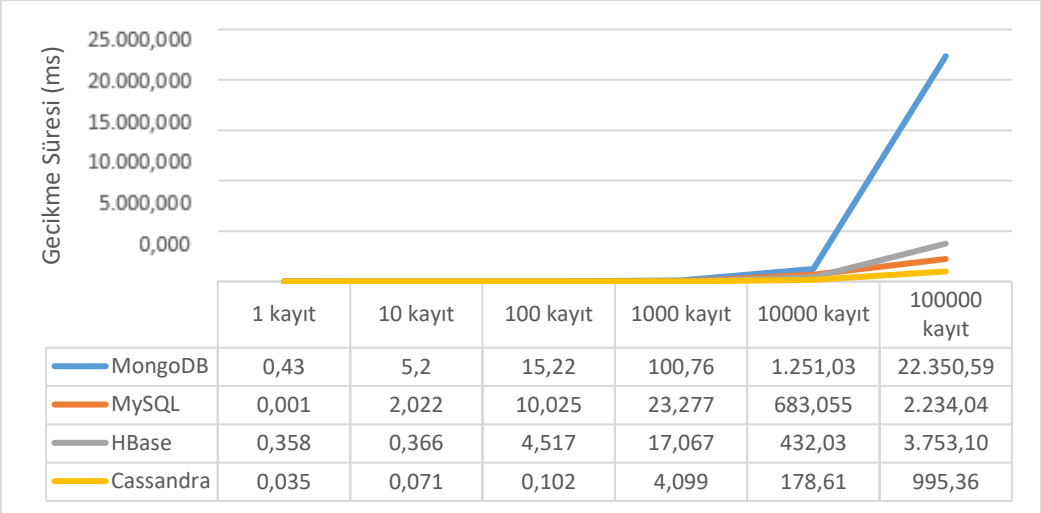


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,220	2,350	21,300	130,257	2.233,088	18.541,50
MySQL	0,0015	1,99	3,25	15,254	683,13	1.457,04
HBase	0,102	0,355	1,485	17,255	229,03	3.213,07
Cassandra	0,054	0,0651	0,905	8,201	122,06	689,03

Şekil 4.18. Yerel bilgisayarda Ubuntu-Php‘de kayıt güncelleme gecikme süreleri.

4.2.3.3. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.19’daki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

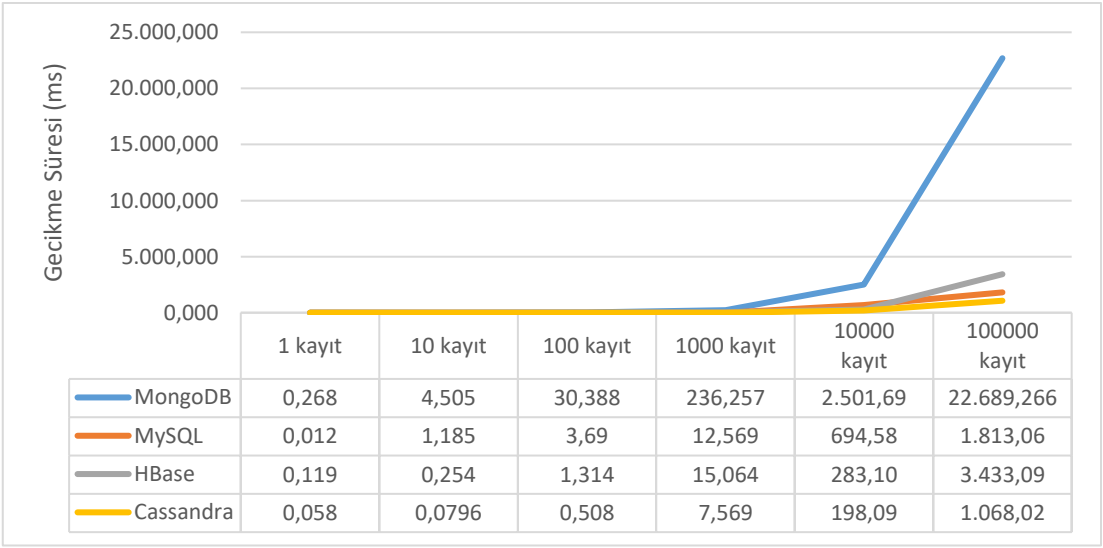


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,43	5,2	15,22	100,76	1.251,03	22.350,59
MySQL	0,001	2,022	10,025	23,277	683,055	2.234,04
HBase	0,358	0,366	4,517	17,067	432,03	3.753,10
Cassandra	0,035	0,071	0,102	4,099	178,61	995,36

Şekil 4.19. Yerel bilgisayarda Windows-Node.Js‘de kayıt ekleme gecikme süreleri

4.2.3.4. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.20’daki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

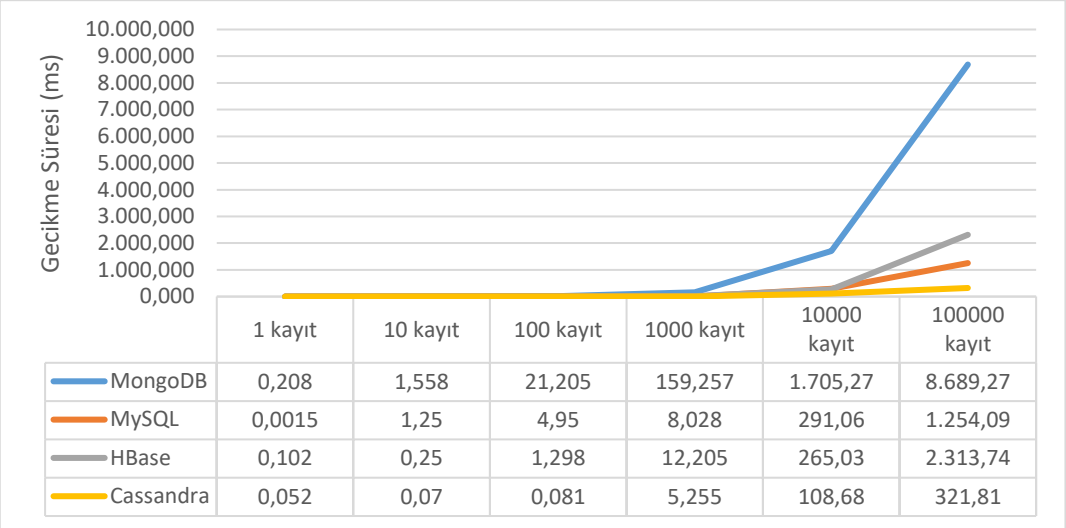


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,268	4,505	30,388	236,257	2.501,69	22.689,266
MySQL	0,012	1,185	3,69	12,569	694,58	1.813,06
HBase	0,119	0,254	1,314	15,064	283,10	3.433,09
Cassandra	0,058	0,0796	0,508	7,569	198,09	1.068,02

Şekil 4.20. Yerel bilgisayarda Windows-Php‘de kayıt Güncelleme gecikme süreleri.

4.2.3.5. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.21’deki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

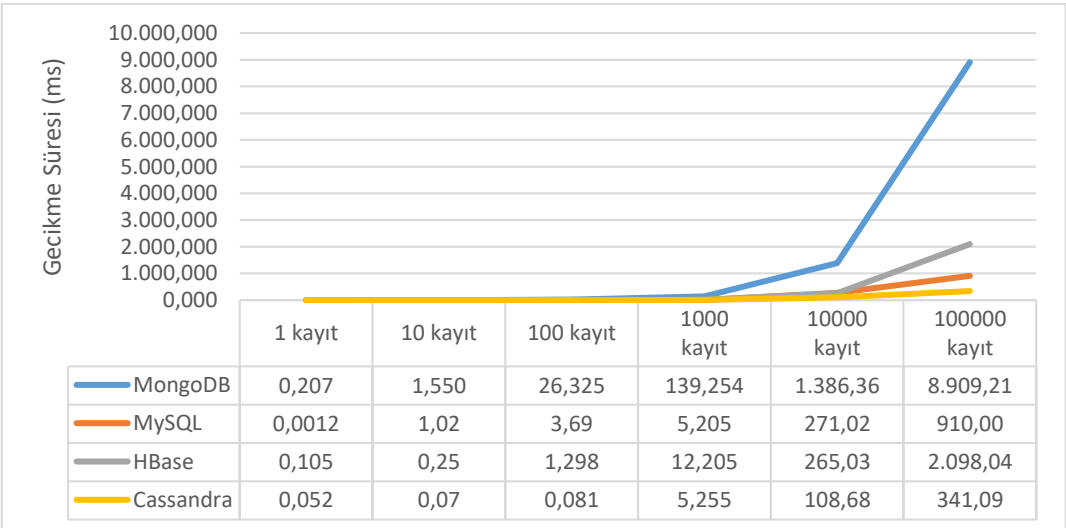


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,208	1,558	21,205	159,257	1.705,27	8.689,27
MySQL	0,0015	1,25	4,95	8,028	291,06	1.254,09
HBase	0,102	0,25	1,298	12,205	265,03	2.313,74
Cassandra	0,052	0,07	0,081	5,255	108,68	321,81

Şekil 4.21. Bulut ortamında Ubuntu-Node.Js’de kayıt güncelleme gecikme süreleri.

4.2.3.6. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.22’deki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

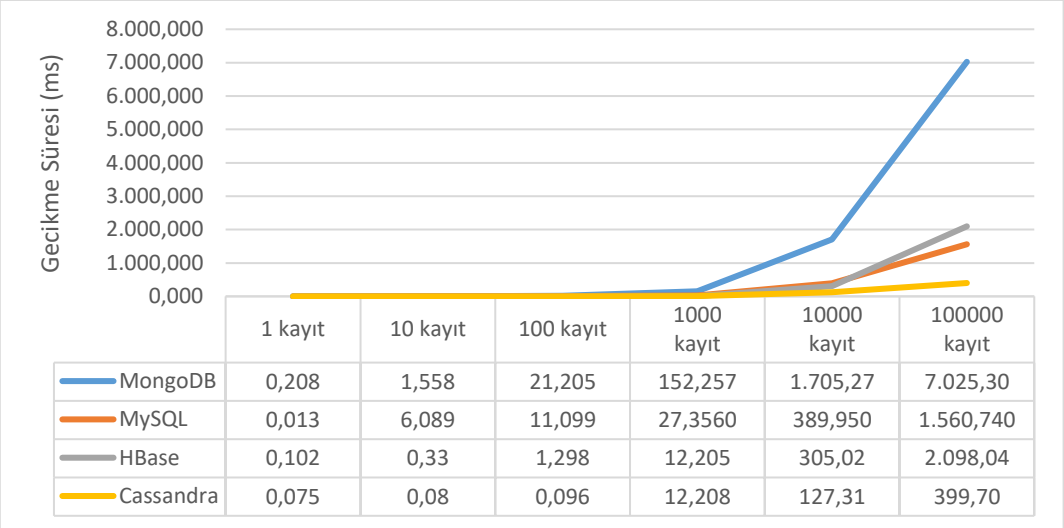


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,207	1,550	26,325	139,254	1.386,36	8.909,21
MySQL	0,0012	1,02	3,69	5,205	271,02	910,00
HBase	0,105	0,25	1,298	12,205	265,03	2.098,04
Cassandra	0,052	0,07	0,081	5,255	108,68	341,09

Şekil 4.22. Bulut ortamında Ubuntu-Php’de kayıt güncelleme gecikme süreleri.

4.2.3.7. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.23'teki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.

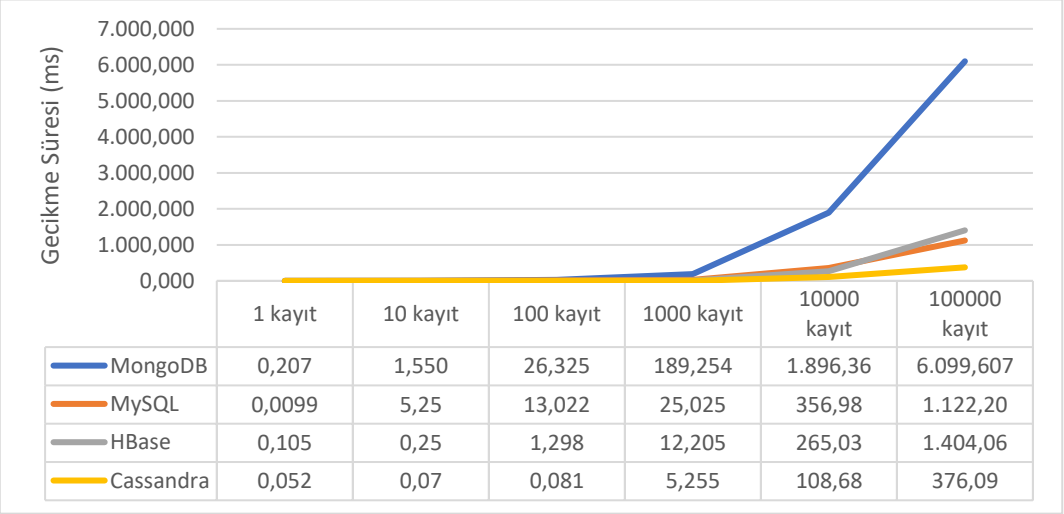


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,208	1,558	21,205	152,257	1.705,27	7.025,30
MySQL	0,013	6,089	11,099	27,3560	389,950	1.560,740
HBase	0,102	0,33	1,298	12,205	305,02	2.098,04
Cassandra	0,075	0,08	0,096	12,208	127,31	399,70

Şekil 4.23. Bulut ortamında Windows-Node.Js'de kayıt güncelleme gecikme süreleri.

4.2.3.8. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.24'teki sonuçlar incelendiğinde Cassandra veritabanının en iyi gecikme süresine sahip olduğu görülmektedir. MongoDB veritabanı en kötü sonuçları vermiştir. Diğer iki veritabanı içerisinde MySQL, HBase veritabanından daha iyi sonuçlar vermiştir.



	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,207	1,550	26,325	189,254	1.896,36	6.099,607
MySQL	0,0099	5,25	13,022	25,025	356,98	1.122,20
HBase	0,105	0,25	1,298	12,205	265,03	1.404,06
Cassandra	0,052	0,07	0,081	5,255	108,68	376,09

Şekil 4.24. Bulut ortamında Windows-Php'de kayıt güncelleme gecikme süreleri.

Kayıt güncelleme işlemleri için yapılan sorgulara ait sonuçlar yukarıda bulunan

grafiklerde belirtilmiştir. Tabloları baz alarak sonuçları işletim sistemleri değişkenine bakarak yorumladığımızda Ubuntu işletim sisteminin Windows işletim sisteminden daha iyi sonuçlar verdiği fakat aralarında ciddi farkların bulunmadığı görülmektedir.

Tabloları baz alarak web teknolojileri bazında değerlendirildiğinde ise Php ile Node.Js arasında belirgin farklar bulunmamakla beraber Php teknolojisinde daha iyi sonuçların alındığı görülmüştür. Tabloları baz alarak bir diğer değişken olan bilgisayar sistemleri incelendiğinde Cloud yapısında yapılan testlerin daha iyi sonuçlar verdiği görülmüştür. Yapılan deneylerde ilk bin kayıta kadar ciddi farkların olmadığı da görülmüştür.

Bu bağlamda bakıldığında Cassandra ile yapılan ölçümlerin diğer veritabanı sistemlerine göre daha hızlı olduğu görülmüştür. Yapılan testlerde diğer veri tabanlarına göre daha yavaş olan sistem ise MongoDB olarak görülmüştür. İlişkisel olmayan veritabanı olan HBase ise MongoDB'nin ardından en yavaş veritabanı sistemidir. Çizelge 4.7 de kayıt güncelleme senaryosunun genel değerlendirmesi yer almaktadır.

Çizelge 4.7. Kayıt güncelleme işleminin genel değerlendirmesi.

İşletim sistemi	Web Teknolojileri	Çalışma Ortamı	En iyi gecikme süresine sahip veritabanı	En kötü gecikme süresine sahip veritabanı
Ubuntu	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Ubuntu	Php	Yerel Bilgisayar	Cassandra	MongoDB
Windows	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Windows	Php	Yerel Bilgisayar	Cassandra	MongoDB
Ubuntu	Node.Js	Bulut Sistemi	Cassandra	MongoDB
Ubuntu	Php	Bulut Sistemi	Cassandra	MongoDB

Çizelge 4.8. Devam. (Kayıt güncelleme işleminin genel değerlendirmesi.)

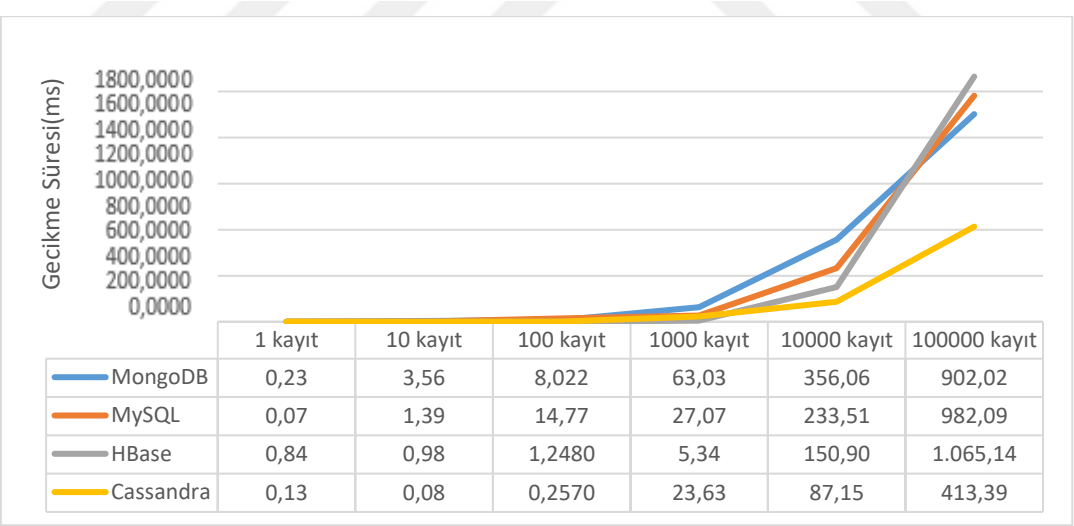
İşletim sistemi	Web Teknolojileri	Çalışma Ortamı	En iyi gecikme süresine sahip veritabanı	En kötü gecikme süresine sahip veritabanı
Windows	Node.Js	Bulut Sistemi	Cassandra	MongoDB
Windows	Php	Bulut Sistemi	Cassandra	MongoDB

4.2.4. Kayıt Silme İşlemleri

Kayıt silme işlemlerine ait sonuçların bulunduğu grafiklere bu bölümde yer verilmiştir.

4.2.4.1. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.25. incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.

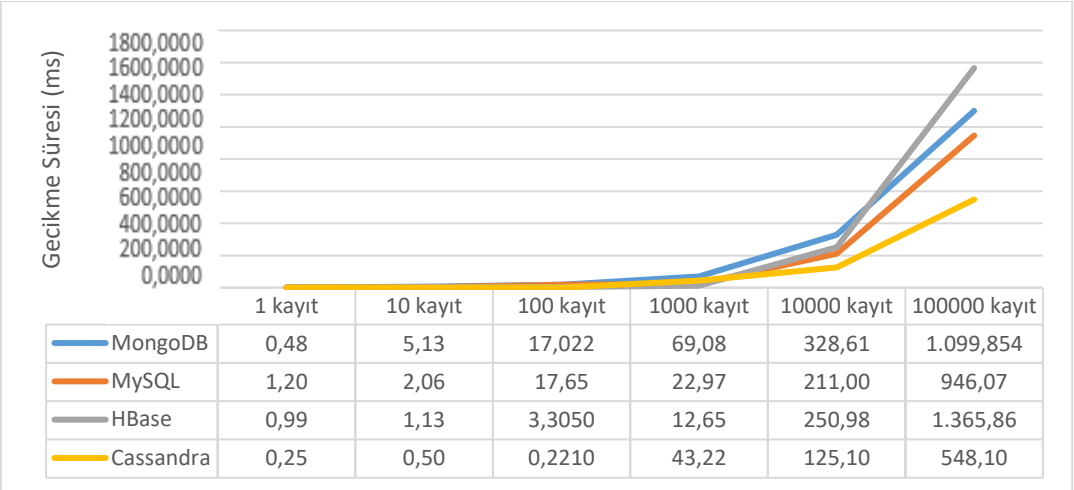


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,23	3,56	8,022	63,03	356,06	902,02
MySQL	0,07	1,39	14,77	27,07	233,51	982,09
HBase	0,84	0,98	1,2480	5,34	150,90	1.065,14
Cassandra	0,13	0,08	0,2570	23,63	87,15	413,39

Şekil 4.25. Yerel bilgisayarda Windows-Php'de kayıt silme gecikme süreleri.

4.2.4.2. *Ubuntu İşletim Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.26. incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.

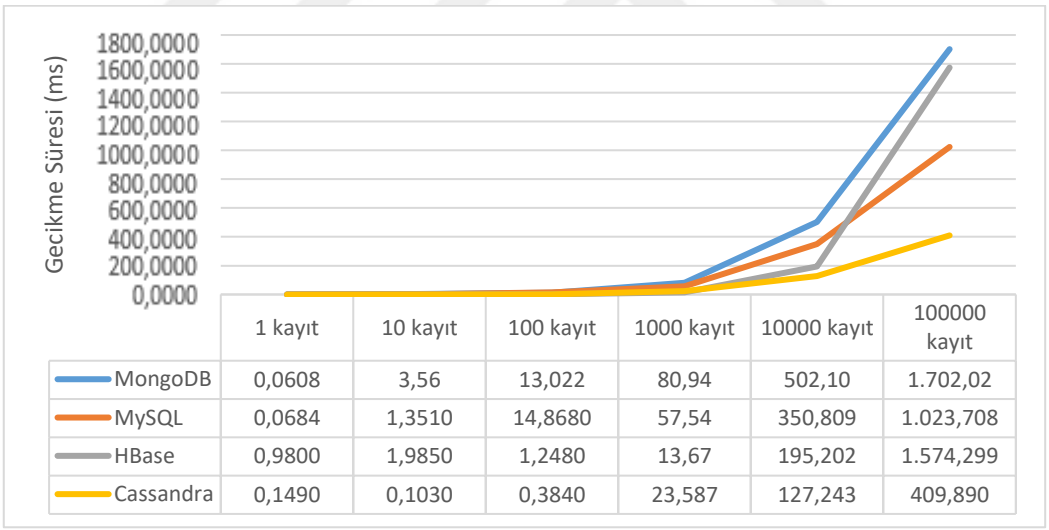


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,48	5,13	17,022	69,08	328,61	1.099,854
MySQL	1,20	2,06	17,65	22,97	211,00	946,07
HBase	0,99	1,13	3,3050	12,65	250,98	1.365,86
Cassandra	0,25	0,50	0,2210	43,22	125,10	548,10

Şekil 4.26. Yerel bilgisayarda Ubuntu-Php'de kayıt silme gecikme süreleri.

4.2.4.3. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.27. incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. MongoDB veritabanı en yüksek gecikme sürelerini vermiştir.

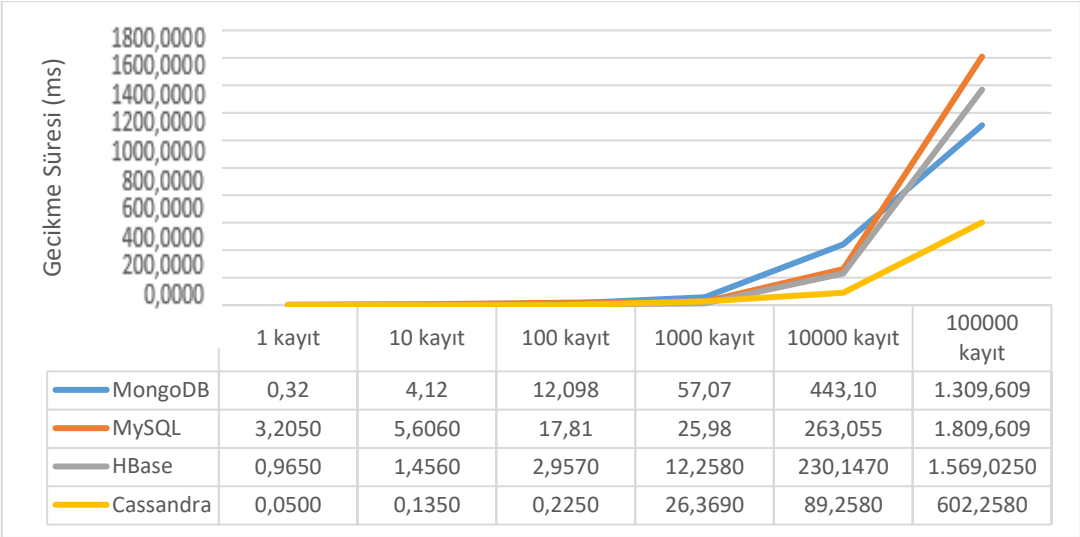


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0608	3,56	13,022	80,94	502,10	1.702,02
MySQL	0,0684	1,3510	14,8680	57,54	350,809	1.023,708
HBase	0,9800	1,9850	1,2480	13,67	195,202	1.574,299
Cassandra	0,1490	0,1030	0,3840	23,587	127,243	409,890

Şekil 4.27. Yerel bilgisayarda Windows-Node.Js'de kayıt silme gecikme süreleri.

4.2.4.4. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Yerel*

Şekil 4.28 incelendiğinde bakıldığında Cassandra veritabanının en düşük sonuçları verdiği, MySQL veritabanının ise en yüksek gecikme sürelerine sahip olduğu görülmektedir. Diğer veritabanları ise birbirine yakın sonuçlar vermiştir.

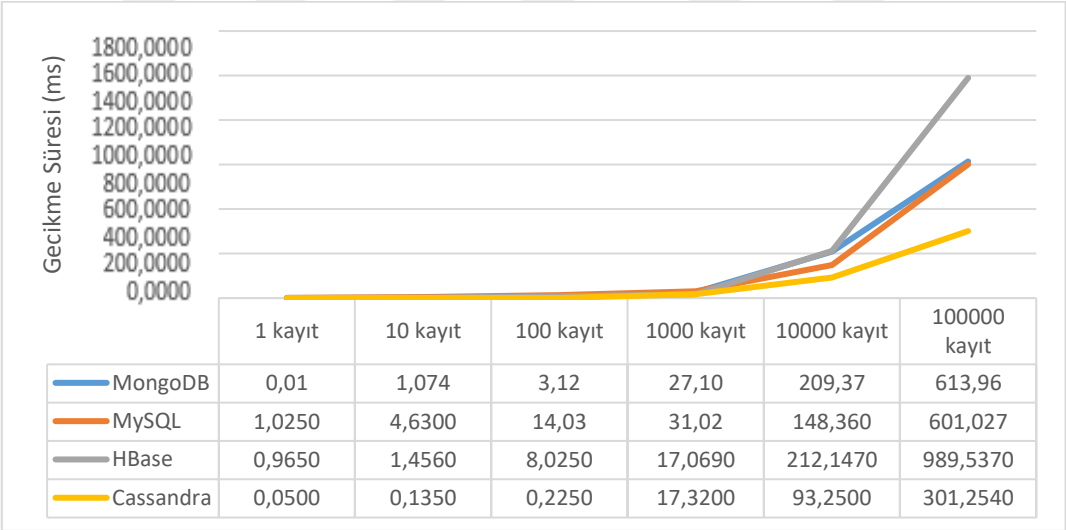


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,32	4,12	12,098	57,07	443,10	1.309,609
MySQL	3,2050	5,6060	17,81	25,98	263,055	1.809,609
HBase	0,9650	1,4560	2,9570	12,2580	230,1470	1.569,0250
Cassandra	0,0500	0,1350	0,2250	26,3690	89,2580	602,2580

Şekil 4.28. Yerel bilgisayarda Windows-Php'de kayıt silme gecikme süreleri

4.2.4.5. *Ubuntu İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.29 incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.

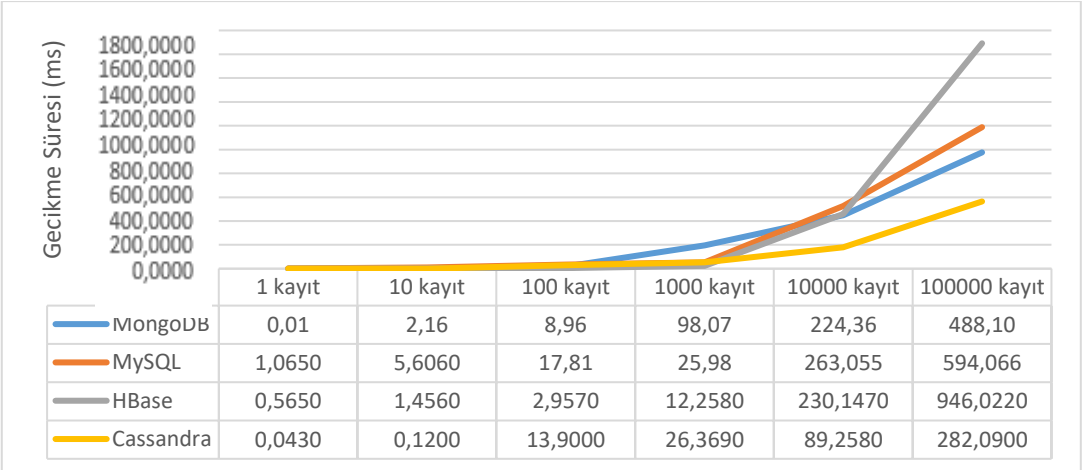


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,01	1,074	3,12	27,10	209,37	613,96
MySQL	1,0250	4,6300	14,03	31,02	148,360	601,027
HBase	0,9650	1,4560	8,0250	17,0690	212,1470	989,5370
Cassandra	0,0500	0,1350	0,2250	17,3200	93,2500	301,2540

Şekil 4.29. Bulut ortamında Ubuntu-Node.Js'de kayıt silme gecikme süreleri.

4.2.4.6. *Ubuntu İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.30 incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.

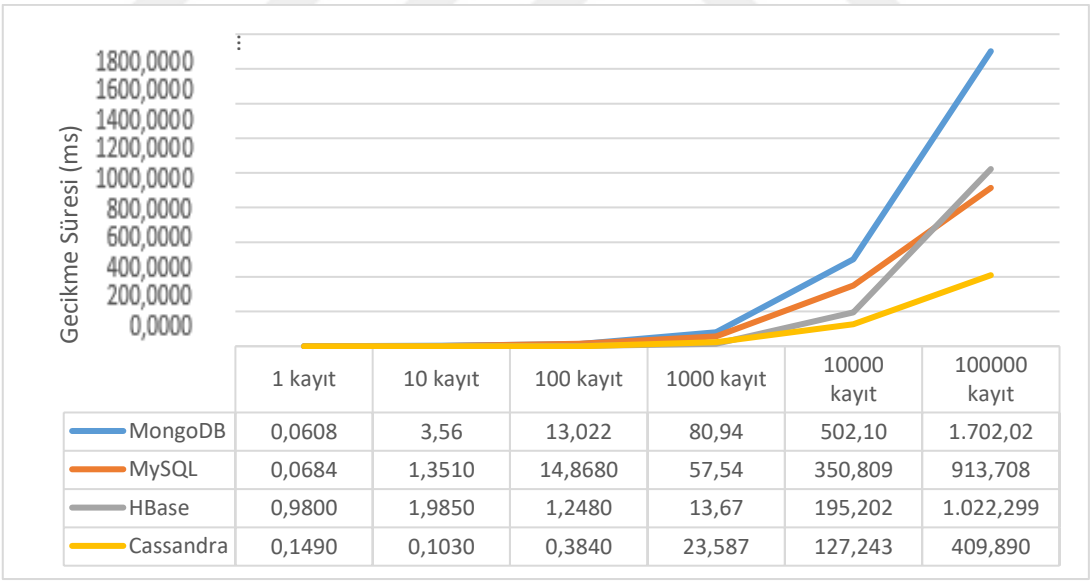


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,01	2,16	8,96	98,07	224,36	488,10
MySQL	1,0650	5,6060	17,81	25,98	263,055	594,066
HBase	0,5650	1,4560	2,9570	12,2580	230,1470	946,0220
Cassandra	0,0430	0,1200	13,9000	26,3690	89,2580	282,0900

Şekil 4.30. Bulut ortamında Ubuntu-Php'de kayıt silme gecikme süreleri.

4.2.4.7. *Windows İşletim Sisteminde Node.Js Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.31 incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.

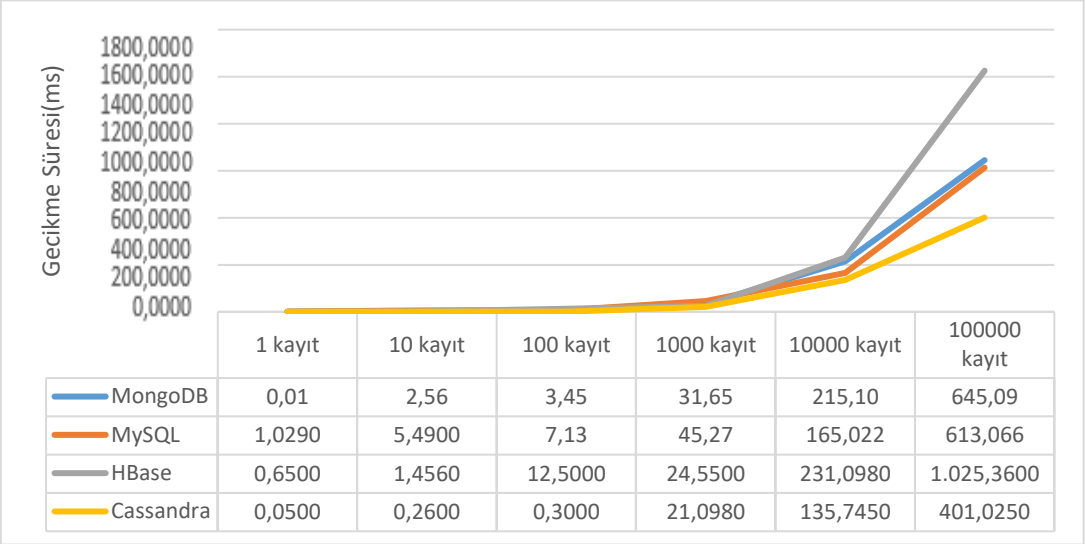


	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,0608	3,56	13,022	80,94	502,10	1.702,02
MySQL	0,0684	1,3510	14,8680	57,54	350,809	913,708
HBase	0,9800	1,9850	1,2480	13,67	195,202	1.022,299
Cassandra	0,1490	0,1030	0,3840	23,587	127,243	409,890

Şekil 4.31. Bulut ortamında Windows-Node.Js'de kayıt silme gecikme süreleri.

4.2.4.8. *Windows İşletim Sisteminde Php Teknolojisi Altındaki Sonuçlar-Bulut*

Şekil 4.32 incelendiğinde Cassandra veritabanına ait gecikme sürelerinin daha düşük olduğu diğer veri tabanlarının ise birbirine yakın sonuçlar ortaya çıkardığı görülmektedir. HBase veritabanı en yüksek gecikme sürelerini vermiştir.



	1 kayıt	10 kayıt	100 kayıt	1000 kayıt	10000 kayıt	100000 kayıt
MongoDB	0,01	2,56	3,45	31,65	215,10	645,09
MySQL	1,0290	5,4900	7,13	45,27	165,022	613,066
HBase	0,6500	1,4560	12,5000	24,5500	231,0980	1.025,3600
Cassandra	0,0500	0,2600	0,3000	21,0980	135,7450	401,0250

Şekil 4.32. Bulut ortamında Windows-Php'de kayıt silme gecikme süreleri.

Kayıt silme işlemleri için yapılan sorgulara ait sonuçlar aşağıda bulunan tablolarda belirtilmiştir. Tabloları baz alarak sonuçları işletim sistemleri değişkenine bakarak yorumladığımızda Ubuntu işletim sisteminin Windows işletim sisteminden daha iyi sonuçlar verdiği fakat aralarında ciddi farkların bulunmadığı görülmektedir.

Tabloları baz alarak web teknolojileri bazında değerlendirildiğinde ise Php ile Node.Js arasında belirgin farklar bulunmamakla beraber Php teknolojisinde daha iyi sonuçların alındığı görülmüştür.

Tabloları baz alarak bir diğer değişken olan bilgisayar sistemleri incelendiğinde Cloud yapısında yapılan testlerin daha iyi sonuçlar verdiği görülmüştür.

Bu bağlamda bakıldığında Cassandra ile yapılan ölçümlerin diğer veritabanı sistemlerine göre daha hızlı olduğu görülmüştür. Yapılan testlerde diğer veritabanları arasında çok belirgin farklar olmamakla beraber HBase veritabanının silme testlerinin genelinde diğer veritabanlarına göre daha düşük bir performans verdiği görülmektedir. Çizelge 4.8'de Kayıt silme senaryosuna it bilgilerin genel değerlendirilmesi bulunmaktadır.

Çizelge 4.9. Kayıt silme işleminin genel değerlendirmesi.

İşletim sistemi	Web Teknolojileri	Çalışma Ortamı	En iyi gecikme süresine sahip veritabanı	En kötü gecikme süresine sahip veritabanı
Ubuntu	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Ubuntu	Php	Yerel Bilgisayar	Cassandra	HBase
Windows	Node.Js	Yerel Bilgisayar	Cassandra	MongoDB
Windows	Php	Yerel Bilgisayar	Cassandra	MySQL
Ubuntu	Node.Js	Bulut Sistemi	Cassandra	HBase
Ubuntu	Php	Bulut Sistemi	Cassandra	HBase
Windows	Node.Js	Bulut Sistemi	Cassandra	MongoDB
Windows	Php	Bulut Sistemi	Cassandra	HBase

5. TARTIŞMA

Bölüm 4'de yapılan deney sonuçları baz alınıp ortaya çıkan sonuçlar yorumlandığında işletim sistemleri özelinde bakıldığı zaman, Ubuntu işletim sistemi ve Windows işletim sistemi ile yapılan test sonuçlarına göre genel manada, Ubuntu işletim sistemiyle alınan sonuçların Windows işletim sistemiyle alınan sonuçlardan daha iyi olduğu saptanmaktadır. İşletim sistemi parametresi bazında bakıldığında her ne kadar Ubuntu işletim sistemi daha iyi sonuçlar verse de performans bazında alınan sonuçlarda ciddi farklılıkların bulunmadığı saptanmıştır.

Bir diğer parametre olan Web teknolojisi bazında bakıldığında ise Php web programlama teknolojisinin genel manada testlerde Node.Js teknolojisine göre daha başarılı olduğu söylenebilir. Oluşan fark genel manada değerlendirilmiş olup performanslar arasında ciddi farklar oluşmadığı görülmektedir.

Bir diğer değişken olan çalışma ortamı özelinde sonuçlara bakıldığında ise Bulut ortamında yapılan deneylerin yerel ortamda yapılan deneylere göre pozitif anlamda önemli farklılıklar oluşturduğu görülmektedir.

Veri hacmi açısından bakıldığında genel olarak on bin kayıta kadar yakın sonuçların alındığı on bin kayıt ve üstünde ise farklılaşmaların başladığı görülmektedir.

Yapılan deneylerde temel veri tabanı işlemleri olan ve CRUD olarak adlandırılan okuma, ekleme, güncelleme ve silme işlemleri yapılmıştır. CRUD işlemleri bazında sonuçlar değerlendirildiğinde kayıt okuma işleminde en iyi performans gecikmesinin MongoDB, en kötü performans gecikmesinin ise HBase veritabanına ait olduğu görülmektedir.

Kayıt ekleme işlemi bazında bakıldığında yapılan deney sonuçlarına göre en iyi performans gecikmesinin Cassandra veritabanına, en kötü performans gecikmesinin ise MongoDB veritabanına ait olduğu görünmektedir.

Kayıt güncelleme işlemi bazında bakıldığında yapılan deney sonuçlarına göre de en iyi performansın Cassandra veritabanına en kötü performansın ise MongoDB veritabanına ait olduğu görünmektedir.

Kayıt silme bazında bakıldığında ise en iyi performans gecikmesinin Cassandra

veritabanına ait olduğu görünmektedir. Performans gecikmesinin en kötü olduğu veritabanına bakıldığında ise Çizelge 4.8'de de görüldüğü üzere HBase, Cassandra ve MySQL veritabanlarının kötü performanslar aldığı sonuçlarla karşılaşılmış olmakla beraber birbirine yakın sonuçlar alındığı gözlemlenmiştir.

6. SONUÇLAR VE ÖNERİLER

İşletim sistemi, web teknolojileri ve çalışma ortamı parametreleri baz alınarak yapılan deneylerde veri hacmi 10'un katları şeklinde arttırılmış ve yüz bin veride sınırlandırılmıştır. İşletim sistemi olarak Ubuntu ve Windows, web teknolojisi olarak Node.Js ve Php, çalışma ortamı olarak ise yerel bilgisayar ve Google Cloud ortamı kullanılmıştır. Deneylerde ilişkisel veritabanı olarak MySQL NoSQL veritabanı olarak ise Cassandra, HBase ve MongoDB kullanılmıştır. Cassandra, dağıtık mimarisi ve yatay ölçeklenebilirliği sayesinde yüksek okuma hızları sağlayabilmektedir.

Veritabanı performanslarını ölçmek için temel veritabanı işlemleri olan ve CRUD olarak bilinen kayıt okuma, kayıt ekleme, kayıt güncelleme ve kayıt silme işlemleri uygulanmış ve deney sonuçları bölüm 5'de açıklanmıştır. Buna göre kayıt okuma senaryolarının olduğu bir çalışmada MongoDB, kayıt ekleme, kayıt güncelleme ve kayıt silme senaryolarının olduğu bir çalışmada ise Cassandra veritabanı performans gecikmesi baz alındığında öne çıkmaktadır. Yapılan çalışmalarda genellikle on bir veriye kadar ciddi farklılıklar oluşmamıştır bu sebeple veri miktarının nispeten küçük olduğu uygulamalarda ya da çalışmalarda ilişkisel veritabanları kullanımı da mantıklı bir seçene olacaktır. Yine aynı senaryoya göre özellikle ciddi bir maliyet kalemi olan bulut sistemleri yerine alternatif sistemler de kullanılabilir.

Sonuç olarak, bu çalışma farklı veritabanı sistemlerinin işletim sistemleri, çalışma ortamları ve web teknolojileriyle olan etkileşimlerini inceleyerek performanslarını karşılaştırmıştır. Çalışmanın sonuçlarına göre, veritabanı performansı çeşitli faktörlere bağlı olarak değişebilmektedir ve farklı iş yükleri için farklı veritabanı sistemleri daha uygun olabilmektedir. İşletim sistemi seçimi ve çalışma ortamı da performans üzerinde belirli bir etkiye sahip olabilir. Bu sonuçlar, veritabanı sistemleri ve ilişkili teknolojilerle çalışan geliştiriciler ve sistem yöneticileri için önemli bir bilgi kaynağı olabilir.

6.1. GELECEK ÇALIŞMALAR

Bu çalışma, ilişkisel ve NoSQL veritabanlarının işletim sistemleri, çalışma ortamları ve

web teknolojileriyle olan performans etkileşimlerini incelemekte ve karşılaştırmaktadır. Elde edilen sonuçlar, veritabanı performansı üzerinde etkili olan çeşitli faktörleri ortaya koymaktadır. Ancak, bu çalışma kapsamında daha ileri araştırmaların yapılmasına olanak sağlayacak bazı potansiyel araştırma alanları mevcuttur. Bu çalışmada, veritabanı performansını etkileyen faktörler genel bir bakışla ele alınmıştır. Gelecekteki çalışmalarda, veri indeksleme yöntemlerinin daha ayrıntılı bir analizi yapılabilir ve farklı veri parçalama stratejilerinin performans üzerindeki etkisi araştırılabilir. Ayrıca, veri tabanı optimizasyonu için yeni algoritmalar ve yöntemler geliştirilebilir.

Dağıtık veritabanı sistemleri, büyük ölçekli veri işleme ve ölçeklenebilirlik sağlama konusunda önemli bir rol oynamaktadır. Gelecekteki çalışmalarda, dağıtık veritabanı sistemlerinin performansı ve veri parçalama stratejileri üzerine odaklanılabilir. Ayrıca, veri tutarlılığı ve veri bütünlüğü gibi kritik konuları ele alan yeni dağıtık veritabanı protokolleri ve algoritmaları geliştirilebileceğinden yapılan deneyler bu durumlarda farklı sistemlerle tekrarlanabilir.

Bulut tabanlı veritabanı hizmetleri, günümüzde popüler bir seçenek haline gelmiştir. Gelecekteki çalışmalarda, farklı bulut sağlayıcılarının veritabanı hizmetlerinin performansı daha detaylı bir şekilde karşılaştırılabilir. Ayrıca, bulut tabanlı veritabanı hizmetlerinin ölçeklenebilirlik, güvenlik ve maliyet-etkinlik açılarından incelenmesi gerekmektedir.

Büyük veri analitiği, günümüzde birçok alanda önemli bir rol oynamaktadır. Gelecekteki çalışmalarda, büyük veri analitiği için kullanılan veritabanı sistemlerinin performansını artırmak için yeni yöntemler ve teknikler üzerine odaklanılabilir. Ayrıca, veri sorgulama ve analiz süreçlerini hızlandırmak için paralel veri işleme teknikleri ve dağıtık hesaplama algoritmaları araştırılabilir.

Bu çalışmada, farklı web teknolojilerinin veritabanı performansı üzerinde önemli bir fark yaratmadığı sonucuna varılmıştır. Ancak, gelecekteki çalışmalarda daha geniş bir web teknolojileri yelpazesinin değerlendirilmesi ve performans karşılaştırmalarının yapılması önerilebilir. Örneğin, Python Django, Ruby on Rails gibi popüler web çerçeveleri ve veritabanı ile entegrasyonlarının performansı daha ayrıntılı olarak incelenebilir.

7. KAYNAKÇA

[1] Anonim, (Ocak, 2023). *World ınternet usage and populatıon statıstıcs 2023 year estimates.* [Online]. Erişim:https://www.internetworldstats.com/stats.htm.

[2] Ö. Akbulut, A. Kaygısız, & İ. Yılmaz, "A Comparative Research on Data Analysis with Factorial Anova, Logistic Regression and Chaid Classification Tree Methods," *Black Sea Journal Agriculture*, c. 1, sayı. 2, ss. 314–322, 2022.

[3] P. E. Ceruzzi," A History of Modern Computing," USA:The MIT Press, 2023, ss. 75.

[4] Y. Gökşen & H. Aşan, "Veri Büyüklüklerinin Veritabanı Yönetim Sistemlerinde Meydana Getirdiği Değişim: NOSQL," *Bilişim Teknoloji Dergisi*, c. 8, sayı. 3, ss. 125-132, 2015.

[5] E.Aktan, "Büyük Veri: Uygulama Alanları, Analitiği ve Güvenlik Boyutu," *Bilgi Yönetimi Dergisi*. c. 1, sayı. 1, ss. 1-22, 2018.

[6] G. D. Samaraweera & J. M. Chang, "Security and privacy implications on database systems in big data era: A survey," *Ieee Transactions On Knowledge And Data Engineering*, c. 33, sayı. 1, ss. 239–258, 2021.

[7] Anonim, (Kasım, 2022). *Database What Is Database ?,* [Online] Erişim= https://www.oracle.com/tr/database/what-is-database/

[8] V. D. Jogi & A. Sinha, "Performance evaluation of MySQL, Cassandra and HBase for heavy write operation," *International Conference on Recent Advances in Information Technology*, c. 3, sayı. 1, ss. 586–590, 2016.

[9] E. S. Pramukantoro, D. Primanita Kartikasari & R. A. Siregar, "Performance evaluation of MongoDB, Cassandra, and HBase for heterogenous IoT data storage," *2nd International Conference on Applied Information Technology and Innovation.*, 2019, doi: 10.1109/ICAITI48442.2019.8982159.

[10] A. Oracle & W. Paper,(2022, 1 Kasım). *Oracle NoSQL Database,*. [Online]. Erişim:http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-wp-1436762.pdf

[11] S. Sotnik, V. Manakov, & V. Lyashenko, "Overview: PHP and MySQL Features for Creating Modern Web Projects," *International Journal of Academic Information Systems Research.*, c. 7 sayı. 1 ss.11-17., 2023.

[12] S. Gülburun & M. Dener, "Bulut Bilişim Güvenliğindeki Zorluklar ve Güncel Çalışmalar Üzerine Bir İnceleme," *Bilişim Teknolojileri. Dergisi.*, c. 15, sayı. 1, ss. 45–53, 2022.

[13] S. Öztürk & H. E. Atmaca, "İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi," *Bilişim Teknolojileri. Dergisi.*, c. 10 sayı.1 ss. 199–199, 2017.

[14] K. Eren, "Bulut Bilişim Teknolojileri Ve Nosql Veritabanları Kullanarak Türkiye'de Terör Olaylarının İncelenmesi," Yüksek lisans tezi, Yönetim Bilişim Sistemleri, Bilgisayar Mühendisliği Bölümü, Sakarya Üniversitesi, Sakarya, Türkiye, 2017.

[15] N. Anjum & S. Alam, "A Comparative Analysis on Widely used Web Frameworks to Choose the Requirement based Development Technology (*Iarjset)*," c. 6, sayı. 9, ss. 16–24, 2019.

[16] T. Capris, P. Melo, N. M. Garcia, I. M. Pires, & E. Zdravevski, "Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation," *2022 International Conference on Data Analytics for Business and Industry, 2022,* doi 10.1109/ICDABI56818.2022.10041513

[17] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, & R. Győrödi, "A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management," *Big Data Cogn. Comput.*, c. 6, sayı. 2, ss. 1-19 2022

[18] B. Dumanlı, "En Çok Kullanılan İlişkisel ve NoSQL Veritabanı Yönetim Sistemlerinin Performans Karşılaştırılması, Bilgisayar Mühendsiliği, "Trakya Üniversitesi,Edirne,Türkiye, 2019.

[19] S. Palanisamy & P. Suvithavani, "A survey on RDBMS and NoSQL Databases MySQL vs MongoDB," *2020 International. Conference. Computing. Communication. Informatics,* 2020, doi: 10.1109/ICCCI48352.2020.9104047.

[20] C. Çelik,S.Çakır & N.Yalçın, "Bulut Tabanlı Bir Mesajlaşma Uygulaması Tasarımı ve Gerçekleştirimi,"*3rd International Conference on Applied Engineering and Natural Sciences,* c. 3, sayı. ss. 975–979, 2022.

[21] M. Samar & H. Hassan, "Comparison between Amazon S3 and Google Cloud Drive," *2nd International Conference on Communication and Information Systems,* 2017, doi: 10.1145/3158233.3159371.

[22] A. D. D. Erazo, M. R. M. Morales, V. K. P. Chávez, and S. L. M. Cardoso, "Comparative Analysis of performance for SQL and NoSQL Databases," *17th Iberian Conference on Information Systems and Technologies.*, 2022, doi: 10.23919/CISTI54924.2022.9820292.

[23] C. Coronel, & S.Morris &P. Rob, *Database Systems (9th Edition)*,Boston:Cengage Learning, 2010, ss. 724.

[24] B. Gültekin, S. Biroğul, & ;İ.Yücedağ, "İşe Alım Süreci Aday Ön Tesbitinde Bulanık Mantık Tabanlı SQL Sorgulama Yönteminin İncelenmesi," *Düzce Üniversitesi Bilim ve Teknol. Dergisi,* c. 3, sayı. 1, ss. 198–209, 2015.

[25] B. Namdeo & U. Suman, "A Model for Relational to NoSQL Database Migration: Snapshot-Live Stream DB Migration Model," *7th International Conference on Advanced Computing and Communication Systems,* 2021, doi: 10.1109/ICACCS51430.2021.9441829.

[26] A. Krechowicz, S. Deniziak, & G. Lukawski, "Highly Scalable Distributed Architecture for NoSQL Datastore Supporting Strong Consistency," *Institute of Electrical and Electronics Engineers Access,* c. 9, sayı.1, ss. 69027–69043, 2021.

[27] Y. Ceran & M. Bezirci, "A Strategical Approach to Relations Between Marketing Information System - Accounting Information System: Strategical Marketing Accounting," *Selçuk Üniversitesi Sosyal. Bilimler Enstitüsü Dergisi.*, c. 1, sayı. 26, ss. 103–115, 2011.

[28] A. Y. Muhammad & F. N. Azizah, "Conversion of Entity-Relationship Model to NoSQL Document-Oriented Database Logical Model Using Workload Information and Entity Update Frequency," *9th International Conference on Advanced Informatics Concepts Theory and Applications,* 2022, doi: 10.1109/ICAICTA56449.2022.9932986

[29] S. Eken, F. Kaya, A. Sayar, & A. Kavak, "Doküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB yatay ölçeklenebilirlik karşılaştırması," *7. Mühendislik ve Teknoloji. Sempozyumu*, c. 7, sayı. 1, ss. 1–7, 2014.

[30] Anonim, (Şubat,2023), "*MySQL: Why My?*," [Online]. Erişim: https://vertabelo.com/blog/mysql-history/

[31] M. Dawodi, M. H. Hedayati, J. A. Baktash, & A. L. Erfan, "Facebook MySQL Performance vs MySQL Performance," 1*0th Annual Information Technology, Electronics and Mobile Communication Conference*,2019, doi: 10.1109/IEMCON.2019.8936259.

[32] A. Lith & J. Mattsson, "Investigating storage solutions for large data," Master thesis, Department of computer science and engineering, Chambers university, Sweeden, 2010.

[33] M. Deepika, V. Shetty, M. Sana, & J. Chidimar, "Comparative Study of SQL and NoSQL Databases to Evaluate Their Suitability For Big Data Application," *International Journal of Computer Science and Information Technology Research.*, c. 4, sayı. 2, ss. 314–318, 2016.

[34] M. Arslan & F. Kayaalp, "Web Teknolojileri Alt Yapısını Kullanarak İlişkisel Ve İlişkisel Olmayan Veri Tabanlarının Performanslarının İncelenmesi, *2.Uluslararası Bilimsel Gelişmeler Kongresi*," c.2, sayı.1, ss. 486-494, 2022.

[35] Anonim, (Mart, 2023), "*Document Databases in NoSQL"* [Online]. Erişim:. https://www.geeksforgeeks.org/document-databases-in-nosql/

[36] Anonim, (Mart, 2023), "What is *MongoDB*," .[Online]. Erişim: https://www.mongodb.com/what-is-mongodb

[37] Anonim, (Mart, 2023), "*Key-Value Data Model in NoSQL*," 2022, [Online]. Erişim: https://www.geeksforgeeks.org/key-value-data-model-in-nosql/

[38] P. Menon, T. M. Qadah, T. Rabl, M. Sadoghi, & H. A. Jacobsen, "LogStore: A Workload-Aware, Adaptable Key-Value Store on Hybrid Storage Systems," I*nternational Journal of Computer Science and Information Technology*., c. 34, sayı. 8, ss. 3867–3882, 2022.

[39] Anonim, (Nisan, 2023), "W*elcome to Apache HBase*," [Online]. Erişim: https://hbase.apache.org/

[40] Anonim, (Nisan, 2023), "*About Node.js*," [Online]. Erişim: https://nodejs.org/en/about

[41] Anonim, (Nisan, 2023), *"About Php"*, [Online]. Erişim: https://www.php.net/manual/tr/preface.php

[42] Anonim, (Nisan, 2023), *"What Is The Cloud Computing ? ,"* [Online]. Erişim: https://cloud.google.com/learn/what-is-cloud-computing

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Mehmet ARSLAN

Yabancı Dili : İngilizce

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Yüksek Lisans	Elektrik Elektronik ve Bilgisayar Mühendisliği	Düzce Üniversitesi	2023
Lisans	Bilgisayar Öğretmenliği.	Mersin Üniversitesi	2011
Lisans	Kamu Yönetimi.	Anadolu Üniversitesi	2009
Ön Lisans	Bilgisayar Teknolojileri ve Programlama.	Korkut Ata Üniversitesi	2008
Lise	Bilişim Teknolojileri.	75.Yıl D.M.O Anadolu Bilgisayar Teknik Lisesi	2003

YAYINLAR

1. M. Arslan & F. Kayaalp, "Web Teknolojileri Alt Yapısını Kullanarak İlişkisel Ve İlişkisel Olmayan Veri Tabanlarının Performanslarının İncelenmesi", *2.Uluslararası Bilimsel Gelişmeler Kongresi(ICONSAD'22),*" vol.2, c.1, pp. 486-494, 2022.