T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

# YÜKSEK LİSANS TEZİ

# BLOK ZİNCİRİ TEMELLİ VERİ TABANLARININ WEB UYGULAMALARINDA KULLANIMININ ANALİZİ

Orhun Ceng BOZO

DANIŞMAN
Doç. Dr. Rüya ŞAMLI

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2021

Bu çalışma, [25.08.2021] tarihinde aşağıdaki jüri tarafından [Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programı]nda [Yüksek Lisans tezi] olarak kabul edilmiştir.

**Tez Jürisi**

[Doç. Dr.] [Rüya ŞAMLI](Danışman)
[İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi]

[Dr. Öğr. Üyesi] [Emel ARSLAN]
[İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi]

[Dr. Öğr. Üyesi] [Mehmet Ali ERTÜRK]
[İstanbul Üniversitesi
Ulaştırma ve Lojistik Fakültesi]

[Unvan] [Adı SOYADI]
[Üniversite
Fakülte]

[Unvan] [Adı SOYADI]
[Üniversite
Fakülte]

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

# ÖNSÖZ

Yüksek lisans tez çalışmalarım boyunca bana yol gösteren, bilgi ve desteğini esirgemeyen değerli hocam ve tez danışmanım Doç. Dr. Rüya ŞAMLI'ya,

Yüksek lisans eğitimim sırasında tecrübe ve bilgileriyle çalışmalarıma katkı sağlayan diğer hocalarıma başta Doç. Dr. Muhammed Ali Aydın olmak üzere,

Ayrıca sabır ve destekleriyle her zaman yanımda olan aileme ve hayat arkadaşıma

Sonsuz teşekkürler.

Ağustos 2021 Orhun Ceng BOZO

# İÇİNDEKİLER

**Sayfa No**

**ÖNSÖZ** ..... **iv**
**İÇİNDEKİLER** ..... **v**
**ŞEKİL LİSTESİ** ..... **vii**
**TABLO LİSTESİ** ..... **viii**
**SİMGE VE KISALTMA LİSTESİ** ..... **ix**
**ÖZET** ..... **xi**
**SUMMARY** ..... **xii**
**1. GİRİŞ** ..... **1**
**2. GENEL KISIMLAR** ..... **3**
2.1 WEB UYGULAMALARI ..... 3
2.1.1 Web Tarayıcı ..... 3
2.1.2 Web Uygulamalarının Yapısal Modeli ..... 7
2.2 VERİ TABANLARI ..... 8
2.2.1 Geleneksel Dosya İşleme Sistemleri ..... 8
2.2.2 Veri Tabanı Yaklaşımı ..... 8
2.2.3 İlişkisel Veri Tabanları ..... 9
*2.3.3.1 Tablo Yapısı* ..... *9*
*2.3.3.2 SQL* ..... *10*
*2.3.3.3 İlişkisel Veri Tabanı Yönetim Sistemi* ..... *11*
2.2.4 İlişkisel Olmayan Veri Tabanları ..... 13
2.3 BLOK ZİNCİRİ ..... 15
2.3.1 Blok Zinciri ve Veri ..... 16
2.3.2 Blok Zinciri Türleri ..... 16
*2.3.2.1 Açık (İzin Gerektirmeyen) Blok Zincirleri* ..... *16*
*2.3.2.2 Gizli (İzinli) Blok Zincirleri* ..... *16*
2.3.3 Blok Zinciri Tabanlı Web Uygulamaları ..... 16
2.3.4 Blok Zinciri Mutabakat Algoritmaları ..... 17
*2.3.4.1 Proof of Work (Emek Kanıtı)* ..... *18*
*2.3.4.2 Proof of Stake (Hisse Kanıtı)* ..... *18*
*2.3.4.3 Diğer Mutabakat Algoritmaları* ..... *18*

2.3.5 Bizans Generalleri Problemi ..... 18
**3. MALZEME VE YÖNTEM ..... 20**
3.1 SQL TABANLI UYGULAMA ..... 20
3.2 NOSQL TABANLI UYGULAMA ..... 22
3.3 BLOK ZİNCİR TABANLI UYGULAMA ..... 22
3.4 VERİ TABANLARININ KARŞILAŞTIRILMASI ..... 23
3.4.1 Kayıt Ekleme ..... 23
3.4.2 Kayıt Sorgulama ..... 25
3.4.3 Performans Analizi İçin Zaman Ölçümü ..... 25
**4. BULGULAR ..... 26**
4.1 İŞLEVSELLİK ..... 26
4.1.1 Kayıt Ekleme ..... 26
4.1.2 Kayıt Sorgulama ..... 29
4.2 YAZILIM GELİŞTİRME ..... 30
4.3 GÜVENLİK ..... 31
**5. TARTIŞMA VE SONUÇ ..... 33**
**KAYNAKLAR ..... 35**

# ŞEKİL LİSTESİ

**Sayfa No**

**Şekil 2.1:** TCP-TLS anlaşması sırasındaki veri alışverişi (MDN Web Docs, 2021) ..... 5

**Şekil 2.2:** DOM örnek eleman ağacı yapısı (MDN Web Docs, 2021) ..... 6

**Şekil 2.3:** Web Uygulamaları Yapısal Modeli ..... 7

**Şekil 2.4:** SQL ile örnek bir sorgu ..... 10

**Şekil 2.5:** İlişkisel veri tabanı yönetim sisteminin temel yapısı (Andreas Meier, 2019) ..... 11

**Şekil 2.6:** Farklı ilişkisel olmayan veri tabanı yapıları (Andreas Meier, 2019) ..... 15

**Şekil 2.7:** Blok zincirinin merkle tree yapısı ..... 15

**Şekil 2.8:** Blok zinciri uygulamasının katmanları ..... 17

**Şekil 3.1:** Sanat galerisi uygulamasının ilişkisel veri tabanı diyagramı ..... 20

**Şekil 3.2:** 150 satırlık sanat galerisi örnek envanter verisi ..... 24

**Şekil 4.1:** SQL sistem ile 150 satırlık örnek verinin eklenmesi işlemi ve süresi ..... 27

**Şekil 4.2:** NoSQL sistem ile 150 satırlık örnek verinin eklenmesi işlemi ve süresi ..... 27

**Şekil 4.3:** Blok zinciri tabanlı sisteme tek girdi eklenmesi ve geçen süre ..... 28

**Şekil 4.4:** Starport config.yml dosyası ..... 31

# TABLO LİSTESİ

**Sayfa No**

**Tablo 3.1:** SQL, NoSQL ve Blok zinciri kayıt arama sorguları ..... 25

**Tablo 4.1:** SQL, NoSQL ve Blok zinciri tabanlı sistemlerde sorguların tamamlanması için geçen süreler ..... 29

**Tablo 4.2:** SQL, NoSQL ve Blok zinciri tabanlı sistemlerde sorguların tamamlanması için geçen süreler ..... 29

# SİMGE VE KISALTMA LİSTESİ

| Kısaltmalar | Açıklama |
|---|---|
| API | : Uygulama proglama arayüzü (Application programming interface) |
| ACID | : Bölünemez olmak, tutarlılık, yalıtım, dayanıklılık (Atomicity, consistency, isolation, durability) |
| BFT | : Bizans hata toleransı (Byzantine fault tolerance) |
| CAP | : Tutarlılık, erişilebilirlik, ağ sorunlarının üstesinden gelebilme kabiliyeti (Consistency, availability, partition tolerance) |
| CPU | : Merkezi işlem birimi (Central processing unit) |
| CRUD | : Oluştur, oku, güncelle, sil (Create, read, update, delete) |
| CSS | : Basamaklı stil şablonları (Cascading style sheets) |
| CSSOM | : Basamaklı stil şablonları nesne modeli (Cascading style sheets object model) |
| CSV | : Virgülle ayrılmış değerler (Comma seperated values) |
| DCS | : Merkeziyetsiz, tutarlı, ölçeklenebilir (Decentralized, consistent, scalable) |
| DNS | : Alan isimlendirme sistemi (Domain name system) |
| DOM | : Doküman nesne modeli (Document object model) |
| GB | : Gigabyte |
| HTML | : Hiper metin işaretleme dili (Hyper text markup language) |
| HTTP | : Hiper metin transfer protokolü (Hyper text transfer protocol) |
| IP | : İnternet protokolü |
| JSON | : Javascript nesne gösterimi (Javascript object notation) |
| MB | : Megabyte |

| | |
|---|---|
| NoSQL | : İlişkisel olmayan sorgulama dili<br>(Non-relational structured query language) |
| PoS | : Hisse kanıtı<br>(Proof of stake) |
| PoW | : Emek kanıtı<br>(Proof of work) |
| SQL | : Yapısal sorgulama dili<br>(Structured query language) |
| TCP | : İletişim kontrol protokolü<br>(Transmission control protocol) |
| TLS | : Taşıma katmanı güvenliği<br>(Transport layer security) |
| URL | : Tekdüzen kaynak konumlayıcı<br>(Uniform resource locator) |
| WWW | : Dünya çapında ağ<br>(World wide web) |

# ÖZET

## YÜKSEK LİSANS TEZİ

## BLOK ZİNCİRİ TEMELLİ VERİ TABANLARININ WEB UYGULAMALARINDA KULLANIMININ ANALİZİ

**Orhun Ceng BOZO**

**İstanbul Üniversitesi-Cerrahpaşa**

**Lisansüstü Eğitim Enstitüsü**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Danışman: Doç. Dr. Rüya ŞAMLI**

Bu çalışmada web uygulamalarında ilişkisel, ilişkisel olmayan veri tabanları ve blok zinciri tabanlı veri tabanlarının işlevleri karşılaştırılmıştır. Farklı yeteneklere sahip bu sistemlerin kullanıcılara yansıyan performans farklılıklarının olup olmadığı, kullanıcı ihtiyaçlarını nasıl karşıladığı, hangi güvenlik konusundaki farklılıkları ve blok zinciri tabanlı veri tabanı sistemiyle web uygulaması geliştirilirken karşılaşılan durumlar değerlendirilmiştir.

Blok zinciri teknolojilerinin türleri ve algoritma seçiminin yansımaları, web uygulamalarında işlevsellik, güvenlik, performans açısından analiz edilmiştir. Blok zinciri temelli veri tabanı sisteminin geleneksel veri tabanlarının temel işlevlerini bazı yolları dayatarak da olsa yerine getirdiği gözlenmiştir. Blok zinciri temelli veri tabanı sisteminin çalışırken performans tarafındaki dezavantajları, güvenlik ve sağlamlık tarafındaki yansımaları analiz edilmiştir.

Ağustos 2021, 49 sayfa.

**Anahtar kelimeler:** Veri tabanı, SQL, NoSQL, Blok zinciri, Blok zinciri tabanlı veri tabanı, Web uygulamalarında blok zinciri

# SUMMARY

## M.Sc. THESIS

### ANALYSIS of USING BLOCKCHAIN BASED DATABASES in WEB APPLICATIONS

**Orhun Ceng BOZO**

**Istanbul University-Cerrahpasa**

**Institute of Graduate Studies**

**Department of Computer Engineering**

**Supervisor : Assoc. Prof. Dr. Rüya ŞAMLI**

In this study, the functions of relational and non-relational databases and blockchain-based databases in web applications are compared. Whether these systems with different capabilities have performance differences reflected on users, how they meet user needs, differences in security and situations encountered while developing a web application with a blockchain-based database system are evaluated.

Types of blockchain technologies and reflections of algorithm selection are analyzed in terms of functionality, security, performance in web applications. It has been observed that the blockchain-based database system fulfills the basic functions of traditional databases, albeit by imposing some solutions. The disadvantages of the blockchain-based database system on the performance side, and its reflections on the security and robustness side are analyzed.

August 2021, 49 pages.

**Keywords:** Database, SQL, NoSQL, Blockchain, Blockchain based database, Blockchain in web applications

# 1. GİRİŞ

İlk bilgisayarlarda, o dönemde düşük miktarda olan verileri işlemek ve kaydetmek için dosyalar, manyetik filmler, kasetler, diskler gibi yapılar kullanılmaktaydı. Zaman içerisinde veri miktarı akıl almaz bir hızla büyürken bu yapılar yetersiz gelmeye başlamış ve verilerin yönetildiği sistemler hızlı bir gelişme göstermiştir. Günümüzde kullanılan veri tabanları ve veri tabanı yönetim sistemleri veriler konusundaki bu hıza ayak uydurmaktadır. Yapısal sorgulama dili (structured query language – SQL) yapıdaki veri tabanları web ile bilgisayarlar birbirlerine bağlandığında geliştirilen web uygulamalarının tüm ihtiyaçlarını sağlamıştır. Ardından artan sunucu yetenekleri, ucuzlayan donanım maliyetleri, web 2.0 gibi kavramlar ve artan internet nüfusu ile farklı ihtiyaçlara da cevap veren veri tabanları ve veri yönetim sistemleri de geliştirilmiş ve ilişkisel olmayan sorgulama dili (non-relational structured query language - NoSQL) veri tabanları da yaygın olarak kullanılmaya başlanmıştır. Artan sunucu yeteneklerinin, ucuzlayan donanım maliyetlerinin artan bir hızla ilerleyişi ve finans sektöründe devrime de yol açan Bitcoin makalesi ile blok zinciri teknolojisi ile veri tabanı ve veri tabanı yönetim sistemleri tekrar vites değiştirerek kendini geleceğe hazırlamaya devam etmiştir. Bitcoin'in icadı ile popülerleşen, merkezi bir otorite ihtiyacı olmadan, eşler arası işlemleri sağlayan, verileri dağıtık defterle eşlerde tutan blok zinciri yapıları tüm dünya piyasalarını değişime zorlamıştır. Blok zinciri teknolojisi, çeşitli web uygulamalarında SQL ve NoSQL veri tabanı sistemlerine alternatif olarak zaman zaman da beraber kullanılmaya başlanmıştır.

SQL ve NoSQL tabanlı geleneksel veri tabanı sistemleri; merkezi yapılar, yetki/izin bazlı ve yönetici gereksinimi duyan sistemler sunarken blok zinciri kullanılan sistemler merkeziyetsiz, izne ve yöneticiye gerek duymayan yapılara imkân sağlayarak mevcut teknoloji çözümlerinden kendilerini ayırmış ve blok zinciri teknolojileri yaygınlaştıkça daha çok sektör ve geliştirici tarafından anlaşılmaya, kullanılmaya ve geliştirilmeye başlanmıştır. Bu tez çalışmasında da web uygulaması geliştirirken geleneksel veri tabanı sistemleriyle blok zinciri arasındaki seçimin sonuçları ya da hangi durumlarda birbirilerine göre avantajlı oldukları analiz konusu olarak seçilmiştir.

Literatürde bu tez çalışmasında gerçekleştirilen işlemlere kaynak sağlayan çeşitli çalışmalar mevcuttur:

Coşkun (2019) çalışmasında SQL ve NoSQL veri tabanları karşılaştırılarak performans analizi yapılmış ve çeşitli kompleks sorgularla ilişkisel ve ilişkisel olmayan veri tabanlarının sorgulara ürettiği çıktılar incelenmiştir.

Mayank vd. (2020) çalışmasında, veri tabanı ve blok zincirinin gelişimlerinde trendler, CAP (consistency, availability, partition tolerance - tutarlılık, erişilebilirlik, ağ sorunlarının üstesinden gelebilme kabiliyeti), ACID (atomicity, consistency, isolation, durability – bölünemez olmaz, tutarlılık, yalıtım, dayanıklılık), DCS (decentralized, consistent, scalable – merkeziyetsiz, tutarlı, ölçeklenebilir) gibi teoremler ve merkeziyetsizleşme fikirleri, 20 tane farklı veri tabanı sistemi ile çeşitli başlıklarda karşılaştırılmış ve incelenmiştir.

Blok zinciri mutabakat algoritmalarını inceleyen inceleme çalışmasında Ferdous (2020), blok zincirinde mutabakatın hangi katmanlar arasında oluştuğu, farklı algoritmaların nasıl davrandığı incelenmiştir.

İncelemelerle de elde edilen bilgiyle bu tez çalışmasında blok zinciri kullanan, SQL tabanlı ve NoSQL tabanlı sistemlerle oluşturulmuş üç farklı yapıdaki bir sanat galerisi web uygulamasının işlevleri çeşitli alanlarda karşılaştırılmıştır. Bu karşılaştırma ile blok zincirinin web uygulamalarda veri tabanı sistemi olarak kullanılırken hangi alanlarda işlevsel olduğu, nasıl performans sergilediği gözlenmek istenmiştir. Bir veri tabanı olarak blok zincirinin hangi alanlarda geleneksel teknolojilerin önüne geçtiği hangi alanlarda gelişmesi gerektiği incelenmiştir.

# 2. GENEL KISIMLAR

## 2.1 WEB UYGULAMALARI

Web uygulamaları, her tür web tarayıcısı program sayesinde internet üzerinden kullanılan uygulamalardır. 1994 yılında internet (web) denen ağın oluşması, 1995 yılında da WWW olarak kısaltılan "World Wide Web" (Dünya çapında ağ) adreslemesiyle birbirine bağlanan bilgisayarlar üzerinden web uygulamaları da insanların hayatına girmiştir. Web uygulamaları temelde, internet bağlantısı gerektirmeden bilgisayarların işletim sistemi yardımıyla yerelde (lokalde) çalışan uygulamaların bir sunucu bilgisayarında çalıştırılıp sadece sonuçların web tarayıcılarla alıcılara gönderilmesini ya da uygulama yine lokalde de çalışsa sunucuda tutulan veri tabanı ile birden fazla bilgisayarın (kişinin) web üzerinden aynı veriyi kullanması ve üzerinde çalışma yapmasını sağlamaktadır.

Web uygulamaları istemci-sunucu işlemlerine dayalıdır. Bu işlemler, şu 3 ana kavram üzerinde kurulmalıdır: bir isimlendirme ve referans metodolojisi (URL, uniform resource locator - tekdüzen kaynak konumlayıcı) veri ve bilginin aktarımı için ara yüzün oluşturulacağı bir yazılım dili (HTML, hyper text markup language, hiper metin işaretleme dili), alıcı ve sunucu bilgisayarların bağlantı kurarken kullanacağı bir protokol (HTTP, hyper text transfer protocol, hiper metin transfer protokolü), (Jazayeri, 2007).

### 2.1.1 Web Tarayıcı

Bilgisayarlardaki dosyalara grafik ara yüz üzerinde erişmek için dosya gezgini uygulamasından faydalanılır. Bu ara yüz ile bilgisayarlar üzerinde bulunan dosyalar arasında gezinmek mümkün olur. Yine masaüstü uygulamaları yardımıyla çeşitli kısa yollardan faydalanarak bu dosyalar arasında daha erişilebilir şekilde gezinilebilir. Sunucularda kurulu olan web uygulamalarına erişmek ve onları görüntüleyip işlem yapabilmek için de web tarayıcıları kullanılır.

Tarayıcılar, istemci-sunucu iletişimine aracılık ederler. URL-HTML-HTTP kavramlarıyla kurulan iletişim, tarayıcının çalışmasını yansıtır.

Tarayıcıların çalışma mekanizmasını anlayabilmek için adres, DNS (domain name system - alan isimlendirme sistemi) araması, TCP (transmission control protocol - iletişim kontrol protokolü), TLS (transport layer security - taşıma katmanı güvenliği) anlaşması, ilk istek, ayrıştırma, DOM (document object model - doküman nesne modeli) ve CSSOM (cascading style sheets object model - basamaklı stil şablonları nesne modeli) ve oluşturma kavramlarından faydalanılır.

**Adres:** Öncelikle web uygulamalarına erişmek için hangi adresteki uygulamaya gidileceğini belirten web uygulamasının URL'sini tarayıcının adres çubuğuna yazılması ya da uygulamanın bağlantısına (link) tıklanması gerekir.

**DNS Araması:** URL genelde "ornek.com" gibi hatırlaması ve yazması insanlar için kolay bir adres ifadesidir. Metinsel ifadelerden oluşan bu adres, arka planda 99.876.321.00 gibi bir sunucu IP (internet protokolü) adresini işaret eder. Kullanıcı tarafından ilk kez ziyaret edilen her URL için bir DNS sunucusunda IP adresi için arama yapılır. DNS sunucusu "ornek.com" URL'sine karşılık gelen IP adresini bulur ve kullanıcıyı bu adrese yönlendirir.

**TCP-TLS Anlaşması:** Web tarayıcıyı kullanan kullanıcı ile sunucunun veri alışverişinin öncesinde TCP anlaşması sağlanmalıdır. Bu anlaşma, iki insanın birbiriyle konuşmaya başlamadan önce selamlaşması/el sıkışması gibi bir anlaşmadır. Bu anlaşma sırasındaki istekler Şekil 2.1'de gösterilmiştir.

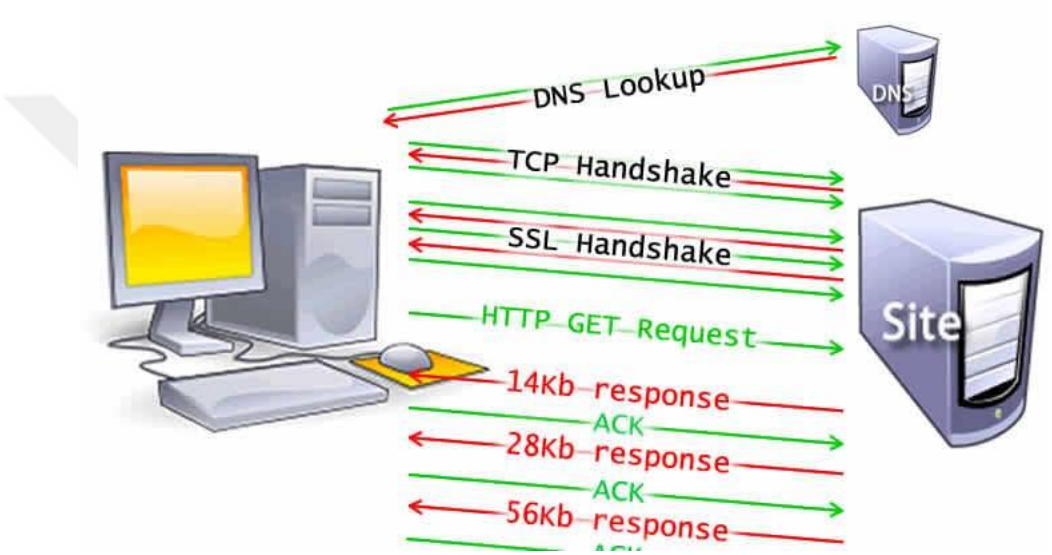
HTTPS üzerinden sağlanan güvenli bağlantılarda TLS üzerinden sunucu doğrulaması ve bir başka anlaşma/el sıkışması daha yapılır.

TCP ve TLS protokolleri sağlanıp anlaşma sağlandıktan sonra sunucu tarayıcının isteğine cevap vermeye hazır hale gelir.

**İlk İstek:** Bağlanmak istenilen sunucuya bağlantı sağladıktan sonra tarayıcı sunucuya bir GET isteği gönderir. Bu isteğe göre sunucu, protokollere uygun olarak başlıkları (headers) ve web sayfalarının yapısını anlatan HTML içeriğini cevap olarak gönderir. 14 kilobaytlık ilk TCP verisi ile başlayıp iletişim tamamlanana kadar tarayıcı ve sunucu arasında iletim devam eder.

**Ayrıştırma:** Sunucudan alınan ilk veriyle beraber tarayıcı sunucundan gelen veriyi ayrıştırmaya başlar. Verilerle DOM ve CSS (cascading style sheets - basamaklı stil şablonları) modelleri oluşturur. DOM ve CSS modelleri ile tarayıcı sayfayı oluşturur ve stilleri düzenler.

DOM, HTML veri içerisinde belirtilen elemanların hiyerarşik şekilde oluşturulduğu modeldir, tarayıcılar DOM'u kullanarak kullanılan elemanları CSS stilleri ile düzenleyip web sayfalarını ekranda oluştururlar. DOM'a Javascript ile müdahale edilebilir. Şekil 2.2'de örnek bir DOM eleman ağacı gösterilmiştir.



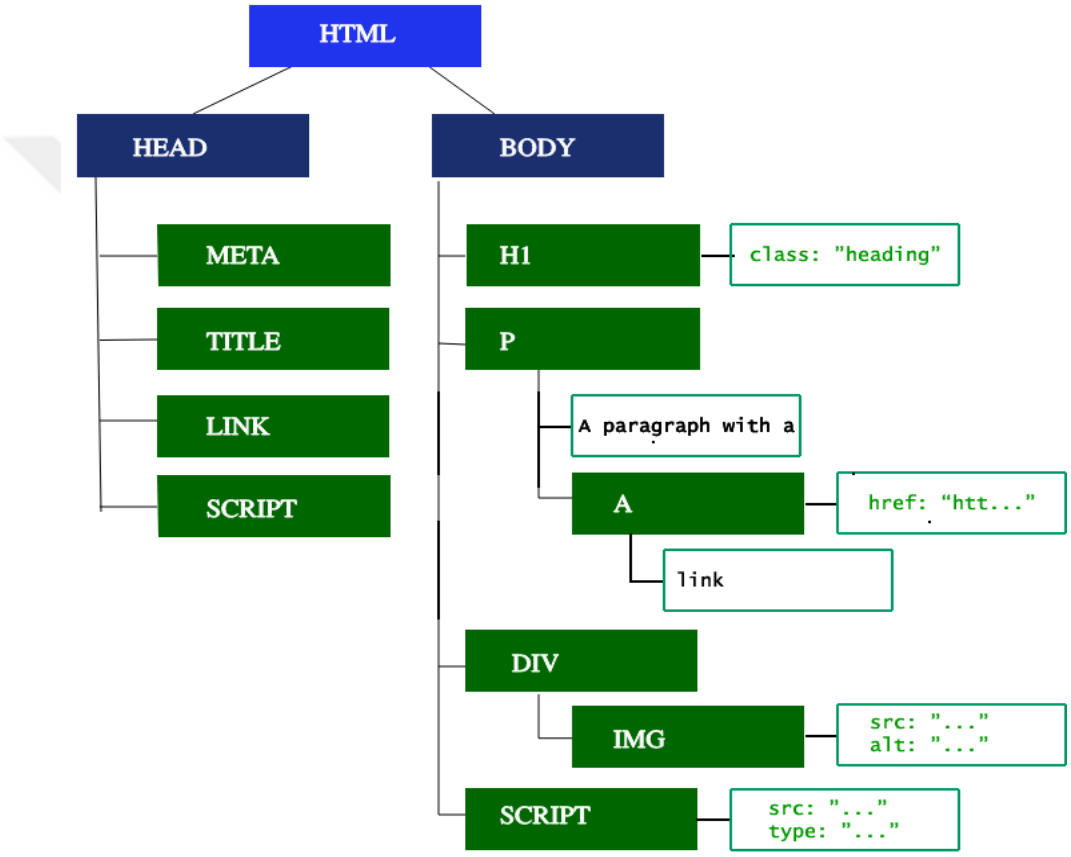

**Şekil 2.1:** TCP-TLS anlaşması sırasındaki veri alışverişi (MDN Web Docs, 2021)

**DOM ve CSSOM:** DOM, hiyerarşik bir ağaç şeklinde oluşturulur. İçerisinde sayfadaki tüm elemanlar ve içerikler mevcuttur. "<html>" elemanı ağacın ilk ve kök elemanıdır. DOM bu etiketleri ve aralarındaki hiyerarşileri içerir. Bir etiket içerisine yerleştirilmiş etiketler alt düğümlerdir. Düğümlerin sonunda görseller, videolar gibi kaynaklar olduğunda kaynakların adresleri ile yeni isteklerde bulunarak bunların sayfalarda gösterilmesi için tarayıcının ön yükleme tarama aracı sorgular oluşturur. "<script>" etiketi içerisindeki Javascript kodları tarayıcının ayrıştırma işlemini asenkron yazılmış kodlar dışında durdurur. Javascript kodları okunup ilgili elemanlar düzenlendikten sonra ayrıştırma devam eder.

Görsel kaynaklar, CSS dosyaları, Javascript dosyaları gibi dış kaynaklar tarayıcının ön yükleme tarama aracı ile sorgulama yaparak ilgili kaynakları getirir. Görsel kaynaklar ve CSS dosyaları

sayfanın oluşmasını engellemezler. Ancak Javascript dosyaları DOM içerisindeki elemanların oluşması ve gösterimini düzenledikleri için dosya getirilip okunduktan sonra DOM ve sayfa oluşumu devam eder.

CSS yapı dosyası CSSOM da DOM gibi hiyerarşik bir yapı dosyasıdır. Tarayıcı DOM gibi CSSOM da aynı şekilde sunucudan gelen verilerle oluşturur. Gelen veriler CSS kurallarını bir harita şeklinde tarayıcının işleyebileceği ve sayfayı oluşturabileceği yapıya dönüştürülür.

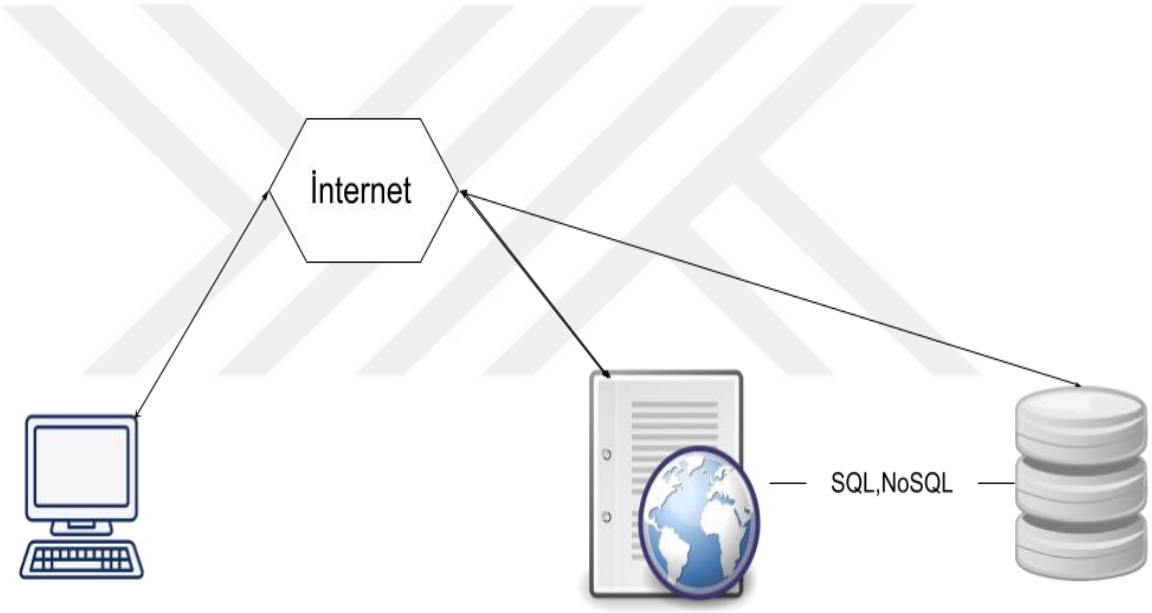



**Şekil 2.2:** DOM örnek eleman ağacı yapısı (MDN Web Docs, 2021)

**Oluşturma:** Web sayfasının tarayıcı ekranında oluşturulması (ekranın ilgili elementlerle boyanması) süreci DOM ve CSSOM tek bir oluşturma ağacında çaprazlanarak birleştirilmesi ile başlar. DOM'un kök düğümünden başlanarak ekranda görünmesi gereken ilgili elementlerin stilleri işlenerek tarayıcı tarafından ekranda oluşturulur. Oluşturma sürecinde sırasıyla CSS kuralları elemanlarla eşleştirilir, ardından görünür elemanların genişlik, yükseklik ve şekil gibi geometrik özellikleri hesaplanır. Son olarak da sayfa elemanlar ile boyanır.

### 2.1.2 Web Uygulamalarının Yapısal Modeli

Web içeriği, özellikle 2000'lerin başındaki gelişmelerle özel olarak adlandırılan "Web 2.0" dönemiyle beraber kullanıcılar tarafından dinamik olarak oluşturulmaktadır. İnternetin ilk dönemlerinde içerikler önceden oluşturulmuş ve sayfalara işlenmiş statik verilerdir. Statik web sayfalarının içeriği önceden oluşturulmuş ve işlenmiş sayfalardır. Statik bir sayfaya erişen kullanıcı sürekli aynı verileri görecektir. Dinamik sayfalarda ise sayfadaki veriler kullanıcıya özel, kullanıcının girişlerine göre ve dış kaynaklardan gelen verilerden oluşur. Dinamik web içeriği, Şekil 2.3'te gösterildiği gibi bir tarayıcı yardımıyla internete bağlı kullanıcının web sunucusuna bağlanıp veri tabanına içerik göndermesiyle oluşturulur.




**Şekil 2.3:** Web Uygulamaları Yapısal Modeli

Sayfanın, uygulamanın dinamik içeriği veri tabanında tutulur. Web uygulaması, çeşitli mantık uygulamalarıyla bu içerik için erişim, düzenleme, oluşturma ve silme gibi işlemleri sağlar. Web sunucusu da uygulamanın çalışmasını, veri tabanının çalışmasını, veri tabanına ulaşılmasını sağlar. Web uygulaması üzerinden kullanıcının çeşitli sorgular çalıştırarak veri tabanından erişilen verilerle beraber web sunucusu kullanıcıya bir HTML sayfa gönderir ve bu dinamik sayfayı kullanıcı tarayıcı üzerinden kullanır.

## 2.2 VERİ TABANLARI

Uygulamalar çeşitli verileri ve bilgileri kullanıcılara göstermek, sistemin işlemesini devam ettirmek ve kayıt tutmak gibi amaçlarla veri işleme sistemleri kullanırlar. Verilerin tutulduğu yapılar da veri tabanlarıdır. Veri tabanı, mantıksal olarak ilişkili verilerden oluşan düzenli bir koleksiyondur (Abraham Silberschatz, 2010).

### 2.2.1 Geleneksel Dosya İşleme Sistemleri

Web sunucularına kurulan ve birden fazla noktadan erişilen günümüzdeki veri tabanı yapılarından önce de bilgisayarlarda kullanılan yerel yazılımlarda veri iletişimi için geleneksel dosya işleme sistemleri kullanılmaktaydı. Bu geleneksel dosya işleme sistemlerinin dezavantajları aşağıda sıralanmıştır (Abraham Silberschatz, 2010).

**1. Program-Veri Bağımlılığı:** Yerel yazılımlar içerisinde üretilen veriler sadece aynı yazılım ile okunabilir, aranabilir ve düzenlenebilir durumdadır.

**2. Tekrarlı Veri:** Farklı işler için ayrıca geliştirilen yerel yazılımlar için formlar tekrar tekrar aynı verilerle doldurulmalıdır ve ilişkiler baştan kurulmalıdır.

**3. Uzun Geliştirme Süreleri:** Veri dosyaları için her seferinde veri ilişki mimarisi yazılıma özel olarak yazılımcı tarafından oluşturulmalıdır bu da geliştirme süresini uzatmaktadır.

**4. Karmaşık Yazılım Bakımları:** İhtiyaca yönelik eklenmesi gereken bir veri sütunu ya da program akışındaki bir değişiklik ile mevcut verilerin bakımı çok uzun süreler alabilir.

**5. Tek Kullanıcı:** Dosya işleme sistemleri tek kullanıcı için tasarlanmıştır. Günümüzde aynı veri tabanına birden fazla kullanıcı bağlanarak verilere erişir ve veri sorgulaması yapabilmektedir.

### 2.2.2 Veri Tabanı Yaklaşımı

Geleneksel dosya işleme sistemlerinden sonra kullanılan veri tabanı sisteminin en büyük amacı kullanıcılara verinin soyutlaştırılarak kullanımını ve düzenlenmesini basitleştirmesidir, (Abraham Silberschatz, 2010). Veri tabanı, mantıksal olarak ilişkili verilerden oluşan düzenli bir koleksiyondur (Jeff Hoffer, 2016). Veri tabanı yaklaşımı ile verinin nasıl saklandığı, nerede

saklandığı ve hangi kullanıcının erişme ve düzenleme yetkisinin olduğu görülecek şekilde tutulup soyutlaştırılması sağlanmıştır.

### 2.2.3 İlişkisel Veri Tabanları

Verileri kaydetmenin ve sunmanın en basit ve sezgisel yolu, bir tablo kullanmaktır. Tüm örnek ve kavram olarak bilinen veriler ekstra bir açıklamaya gerek olmadan bir tablodan rahatlıkla anlaşılmakta ve okunabilmektedir. İlişkisel veri tabanı yaklaşımında hem veriler hem de veriler arasındaki ilişki tablolarda sunulmaktadır.

#### *2.3.3.1 Tablo Yapısı*

Verileri veya veriler arasında ilişkileri satırlarla tablo ile tutan yapının özelliklerini ifade eder. Bu özellikler aşağıdaki şekilde özetlenebilir:

**Tablo ismi:** Veri tabanı içerisindeki her tablonun farklı bir ismi olmalıdır. Tablo isimleri belirttikleri tabloyu niteler bir biçimde verilmelidir. Ayrıca her veri tabanı yönetim sisteminde tablo isimleri ile ilgili bazı şekilsel kurallar bulunmaktadır.

**Nitelikler (Attributes):** Niteliklerin isimleri tablo içerisinde benzersizdir ve nitelikler bir sütunu etiketlemek için kullanılmaktadır.

**Birincil anahtar:** Bir nitelik ya da niteliklerin kombinasyonu benzersiz şekilde veriyi tablodaki diğer satırlardan ayırır, bu birincil anahtar niteliktir.

**Sıralar:** Sütunların ve satırların sırası önemsizdir.

Genel olarak özetleyecek olursak;

- Tablolar sütunlardan (kolonlardan) oluşur.
- Her bir sütunun ayrı bir adı vardır.
- Her bir sütun, aynı niteliğin tanımladığı aynı etki alanının (domain) belirlediği değerleri içerir.
- Her bir satır birbirinden farklıdır.
- Sütunların sırası önemsizdir.
- Satırların sırası önemsizdir.

### *2.3.3.2 SQL*

İlişkisel yapılar bilgiyi tablolar halinde saklar. Dolayısıyla her tablo aynı yapıda olan veri demetleridir. Tablolardaki verinin aynı yapıda olması çeşitli sorgularla istenen veriye ulaşmayı ve değiştirmeyi ve sonucu yine tablo halinde almayı mümkün kılmıştır. İlişkisel veri tabanlarının sorgulandığı ve değiştirildiği dil, SQL olarak kısaltılan "Structured Query Language" dır. SQL ile istenen veri açıklanarak veri tabanı yönetim sistemine sorgu gönderilip ve sonuç tablosu alınmaktadır. En önemli SQL deyimi İngilizce SEÇ anlamına gelen SELECT deyimidir. Bu deyim, veri tabanı yönetim sistemleri tablolarını sorgulamak için kullanılır.




**Şekil 2.4:** SQL ile örnek bir sorgu

Şekil 2.4'te gösterilen örnek sorgu, tablodan koşulun sağlandığı satırlardaki nitelik değerlerini içeren bir tablo geri döndüren sorgudur.

SQL deyimleri düzenlenirken her programlama dilinde olduğu gibi bazı kurallara dikkat etmek gereklidir:

1. SQL deyimleri büyük-küçük harf ayrımından etkilenmez. Bunun anlamı, SQL deyiminin bir kısmı büyük, diğer bir kısmı da küçük harflerle ifade edilebilir. SELECT ile Select aynı anlama geldiği gibi TABLO ADI, Tablo Adı ya da tablo adı da aynı anlama gelmektedir.

2. Bir SQL deyimi bir veya daha fazla satırda yer alabilir. Bilgisayar programlarında önemli olmayan boşluk kavramı SQL için de önemli değildir.

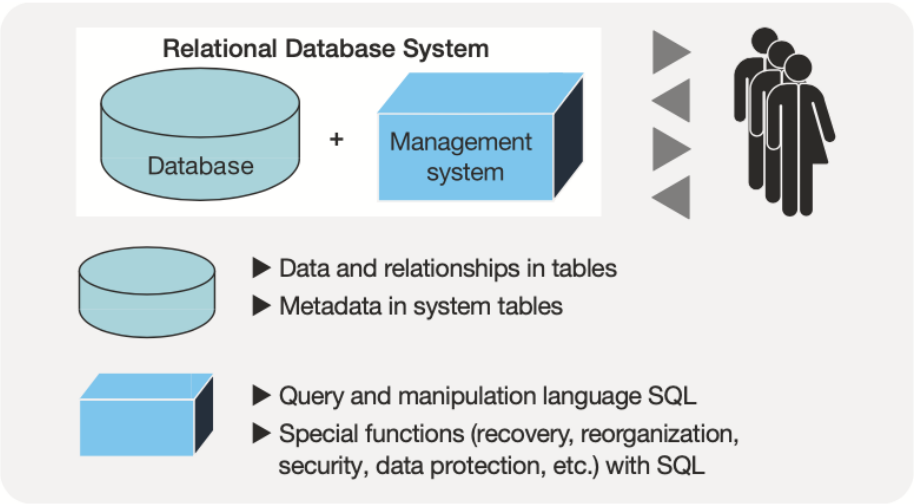
3. SQL anahtar kelimeleri kısaltılamaz veya yarıda kesilerek ayrı ayrı satırlara bölünemez. SQL deyimi satırlara yayılabilir ancak kelimeler satırlara yayılamaz.

4. Deyimin bittiği yeri belirlemek için en sonuna (;) işareti kaydedilebilir.

5. Anahtar kelimeler genellikle ayrı satırlara kaydedilerek daha elverişli bir görünüm sağlanabilir.

### *2.3.3.3 İlişkisel Veri Tabanı Yönetim Sistemi*

İlişkisel veri tabanları, bilişim sistemlerinin geliştirilmesinde ve bu sistemlerin operasyonlarında verinin merkezi şekilde, kalıcı olarak ve belirli bir yapıda kaydedilip işlenmesi için kullanılmaktadır. Veri tabanları, uygulamalar aracılığıyla kullanıcılarla ile iletişime geçmesi için bir yönetim sistemi ile kullanılmaktadır. Şekil 2.5'te gösterildiği gibi veri tabanı yönetim sistemi bir depolama alanı ve yönetim katmanı içermektedir. (Andreas Meier, 2019)




**Şekil 2.5:** İlişkisel veri tabanı yönetim sisteminin temel yapısı (Andreas Meier, 2019)

Depolama alanı, veri tabanı olarak bahsedilen verilerin ve ilişkilerin depolandığı alandır. Bunlara ek olarak sistem tabloları da bu alanda tutulmaktadır. Sistem dosyaları veri tabanı operasyonları için gerekli tabloları içermektedir. Bu bilgiler kullanıcılar tarafından okunabilir ancak değiştirilemez durumdadır.

Yönetim katmanının görevi; verinin tanımlanması, seçilmesi ve düzenlenmesi işlemlerini, SQL aracılığıyla yapılan sorguları işleyerek gerçekleştirmesidir. Ayrıca yönetim katmanı, hatalardan sonra veri kurtarma, veri koruma ve yedekleme servis fonksiyonlarını da yerine getirmektedir.

Veri tabanı yönetim sistemlerinde veri bütünlüğünü sağlamak için birçok yol bulunmaktadır. Bunlardan en önemlisi, "bütünlük sınırlamaları" adı verilen ve veri tabanı yönetim sistemi üzerinde gerçekleştirilen çeşitli sınırlamalardır. Bütünlük sınırlamaları, Veri tabanı yönetim sistemi veya uygulama programları tarafından tanımlanır. Sözü edilen sınırlamalar, kullanıcı

tarafından yapılması gereken ekleme, silme ve güncelleştirme işlemlerinden önce bir denetim yapılmasına neden olur. Bu denetim sonucunda, sadece sınırlamalara uyan işlemlerin yapılmasına izin verilir, diğerleri reddedilir. Bu da yapılan tüm işlemlerin belirli bir kritere uyması anlamına gelmektedir. Veri tabanı yönetim sistemi sınırlamaları temel olarak aşağıdaki gibi ifade edilebilir:

- Anahtar Sınırlamaları
- Veri Değerleri Sınırlamaları
- İşletme Kuralları Sınırlaması
- Nitelikler Arası Bağımlılıklar

**Anahtar Sınırlamaları:** Bütünlük sınırlamalarının sağlanmasında anahtarlar önemli rol oynar. Anahtar türü belirlenerek, bu sınırlamaların Veri tabanı yönetim sistemi tarafından otomatik yapılması sağlanır. Bu sınırlamalar, birincil anahtar (primary key) veya dış anahtar (foreign key) sınırlamaları şeklinde gruplandırılabilir.

***Birincil Anahtar Sınırlamaları:*** İlişkisel veri tabanı yönetim sisteminde bir tablonun benzer değerler içermeyen, özgün bir sütunu ya da birkaç sütunu birlikte "birincil anahtar" olarak tanımlanabilir. Birincil anahtar, esasında bir aday anahtarıdır ve söz konusu varlığın kayıtlarını en iyi biçimde karakterize eder. Birincil anahtar tanımlandığında, bu anahtarın NULL ya da birbirinin aynı değerleri içermemesi gibi bir sınırlandırma konulmuş olur.

***Dış Anahtar Sınırlamaları:*** Bir dış anahtar, bir sütun veya çok sayıdaki sütunların birleşiminden oluşur. Dış anahtar, aynı tablo ya da bir başka tabloda yer alan birincil anahtarla eşleştirilir. Dış anahtarın değeri bir birincil anahtar değeriyle eşleşmeli veya NULL olmalıdır. Eğer dış anahtar birincil anahtarın bir kısmını oluşturuyorsa, doğal olarak NULL değer içeremez.

***Veri Değerleri Sınırlamaları:*** Tablonun herhangi bir sütununda yer alacak verilerin hangi değerlere sahip olabileceği konusunda sınırlamalar getirebilir. Eğer herhangi bir tabloda boş bırakılmaması istenen bir alan varsa bu alan girişi ile ilgili NULL olmaması gerektiği belirtilebilir.

***İşletme Kuralları Sınırlaması:*** Tablolar üzerinde işletme kurallarına dayalı sınırlamalar yapılabilir. Bu sayede işletme kurallarına uygun olarak veri bütünlüğü söz konusu olabilir. Örneğin, işletmenin muhasebe bölümünde "programcı" kadrosuna sahip eleman çalıştırmak

anlamlı değildir, hatta hatalara, yetkisiz erişimlere vb. sebep olacağından önlenmesi gerekmektedir. Veri tabanı yönetim sisteminde bilgi girişleri esnasında muhasebe bölümüne yeni personel kaydı eklemek gerektiğinde, söz konusu işletme kuralı, uygunsuz veri girişlerini engelleyecektir. Bu tür sınırlamalar çoğunlukla uygulama programları aracılığıyla denetlenir.

***Nitelikler Arası Bağımlılıklar:*** Veri bütünlüğünün ve tutarlılığının sağlanmasında, nitelikler arasındaki bağımlılıkların kullanılması yararlı olabilir. Niteliklerin birbirleriyle olan bağımlılığı göz önüne alınmadan veri tabanı tasarımının yapılması sorunlara neden olacaktır. Nitelikler arasındaki fonksiyonel bağımlılıklar belirlenerek, şema üzerinde düzenlemeler yapılır.

### 2.2.4 İlişkisel Olmayan Veri Tabanları

İlişkisel olmayan veri tabanları, Şekil 2.6'da gösterildiği gibi farklı formatlarda olabilirler.

İlişkisel olmayan veri tabanı yönetimi (NoSQL) kavramı, ilişkisel olmayan ve aşağıdaki iki kriteri ortak olarak sağlayan veri yönetim sistemleri için kullanılmaktadır. (Andreas Meier, 2019)

1. Veri tablolarda tutulmaz.

2. Veri tabanı sorgu dili SQL değildir.

NoSQL yapılar, çok geniş kullanım alanına sahiptir ve kompleks web uygulamalarında ilişkisel veri tabanı yönetim sistemleriyle beraber de kullanılmaktadır. Uyumlu yapıları ile ana hedefleri hızlı ölçeklenebilir yapıları desteklemek, tabloların sınırlamalarından sistemi kurtarmak ve API'lere (application programming interface, uygulama proglama arayüzü) ile kolay erişim sağlamaktır. Belli başlı NoSQL Veri tabanı yönetim sistemi türleri şunlardır:

- Anahtar-Değer Veri Tabanı Yönetim Sistemleri
- Belge Veri Tabanı Yönetim Sistemleri
- Grafik Veri Tabanı Yönetim Sistemleri
- Bellek İçi Veri Tabanı Yönetim Sistemleri
- Arama Veri Tabanı Yönetim Sistemleri

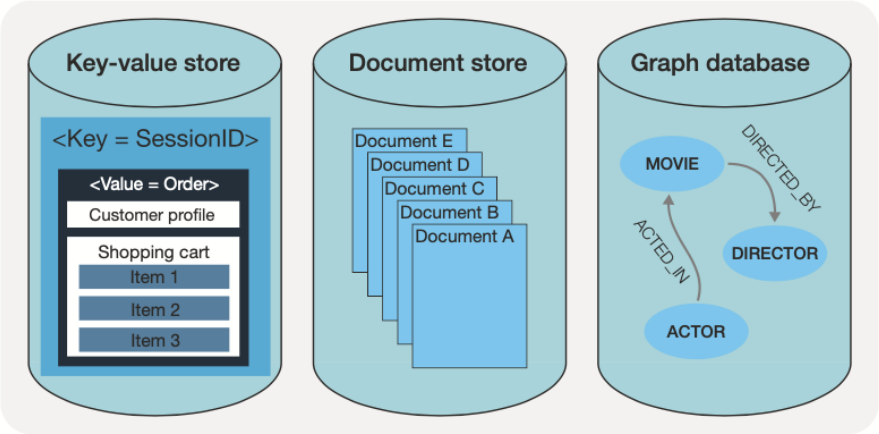
***Anahtar-Değer Veri Tabanı Yönetim Sistemleri:*** Anahtar-değer veri tabanı yönetim Sistemleri yüksek oranda bölümlendirilebilir ve diğer veri Tabanı Yönetim Sistemleri'nin ulaşamayacağı ölçeklerde yatay ölçeklendirmeye imkan tanır. Anahtar-değer veri modeli için özellikle de oyun, reklam teknolojisi ve nesnelerin interneti gibi kullanım örnekleri uygundur.

***Belge Veri Tabanı Yönetim Sistemleri:*** Geliştiriciler için verimli ve sezgisel bir veri modeli olduğundan veriler, uygulama kodunda genellikle bir nesne veya JSON (Javascript nesne gösterimi - Javascript object notation) benzeri bir belgeyle temsil edilir. Belge Veri Tabanı Yönetim Sistemleri, geliştiricilerin uygulama kodlarında kullandıkları aynı belge modeli biçimini kullanarak bir veri tabanı yönetim sisteminde veri depolamasını ve sorgulamasını kolaylaştırır. Belgelerin ve belge veri tabanı Yönetim sistemlerinin esnek, yarı yapılandırılmış ve hiyerarşik yapısı sayesinde geliştiriciler, uygulamaların gereksinimleri doğrultusunda geliştirme yapabilirler. Bu belge modeli; kataloglar, kullanıcı profilleri ve her belgenin hem benzersiz olduğu hem de zamanla geliştiği içerik yönetim sistemleri ile birlikte oldukça kullanışlıdır.

***Grafik Veri Tabanı Yönetim Sistemleri:*** Bir grafik veri tabanı yönetim sisteminin amacı, yüksek oranda bağlı veri kümeleriyle çalışan uygulamalar oluşturup çalıştırmayı kolaylaştırmaktır. Grafik veri tabanı yönetim sistemlerinin yaygın olarak karşılaşılan kullanım örnekleri arasında sosyal ağlar, öneri altyapıları, dolandırıcılık algılama ve bilgi grafikleri yer alır. Amazon Neptune örnek olarak verilebilir.

***Bellek İçi Veri Tabanı Yönetim Sistemleri:*** Oyun ve reklam teknolojisi uygulamalarının puan tabloları, oturum depoları ve gerçek zamanlı analitik gibi mikro saniye düzeyinde yanıt süreleri gerektiren ve her an ani trafik artışları yaşanabilen kullanım örnekleri vardır. Bu gibi durumlar için bellek içi veri tabanları kullanılır.

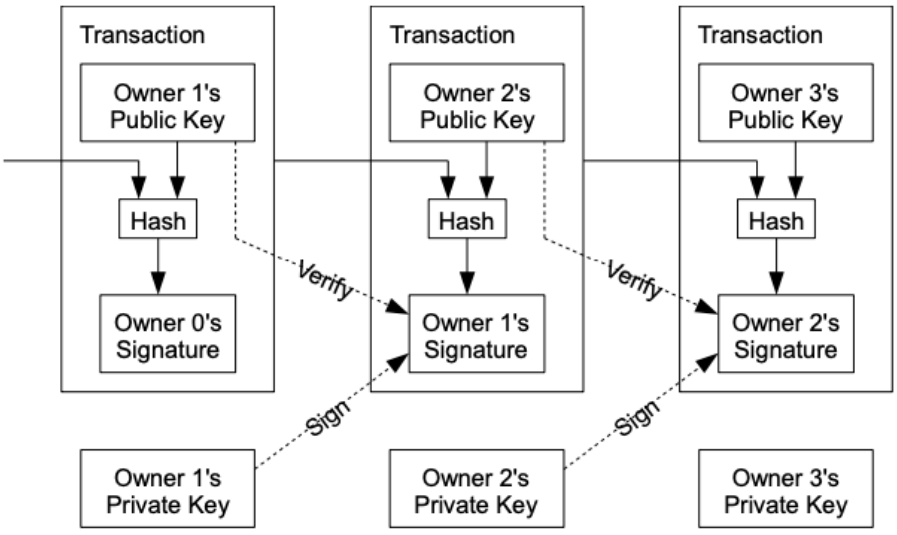
***Arama Veri Tabanı Yönetim Sistemleri:*** Birçok uygulama, yazılım geliştiricilerin sorunları gidermesine yardımcı olmak için günlük oluşturur. Özellikle yüksek seviyede veri kullanan sistemlerde arama Veri tabanı yönetim sistemi yapıları sıklıkla karşımıza çıkmaktadır. Amazon Elasticsearch Service (Amazon ES), yarı yapılandırılmış günlükleri ve ölçümleri dizine ekleyerek, biriktirerek ve bunlarda arama yaparak makineler tarafından oluşturulan veriler için neredeyse gerçek zamanlı görselleştirmeler ve analitik sağlama amacına özel olarak tasarlanmıştır.




**Şekil 2.6:** Farklı ilişkisel olmayan veri tabanı yapıları (Andreas Meier, 2019)

## 2.3 BLOK ZİNCİRİ

Bitcoin'in icadı ile (Nakamoto, 2008) popülerlik kazanan blok zinciri teknolojisi, SQL ve NoSQL veri tabanları ile oluşturulan veri tabanı yönetim sistemlerine göre bazı farklı yetkinlikler sunduğu için çeşitli uygulamalarda kullanılmıştır. Blok zinciri teknolojisi; genel olarak merkezi bir otorite ihtiyacı olmadan, eşler arası işlemleri sağlayan, verileri Şekil 2.7'de gösterildiği gibi merkle tree yapısında birbirine zincirleme olarak değiştirilemez şekilde bağlı halde, dağıtık defterle eşlerde tutan blok zinciri, geleneksel veri tabanlarına alternatif olarak ve/veya geleneksel veri tabanlarıyla beraber kullanılmaktadır.




**Şekil 2.7:** Blok zincirinin merkle tree yapısı

### 2.3.1 Blok Zinciri ve Veri

En popüler blok zinciri Bitcoin'in blok zincirinin toplam boyutu Nisan 2021 itibariyle 340 GB (Gigabyte), ortalama bir blok boyutu da 1.1 MB (megabyte) (Blockchain.com, 2021) veri boyutuna ulaşmıştır. Blok zinciri verileri, merkeziyetsiz şekilde işlemleri imzalayan tüm eşlerin sahip olduğu dağıtık defterlerde tutulmaktadır. Bitcoinlerin sahiplik bilgisi de kullanıcılarının dijital cüzdanlarında tutulmaktadır.

Geleneksel veri tabanı yönetim sistemleri, bilgisayarların icadı ile senelerdir ihtiyaçlara göre kendisini geliştirmiş bir mirastan güç almaktadır. Blok zinciri teknolojisi ile geleneksel yaklaşımının beraber geliştirilerek uygulamalarda kullanılmaktadır. Yapılan çalışmalarda (Mayank Raikwar, 2020) geleneksel veri tabanı yönetim sistemlerinin ve blok zinciri alanındaki teknolojilerinin birbirinden çeşitli özellikler kazanıp etkilenerek geliştiği belirtilmiştir.

### 2.3.2 Blok Zinciri Türleri

#### *2.3.2.1 Açık (İzin Gerektirmeyen) Blok Zincirleri*

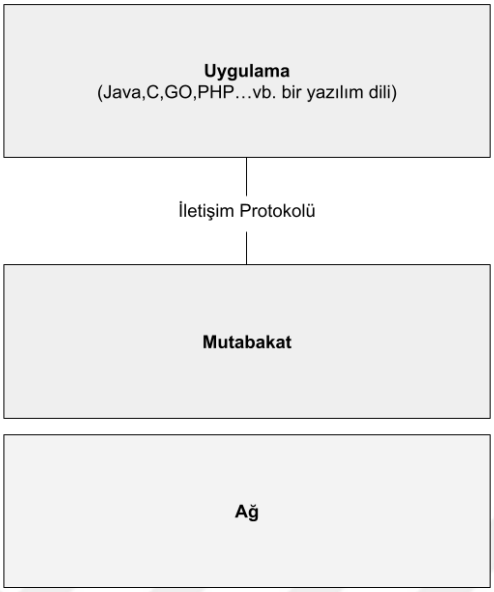
Herkesin katılabileceği, işlemler oluşturabileceği, bu işlemleri doğrulayabileceği, blok zincirinin durumunu güncelleyebileceği yapılardır. Tüm işlemler ve zincirin durumu şeffaftır ve herkes tarafından erişilebilir.

#### *2.3.2.2 Gizli (İzinli) Blok Zincirleri*

Açık zincirlerin tam tersi özeliklerine sahip yapılardır. Blok zincirine sadece yetkilendirilmiş kullanıcıların eriştiği ve verilerin serbest şekilde erişime gizlendiği yapılardır.

### 2.3.3 Blok Zinciri Tabanlı Web Uygulamaları

Blok zinciri tabanlı web uygulamaları Şekil 2.7'de gösterildiği gibi mimari açıdan uygulama, mutabakat ve ağ katmanları ile incelenebilir.




**Şekil 2.8:** Blok zinciri uygulamasının katmanları

**Uygulama Katmanı:** Blok zincirinin mantık kısmını tanımlayan, durum geçişlerini düzenleyen yapıdır.

**Mutabakat Katmanı:** Düğümlerin blok zincirinin durumunda hem fikir olmalarını sağlayan ve merkeziyetsiz yapının karar verme mekanizmasını oluşturan katmandır.

**Ağ Katmanı:** İşlemlerin, durum geçişleri mesajlarının ve fikir birliği iletilerinin çoğaltılıp yayılmasından sorumlu katmandır. (Tendermint Inc., 2017)

### 2.3.4 Blok Zinciri Mutabakat Algoritmaları

Blok zinciri algoritması seçimi -blok zinciri tabanlı bir sistem oluşturmak ile geleneksel veri yönetim sistemleri kullanmak arasındaki seçim gibi- temel bir seçim noktasıdır. Uygulamanın yeteneklerini, performansını, güvenliğini ve işleyişini köklü şekilde etkilemektedir. (Ferdous, 2020)

#### *2.3.4.1 Proof of Work (Emek Kanıtı)*

Blok zinciri durum değişimleri, hesaplama gücü gerektiren kaynakların kullanılması ile gerçekleştirilir. Yeni bir halkanın ya da yeni bir işlemin oluşturulması için gereken kanıt kaynağı, oluşturulmuş bir kanıtın doğrulanması için gereken kaynaktan daha fazladır. Bu asimetrinin ardındaki fikir sistemin hileli bir işlemle kandırılmasını engellemektir.

En popüler blok zincir olan Bitcoin de PoW (emek kanıtı, proof of work) algoritması ile çalışmaktadır. 2021 yılı içerisinde Bitcoin'in ortalama onay süresi (bir işlemin madencilikle çıkarılan bir bloğa dahil edilmesi ve halka açık deftere eklenmesi için ortalama süre) aylık ortalama 800 dakikanın üzerine çıkmıştır. (Blockchain.com, 2021)

#### *2.3.4.2 Proof of Stake (Hisse Kanıtı)*

PoW algoritmasının limitlerine karşı, 2011 senesinde bir forumda ilk önerisine rastlanan (QuantumMechanic, 2011 ) Proof of Stake (hisse kanıtı, proof of stake) algoritmasının önerisi, blok oluşturma sürecine katılmak isteyen yeni bir düğümün ilk başta belirli sayıda ilgili birim hissesine sahip olduklarını kanıtlamaları ve emanet hesabına belirli miktar değeri kilitlemeleri gerektiği kavramlarına göre şekillenmiştir. Kilitlenen tutar, işlemin güvenirliliğini sağlamak için bir emanettir. İlgili işlemi yapan düğüm uygunsuz davranırsa emanet olarak kitlenen değeri kaybedebilir ve kurallara göre bir daha blok durumunu değiştirme işlemlerinin hiç birine katılamayabilir.

#### *2.3.4.3 Diğer Mutabakat Algoritmaları*

PoW ve PoS algoritmalarını hibrit bir şekilde kullanan ya da baştan sona farklı mekanizmalarla geliştirilen ve önerilen mutabakat algoritmalarıyla uygulamalar geliştirilmektedir. Bu algoritmalar bazen ihtiyaçlar üzerinden geliştirilirken bazen de kısıtlamalar üzerinden düşünülerek kullanılmaktadır. Her geçen gün yeni algoritmalarla blok zinciri uygulamaları hayata geçerek kitlelerin kullanımına sunulmaktadır (Ferdous, 2020).

### 2.3.5 Bizans Generalleri Problemi

Dağıtık veri sistemlerinin ana problemlerinden olan Bizans Generalleri Problemi, 1982'de isminin verildiği bir çalışmada (Leslie Lamport, 1982) tanımlanmıştır. Daha önce gereksiz maillerin filtrelenmesi ve hava kontrol sistemlerinin hata toleransları için de gündeme gelen bu

problem genel olarak dağıtık karar alan ve iletişim kuran düğümlerin aksiyonları arasındaki hata toleransı ile ilgilidir.

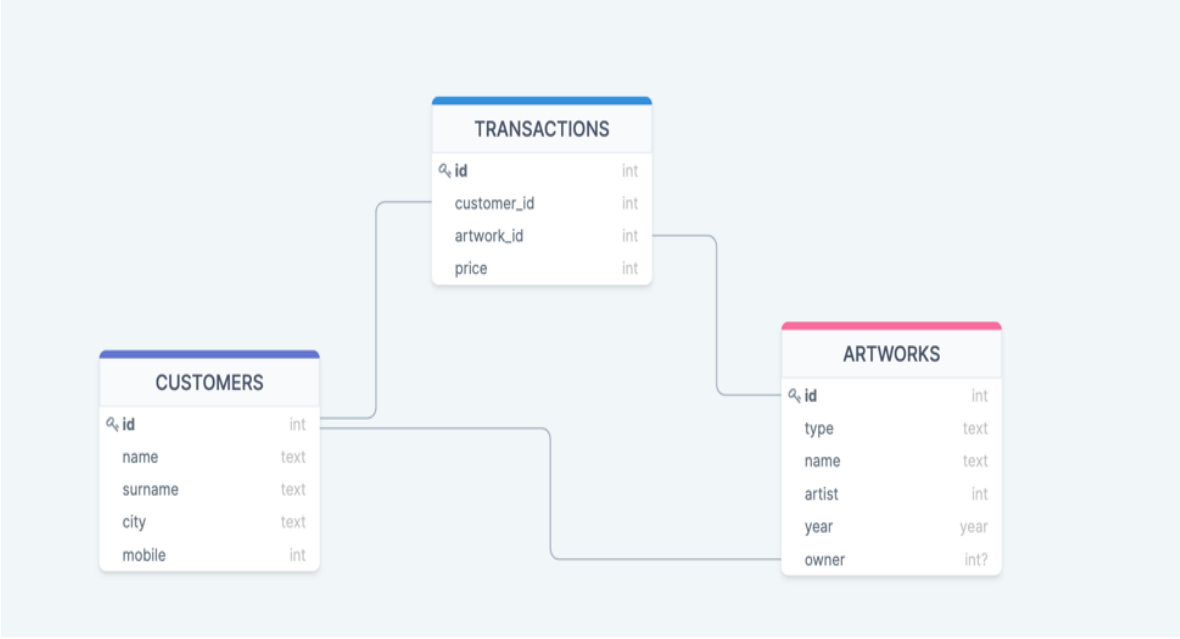
Güvenilir sistemler, elemanlarının yanlış çalışmalarına karşı önlemleri olan ve hatalı girdileri tolere etmelidir. Bizans generalleri problemi bir mutabakat problemidir. Bir şehri ele geçirecek tüm generaller ya da sayıca en kalabalık ordu ile harekete geçen generaller aynı anda saldırdığında şehir düşecektir. Eğer birbirlerinden uzak generaller, haberleştiği habercilerle saldıracaklarına ya da bekleyecekleri kararına göre beraber hareket etmezlerse savunma kazanacaktır. Hileci bazı haberciler bizanslı generallerin birbirlerine gönderdiği mesajları değiştirip orduyu kolayca yenilgiye uğratabilir. Bu problem sensörlerden veri toplayan kontrol sistemlerinde, gereksiz e-posta filtrelerinde, bankaların dolandırıcılık takip sistemleri gibi sistemler ve dağıtık veri sistemlerinde yaşanmaktadır. Blok zinciri mutabakat algoritmaları, bu probleme çeşitli şartlarla çözümler sunmaktadır.

# 3. MALZEME VE YÖNTEM

Bu bölümde sanat eserlerinden oluşan bir envantere sahip sanat galerisi uygulaması hem Mysql (SQL), hem Mongodb (NoSQL) hem de blok zinciri tabanlı bir yapı ile oluşturulmuştur. Sanat galerisi uygulaması ile galeri sahibi envanterine yeni sanat eserleri ekleyebilir, silebilir ve eserlerin belirlenmiş bilgilerini güncelleyebilmektedir.

## 3.1 SQL TABANLI UYGULAMA

Şekil 3.1'de gösterilen diyagramdaki tablolarla oluşturulan SQL veri tabanı yapısının gerçekleştirilmesi için ilişkisel veri tabanı yönetim sistemi Mysql 8.0.21 ve PhpMyAdmin 5.0.3 kullanılmıştır. Sunucu yönetimi için Apache 2.4.41 ve uygulamalar için PHP 7.4 kullanılmıştır. Sunucu olarak DigitalOcean firmasının 2 GB hafıza / 1 CPU (merkezi işlem birimi, central processing unit) olan sunucuları, Ubuntu 20.04 işletim sistemi ile kullanılmıştır.




**Şekil 3.1:** Sanat galerisi uygulamasının ilişkisel veri tabanı diyagramı

Bu yapıdaki tabloların açıklamaları aşağıdaki gibidir:

CUSTOMERS TABLOSU: Müşteri bilgilerini tutan tablodur, içerisinde id, name, surname, city ve mobile verileri bulunmaktadır. Bu verilerin açıklamaları aşağıdaki gibidir:

- id: Müşterilere verilen özgün, unik bir sayıdır, int yani tamsayı türünde tutulmaktadır ve tablonun anahtarı konumundadır.
- name: Müşterilerin adını tutan bir değerdir, text yani metin türünde tutulmaktadır.
- surname: Müşterilerin soyadını tutan bir değerdir, text yani metin türünde tutulmaktadır.
- city: Müşterilerin şehrini tutan bir değerdir, text yani metin türünde tutulmaktadır.
- mobile: Müşterilerin mobil telefonunu tutan bir değerdir, int yani tamsayı türünde tutulmaktadır.

TRANSACTIONS TABLOSU: İşlem bilgilerini tutan tablodur, içerisinde id, customer_id, artwork_id ve price verileri bulunmaktadır. Bu verilerin açıklamaları aşağıdaki gibidir:

- id: İşlemlere verilen özgün, unik bir sayıdır, int yani tamsayı türünde tutulmaktadır ve tablonun anahtarı konumundadır.
- customer_id: İşlemi yapan müşteriye verilen özgün, unik bir sayıdır, int yani tamsayı türünde tutulmaktadır, CUSTOMER tablosunun anahtarı olan id ile aynı işlevi görmektedir ve bu iki tablo arasında bağlantı sağlamaktadır.
- artwork_id: İşlemin üzerinde yapıldığı sanat eserine verilen özgün, unik bir sayıdır, int yani tamsayı türünde tutulmaktadır, ARTWOKS tablosunun anahtarı olan id ile aynı işlevi görmektedir ve bu iki tablo arasında bağlantı sağlamaktadır.
- price: Yapılan işlemin ücretini tutan bir değerdir, int yani tamsayı türünde tutulmaktadır.

ARTWORKS TABLOSU: Sanat eserlerinin bilgilerini tutan tablodur, içerisinde id, type, name, artist, year ve owner verileri bulunmaktadır. Bu verilerin açıklamaları aşağıdaki gibidir:

- id: Sanat eserlerine verilen özgün, unik bir sayıdır, int yani tamsayı türünde tutulmaktadır ve tablonun anahtarı konumundadır. TRANSACTIONS Tablosu'ndaki artwork_id'ye karşılık gelmektedir ve bu iki tablo arasında bağlantı sağlamaktadır.
- type: Sanat eserinin türünü ifade etmektedir, text yani metin türünde tutulmaktadır.
- name: Sanat eserinin adını ifade etmektedir, text yani metin türünde tutulmaktadır.
- artist: Sanat eserini oluşturan sanatçıyı ifade etmektedir, int yani tamsayı türünde tutulmaktadır. Buna göre her sanatçının bir numarası bulunmaktadır.

- year: Sanat eserinin yapıldığı yılı ifade etmektedir, veriye uygun biçimde year yani yıl türünde tutulmaktadır.
- owner: Sanat eserinin sahibi olan müşteriyi ifade etmektedir. int yani tamsayı türünde tutulmaktadır. CUSTOMER Tablosu'ndaki id'ye karşılık gelmektedir ve bu iki tablo arasında bağlantı sağlamaktadır.

## 3.2 NOSQL TABANLI UYGULAMA

Sanat galerisi uygulamasının Mongodb ile çalışan versiyonu için Mongo Atlas ile Mongodb 4.4.6 sürümü kullanılmıştır. Sunucu tarafında Ubuntu 20.04 işletim sistemi ve DigitalOcean firmasının 2 GB hafıza / 1 CPU olan sistem, içerik yönetim sistemi olarak Strapi 3.6.5 kullanılmıştır.

Mongodb'de "Strapi" üzerinden 3 ana koleksiyon oluşturulmuştur: Bu koleksiyonlar: "Customers", "Artworks" ve "Transactions" şeklinde şekillendirilmiştir.

## 3.3 BLOK ZİNCİR TABANLI UYGULAMA

Sanat galerisi web uygulamasının blok zincir destekli versiyonu için DigitalOcean'ın 2 GB hafızalı, 1 CPU barından sanal makinesinde Ubuntu 20.04 işletim sistemi kullanılmıştır. Blok zinciri oluşturmak ve yönetimi için Starport 0.16 kullanılmıştır. Starport'un çalıştırılması için "Go 1.16" kurulumu yapılmıştır. Starport'un önyüz uygulaması "Vue.js" ile çalışmaktadır. Sunucuda bu paketlerin çalıştırılması için "Node.js" ve "npm" de kurulmuştur.

Tendermint BFT (bizans hata toleransı, byzantine fault tolerance) PoS, popüler blok zinciri Bitcoin'in kullandığı yüksek hesaplama kaynağı gerektiren PoW mutabakat algoritması yerine Bizans Generalleri Problemi'ne çözüm sağlayan bir PoS mutabakat algoritmasıdır (Kwon, 2014).

Çalışmada kullanılan Starport ara yüzü, Cosmos SDK ile blok zinciri tabanlı uygulamalar geliştirilmek için kullanılmıştır. Cosmos SDK, Tendermint firmasının geliştirdiği açık kaynak kodlu popüler bir blok zinciri geliştirme yazılımcı kitidir. Cosmos SDK de mutabakat algoritması olarak Tendermint BFT PoS algoritmasını kullanmaktadır.

## 3.4 VERİ TABANLARININ KARŞILAŞTIRILMASI

Sanat galerisi web uygulamasında galeri sahibi envanterini güncellerken, galerinin müşterileri de sahip oldukları sanat eserinin envanter bilgisini ve aldıkları galeriden onaylı sahipliklerini sorgulayabilecekleri bir yazılım kurgulanmıştır. Bu yazılım 3 farklı veri tabanı sistemi kullanılarak oluşturulmuştur.

Veri tabanlarının karşılaştırılması için üç sistemin de benzer işlemler yapması hedeflenmiş ve bu işlemlerin gerçekleşmesi sırasında performans, işlevsellik, güvenlik, yazılım geliştirme yönünden farklarının analiz edilmesi amaçlanmıştır.

### 3.4.1 Kayıt Ekleme

Galeri sahibi, sanat galerisi uygulamasında envantere yeni bir sanat eseri eklemek istediğinde üç ana yöntem kullanabilmektedir, bunlar:

1. Komut ara yüzü ile direkt veri tabanlarına veri eklemek

2. Veri tabanı yönetim sistemlerinin grafik ara yüzünü kullanarak veri eklemek (Phpmyadmin, Strapi, Cosmos SDK)

3. Özel olarak geliştirilen uygulama kullanıcı ara yüzünü kullanarak veri eklemek

şeklindedir.

Bir JSON dosyası ya da CSV (virgülle ayrılmış değerler, comma seperated values) dosyası toplu şekilde girdi eklemek için kullanılabilir.

Yeni kayıt eklemek, sistemlerin performanslarının karşılaştırılabileceği görevlerden biridir. Örnek veri olarak Şekil 3.2'de gözüken 150 satırlık bir CSV dosyası, toplu olarak sanat galerisi sahibinin elindeki envanteri web uygulamasına tek seferde eklemesi senaryosunun üç sistemin de performans olarak anlamlı şekilde ölçülebildiği -arada başka bir ara sistem olmadığından- komut ara yüzü kullanılarak gerçekleştirilmesi için hazırlanmıştır. Bu CSV dosyası dosya gezgini ile sunuculara yüklenmiştir.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | type | name | artist | year | owner |
| 2 | Painting | Name 1 | Artist 1 | 1999 | 1 |
| 3 | Painting | Name 2 | Artist 2 | 1999 | 1 |
| 4 | Painting | Name 3 | Artist 3 | 1999 | 1 |
| 5 | Painting | Name 4 | Artist 4 | 1999 | 1 |
| 6 | Painting | Name 5 | Artist 5 | 1999 | 1 |
| 7 | Painting | Name 6 | Artist 6 | 1999 | 1 |
| 8 | Painting | Name 7 | Artist 7 | 1999 | 1 |
| 9 | Painting | Name 8 | Artist 8 | 1999 | 1 |
| 10 | Painting | Name 9 | Artist 9 | 1999 | 1 |
| 11 | Painting | Name 10 | Artist 10 | 1999 | 1 |
| 12 | Painting | Name 11 | Artist 11 | 1999 | 1 |
| 13 | Painting | Name 12 | Artist 12 | 1999 | 1 |
| 14 | Painting | Name 13 | Artist 13 | 1999 | 1 |
| 15 | Painting | Name 14 | Artist 14 | 1999 | 1 |
| 16 | Painting | Name 15 | Artist 15 | 1999 | 1 |
| 17 | Painting | Name 16 | Artist 16 | 1999 | 1 |
| 18 | Painting | Name 17 | Artist 17 | 1999 | 1 |
| 19 | Painting | Name 18 | Artist 18 | 1999 | 1 |
| 20 | Painting | Name 19 | Artist 19 | 1999 | 1 |
| 21 | Painting | Name 20 | Artist 20 | 1999 | 1 |
| 22 | Painting | Name 21 | Artist 21 | 1999 | 1 |
| 23 | Painting | Name 22 | Artist 22 | 1999 | 1 |
| 24 | Painting | Name 23 | Artist 23 | 1999 | 1 |
| 25 | Painting | Name 24 | Artist 24 | 1999 | 1 |
| 26 | Painting | Name 25 | Artist 25 | 1999 | 1 |
| 27 | Painting | Name 26 | Artist 26 | 1999 | 1 |
| 28 | Painting | Name 27 | Artist 27 | 1999 | 1 |
| 29 | Painting | Name 28 | Artist 28 | 1999 | 1 |
| 30 | Painting | Name 29 | Artist 29 | 1999 | 1 |
| 31 | Painting | Name 30 | Artist 30 | 1999 | 1 |

**Şekil 3.2:** 150 satırlık sanat galerisi örnek envanter verisi

### 3.4.2 Kayıt Sorgulama

Sanat galerisi uygulamasında galerideki sanat eserlerini içeren örnek 150 satırlık veri, veri tabanlarına eklendikten sonra içerisinden “Name 1” ifadesi ile başlayan sanat eseri isimlerinin veri tabanında ilgili tablodan sorgulanması Tablo 3.1’deki sorgular yapılmıştır. Bu sorgu ile veri tabanlarındaki belirli bir metin aramasında ve veri tabanlarından beklenen temel görevlerden olan verileri kullanıcılara gösterme işleminde nasıl performans gösterdikleri izlenmiştir. Oluşturulan senaryoda sanat galerisi sahibi ve müşterileri tarafından belirli bir eserin envanterdeki durumunu sorgulayan işlevin gerektirdiği sorgudur.

| **Sistem** | **Sorgu** |
|---|---|
| SQL | select * from artworks where name like 'Name 1%' |
| NoSQL | db.artworks.find( {"name":{ $regex : /^Name 1/ } }) |
| Blok zinciri | artshopd query artshop search-artwork ‘Name 1’ |

**Tablo 3.1:** SQL, NoSQL ve Blok zinciri kayıt arama sorguları

### 3.4.3 Performans Analizi İçin Zaman Ölçümü

İşlemlerin kullanıcıya yansıyan sistem sürelerini ölçmek için sunucuda kurulu olan “Node.js” ve “npm” ile “time” paketi (TooTallNate, 2017) de terminallere kurulmuştur. Bu paket ile çeşitli zaman damgaları oluşturulmaktadır. Çalışma için “time İŞLEM KOMUTU…” şeklinde terminal sorgularından önce “time” komutu eklenerek işlem cevabından sonra tamamlanma süresinin terminalde gözükmesi sağlanmıştır.

# 4. BULGULAR

Bu kısımda, tez çalışması içerisinde tasarlanmış olan veri tabanı yapıları, işlevsellik, yazılım geliştirme ve güvenlik kriterlerine göre karşılaştırılmış ve ortaya çıkan sonuçlar sunulmuştur.

## 4.1 İŞLEVSELLİK

Veri tabanları, mantıksal olarak ilişkili verilerden oluşan düzenli koleksiyonlardır. Bu koleksiyonların web uygulamalarındaki temel rolleri; uygulamanın durumunu, uygulamanın çalışması sırasında güncellenmiş veriyi ya da çalışması için gereken veriyi tutması, verileri aramaya müsaade etmesi, güncelleyebilmesi ve gerektiğinde silmesidir.

SQL ve NoSQL olarak geçmişten günümüze veri tabanı sistemlerinin gelişimiyle çok çeşitli uygulamalara çözüm üreten veri tabanı yönetim sistemleri oluşturulmuştur. Blok zinciri öncelikle taraflar arasındaki güven sorununu çözme ve merkeziyetsiz bir yapı olma hedefiyle geliştirilen finans çözümü Bitcoin ile ünlenip geliştirilmiştir. Yapılan çalışmalarda da görüldüğü üzere blok zinciri teknolojisi ve geleneksel veri tabanı sistemleri birbirlerinden etkilenerek gelişmeye devam etmektedir. Açık, gizli veya PoW, PoS gibi hem blok zinciri türü hem de mutabakat algoritması blok zinciri ile oluşturulan sistemlerin işleyişini köklü olarak değiştirmektedir.

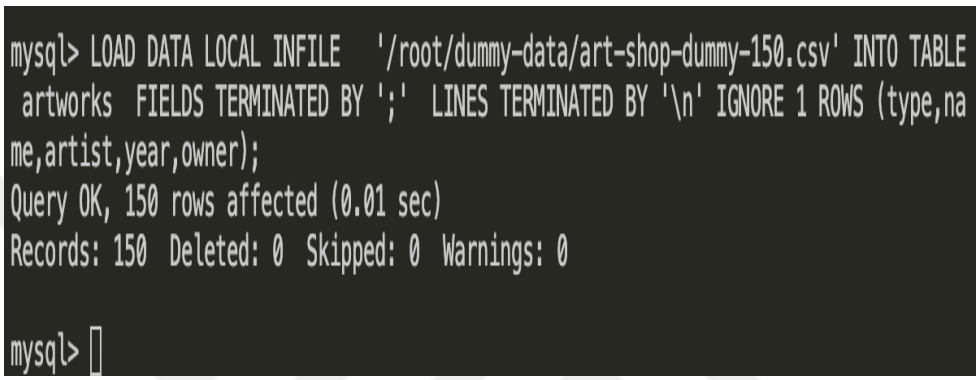
Çalışmada kullanılan açık (izin gerektirmeyen) ve PoS mutabakat algoritmasını kullanan Starport ile oluşturulan blok zinciri geleneksel veri tabanı sistemlerinden işlevleri kendine has şekilde yerine getirirken yapısal kurallarından kaynaklanan bazı yöntemleri uygulamaya dayatmaktadır. Dayatılan yöntemler güvenlik tarafında ve performans tarafında yansımalara sebep olmaktadır. Blok zinciri tabanlı veri tabanı sistemi için kayıt ekleme, sorgulama gibi işlevler test edilmiş ve performansları incelenmiştir.

### 4.1.1 Kayıt Ekleme

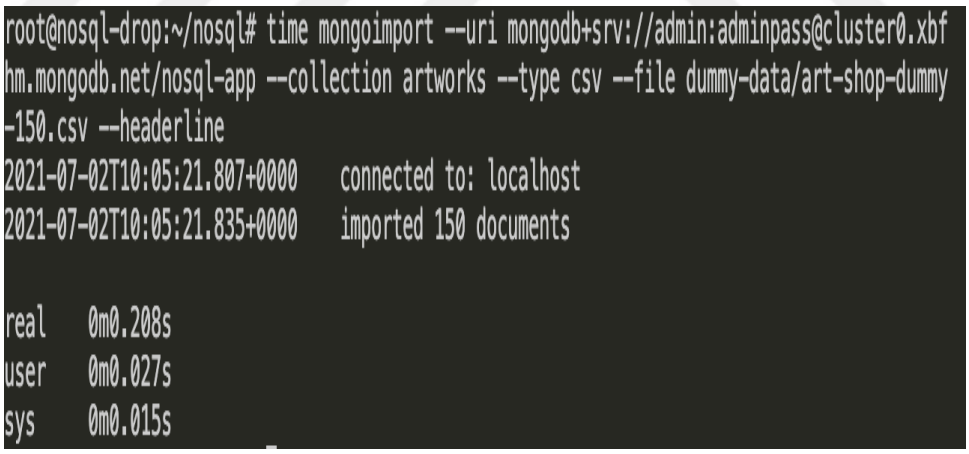
SQL ve NoSQL sistemler, toplu olarak kayıt ekleme görevini sorunsuz şekilde gerçekleştirmiştir. Kayıt ekleme görevini SQL ve NoSQL temelli sistemler, Şekil 4.1 ve Şekil 4.2'te görülebilen terminal çıktılarını üreterek tamamlamışlardır. SQL sistem, NoSQL sisteme göre toplu veri ekleme görevini daha hızlı yerine getirmiştir. Web uygulamasını kullanan bir

kullanıcı gözlemiyle 150 satırlık verideki fark SQL ve NoSQL için günlük kullanımda fark edilebilir seviyede değildir. Performansları birbirine çok yakındır.

SQL için “LOAD DATA LOCAL INFILE” komutu direkt sunucu üzerinden kullanılırken, NoSQL yapı için Mongodb Atlas sunucularına yerel bilgisayar üzerinden bağlantı sağlayan “mongoimport” komutu kullanılmıştır. Tablo 4.1’de sonuç süreleri ve sorguları ayrıca gösterilmiştir.




**Şekil 4.1:** SQL sistem ile 150 satırlık örnek verinin eklenmesi işlemi ve süresi




**Şekil 4.2:** NoSQL sistem ile 150 satırlık örnek verinin eklenmesi işlemi ve süresi

Girdilerin eklenmesi sırasında performans testinin anlamlı olabilmesi için SQL ve NoSQL sistemlerde çoklu veriler kullanılmıştır ancak blok zinciri tabanlı sistemde kullanılan Starport, Tendermint’in BFT PoS şeklinde anılan mutabakat algoritmasını kullandığı için girdi eklemeleri -popüler tüm algoritmalarda olduğu gibi- tek tek olmalıdır. Bir düğümün birden fazla kayıt eklemesi için de mesajların zincir yapısına benzersiz kayıt anahtarı ile eklenmesi ve güvenilirliğinin her seferinde tek tek kanıtlanması gereklidir.

Yapısı gereği tek tek verilerin eklendiği Starport altyapısını kullanan blok zincir tabanlı sistemde analize başlamak için 150 satırlık örnek veri setinden tek bir satırlık girdi blok zincirine eklenmiştir. Bu eklemeyi yaparken öncelikle blok zincirinde dijital varlığın CRUD (oluştur, oku, güncelle, sil - Create, Read, Update, Delete) aksiyonları için fonksiyonların oluşturulması gerekli olduğundan sanat eseri yapısının blok zincirinde bir dijital varlık olarak oluşturulması "starport type artwork Arttype name artist year owner" Starport komutu ile gerçekleştirilmiştir. Ardından Starport'un web uygulamasındaki API sistemi için çeşitli proto dosyalarında ve yapısal dosyalarında ayarlamalar yapılmıştır. Ardından "time artshopd tx artshop create-artwork 'Painting' 'Name2' 'Artist 2' '1999' '0' --from=dbtests" komutu ile "dbtests" kullanıcısı adına blok zincirinin ilk kaydı yapılmıştır. Şekil 4.3'te blok zincirine zincirin onaylayıcısı tarafından eklenen bu ilk kayıt ile ilgili terminal sorgusu ve çıktıları gösterilmektedir. Tablo 4.1'de görülebileceği gibi blok zinciri için tek bir kayıt ekleme süresi, SQL ve NoSQL sistemlerin 150 satırlık veriyi ekleme sürelerinde de uzun sürmüştür.

Blok zinciri için kayıt ekleme analizi -sorgu analizinin belirli bir satır arasından yapılması için- 150 kaydın eklenmesiyle yani tek tek sorguların çalıştırılması şeklinde toplamda 450 saniye süren tekrarlı işlemler bütünüyle sağlanmıştır. Yapılan testlere göre girdi eklenmesi için hızlıdan yavaşa göre sıralama SQL, NoSQL, blok zinciri şeklindedir. Süreler ve sorgular Tablo 4.1'de gösterilmiştir.

```
gitpod /workspace/starport/artshop $ time artshopd tx artshop create-artwork 'Painting' 'Name2' 'Artist 2' '1999' '0'  --from=dbtests
{"body":{"messages":[{"@type":"/bozonet.artshop.artshop.MsgCreateArtwork","creator":"cosmos1u04hd2m8laexpj0msajps8smt9yf9d9kx8cvv8","
Arttype":"Painting","name":"Name2","artist":"Artist 2","year":"1999","owner":"0"}],"memo":"","timeout_height":"0","extension_options"
:[],"non_critical_extension_options":[]},"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":
""}},"signatures":[]}

confirm transaction before signing and broadcasting [y/N]: y
{"height":"331","txhash":"A39468C88C03338591BC555B64550BE66B33135DFCB7C945E5E3A8EC84D292BF","codespace":"","code":0,"data":"0A130A0D4
372656174654172747 76F726B12020801","raw_log":"[{\"events\":[{\"type\":\"message\",\"attributes\":[{\"key\":\"action\",\"value\":\"Cre
ateArtwork\"}]}]}]","logs":[{"msg_index":0,"log":"","events":[{"type":"message","attributes":[{"key":"action","value":"CreateArtwork"
}]}]}],"info":"","gas_wanted":"200000","gas_used":"45824","tx":null,"timestamp":""}

real    0m1.807s
user    0m0.071s
sys     0m0.032s
```

**Şekil 4.3:** Blok zinciri tabanlı sisteme tek girdi eklenmesi ve geçen süre

| Sistem | Sorgu | Süre |
|---|---|---|
| SQL | LOAD DATA LOCAL INFILE '/root/dummy-data/art-shop-summy-150.csv' INTO TABLE artworks FIELDS TERMINATED BY ';' LINES TERMINATED BY '\n' IGNORE 1 ROWS (type,name,artist,year,owner); | 0,010 saniye |
| NoSQL | mongoimport --uri <MongoAtlasAdres> --collection artworks --type csv --file dummy-data/art-shop-dummy-150.csv --headerline | 0,015 saniye |
| Blok zinciri | artshopd tx artshop create-artwork 'Painting' 'Name2' 'Artist 2' '1990' '0' --from=dbtests | 0,032 saniye |

**Tablo 4.1:** SQL, NoSQL ve Blok zinciri tabanlı sistemlerde sorguların tamamlanması için geçen süreler

### 4.1.2 Kayıt Sorgulama

Sanat galerisi uygulamasındaki sanat eserlerini içeren örnek 150 satır içerisinden "Name 1" ifadesi ile başlayan sanat eseri isimlerinin veri tabanında ilgili tablodan sorgulanması için yapılan sorgular ve operasyon süreleri Tablo 4.2'de gösterilmiştir.

| Sistem | Sorgu | Süre |
|---|---|---|
| SQL | select * from artworks where name like 'Name 1%' | 0.0010 saniye |
| NoSQL | db.artworks.find( {"name":{ $regex : /^Name 1/ } }) | 0.046 saniye |
| Blok zinciri | artshopd query artshop search-artwork 'Name 1' | 0.072 saniye |

**Tablo 4.2:** SQL, NoSQL ve Blok zinciri tabanlı sistemlerde sorguların tamamlanması için geçen süreler

NoSQL sistem için kullanılan Mongodb komut ara yüzünde varsayılan olarak ayrıntılı operasyon süresi gösterilmemekteydi. Sorgu öncesinde, operasyon süresinin işlem sonunda gösterilmesi için "setVerboseShell(true)" komutu çalıştırılmıştır.

Blok zinciri tabanlı veri tabanı sistemi için kullanılan Starport'ta varsayılan olarak "type" komutuyla oluşan CRUD fonksiyonlarda sorgu tarafında sadece aşağıdaki komutlar bulunmaktadır. Go dili kullanılarak oluşturulmuş bu sistemde gerçekleştirilen sorgunun çalıştırılması için ekstra olarak sisteme sorgunun geliştirilme yapılarak eklenmesi gerekmektedir.

Starport'ta "type" komutu ile varsayılan olarak gelen sorgu komutları aşağıdaki gibidir:

1. **"list-artwork":** Tüm girdileri çıktı olarak gönderir.
2. **"show-artwork":** Gönderilen kimlik numarasına sahip bir girdi varsa tek bir girdiyi çıktı olarak gönderir.

## 4.2 YAZILIM GELİŞTİRME

SQL tabanlı sistem için kullanılan Mysql veri tabanının ilk yayınlanan versiyonu 1995 yılında, NoSQL sistem için kullanılan Mongodb'nin de ilk versiyonu 2005 yılında yayınlanmıştır. Blok zinciri tabanlı sistem için kullanılan Tendermint firmasının geliştirdiği açık kaynak kodlu Starport ise 2020 yılının başında ilk versiyonunu yayınlamıştır.

Geçmişten mirasları olan geleneksel veri tabanı sistemleri için geliştirilen veri tabanı yönetim sistemleri uygulamaları, sunucu firmalarında tek tık ile kurulan kurulum dosyaları ve bulut ile yönetilen hazır veri tabanı sunucuları bulunmaktadır. Blok zinciri tarafında bulut sunucuda yönetilen sunucu hizmeti sağlayan sınırlı alternatifler bulunmaktadır. Bunlar Oracle (Oracle, 2021) ve Amazon AWS (AWS, 2021) çözümleridir.

Yazılımların geliştirilmesi için geçmişten gelen mirastan dolayı SQL ve NoSQL ile ilgili kaynaklar daha fazladır. Blok zinciri tabanlı veri tabanı sistemleri için Tendermint mutabakat algoritması, Cosmos SDK ve Starport çözümleri uygulama, mutabakat ve ağ katmanları için bir başlangıç noktası önerse de uygulama ihtiyaçlarına göre uzmanlık gerektiren geliştirmeler ve düzenlemeler geliştirmek gerekmektedir. API desteğiyle ve ön yüz uygulaması ile Cosmos

SDK web uygulamalarında blok zinciri tabanlı web uygulamaların geliştirilmesini kolaylaştırmıştır.

## 4.3 GÜVENLİK

Web uygulamalarında üyelik, işlem yetkileri gibi kısımlarındaki yönetim ve işlevsellik; blok zinciri türü seçiminden, algoritma seçimi gibi temel konulara dayanmaktadır. Kullanılan Tendermint BFT PoS mutabakat algoritması ile uygulamaya üyelikler Şekilde 4.4'te gösterilen "config.yml" dosyasındaki tanımlamalarla başlatılmaktadır. Algoritma gereği belirli bir dijital birimi kilitli emanet halde bulunduran düğümler varlık transferi ve transfer onayı gerçekleştirip ödül kazanmaktadır. Yani sisteme üye olan herkesin açık kimlik numaraları ve sahip oldukları hisseler tüm düğümlerle paylaşılmalıdır. Sistemdeki bir girdinin blok zincirindeki onayı için tüm onaylayıcıların en az üçte ikisi, transferi onaylamalıdır (Kwon, 2014).

config.yml > ...

```yaml
accounts:
  - name: dbtests
    coins: ["20000token", "200000000stake"]
  - name: bob
    coins: ["10000token", "100000000stake"]
validator:
  name: dbtests
  staked: "100000000stake"
client:
  vuex:
    path: "vue/src/store"
  openapi:
    path: "docs/static/openapi.yml"
faucet:
  name: bob
  coins: ["5token", "100000stake"]
```

**Şekil 4.4:** Starport config.yml dosyası

SQL ve NoSQL sistemlerde veri tabanı yönetim sistemleri katmanında veri tabanı sunucusuna bağlanabilecek IP adresleri tanımlanabilmektedir. Bu IP adreslerinden gelen istekler dışında tüm istekler engellenebilmektedir. Bunun yanında IP adresi dışında hem veri tabanı hem de uygulama katmanında kullanıcı hesapları tanımlanarak belirli IP adresinden gelen ve kullanıcı adı/parola ile yetkilendirilmiş kullanıcılara yetkilendirme de yapılabilmektedir. Blok zinciri tarafında PoS algoritmasında, girdileri onaylayanlar tarafından eğer hileli bir işlem fark edilirse kilitli şekilde emanet edilen değer, düğüme iade edilmeyebilir ve hileci düğümün hesabı zincirden tamamen silinebilir.

Blok zinciri tabanlı veri tabanlarında blok zinciri teknolojisi sayesinde veri geçmişi baştan sona özgündür ve değiştirilemez. Zincire dahil olan hiçbir sistem kullanıcısı oluşturulmuş ve dallanmış bir zincirin geçmişini değiştiremez. SQL ve NoSQL sistemlerdeki güvenlik duvarları yıkılırsa geçmiş veriler değiştirilebilir. Devasa sayılı satırlara sahip SQL ve NoSQL sistemlerde ancak verilerin değiştirildiği özel olarak incelenirse fark edilebilir.

Blok zinciri tabanlı sistemler kurulurken mimari yapı her zincire özgün olarak kurgulanabilir ancak Starport kullanarak geliştirilen blok zinciri sistemini çalıştıran sunucu kapanırsa ve dosyalar yedeklenmemişse zincir ve zincirin kaldığı durum kalıcı olarak silinecektir. Bu SQL ve NoSQL veri tabanları için de geçerlidir. Veri tabanlarının hepsi yedeklenebilir ve yedeklendiği yerden tekrar başlatılabilir. Cosmos SDK anahtar-değer depolamalı LevelDB'nin Go dili uyarlamasını (Tendermint, 2019) kullanmaktadır. Blok zincirinin kaldığı durum, bir SQL ya da NoSQL veri tabanına yazılarak yedeklenebilir böylelikle sistemde hiçbir düğüm çalışır olmasa da veri tabanındaki yedek ile tekrar blok zinciri yayına başlanabilir.

# 5. TARTIŞMA VE SONUÇ

Devlet uygulamaları, sağlık, finans ve eğlence gibi günlük yaşantının temel alanlarının tamamı vazgeçilmez ve geriye dönülemez şekilde web uygulamalarıyla yönetilmektedir. Birbirine bağlanan cihazların sayısı artarken bu kalabalığın iletişimi, güvenliği ve hızı önem kazanmaktadır. Bu sayının artışıyla beraber merkezi yapıların günlük yaşantı hızına yetişememesi, cihazların otonom karar alma gerekliliği gibi ihtiyaçlardan dolayı merkeziyetsiz ve güven problemi olmayan blok zinciri tabanlı yapıların popülerliği artmaktadır.

Geleneksel veri tabanı çözümlerinde; verilerin tutulduğu birimlere, verilerin birbirleriyle ilişkilerine göre sınıflandırılmıştır. Blok zincirinde ise sisteme katılım ve mutabakat algoritmasına göre sınıflandırılmaktadır. Blok zinciri tabanlı sistemlerin merkeziyetsiz olması, tutarlı olması ve güven problemini ortadan kaldırması ön plana çıkmıştır.

SQL ve NoSQL yapılar, uygulama katmanı kararlarını sorgulamadan veri biçimi uygun olan ve yetki duvarını aşan tüm verileri kaydeder gönderir ve işlerler. Veri tabanlarının sahip oldukları kendi fonksiyonların kullanımı, web dillerindeki gelişmenin hızlılığı ve web dillerinin esnekliğinden dolayı çok yaygın değildir. Verilerin işlenmesi daha çok uygulama katmanında yapılmaktadır. Blok zinciri tabanlı çözümlerde, verilerin sıralanması, aranması gibi işlemler uygulama katmanında yapılmak zorunda olunduğu görülmüştür. Uygulama katmanının zincire erişimi ve yetkileri, blok zincir teknolojisinin türüne (açık, gizli vb.) ve mutabakat algoritmasına (PoW, PoS vb.) bağlıdır. Blok zinciri teknolojileri özellikle mutabakat algoritmaları tarafında çeşitlenmiş durumdadır. Yetkiler ve zincire katılım, birden fazla paydaşın güvenle kullanması amaçlanan web uygulamalarında verilerin işlenmesi için en kritik noktalardan biridir.

İncelenen izin gerektirmeyen (açık), Tendermint BFT PoS algoritmasını kullanan blok zinciri tabanlı veri tabanı sistemi, web uygulamalarında kullanılmak üzere geliştirilmiş kütüphaneler kullanılarak oluşturulmuştur. Analizin yapılması için yapılan çalışmalara ve sonuçlara göre özellikle yazılım geliştirmek için blok zinciri teknolojileri SQL ve NoSQL sistemlerine göre oldukça az gelişmiş kalmaktadır. SQL ve NoSQL yapılar, tek tuşla kurulabilen yapılar halinde bulut çözümler içerisinde yer alırken hem sistemsel kararların karmaşıklığı gereği hem de geleneksel çözümlere göre yeni bir çözüm olmasından dolayı blok zinciri ile bir yazılım geliştirmek daha fazla uzmanlık ve zaman gerektirmektedir. Hızlıca denemeler yaparak pazar

ve müşteri uyumunu yakalamayı hedefleyen web uygulamaları için blok zinciri ile kurulmuş bir sistem daha fazla maliyet demektir.

Performans tarafında yapılan testlerde de açık şekilde görüldüğü gibi blok zinciri teknolojisi SQL ve NoSQL veri tabanı teknolojilerine göre yapısal özelliklerinden dolayı oldukça yavaş kalmaktadır. Web uygulamalarında blok zinciri teknolojisinin tercih sebebi olabileceği kavramlar merkeziyetsiz karar verme olanağı, verilerin geçmişinin değiştirilmemesi ve güven problemini zorlu matematik formülleriyle ortadan kaldırması olarak belirtilebilir.

Blok zinciri tabanlı sistemlerin karmaşıklığı, yazılım geliştirme bariyerinin yüksek olması ve işlem performansındaki yavaşlığa rağmen dünyayı değiştiren birçok büyük uygulamada tercih edilmesinin en büyük nedeni güvenlik ve tutarlılık konusundaki sağlamlık olduğu görünmektedir. SQL ve NoSQL güvenlik duvarları geçildikten sonra ya da sistem yöneticisi geçmiş bir veriyi dilediği gibi değiştirebilir. Blok zincirine işlenmiş geçmiş bir işlem işlendiği imzalarla ve bilgilerle zincir var oldukça değiştirilemez şekilde kalacaktır. Algoritmaların hata toleransı farklı da olsa blok zinciri mutabakat algoritmaları hileci düğümlere karşı çözüm sağlamaktadır. Sayı olarak üstünlük hileci olmayan düğümlerde olduğu sürece blok zincir tabanlı sistem yanlış veriyle yanıltılamaz durumdadır. Kişiler ve kurumlar merkezi bir yapıya güven duymak zorunda olmadan veri transferi sağlayabilmektedir. Bu ihtiyaçlar için blok zinciri tabanlı veri tabanı sistemlerin, geleneksel veri tabanlarıyla beraber gelişerek daha fazla web uygulamasında kullanılacağı gözükmektedir.

Yapılan tez çalışmasında SQL, NoSQL ve blok zinciri tabanlı veri tabanı sistemi karşılaştırılırken Tendermint BFT PoS algoritmasını kullanan açık bir blok zinciri kullanılmıştır. Bu sistemlerle bir sanat galerisi uygulaması için çeşitli veri tabanı işlemleri incelenmiştir. Farklı blok zinciri mutabakat algoritmaları, farklı uygulama ihtiyaçları ve farklı veri tabanları işlevleri analiz edilerek bu çalışma genişletilebilir.

# KAYNAKLAR

Jazayeri, M., 2007. Some Trends in Web Application Development. *Future of Software Engineering (FOSE '07)*.

MDN Web Docs, 2021. *Populating the page: how browsers work.* [Çevrimiçi] https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work [Erişim Tarihi: 21 Mart 2021].

Abraham Silberschatz, H. F. K. S. S., 2010. Chapter 1. %1 *Database System Concepts, 6th Edition.* pp. 6-15.

Jeff Hoffer, R. V. H. T., 2016. The Database Environment and Development Process. %1 *Modern Database Management.* Pearson Education Limited, p. 41.

Andreas Meier, M. K., 2019. %1 içinde*SQL & NoSQL Databases - Models, Languages, Consistency Options and Architectures for Big Data Management,* pp. 6-13.

Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Çevrimiçi] https://bitcoin.org/bitcoin.pdf, [Erişim Tarihi: 10 Nisan 2021]

Blockchain.com, 2021. *Bitcoin Blok Zinciri Boyutu.* [Çevrimiçi] https://www.blockchain.com/charts/blocks-size, [Erişim Tarihi: 15 Nisan 2021].

Mayank Raikwar, D. G. G. V., 2020. *Trends in Development of Databases and Blockchain.* Seventh International Conference on Software Defined Systems.

Tendermint Inc., 2017. *What is Cosmos SDK?.* [Çevrimiçi] https://v1.cosmos.network/intro, [Erişim Tarihi: 5 Mayıs 2021].

Ferdous, M. S. &. C. M. &. H. M. &. C. A., 2020. Blockchain Consensus Algorithms: A Survey.

Blockchain.com, 2021. *Ortalama Onay Süresi.* [Çevrimiçi] https://www.blockchain.com/charts/avg-confirmation-time, [Erişim Tarihi: 5 Mayıs 2021].

QuantumMechanic, 2011 . *Proof of stake instead of proof of work.* [Çevrimiçi] https://bitcointalk.org/index.php?topic= 27787.0, [Erişim Tarihi: 11 Mayıs 2021].

Kwon, J., 2014. *Tendermint: Consensus without Mining.* [Çevrimiçi] https://tendermint.com/static/docs/tendermint.pdf, [Erişim Tarihi: 5 Mayıs 2021].

TooTallNate, 2017. [Çevrimiçi] https://www.npmjs.com/package/time, [Erişim Tarihi: 1 Mayıs 2021].

Oracle, 2021. *Oracle Blockchain Platform Cloud Service.* [Çevrimiçi] https://www.oracle.com/blockchain/cloud-platform, [Erişim Tarihi: 10 Mayıs 2021].

AWS, 2021. *Amazon Managed Blockchain.* [Çevrimiçi] https://aws.amazon.com/tr/managed-blockchain, [Erişim Tarihi: 10 Mayıs 2021].

Tendermint, 2019. *Tendermint DB.* [Çevrimiçi] https://github.com/tendermint/tm-db, [Erişim Tarihi: 20 Nisan 2021].

Leslie Lamport, R. S. M. P., 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3), pp. 382-401.

Coşkun, Ö., 2019. *İlişkisel ve ilişkisel olmayan veritabanı yönetim sistemlerinin karşılaştırılması ve performans analizi,* İÜC, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Ana Bilim Dalı.