T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

NESNELERİN İNTERNETİ ODAĞINDA BÜYÜK VERİ İÇİN TAHMİNE DAYALI ANALİTİK MİMARİSİ

YÜKSEK LİSANS TEZİ

Oğuzhan YÜCE

Akıllı Sistemler Mühendisliği Anabilim Dalı

Tezli Yüksek Lisans Programı

ARALIK 2022

T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

NESNELERİN İNTERNETİ ODAĞINDA BÜYÜK VERİ İÇİN TAHMİNE DAYALI ANALİTİK MİMARİSİ

YÜKSEK LİSANS TEZİ

Oğuzhan YÜCE
(19324814007**)**

Akıllı Sistemler Mühendisliği Anabilim Dalı

Tezli Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Ergün GÜMÜŞ

Aralık 2022

BTÜ, Lisansüstü Eğitim Enstitüsü'nün 19324814007 numaralı Yüksek Lisans Öğrencisi Oğuzhan YÜCE, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "NESNELERİN İNTERNETİ ODAĞINDA BÜYÜK VERİ İÇİN TAHMİNE DAYALI ANALİTİK MİMARİSİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Dr. Öğr. Üyesi Ergün GÜMÜŞ**
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Dr. Öğr. Üyesi Ergün GÜMÜŞ**
Bursa Teknik Üniversitesi

Doç. Dr. Erdem YAVUZ
Bursa Teknik Üniversitesi

Doç. Dr. Metin BİLGİN
Bursa Uludağ Üniversitesi

Teslim Tarihi : Aralık 2022
Savunma Tarihi : 20 Aralık 2022

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

İNTİHAL BEYANI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Oğuzhan YÜCE

İmzası :

Eşime,

ÖNSÖZ

Günümüzde veri en önemli bileşenlerden biri olarak hemen her iş sürecine katkı koymakta. Verinin saklanması, aktarılması, işlenmesi ve elde edilen öngörülerin raporlanması gibi konulardaki imkan ve kabiliyetler şirketler için birer rekabet unsuruna dönüşmüş durumda. Çoğu faaliyet alanı veriden etkin faydalanabilmek için süreçlerini dijitalleştirmekte ve verimliliği artıracak yapay zeka projeleri yapmayı hedeflemektedir. Nesnelerin interneti yaklaşımının üretim alanlarında benimsenmesi ve yaygınlaşması neticesinde büyük veri yönetimi ve büyük veri üzerinde analitik süreçlerin işletilmesi belirli gereksinimleri de beraberinde getirmektedir. Bu yönüyle, tez çalışmamın verinin yaşam döngüsündeki hemen her bir unsura temas ederek bu alanda araştırma yapan insanlara rehberlik edebilecek bir çalışma olduğu kanaatindeyim.

Birçok kişinin emeğinin bir yansıması olan çalışmamda ilk olarak, kıymetli danışman hocam Dr. Öğr. Üyesi Ergün GÜMÜŞ'e teşekkürlerimi sunmak istiyorum. Tez konusunu seçerken ilgi alanlarıma ve isteklerime yüksek hassasiyet gösteren, yüksek lisans eğitimim boyunca desteklerini esirgemeyen ver her durumda hoşgörüyle yaklaşan saygıdeğer hocama sonsuz teşekkürlerimi sunarım.

Teknik imkanların sağlanması noktasında desteğini her zaman hissettiğim, her yönüyle önümü açan saygıdeğer yöneticim Sn. Vedat DAVARCIOĞLU'na teşekkürü borç bilirim. CITS Bilişim ve Yazılım A.Ş firmalarına, tüm yöneticilerime ve çalışma arkadaşlarıma çok teşekkür ederim.

Hemen her konuda birlikte gayret gösterdiğim kıymetli meslektaşım Utku ASLAN'a paylaşımcı kişiliği ve çalışmama olan katkılarından dolayı ayrıca teşekkür ederim.

Hayatım boyunca koşulsuz ve sonsuz sevgileriyle yanımda olan biricik annem ve babama, tanıştığım günden beri hayatımın her alanında bana eşlik eden, ışıltısıyla ruhumu aydınlatan değerli eşime, sırtımı dayadığım kıymetli kardeşime sonsuz teşekkürler.

Aralık 2022

Oğuzhan YÜCE
Ar-Ge Yapay Zeka Uzmanı

İÇİNDEKİLER

Sayfa

KISALTMALAR

ATYD : Atomik, Tutarlılık, Yalıtım, Dayanıklılık
API : Application Programming Interface
CRUD : Create, Read, Update, Delete
CSV : Comma Separated Values
GİD : Genişletilebilir İşaretleme Dili
HTTP : Hyper Text Transfer Protocol
IoT : Internet Of Things
IP : Internet Protokol
JNG : JavaScript Nesne Gösterimi
JSON : JavaScript Object Notation
MQTT : Message Queuing Telemetry Transport
PBF : Protocolbuffer Binary Format
REST : Representational State Transfer
SaaS : Software as a Service
SQL : Structured Query Language
TCP : Transmission Control Protocol
UPA : Uygulama Programlama Arayüzü
VAD : Virgülle Ayrılmış Değerler
XML : Extensible Markup Language

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

NESNELERİN İNTERNETİ ODAĞINDA BÜYÜK VERİ İÇİN TAHMİNE DAYALI ANALİTİK MİMARİSİ

ÖZET

Teknolojik gelişmeler, araştırmacıların imkan ve kabiliyetlerini gün geçtikçe arttırmaktadır. Yapay zeka ve ilintili alanlarda donanımsal, yazılımsal ve hesapsal tekniklerdeki gelişmelerle birlikte veri merkezli iş süreçleri önem kazanmıştır. Bu durum veri saklama, veri aktarma ve veri işleme süreçlerinin yönetilmesi gereksinimini doğurmuştur. Üretim alanlarındaki her bir cihazın veri kaynağı olarak görüldüğü nesnelerin interneti yaklaşımı ile beraber büyük miktarda veri oluşması ve bu verilerden gerçek zamanlı veya gerçeğe yakın zamanlı çıktılar üretilmesine yönelik birçok çözüm literatürde yer bulmuştur. Bulut teknolojileri kullanılarak büyük veri ve nesnelerin interneti başlıkları altında birçok çalışma yapılmış ve çeşitli problemlere odaklanan çözümler ortaya atılmıştır. Bu gelişmeler kurumlarda paradigma değişimlerini de beraberinde getirmiş ve büyük veri enstrümanlarını kullanabilen, analitik süreçleri işletebilen anahtar kişilerin endüstriyel ortamlarda konumlandırılması yönünde bir eğilim oluşmuştur. Kullanıcılar özelinde programlama becerisinin öneminin azaldığı, kodsuz veya az-kodlu yazılım araçlarını kullanma becerisinin ise önem kazandığı görülmektedir. Verinin elde edilmesinden, iş zekası veya operasyonel gösterge araçlarınca raporlanmasına kadar birbirine entegre birçok süreç mevcuttur. Dijitalleşme, izlenebilirlik ve analitik kabiliyetin arttırılması gibi hedefler kuruluşların öncelikleri arasındadır. Analitik kabiliyetler bağlamında makine öğrenmesi algoritmalarının işletilmesi, modellerin eğitilmesi, eğitilen modellerin yazılım servisi olarak sunulması ve çıktılarının çeşitli dijital ortamlara raporlanması başlıkları da büyük veri süreçleriyle entegre bir şekilde çalışmak durumundadır. Bu tez çalışmasında bulut-yerel ortam gereksinimleri de dikkate alınarak nesnelerin interneti odağında büyük veri için tahmine dayalı analitik mimarisi paylaşılmıştır. İlgili mimari yapının ortaya konmasında, tekniğin geldiği son seviye üzerinden işlevsel kırılımlarla birlikte çeşitli açık kaynak kodlu yazılımlara ve yazılım geliştirme platformlarına da işaret edilmiştir. Büyük verinin saklanması ve işlenmesi konusunda ortaya atılmış veri gölü, gölevi benzeri kavramların yanı sıra akan veri işleme konusundaki tasarımsal yaklaşımlar izah edilmiş, kavramların karşıladığı anlama hizmet eden yazılım araçları da örneklenmiştir. Benzer şekilde nesnelerin interneti yaklaşımında literatürde yer bulmuş, olgunluk kazanmış kavramlar ve yazılım araçları da örneklenmiştir. Sunulan mimari hem yazılım geliştiricilerin, hem anahtar kullanıcıların hem de son kullanıcıların faydalanabileceği verinin tüm hikayesini kapsayacak nitelikte bileşenler içermektedir.

Anahtar kelimeler: Büyük Veri, Nesnelerin İnterneti, Akan Veri, Makine Öğrenmesi, Veri Gölü, Gölevi

PREDICTIVE ANALYTICS ARCHITECTURE FOR IOT FOCUSED BIG DATA

SUMMARY

Technological developments increase possibilities and capabilities of researchers day by day. Data-centered business processes gain importance by the developments in hardware, software and computational techniques in artificial intelligence and related fields. This situation leads to requirement of managing data storage, data transfer and data processing. Spread of internet of things approach, where each device in the production areas is seen as a data source, provides solutions for the creation of large amounts of data and the production of real-time or near-real-time outputs. Many studies point out various problems and solutions under the big data and internet of things by using cloud technologies. These developments bring paradigm shifts and a tendency in institutions to position key people who can use big data instruments and operate analytical processes in industrial environments. Importance of programming skills for users decrease, while importance of ability to use software tools with no code or low code increase.There are many integrated processes from obtaining data to reporting by business intelligence or operational dashboarding tools. Goals such as increasing digitalization, traceability and analytical capability are among priorities of the organizations. In the context of analytical capabilities, executing machine learning algorithms, training models, deploying the trained models as software services and reporting their outputs to various digital environments also have to work in integration with big data processes. This thesis, presents a predictive analytics architecture for big data with the focus of the internet of things, taking into account cloud-native environment requirements. Open source softwares and software development platforms are also indicated, along with functional breakdowns over state of the art to reveal the relevant architectural structure. In addition to concepts such as data lake and lakehouse, design approaches to streaming and stream processing are explained theoretical and exemplified by software tools. Similarly, concepts and software tools related with Internet of Things approach are also exemplified. The presented architecture includes components that cover the entire story of the data and can be benefitted from by software developers, key users and end users.

Keywords: Big Data, Internet of Things, Streaming, Machine Learning, Data Lake, Lakehouse

1. GİRİŞ

Büyük veri kavramı pek çok açıdan günümüzdeki araştırmalara ve iş yapış biçimlerine etki etmektedir [1]. Bu etkilerden dolayı büyük veri literatürde çığır açıcı teknolojik gelişme şeklinde nitelendirilmiş ve organizasyonların bu alandaki gelişmeler kaynaklı ortaya çıkan fırsatları sosyoekonomik değere nasıl dönüştürülebileceği üzerine çalışmalar yapılmıştır [2]. Büyük ölçekteki veriden değer üretmek için büyük veri analitiğine ihtiyaç duyulmuştur. Hesaplama ünitelerinin verimli bir şekilde çok yüksek miktardaki veriden analitik yapabilmelerine yönelik yazılımsal ve donanımsal gelişmeler ortaya atılmış, çeşitli algoritmalar ve sistem tasarımları paylaşılmıştır [3]. Nesnelerin interneti, 5G ve bulut teknolojilerinin verinin üretilmesinde, aktarılmasında, saklanmasında ve işlenmesinde önemli katkılar sunması, endüstriyel üretim ortamlarından elde edilen verinin irileşmesine sebep olmuştur [4]. Veri hacmindeki artış incelendiğinde üstel bir grafik seyrettiği görülmektedir [5]. Başka bir açıdan bakıldığında, büyük veri ekseninde dönen ve etkileşen farklı farklı teknolojik kırılımların mevcudiyeti dikkat çekicidir.

1.1 Tezin Amacı ve Önemi

Veri miktarı ve çeşitliliği arttıkça veriyi faydalı hale getirme, veriden çıkarım yapma gibi analitik süreçler kıymet kazanmıştır. Endüstri 4.0 yaklaşımıyla birlikte veri toplama sistemleri endüstrinin günemini yoğun bir şekilde meşgul etmektedir [6]. Şirketlerin endüstriyel olgunluk veya endüstriyel hazırlık seviyelerinin ölçümünde veri toplama, aktarma, saklama, işleme ve iş süreçlerine entegre etme kabiliyetleri önemli bir gösterge olarak kullanılmaktadır [7]. Bu kısıma kadar aktarılmış bilgiler ışığında veriden bilgeliğe uzanan yolda, büyük veri ve büyük veri ekseninde dönen nesnelerin interneti yaklaşımının birbirine entegre olduğu analitik bir sistem tasarımı endüstriyel ortamlar için kritik bir hal almıştır.

Bu tez çalışmasının amacı, bulut teknolojileriyle uyumlu, özel veya genel bulut ortamlarında kullanılabilecek, analitik süreçlerin işletilebileceği bir büyük veri altyapısını nesnelerin interneti odağında bir tasarım olarak paylaşmaktır. Nesnelerin

interneti yaklaşımı da kendi içerisinde bir tasarım sürecini içerdiğinden, tez çıktısı alt sistem kurgusunu içine alan bir sistemler bütünü olarak değerlendirilebilir. Bu yönüyle endüstriyel ortamlarda veriden değer yaratılmasına yönelik önem arz etmektedir.

1.2 Tezin Organizasyonu

Bu tez altındaki çalışmalar 11 başlıkta kategorize edilmiştir.

Giriş bölümünde tezin amacı, önemi ve organizasyonuna yönelik detaylar aktarılmıştır.

İkinci bölümde, nesnelerin interneti odağında büyük veri, nesnelerin interneti ve analitik süreçlere yönelik literatürdeki çalışmalara ait kısa bilgiler paylaşılmıştır.

Üçüncü bölümde veri tabanı, veri ambarı, veri gölü ve gölevi kavramları ele alınmış böylelikle büyük veri saklamayla ilgili bileşenler ifade edilmiştir.

Dördüncü bölüm, akan veri sistemlerinin bileşenlerini içermektedir. Akan veri işleme mimarileri burada paylaşılmıştır.

Beşinci bölümde veri dönüştürme süreçleri tanımlanmıştır. Analitik motor ve veri tablo formatlarının veri dönüşümündeki yerine vurgu yapılmıştır.

Altıncı bölüm, veri görselleştirme ve raporlama araçlarının tanıtıldığı bölümdür. Nesnelerin interneti odağında oluşacak büyük verinin görselleştirmesi bu bölümde aktarılmıştır.

Yedinci bölüm, büyük veri mimari paradigmalarına odaklanır. Büyük veri bileşenlerinin çeşitli değerler zinciri şeklinde ele alındığı mantıksal veri ambarı, veri yapısı ve veri ağı paradigmalarını ifade eden bölümdür.

Sekizinci bölümde, nesnelerin interneti yaklaşımı ve bu yaklaşım etrafında olgunlaşmış veri iletme protokolü, mesaj aracısı gibi kavramları barındıran nesnelerin interneti platformu paylaşılmıştır.

Dokuzuncu bölümde, verinin anlamlandırılmasında en önemli operasyonlardan biri olan makine öğrenmesi operasyonları ve bu operasyonların büyük veri bileşenleriyle etkileşimi izah edilmiştir.

Onuncu bölümde, tez çalışmasının temel amacı olan nesnelerin interneti odağında büyük veri için tahmine dayalı analitik mimarisi önerilmiştir.

Onbirinci bölümde, tez çalışması kapsamındaki araştırmalar neticesinde varılan sonuçlar ve çözüm üretilen noktalar ele alınmış, çözülmeyi bekleyen yeni problemlere ve yeni araştırma alanlarına işaret edilmiştir.

2. LİTERATÜR ARAŞTIRMASI

Literatürde büyük veri çatısı altında veri tabanı, veri ambarı, veri yapısı, veri ağı, veri gölü [8] ve gölevi [9] gibi verinin saklanmasına ve sorgulanmasına yönelik birçok kavram bulunmaktadır. Büyük verinin işlenmesi başlığındaysa kappa & lambda mimarileri [10] üzerinden akan veri hatlarına yönelik tasarımlar literatürde kendine yer bulmuştur. Büyük veriye giden yolda veri tabanları çok sık başvurulan bir enstrüman olarak karşımıza çıkmaktadır. Yapılandırılmış ve yapılandırılmamış verinin saklanması ve sorgulanmasına yönelik farklı veri tabanı çeşitleri çalışılmıştır. Analitik ve raporlama ihtiyaçları arttıkça, yapılandırılmış veri tabanı üzerinde saklanan verilerin sorgulanması süreçlerini optimize edecek tasarımlara yönelinmiştir. Çevrimiçi işlemsel süreçler ile operasyonlarını gerçekleştiren klasik veri tabanı yaklaşımlarından, çevrimiçi analitik süreçler ile operasyonlarını gerçekleştiren analitik veri tabanları örüntüsüne ihtiyaç duyulmuştur. Yapılandırılmış ve dönüştürülmüş veriyi üzerinde tutan, analitik veri tabanı olarak ele alınabilecek bu yaklaşım "veri ambarı" olarak ifade edilmiştir. İki veri tabanı tasarımı arasındaki temel fark, veri ambarı yapısının kolon yönlü olmasından kaynaklanmaktadır.

Analitik veri tabanları verinin yönetimi, iş zekâsı ve raporlama süreçlerinde birçok imkân ve kabiliyet sağlamış olsa da, ileri analitik çalışmalar için ancak limitli kullanım durumunlarında tek başına yeterli gelmektedir. Veri kümesinin büyümesi durumuna ilaveten ses, video ve metin gibi yapılandırılmamış verilerden makine öğrenmesi modellerinin eğitilmesi gibi kullanım durumlarında veri ambarlarının saklama ve sorgulama kabiliyetleri ihtiyaçlara tam olarak cevap verememektedir. Bu problemlerden ötürü ham verilerin düşük maliyetli saklama alanlarında, Apache Parquet [11] gibi açık dosya formatlarında tutulduğu ve bu veri dosyalarına bir uygulama programlama arayüzü (UPA) üzerinden erişim sağlanabilen veri gölü tasarımı ortaya atılmıştır. 2003-2004 yıllarında Google tarafından arama motorlarıyla alakalı paylaşılan bir dizi çalışmada, verileri makineler üzerinde dağıtık bir şekilde saklayabilen Google Dosya Sistemi [12] ve Google MapReduce sistemi [13], bu tasarımın atası olarak nitelendirilmektedir [14]. Apache Hadoop [15] ekosisteminin yaygın kullanımıyla evrimi devam eden bu süreçte, depolama alanı olarak Hadoop Dosya Sistemi kullanılmıştır. 2015'li yıllardan itibaren Hadoop Dosya Sistemi yerini, S3 [16] gibi bulut veri göllerine bırakmıştır. Böylelikle ilk nesil diyebileceğimiz veri

ambarı merkezli analitik platform yaklaşımlarından, ikinci nesil diyebileceğimiz veri ambarı ve veri gölünü birlikte barındıran iki katmanlı analitik platform yaklaşımlarına geçilmiştir. S3 gibi veri gölü depolama sistemleri sadece düşük-seviye nesne deposu olma özelliğini sağlamaktadır. Nesne deposuna, veri tabanlarının sahip olduğu atomik, tutarlılık, yalıtım ve dayanıklılık (ATYD) gibi prensiplerin kazandırılabilmesi noktasında en önemli soyutlama katmanlarından biri, veri dosyalarına ait o verileri tanımlamayan üstveri yönetiminin yapılabilmesidir [17].

Nesne deposuna ilaveten açık dosya formatları ve üstveri katmanının yerleşmesiyle birlikte ortaya çıkan tasarım "gölevi" olarak nitelendirilmektedir. Gölevinde, nesne deposundaki nesneler, ATYD prensiplerine uygun işlem kabiliyetleri kazanmış ve bir tablo versiyonu mantığında tanımlanır hale gelmiştir. Veri gölleriyle, veri tabanlarıyla etkileşir gibi etkileşmeyi sağlayan bu format, "veri tablo formatı", "veri gölü tablo formatı" veya kısaca "tablo formatı" olarak adlandırılmıştır. Açık kaynak kod dünyasında bu alandaki çalışmalar Apache Hive [18] ile başlayıp, hangi nesnelerin tablonun parçası olduğu hakkındaki bilgileri veri gölü üzerinde Parquet formatında işlem günlüğü olarak saklayan ve tablo başına çok sayıda nesneye ölçeklenmesini sağlayabilen Apache Iceberg [19] ile devam etmiştir. Benzeri bir sistemle, Uber çatısı altında başlayıp açık kaynak kod dünyasına kazandırılan Apache Hudi [20] ise akan verinin veri gölünde saklanması odağında çözümler sağlayan bir tablo formatı olarak karşımıza çıkmaktadır.

Verinin saklanmasına ve sorgulanmasına yönelik ortaya çıkan veri ambarı, veri gölü ve gölevi yaklaşımlarının gelişimine parallel olarak önde gelen makine öğrenmesi kütüphanelerinin (Apache Spark Mllib, Tensorflow vb.) birçoğu bu değişime adapte olmuş ve açıklayıcı veri çerçevesi UPA üzerinden Parquet gibi veri gölü dosya formatlarını okuyabilir hale gelmiştir. Açıklayıcı UPA üzerinden çeşitli optimizasyonlar sağlayabilen araçlar sayesinde (Apache Spark SQL vb.) veri dönüşümleri tembelce [21] gerçekleştirilebildiğinden gölevi tasarımları zengin özellikler kazanabilmektedir.

Büyük veri alanında birçok sürecin verimli bir şekilde yönetilmesine yönelik yenilikler ve gelişmeler olsa da, Endüstri 4.0 ve nesnelerin interneti başlıklarında yönetilmesi gereken farklı süreçler kendini göstermektedir. Nesnelerin interneti yaklaşımında herhangi bir sensör veya cihaz, veri kaynağı olarak değerlendirildiği için cihaz yönetimi ve uzaktan ölçüm (telemetri) mekaniklerinin yönetimi önemli bir başlıktır.

Cihaz verilerinin veya sensör mesajlarının aktarılması, bu aktarımın yükte hafif bir şekilde gerçekleşmesini sağlayacak mesaj kuyruklama protokollerinin gelişimini zorunlu kılmıştır. Mesaj Kuyruklama Telemetri İletimi veya literatürde kabul görmüş İngilizce kısa adıyla MQTT [22] protokolü gibi çeşitli protokoller üzerinden veri aktarım süreçleri işletilmektedir. Endüstriyel ortamlarda birçok kontrol mekanizması eşik değer takibi üzerinden gerçekleştirildiği için aktarılan verilerin işlenmesi ve eşik değerler üzerinden bir takım olayların veya çıktıların türetilmesi gerekmektedir. Yine birçok tesiste görsel indikatörler üzerinden kalite kontrol süreçleri işletildiğinden ve telemetri verilerinin gerçek zamanlı görselleştirilmesi üretim alanlarına önemli çıktılar sağladığından, yönetilmesi gereken bir başka başlık olarak karşımıza çıkmaktadır. Nesnelerin interneti verilerinin saklanması, aktarılması, işlenmesi ve görselleştirilmesi gibi ihtiyaçların bütünleşik yönetimi için nesnelerin interneti çözümleri sağlayan birçok açık kaynak kodlu yazılım aracı (SiteWhere, Thingsboard [23] vb.), bu unsurları bir platform [24] kurgusunda ele almıştır. Yapay zeka entegre çalışmalar üretime katma değer sağladığından ileri analitik amaçlar doğrultusunda büyük veri süreçleriyle nesnelerin interneti platformundaki süreçlerin entegre olarak çalışması gerekliliği doğmuştur. Bu bağlamda ham mesaj verilerinin nesnelerin interneti platformundan çıkarılıp veri ambarı, veri gölü veya gölevi unsurlarına aktarılması ileri analitik uygulamalar için bir kilometre taşı niteliğindedir. Apache Kafka [25] gibi akan veri işleme ve olay deposu olarak kullanılan açık kaynak kodlu yazılım araçları kullanılarak nesnelerin interneti platformu ve büyük veri yönetimi unsurları iletişimi sağlayabilmektedir.

3. BÜYÜK VERİ SAKLAMA

3.1 Veri Tabanı

Veri tabanı, kendi kendini tanımlayan entegre kayıtlardan oluşan bir koleksiyon olarak değerlendirilmektedir. Veri tabanlarının ilk kuşak örnekleri 1960'lı yıllarda, kayıtların başka kayıtları işaretçiler aracılığıyla bağlantıda tutması prensibi üzerine kuruluydu. Navigasyonel veri tabanı sistemi şeklinde ifade edilen bu tasarımı, 1970'li yıllarda ortaya atılan, ticari anlamda büyük etki yapmış ve birçok yazılım aracına yön vermiş ilişkisel veri tabanı modeli takip etti [26]. Bu model 1976 yılında Varlık-İlişki veri tabanı modeli ismiyle anılmaya başlandı ve ilişkisel veri tabanı yönetimi sistemi bu vakitlerde ortaya çıktı [27]. 1980'li yıllara gelindiğinde bilgisayar satışlarındaki devasa artış, ilişkisel veri tabanlarının ticarileşmesi ve çok sayıda yazılım ürününün piyasaya sunulması noktasında kaldıraç etkisi yapmış ve Yapılandırılmış Sorgu Dili (YSD / SQL) standart olarak ele alınmaya başlamıştır. Nesne yönelimli veri tabanı yaklaşımlarının kendine yer bulmasıyla nesne yönelimli programlama dilleriyle uyumlu çalışabilen veri tabanı yönetim sistemleri de ortaya çıkmıştır[28] . Birçok dille uyumlu çalışabilmelerine rağmen burada nesne-ilişki eşleme yazılımları daha çok tercih edilmiştir [28]. 1990'lı yılların ortalarında internetin kullanılabilir olmasıyla istemci-sunucu mantığında hizmet veren veri tabanı sistemleri internet bağlayıcıları (bağlaçları) geliştirdiler. Aynı dönemlerde ilişkisel veri tabanı yönetimi araçlarının sahip olduğu karakteristik özelliklerde ayrışmalar belirmiştir. Bunun en önemli örneklerinden biri, çok boyutlu veri yönetimi yapmaya yönelik gruplama ve birleştirme operasyonlarında verimlilik artışı sağlayan çevrimiçi analitik süreçlerle operasyonlarını gerçekleştiren analatik veri tabanlarıdır [29]. Böylelikle üzerinde yapılandırılmış ve dönüştürülmüş veriyi tutmaya yönelik yaklaşım veri ambarı başlığında ifade edilmiştir [30]. İlgili dönemde internet ve ağ teknolojilerindeki gelişmeler neticesinde heterojen verilerde çoğalma yaşanmış, sabit şema düzeninin yetersiz kaldığı durumlar çoğalmış, düzensiz ve yapılandırılmamış veri yönetimine ihtiyaç duyulmuştur. 1996 yılında Genişletilebilir İşaretleme Dili (GİD / XML) ortaya atılmış, hem içiçe yapılandırılmış veriler hem de şemasız veriler için bir çözüm alternatifi olmuştur [31]. İnternet ve multimedya verilerinin artmasıyla, veri tabanı yapılarında da yapılandırılmamış veriyi yönetebilecek yapılara doğru evrimleşen bir süreç yaşanmış ve 1998 yılında NoSQL ifadesi ilk defa Carlo Strozzi tarafından telafuz

edilmiştir [32, 33, 34]. İlk ifadesiyle ilişkisel olmayan veri tabanlarına atıf yapılsa da yaklaşık on yıl sonra Eric Evans tarafından yeniden irdelecek olan terimin "sadece ilişkisel değil" anlamında ele alınması tavsiye edilecektir [35, 36]. Veri tabanı oluşturan ve kullanan kişiler GİD'in yaygın kullanım alanı bulmasına rağmen yarı-yapılandırılmış veri için optimum olmadığı kanaatini taşıyorlardı. 2001 yılında Douglas Crockford, GİD'in daha basitleştirilmiş bir türü olarak değerlendirilebilecek JavaScript Nesne Gösterimi'ni (JNG) sundu [37]. JNG mantıksal bir veri modeli değil, bir serileştirme formatıdır. GİD gibi karmaşık olan içiçe yapılandırılmış veriler içeren kayıtları modelleyebilmesinin yanı sıra, şemasız veriler için de kullanılabilir. JNG, veriye JNG nesnesi üzerinden erişebilirken GİD'in çözümlenmiş veriye ihtiyacı vardır. JNG veriyi bir anahtar-değer eşlemesi mekaniğiyle saklarken, GİD veriyi bir ağaç yapısı mekaniğiyle saklar. JNG gibi formatlar günümüzde NoSQL veri tabanlarıyla yakın ilişkidedir. NoSQL veri tabanları çizge, döküman saklama ve anahtar-değer saklama veri tabanları gibi alt başlıklar içermektedir. JNG ile yakın ilişkide olan alt tür JNG dökümanları saklama veri tabanı şeklinde ifade edilmektedir [38].

NoSQL veri tabanı sistemleri Amazon, Google, Twitter ve Facebook gibi büyük internet şirketleriyle beraber gelişimini sürdürdü. Bu şirketler veriyi yönetirken geleneksel ilişkisel veri tabanı yönetim sistemlerinin baş etmekte zorlandığı farklı farklı durumlara NoSQL çözümler ve alışılagelmiş yaklaşımlar dışındaki çözümler üzerinden cevap aradılar.

2003-2004 yıllarında Google'ın paylaştığı çalışmalar neticesinde dağıtık veri saklayabilen Google Dosya Sistemi ve Google MapReduce sistemi tanıtıldı. Böylece veri tabanı kullanımında konvansiyonel sınırların dışında yeni yaklaşımlar, yeni kırılımlar belirmiş oldu. Bu gelişme sonrasında, veri saklama terminolojisinde değişiklikler yaratarak veri koleksiyonları ve veri gölü gibi yeni terimlerin ön plana çıkmasına sebep olmuş bir gelişme olarak karşımıza çıkmaktadır.

Günümüzde ilişkisel veri tabanları ve NoSQL veri tabanları dağıtık yapılar şeklinde çalışabilen, gösterge araçlarıyla entegre olabilen ve büyük veriyi idare etmeye yönelik güçlü mekanikler içeren araçlar şeklinde halen iyileşmeye devam etmekte ve gelişimini sürdürmektedir. Çokça açık kaynak kodlu veri tabanı aracı geniş kitlelere erişmiş ve kendi alanlarında önemli etkilere sahip olmuşlardır. Açık kaynak kodlu ilişkisel veri tabanı örnekleri olarak, PostgreSQL [39], MariaDB [40] ve ClickHouse

[41] gösterilebilir. Açık kaynak kodlu NoSQL veri tabanı örnekleri olarak Apache Cassandra [42] ve Apache HBASE [43] verilebilir. MongoDB [44] ve Neo4j [45] gibi NoSQL veri tabanları tam olarak açık kaynak kodlu olmasa da topluluk erişimine açık, kodları erişilebilir versiyonları barındırdıklarından bu alana ilave olarak örnek gösterilebilir.

3.2 Veri Ambarı

Veri ambarı, ilişkisel veri tabanlarının kendi içerisinde işlevsel olarak farklılaşması neticesinde ortaya çıkmış, çok boyutlu veri yönetimi yapmaya yönelik gruplama ve birleştirme operasyonlarında verimlilik artışı sağlayan ilişkisel veri tabanlarının bir türü şeklinde nitelendirilebilir. Veri ambarı oluşturma süreci, çeşitli kaynaklardan gelen verinin bir işin detaylı görünümünü oluşturmak amacıyla yönetildiği bir süreç olarak değerlendirilmektedir [46]. Çevrimiçi işlemsel süreçler ile operasyonlarını gerçekleştiren klasik veri tabanı yaklaşımlarından, çevrimiçi analitik süreçler ile operasyonlarını gerçekleştiren analitik veri tabanları örüntüsüne geçişle birlikte, üzerinde bir iş sürecine yönelik yapılandırılmış ve dönüştürülmüş veriyi tutan, günümüzde "analitik veri tabanı" olarak ele alınabilecek bu yaklaşım, iş zekası ve gösterge araçlarıyla entegre kullanılarak raporlama faaliyetlerinde kritik rol üstlenmektedir. Analitik amaçlarla kullanıldığından kolon yönlü operasyonlarda optimumu hedefleyen yapıdadır. Kolon yönlü operasyonlar sayesinde değişkenler ve değişkenler üzerinden formüller temsil edilebildiğinden raporlama süreçlerinde fayda sağlamaktadır. Geleneksel olarak çevrimiçi işlemsel süreçler ile satır yönlü operasyonlar yürüten veri tabanlarında analitik işlemler veya hesaplamalar için bir değişken için sorgu atıldığında, o değişkenin kolon boyu aldığı değerleri getirebilmek için önce diğer kolonların taşıdığı değerleri getirerek ilgili tüm satırları sorgulayıp sonrasında ilgili değişkenin değerlerini kapsülleyen bir davranış söz konusudur. Örneğin, maaş bilgisini maaş kolonunda tutan bir tabloyu kullanarak ortalama maaş hesabı yapacak olursak, analitik veri tabanları kolon yönlü olduğundan ilgili kolondaki değerleri getirecek ve ortalama hesaplayacaktır. Buna karşın satır yönlü veri tabanında bu hesaplamayı yapabilmek için önce bütün satır değerleri maaş kolonu dışında kolonları da içerecek şekilde getirilecek, sonrasında maaş kolonunun değerleriyle ortalama hesaplanacaktır.

Günümüzde veri ambarı olarak nitelendirilen yapılar veri gölü gibi derin saklama alanları, akan veri araçları, operasyonel gösterge araçları ve iş zekâsı araçlarıyla entegre olabilecek bağlantı araçlarını da içinde barındırmaktadır. Bu özelliklere sahip, dağıtık veri deposu olarak da işlev görebilen açık kaynak kodlu veri ambarına örnek olarak Apache Druid [47] gösterilebilir. Birçok sektörde ticari yazılım aracı olarak karşılaşabileceğimiz kolon yönlü çalışabilen veri tabanlarına örnek olarak Google BigQuery [48], Amazon Redshift [49] ve SAP HANA [50] örnek gösterilebilir.

3.3 Veri Gölü

Veri gölü, büyük veri dünyasında veri analitiği süreçleri için önemli bir bileşen olarak son 20 yılda ortaya çıkmış bir terimdir. Tüm ham verinin saklandığı, analiz için organizasyondaki kişilerin erişimine açık tek bir veri deposu olması fikrine dayanan bir kurgu olarak karşımıza çıkmaktadır [51]. Veri gölü yapısı, veri ambarı yapısından farklı olarak şemalı veya yapılandırılmış veriyi tutmaya zorlamaz. Veri kaynağının sağladığı şekliyle veriyi saklamaya yönelik olduğundan, ham veri deposu olarak değerlendirilebilir. Veri ambarları, yapısı gereği analiz için dönüştürülmüş, düzenlenmiş verileri sakladığından organizasyondaki kişileri, tekleştirilmiş bir veri modeli ve şema yapısına zorlamaktadır. Bu haliyle veri ambarları sadece küçük organizasyonlar için bütünüyle bir veri deposu işlevini yerine getirebilmektedir.

Karmaşıklığı yüksek ve büyük organizasyonlarda bir model ortaya koyabilmek için her biri kendi veri modeline sahip olan, etki alanı odaklı veya iş alanı odaklı tasarım desenlerinden “sınırlı bağlam” deseninde [52] birden çok model gerekmektedir [51]. Bu gereksinimler veri ambarı yapısından veri gölü yapısına geçişi zorunlu kılmıştır. Veri gölünün çıkışında önderlik eden bir başka gereksinim de veri analizinin iş süreçlerine yerleşerek ilgi odağı olmasında yatmaktadır. Derin analizler yapacak kişiler çoğunlukla “veri bilimci” ünvanına sahiptir. Bir veri bilimci geçerli bir model ortaya koyabilmek için ön işleme, öznitelik çıkarma gibi bazı operasyonları işletir ve dolayısıyla model için kullandığı verinin nasıl toplandığına ve kalitesine duyarlıdır. Bu bakımdan veri bilimci için dönüştürülmüş veri kadar ham veri üzerinde çalışabilmek de önemlidir.

Google Dosya Sistemi ve Google MapReduce sisteminin ortaya konulmasıyla kapı aralanan veri gölü yaklaşımı, Apache Hadoop ekosistemiyle birlikte yaygınlaşmış ve kabul görmüştür. Halen daha günümüzde veri gölü olarak kullanılan veri depolarında

Hadoop Dosya Sistemi'nin kullanıldığı görülmektedir. Veri gölü yaklaşımı, veriyi formatından bağımsız bir şekilde saklayabilen ve verinin sorgulandığı ana kadar şema veya meta veri gereksinimi duymayan bir karaktere sahiptir [53]. Bu haliyle şema eksikliğinden veya benzer sebeplerden, verinin anlaşılamaz bir hale gelmesi ve veri gölünün "veri bataklığı"'na dönüşmesi kaygısı dikkate değerdir [54]. Veri gölünde saklanacak veriler hakkındaki verileri barındıran meta veri deposu [55] bu kaygıları gidermek noktasında hafifletici bir temel bileşen olsa da bu sorunları tam olarak çözememektedir. Bu problemlere ilave olarak, büyük veri bağlamında düşünüldüğünde verinin göle yazılması, gölden okunması ve sorgulanması başlıklarında da dikkat edilmesi gereken, veri gölünü bataklığa çevirebilecek bir takım noktalar mevcuttur.

Veri gölü kavramı ham veri için bir büyük veri bileşeni olarak konumlandığından, bir çeşit saklama alanı sunar. Uzun süre bu saklama alanı, sunucuların (sanal ve gerçek) dosya sistemleri üzerinden sağlanmış bir alanla sunulmuştur. Halen geçerli kabul edilen Hadoop Dosya Sistemi de bunun örneklerindendir. Bu yapı bir çeşit saklama alanı yönetimi yapmakta ve dosya sistemi üzerinden dağıtık depolamaya yönelik yapılandırmalar içermektedir. Teknolojik gelişmelerle birlikte büyük veri yönetiminde yazılım-tanımlı saklama alanı kurguları ortaya çıkmış ve dosya sistemi dışındaki saklama tiplerinde de ölçeklenebilir ve dağıtık çözümler sunabilen yapılar kendisine yer bulmuştur. Bu çözümlere örnek olarak Ceph [56] gösterilebilir. Akademik ortamlarda tohumları 2004-2005 yıllarında serpilmiş, sonrasında 2014 yılında Red Hat'in girişimleriyle gelişim ve bakım sürekliliği arttırılmış bir yazılım aracı olarak karşımıza çıkan Ceph, 2018'de Linux Vakfı altında Ceph Vakfı'nın başlatılmasıyla gelişimini sürdürmüştür [57]. Açık kaynaklı yazılım-tanımlı depolama teknolojisi olarak Ceph, dosya sistemi tipi dışında blok ve nesne deposu tiplerini de uygulayabilir. Ceph, sunucuların formatsız disk alanlarına yerleşerek bu üç tipi de uygulayabilecek bir dağıtık depolama alanı sunabilecek şekilde ilgili alanları yapılandıran bir yazılım aracıdır.

Dosya sistemi dışındaki tiplerin uygulanabilmesi, veri gölü bağlamında da önemli etkiler doğurmuştur. Veri gölü özelinde en kritik etkiyi nesne deposu yapmıştır. Nesne deposu, döküman, ses, görüntü ve video dosyaları gibi çeşitli formatlara sahip dosyaları, türden veya boyuttan bağımsız olarak "nesne" adı verilen ayrı bir veri birimi şeklinde saklayan bir yapıdır. Tüm nesneler, nesne deposunda hiyerarşik mimariye

sahip klasörler yerine, düz bir şekilde ad uzayıyla / ad boşluğuyla birlikte saklanır [58]. Bu yapı, dosya hiyerarşisine sahip olmadığı için meta veri yönünden hem daha esnek hem daha zengindir [59]. Nesne deposuna REST UPA ile HTTP protokolü üzerinden erişim sağlanabilirken, iletim TCP/IP protokolü üzerinden sağlanabilir. Oluşturma, okuma, güncelleştirme ve silme gibi komutlar HTTP üzerinden işletilebilir [60]. Diğer bir saklama tipi olan blok saklama tipinden farklı olarak nesne deposu, işletim sisteminin doğrudan erişimine açık olmayıp, uygulamalar ya da diğer bir yorumuyla UPA'lar üzerinden erişim sağlanan bir alandır [61]. Erişimin UPA'lar üzerinden yürütülebilmesi ve doğrudan nesneyle ilgili erişim ve işlem kabiliyeti sunmasından kaynaklı olarak günümüz genel bulut şirketlerinin iş yapış biçimi olan "kullandığın kadar öde" gibi iş modellerine de uyumlu bir yapı ihtiva eder. Bu da uygulamaları için sunucu yerine sadece saklama alınana ihtiyaç duyan yazılım geliştiriciler veya bireysel kullanıcılar için maliyet düşürecek bir fayda sağlamaktadır. Benzer şekilde fonksiyonel olarak ayrışmış, saklama ve saklanan nesneye erişme özelliği sunan bir birim olarak ele alındığında sunucusuz modelde [62, 63] hizmet veren yazılımlar açısından da maliyet düşürücü bir etki sunar.

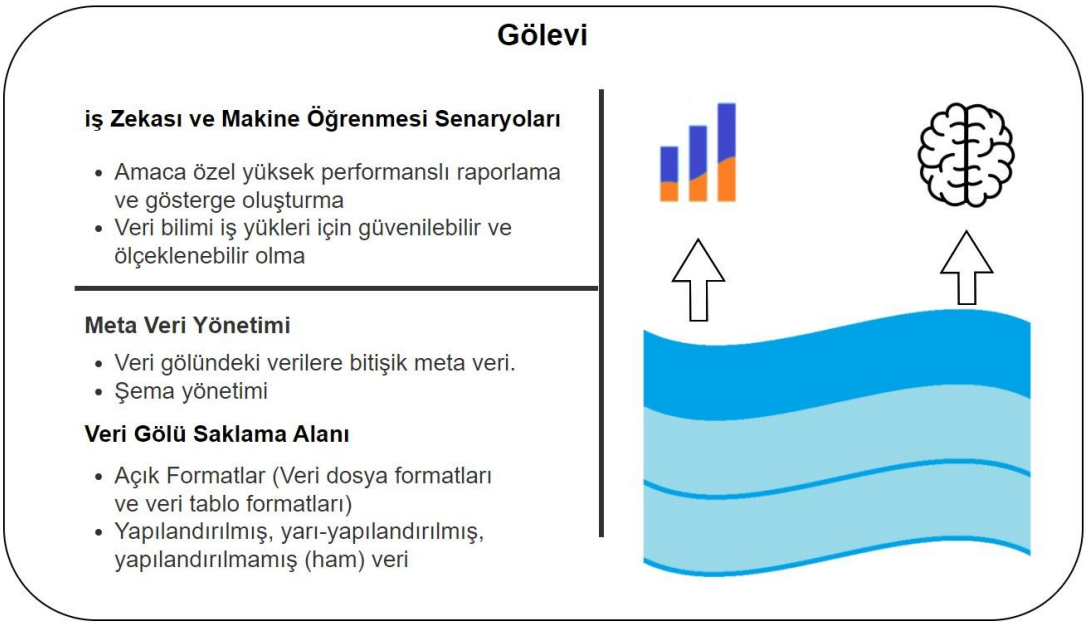
Veri gölünün depo görevi gören saklama alanına göl saklama alanı veya derin saklama alanı da denmektedir [64]. Göl saklama alanının performansı, temelde okuma ve yazmayla ilgili olduğundan bu iki başlıktaki süreçleri optimize edecek çalışmalar yapılmıştır. Veri dosya formatı, meta veri deposu ve veri tablo formatı gibi veri gölünü bir üst seviyeye terfi ettirecek unsurlar 3.4 Gölevi başlığında irdelenmiştir.

3.4 Gölevi

Gölevi [9], veri gölünün yönetiminin maksimum verimde yapıldığı ve veri gölünün metaforik olarak adeta bir veri ambarı kabiliyetlerinde kullanıldığı, bütüncül bir yaklaşımdır. Gölevi tasarımı veri dosya formatı, meta veri deposu ve veri tablo formatı yapılandırmaları tamamlanmış, çeşitli sorgu motorları aracılığıyla sorgu atılabilen ve iş zekası araçlarıyla entegre olabilen bir nesne deposu üzerine yani üst seviye bir veri gölünü merkez alarak inşa edilmiştir [65].

Gölevi, merkezindeki veri gölüne ilave olarak büyük veri işleme araçları (Apache Spark vb.), veri ambarı, iş zekâsı ve makine öğrenmesi araçlarının bir arada kullanıldığı bir platform mantığındadır. Gölevini büyük veri mimari paradigmalarından biri veya bir tasarım örüntüsü olarak ele alan kaynaklar da

mevcuttur [66] ancak bu tez çalışması kapsamında gölevi bir paradigma olarak değil bir platform olarak ele alınmıştır [65]. Gölevi yapısı Şekil 3.1 (esin kaynağı [65]) üzerinden betimlenmiştir.



Şekil 3.1 : Gölevi yapısı

Veri bataklığına dönüşmeyecek bir veri gölünün tasarım süreçleri 4 başlıkta toparlanabilir [67].

- Ham verinin nesne deposunda saklanması.
- Ham veri dosyalarının analiz edilmek amacıyla boyut ve biçim yönünden optimize edilmesi.
- Şema yönetimi ve versiyonlama yapabilecek bir meta veri deposu aracının eklenmesi.
- Optimize edilmiş veri setlerini kullanacak sorgu motorları ve iş zekası araçlarına entegrasyonun sağlanması.

3.4.1 Veri dosya formatı

Verinin göle yazılması veya okunmasını iyileştirmek için tercih edilebilecek dosya formatları mevcuttur. Bu formatlar genel olarak metin tabanlı, kolon yönlü, satır yönlü ve Hadoop'a özel formatlar şeklinde dört başlıkta ele alınmaktadır [68].

Metin tabanlı formatlara örnek olarak JNG (JSON) ve VAD (Virgülle Ayrılmış Değerler, CSV) örnek gösterilebilir. Yaygın kullanılan formatlar olduğu için çoğu programlama dili tarafından serileştirme ve seri durumdan çıkarma işlemlerinde hali hazırda desteklenen formatlardır. Benzer sebeple akan veri süreçlerinde de yaygın bir şekilde kullanılırlar.

Kolon yönlü formatlara Apache Parquet ve Apache ORC [69] örnek olarak verilebilir. Kolon yönlü formatlar, formülleri hesaplamada veya analitik amaçlı sorgularda değişkenleri temsil edebildiğinden veri okumayı optimize eden yapılar olarak değerlendirilebilir. Akan veri süreçlerinde sıklıkla tercih edilen dosya formatlarındandır.

Satır yönlü formatlara Apache Avro [70] ve PBF (Protocolbuffer Binary Format) [71] örnek verilebilir. Avro, çoğunlukla veri serileştirmede kullanılan ve şema açılımı sunan bir saklama formatıdır. Avro, sistemler, programlama dilleri ve veri işleme çatılarıyla veri değişimi yapmaya yardımcı olur. Şema bilgisini, okunması ve yorumlanması kolay olması sebebiyle JNG formatında tutar. Şema kullanımı birkaç durumda kritik önemde olabilir. Örneğin akan veride belirli alanların varlığını mevcut kılmak istiyorsanız veya veri modelinizi ortaya koymak ve sonrasında yaptığınız değişiklikleri idare etmek istiyorsanız şema kullanımının gerekliliği yadsınamaz [72]. LinkedIn şirketindeki bir örnekle ele alacak olursak, veri tabanı değişiklikleri de Avro olaylarının akışı şeklinde temsil edilmektedir [72]. Avro formatına veri yazma operasyonlarında veya kayıtların uygulamalar arasında paylaşıldığı durumlarda sıklıkla başvurulmaktadır.

Veri dosya formatlarının bazıları başlıca özellikleriye Çizelge 3.1'de verilmiştir.

Çizelge 3.1 : Başlıca veri dosya formatları.

Özellikler	Avro	Parquet	ORC
Format Tipi	Satır yönlü	Kolon yönlü	Kolon yönlü
Şema Entegrasyonu	✓	✓	✕
Şema Değerlendirme	✓	✓	✕
Yazma / Okuma	Yazma	Okuma	Okuma

3.4.2 Meta veri deposu

Meta veri, saklanan veri hakkındaki verileri tanımlayan bir ifadedir. Meta terimi bir soyutlama yaratarak yeni bir katmana atıf yapmaktadır. Veri hakkındaki veriler de verinin okuma ve yazma sürçlerini iyileştirmede önemli olmaktadır. Ayrıca veri bilimciler açısından bakıldığında meta veriler makine öğrenmesi modellerinde önemli girdiler oluşturabilecek gruplama ve birleştirme süreçlerine katkı sağlayan unsurlar içerebilmektedir. Veri hakkındaki veriyi oluşturmak, saklamak ve yönetmek de iyi bir büyük veri sistemi yönetimi açısından elzemdir. Bu gibi sebeplerden ötürü, veri gölünde veri dosyalarına ilave meta verilere ait dosyalara da rastlanır [73]. Veri gölü yapısının ana bileşenleri arasında yer alan meta veri yönetimi, veri şemalarının sürdürülebilir yönetimi, hesapsal verimliliğin arttırılması ve veri gölünün daha kullanılabilir olması gibi avantajları beraberinde getirir. Apache Hadoop ekosisteminde kendisine yer etmiş ve genelde SQL sorgu motoru olarak konumlandırılmış Apache Hive, içinde barındırdığı Metastore bileşeniyle günümüzdeki büyük veri ekosistemlerinde de kendisine meta veri deposu olarak yer bulmuştur. Veri tablo formatlarına meta veri yönetimindeki yetenekleriyle katkı sağlayan Hive sayesinde Spark SQL [74], Presto [75] veya Hive sorgu motorları tarafından veri gölüne SQL sorgu mantığında sorgu atılmasına da olanak verir. Bu cümlenin anlaşılması için veri tablo formatı ve sorgu motoru kavramları takip eden başlıklarda açıklanmıştır.

3.4.3 Veri tablo formatı

Veri tablo formatları, veri gölüne alışılagelmiş veri tabanı özellikleri kazandırdığından oldukça caziptir. Dağıtılmış veri dosyalarını tek bir tabloda toplayarak, tıpkı bir veri tabanındaki tablo mantığında işlenmesini sağlar [76]. Bu başlıkta, Apache Hudi [77],

Delta Lake[78] ve Apache Iceberg [79] en çok tercih edilen veri tablo formatları olarak karşımıza çıkmaktadır. Apache Parquet ve Apache Avro veri dosya formatlarını kapsülleyebilen yapısı [80], okuma ve yazma hızını optimize edebilen bileşenler barındırması sayesinde Apache Hudi gelecek vaad eden açık kaynak kodlu bir veri tablo formatıdır. Apache Kafka gibi akan veri araçlarıyla entegre olabilmesi yönüyle de diğer veri tablo formatlarından olumlu yönde ayrıştığı görülmektedir [81].

3.4.4 Sorgu motoru

Sorgu motoru kavramı, büyük veri bağlamında kullanıldığında dağıtık halde saklanmış büyük veriye SQL sorgu diliyle sorgu atabilen motorları karşılamaktadır. Tekniğin bilinen seviyesinde, veri gölündeki verinin iş zekası araçları üzerinden sorgulanabilmesi veya bir takım çıktılar şeklinde raporlanabilmesi gibi kabiliyetler sorgu motorları aracılığıyla sağlanabilmektedir. Bir veri hattı şeklinde örneklemek gerekirse, Hive Metastore ile bir Hudi tablosunu kaydettiğimizde Spark SQL, Hive veya Presto sorgu motorlarıyla ilgili tabloya sorgu atılabilmektedir. Bu sorgu motorlarının, çoğu iş zekası aracına entegre olabilme özelliği dahili olarak mevcuttur. Bu sayede veri gölündeki verinin, bir takım görsel indikatörlere dönüştürülebilmesi ve sunulabilmesi mümkün olmaktadır.

4. AKAN VERİ

Akıllı sistemlerin kulanımının artmasıyla sensör kullanımı ve popülariteleri de artış göstermiştir. Bununla beraber, verinin üretilir üretilmez işlenmesi ihtiyacı mevcut veri sistemlerinin sınırlarını da zorlamıştır. Akan veri ve bu verinin işlenmesi sistemleri bu gibi ihtiyaçların giderilmesi hususunda çözümler geliştiren bir alandır [82].

Akan veri sistemleri özellikle yazılım uygulamaları bağlamında değerlendirildiğinde "olay akışı" gibi bir terim üzerinden ifade edilmektedir. Bir olay, olan bir şey hakkında değişmez bir gerçeği temsil eder ve kendisini açıklayan bir veya daha fazla veri alanını içerir [83]. Gerçek veya gerçeğe yakın zamanlı veri aktarmak, işlemek ve yeniden akış üretmek amaçlarıyla kullanılabilen akan veri sistemleri birçok yazılım mimarisi paradigmasında kendine yer bulmuştur. Günümüzde olay akışı platformları, yazılımların olaylar oluştuğunda tepki vermesini sağlayan bir mimari yapıdadır. Bu yapıya uygun sistemler, "reaktif sistemler" olarak da ifade edilmektedir [84]. Literatüre bakıldığında, büyük veri sistemlerinin 4 gereksinime dayandığı değerlendirilmektedir [85, 86].

- Yazma ağırlıklı iş yükü.
- Değişken istek yükleri (yeni kaynaklar ekleme ve bunları serbest bırakma).
- Hesaplama yoğun analitik (Değişen sorgu iş yükleri ve gecikme talepleri).
- Yüksek erişilebilirlik.

Bahse konu gereksinimler, büyük veri sistemlerinin bir özelliği olmasının yanı sıra, iş süreçlerindeki beklentilerin de ne yöne kaydığının bir göstergesidir. Bu kayma, yazılım geliştirme yaklaşımlarına da yansımıştır. Akan veri yapıları ve olaylar üzerinden kurgulanan süreçler, bu gereksinimlerin karşılanmasında yüksek öneme sahiptir.

Olay güdümlü mimariler ve mikroservis mimarileri, ölçeklenebilir ve çevik yazılım geliştirme yaklaşımları arasında ana akım olarak kabul edilmektedir. Bu iki kavram birbiri yerine kullanılabilen veya birbirinin karşıtı olan kavramlar değildir. Mikroservis yapıda olup olay güdümlü olmayan uygulamalar olabilirken, olay güdümlü olup monolitik yapıda uygulamalar da olabilir. Ayrıca hem olay güdümlü olup hem de mikroservis yapıda uygulamalar da olabilir.

Olay güdümlü mimariler veri üreten ve tüketen bileşenleri mümkün olan en yüksek seviyede birbirinden ayırır. Olayları asenkron (eşzamansız) olarak ele alan ve işleyen,

tekil bir amacı olan olay işleme bileşenlerinden oluşan bir örüntüdür [87]. Monolitik yapının karşıtı olarak ifade edilen mikroservis mimarisiyse, bir uygulamanın bileşenlerinin birbirinden bağımsız olarak dağıtılabilir ve servis edilebilir olduğu bir yazılım geliştirme paradigmasıdır [88].

Olay terimi üzerine inşa edilmiş ve bazı kafa karışıklıklarına sebep olabilecek bir takım önemli kavramlar mevcuttur. Bu hususta "olay kaynağı" [89] ve "olay bildirimi" kavramlarına mutlaka değinilmesi gerekmektedir. Olay güdümlü mimariyle karıştırılan bir kavram olan olay kaynağı, uygulamaların durum bilgilerindeki değişiklikleri olaylar dizisi şeklinde yakalamayı ifade ederken, olay güdümlü mimariyse bileşenler arasında iletişim kurmak için olayları kullanmayı ifade etmektedir. Bu yönüyle bakıldığında olay kaynağında amaç daha çok bir tarihçe tutmak ile ilgiliyken, olay güdümlü örüntüde farklı sınırları olan servislerin daha kolay adapte olabilmesi ve ölçeklenebilirliğin arttırılmasıyla ilgilidir. Kapsam olarak da birbirinden ayrılan bu iki örüntüden, "olay kaynağı"'nda kapsam tek bir uygulama veya sistemken, "olay güdümlü" örüntüde kapsam birden çok uygulama veya tümüyle bir organizasyon olabilmektedir [90]. Olay kaynağının temelindeki fikir, bir sistemin durumunda bir değişiklik yapıldığında, bu değişikliği bir olay şeklinde kaydetmek ve olayları gelecekte herhangi bir zamanda yeniden işleyerek sistem durumunu yeniden inşa edebilmektir. Böylece olay deposu gerçeğin ana kaynağı haline gelir ve sistem durumu oradan yeniden türetilebilir [91].

Karışıklık doğurabilecek bir diğer kavramsa olay bildirimidir. Olay bildirimi, bir sistemin etki alanında olan bir değişikliği, diğer sistemlere bildirmesi için olay mesajları göndermesine denir ve burada bildirimde bulunan kaynak sistem, yanıtla ilgilenmemektedir [91].

Günümüzde olaylar ve akan veri yapıları neredeyse hemhal olmuş ve bu durum çoğu yerde olay akışı ifadesiyle kendine yer bulmuştur Verileri bir olay akışı olarak yapılandırma fikrinden beslenen bu ifade de bir akan veri süreci olarak değerlendirilebilir[92]. Gerçek veya gerçeğe yakın zamanlı veri aktarımı ve filtrelemeler yapmaya yönelik bir takım mekanikler ortaya atılmıştır. Hem olay deposu hem de akan veri işleme platformu olarak kendisini tanımlayan Apache Kafka [93] gibi açık kaynak kodlu sistemler bu mekaniklerin pratikteki en iyi biçimde işlemesine olanak vermektedir. Hareket halindeki veriyi yöntemekle ilgili bu araçlara bir başka

örnek de Confluent [94] aracıdır. Olay akışı sistemlerinde kullanılabilecek işlevlerine göre kategorize edilebilecek çeşitli kompozisyon desenleri de mevcuttur [95].

4.1 Aracı

Verileri aktarma, genellikle pub-sub (yayınla - abone ol) veya üret – tüket yapıları üzerinden olmaktadır. Bu bağlamda yazma ve okuma işlemleriyle aralarında bir analoji kurmak mümkündür. Apache Kafka üzerinden üret – tüket yapılarını somutlaştırarak ifade etmek gerekirse, üreticiler, olayları veya kayıtları "aracı" adı verilen sunuculardaki konulara iten süreçlerdir. Tüketiciler ise bir Kafka konusundan kayıtları çeken süreçlerdir. Eğer aracı sayısı birden fazlaysa bu yapı "Kafka kümesi" adını alır. Aracıların, konuların ve kullanıcıların yönetimi gibi konular bir koordinasyon aracı olan Apache ZooKeeper [96] ile sağlanmaktadır. Farklı bağlamlarda karşılaşılan olay, kayıt veya mesaj terimleri akan veri süreçlerinde hemen hemen denk anlamları taşımaktadır. Kafka aracısı, mesajları belirli konfigürasyonlara (saklama süresi vb.) göre disk üzerinde saklayabilmektedir [97]. Bu yönüyle, akan veri sistemlerine bir tampon, saklama alanı veya veri tabanı benzeri bir işlev kazandırır. Böylece gerçek veya gerçeğe yakın zamanlı veri aktarırken veya işlerken, değişken gecikme süresi taleplerini karşılamaya olanak sağlayarak daha önce bahsedilen büyük veri gereksinimleri için de çözüm üretir.

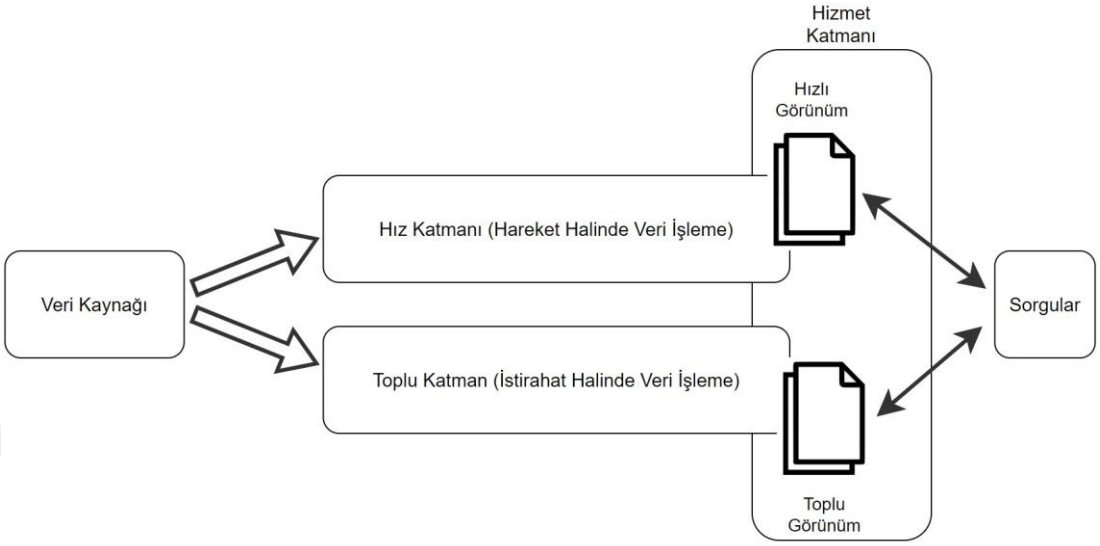
4.2 Akan Veri İşleme

Etki alanı odaklı (dayalı) tasarımı takip eden yazılımların çoğu, meydana gelen sayısız olayı ele almak için çeşitli mimari yaklaşımları benimserler. Lambda ve Kappa mimarileri [10] çeşitli katmanlar üzerinden büyük veri işlemeyi mümkün kılan mimarileridir. Nesne deposu gibi veri gölü bileşenleri, Kafka gibi akan veri araçları, Apache Spark gibi birleşik analitik motoru araçları, akan veri işleme mimarilerinde önemli işlevleri karşılayan araçlar olarak karşımıza çıkmaktadır. Bu mimariler irdelenirken bu araçların tekniğin bilinen seviyesine göre işlevsel olarak konumlandırmaları da yapılacaktır.

4.2.1 Lambda mimarisi

2011 yılında Lambda mimarisi [98] terim olarak ortaya konmuş ve genelleşebilir, ölçeklenebilir ve hataya dayanıklı gerçek zamanlı veri işleme mimarisi olarak

sunulmuştur. Lambda mimarisi “toplu katman”, “hız katmanı” (akış katmanı olarak da bilinir) ve “hizmet katmanı” olmak üzere üç katmandan oluşur [99]. Şekil 4.1’de örnek bir lambda mimarisi görülmektedir.



Şekil 4.1 : Lambda mimarisi

4.2.1.1 Toplu katman

Bu katman temel olarak iki işlevi yerine getirir.

- Değişmez bir ana veri kümesini saklamak.
- Ana veri kümesine dayalı toplu görünümlerin ön hesaplamasını gerçekleştirmek.

Son maddeyi bir başka şekilde ifade edersek, veri gölündeki değişmez bir veri kümesi bütünüyle bir işleve girdi olabilir ve toplu görünüm oluşturabilir [100]. Buradaki süreci yazılım araçları üzerinden örneklendirecek olursak, toplu katmana girdi olabilecek akan veriler, doğrudan Kafka’dan gelebileceği gibi, yeniden oluşturulmuş akan veriler şeklinde Spark aracının akış bileşeninden de gelebilir. Önceleri toplu katmanda ana veri kümesini saklamak için kullanılan bileşen Hadoop Dosya Sistemiyken, günümüz bulut sistemlerinde nesne depoları tercih edilmektedir. Ana veri kümesini kullanarak bir takım toplu görünümler oluşturmakta en çok tercih edilen araçlardan birine örnek olarak, analitik motoru bağlamındaki bileşenleriyle yine Apache Spark verilebilir. Spark’ın çok kabiliyetli yapısı, işlevleri soyut olarak düşünürken kafa karışıklığı oluşturabilir. Bu yüzden vurgulamak gerekirse Spark, Kafka’dan akan veri dinleyebildiği gibi akan veri oluşturup tekrar Kafka’ya veya veri

gölündeki nesne deposuna gönderebilen bileşenlere sahiptir. Ayrıca Spark analitik motor ve sorgu motoru olması özelliğiyle veri gölünde veri dosyaları veya veri tablo formatları şeklinde saklanan verilere de erişebilir. Eriştiği veriye bellek-içi işlemler şeklinde bir takım dönüşümler de yapabilir. Dolayısıyla toplu katmana hem girdi oluşturabilir, hem de ana veri setlerine erişip dönüşümler yaparak hizmet katmanına toplu görünümler şeklinde çıktı da sağlayabilir [101]. Toplu katmanda veri, dinlenme (istirahat veya hareketsizlik anlamında) sırasında işlenir.

4.2.1.2 Hız katmanı

Hız katmanı, gerçek zamanlı görünümleri indeksler [10] ve toplu katmandaki yüksek gecikme süresini telafi etmeyi hedefler. Toplu katmandaki mantıkla ele alırsak, toplu katmanda bulunan devasa veri kümeleri yüzünden en güncel görünümlere ulaşabileceğimiz en son veri kümesine ait görünümlerin hesaplanması zaman alacaktır. Hız katmanı en güncel verileri sorgulamayı verimli hale getirmeyi amaçlayan katmandır. Hız katmanını bir başka şekilde ifade edecek olursak, en son edinilen veriler bir işleve girdi olabilir ve gerçek zamanlı görünüm oluşturabilir. Örneğin, buradaki süreçte hız katmanına girdi olabilecek veriler Kafka'dan gelebilirken, Spark aracının akış bileşeni üzerinden hizmet katmanına çıktı sağlayabilir. Hız katmanında veri, hareket halinde işlenir.

4.2.1.3 Hizmet katmanı

Hizmet katmanı toplu katmandan ve hız katmanından gelen sonuçları birleştirir ve hedefi, birleştirilmiş sonuçlara kullanıcının kolaylıkla erişmesidir.

4.2.2 Kappa mimarisi

Kappa mimarisi 2014 yılında, Lambda mimarisinin iki tane karmaşık dağıtık sistem barındırması ve üzerinde çalıştığı çerçeveye özelleşmiş kodlar yazmaya zorlaması gibi eleştirilerle, bu mimariyi basitleştirmeye yönelik sorgulamalar neticesinde ortaya konulmuştur [102]. Kappa mimarisindeki önerme, tek bir teknolojik yığın üzerinden hem gerçek zamanlı hem de toplu veri işleme gerçekleştirebilmek şeklindedir. Bu yönüyle basitliği ve gerçekliğin tekil kaynağı olmayı hedefler [103].

Kappa mimarisi, gerçek zamanlı katman [104] (hız katmanı olarak da bilinir) ve hizmet katmanı olmak üzere iki katmandan oluşmaktadır. İlgili katmanlar ve

kullanılan yazılım araçları Lambda mimarisiyle aynı olduğundan, katmanları tekrar tanımlamaya gerek yoktur. Katmanlara dikkat edildiğinde Kappa mimarisi, Lambda mimarisinden toplu katmanın çıkartılmış hali şeklinde yorumlanabilir [105].

4.2.3 Lambda mı Kappa mı ?

Lambda veya Kappa mimarisi kullanmakla ilgili tek bir doğru tercih durumu yoktur. Her iki mimari de kendisine özgü avantajlar ve dezavantajlar barındırmaktadır. Uygulama geliştirme, hata ayıklama ve kod bakımı açısından operasyonel kolaylık sağlaması ve tekil yapısı Kappa mimarisine avantaj sağlarken, toplu katmanın varlığı da büyük ölçekli makine öğrenmesi modellerinin eğitilmesi durumlarında yoğun hesaplama kabiliyeti sağlamaktadır.

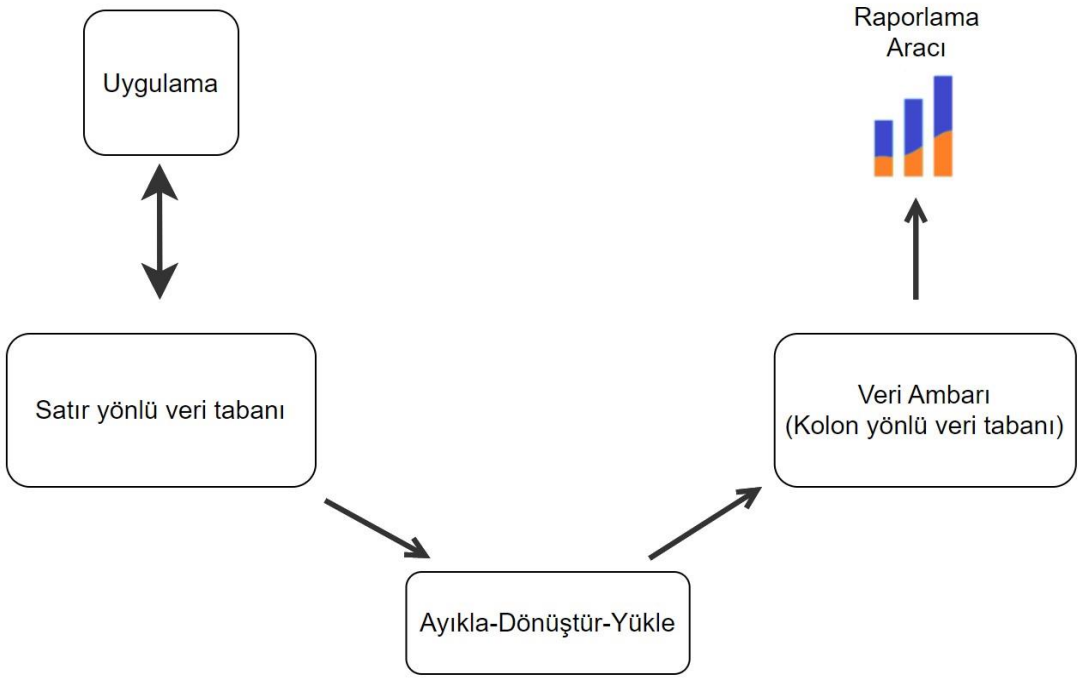
5. VERİ DÖNÜŞTÜRME

Verinin belirli amaçlara yönelik tüketimi belirli koşullar barındırıyor olabilir. Kimi zaman veri gruplarının format uyumsuzluğundan, kimi zamansa verinin ancak bir hesaplama süreci sonunda fayda sağlayacak hale gelmesinden kaynaklı olarak dönüşüm ihtiyaçları oluşabilir. Bu ihtiyaçlar, büyük veri ekosisteminde farklı bileşenlerin kullanımını veya farklı sıralarla dönüştürme yapılmasını gerektirebilir. Literatür bu anlamda büyük veri ekosisteminin unsurlarının kullanımını iki başlıkta ele almaktadır. Bu başlıklar, "Ayıkla-Dönüştür-Yükle" ve "Ayıkla–Yükle–Dönüştür" şeklindedir.

Bu iki başlık aslında, biri veri ambarı diğeri ise veri gölü üzerinden işletilen veri dönüşüm serüvenini ifade etmektedir.

5.1 Ayıkla-Dönüştür-Yükle

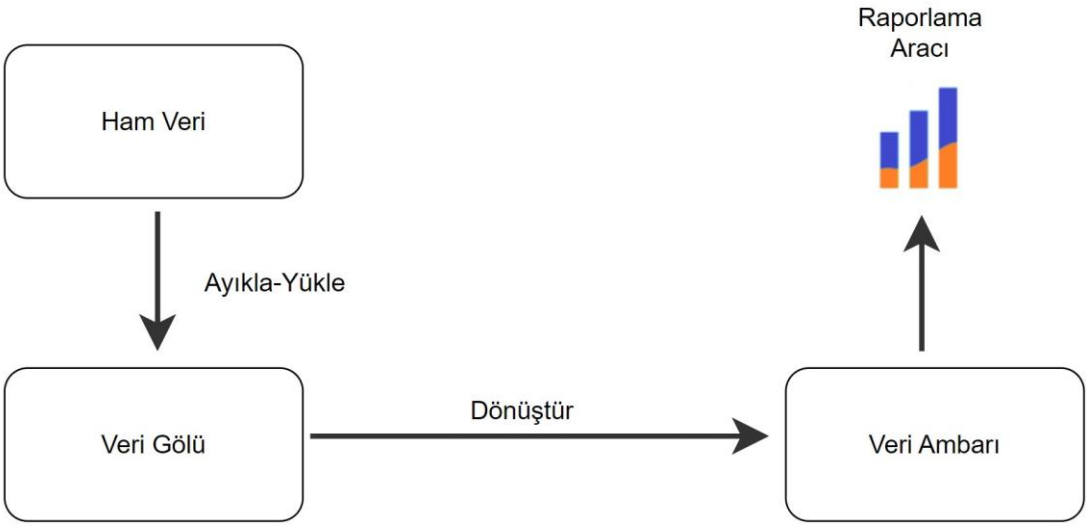
Ayıklanıp dönüştürülmesi sonucu yapılandırılmış ve raporlamaya hazır hale getirilmiş verinin oluşturulması sürecidir. Ayıklama işleminden sonra verinin çoklanmadan, doğrudan dönüştürülmesi ve sonrasında bir saklama alanına yüklenmesi senaryosunumerkeze alan bir kurgudur. Dikkat edildiğinde dönüştürüldükten sonra elde edilen veri yapılandırılmış bir veridir ve yapılandırılmış veriler genelde raporlanma amacıyla gösterge araçları veya iş zekası araçlarıyla entegre haldedir. Bu yapıdaki veriler için veri ambarı en uygun yapıdır. Şekil 5.1'de ayıkla-dönüştür-yükle süreci görülmektedir.



Şekil 5.1 : Ayıkla-Dönüştür-Yükle süreci

5.2 Ayıkla-Yükle-Dönüştür

Verinin dönüşüm serüvenini veri gölü ekseninde ele alanan süreçtir. Ayıklandıktan sonra verinin çoklanması ve veri gölüne yüklenmesiyle devam eder. Ayıklanmış ama raporlamaya uygun bir şekilde yapılandırılmamış verinin bir kopyasının veri gölünde saklanması ve sonrasında dönüştürülmesi sürecidir. Genellikle veri kaynağından ayıklama ve yükleme işlemleriyle veri gölüne çekilen veri üzerinde iş zekası unsurlarınca gösterge veya rapor hazırlanmak istendiğinde, verinin çeşitli dönüşümlere uğraması gerekmektedir. Şekil 5.2'de ayıkla-yükle-dönüştür süreci görülmektedir.



Şekil 5.2 : Ayıkla-Yükle-Dönüştür süreci

5.3 Analitik Motor ve Bellek-İçi İşleme

Veri dönüşümü denildiğinde akla gelebilecek kullanım durumlarından bazıları aşağıda belirtilmiştir:

- Veri formatlarındaki dönüşümler.
- Veri üzerinden formüller işletip, hesaplamalar sonucu yeni bir veri kümesinin elde edilmesi.
- Verideki bazı alanların veya değerlerin ayıklanması.

Yukardaki örnekler, ister akan veri üzerinde isterse dinlenen veri üzerinde işletilsin, her durumda bir analitik motor gereksinimi söz konusudur. Apache Spark gibi kabiliyetli araçlar hem veri gölüne hem de Kafka gibi akan veri araçlarına entegre olabildiğinden hemen hemen her veri dönüşümü sürecinde kullanılabilmektedir. Spark, veri dönüşümünü gerçekleştirirken dağıtık bir şekilde bellek-içi veri işleyebilir. Ölçeklenebilir bir yapısı olduğundan tıpkı Kafka'da olduğu gibi bir küme şeklinde yapılanabilir. Spark'taki veri dönüşümleri, temel değerlendirme tekniği kullanılarak gerçekleştirilir. Böylece, aksiyona ihtiyaç duyulacağı ana kadar işlemin yapılması ertelenir. Spark, aksiyonun işletileceğini tespit ettiğinde tüm dönüşümleri sıralı bir şekilde kaydettiği bir yönlendirilmiş döngüsüz çizge oluşturur [106]. Bu sayede dönüşümler gerektiğinde yapılacak ve bir takım indirgemeler ile optimize edilerek beklenen sonuç alınacaktır.

5.4 Veri Tablo Formatlarının Veri Dönüşümüne Etkisi

Veri tablo formatları, veri gölündeki veri dosyalarını kapsülleyen ve veri gölüne ilişkisel veri tabanı tablosu benzeri özellikler kazandırabildiren yapılardır. Bu özelliklerinden dolayı çeşitli sorgu motorları üzerinden iş zekası araçlarına entegre olabilmektedirler. İş zekası araçları alışılagelmiş haliyle daha çok veri ambarlarıyla entegre olabilmektedir. Hudi gibi veri tablo formatları sayesinde veri gölünde duran verinin, Presto gibi sorgu motorları vasıtasıyla, iş zekası araçlarınca erişilebilmesi ve raporlanabilmesi için veri ambarına çekilmesi gerekliliği ortadan kalkmıştır. Bu da verinin serüveninde çeşitli durak değişikliklerine neden olmaktadır.

6. VERİ GÖRSELLEŞTİRME VE RAPORLAMA

Endüstride birçok firma dijital dönüşüm çalışmaları gerçekleştirmektedir. Çoğu dijital dönüşüm projesinde öncelikli hedeflerden biri de iş sürecinin ve verinin izlenebilirliğidir. Veri görselleştirme, analitik çalışmalar sonucu elde edilen bulguların anlaşılmasında ve performans indikatörlerinin takibinde çok önemli rollere sahiptir. Üretim tesislerinde anomali tespiti, hata tespiti, kondisyon gözlemi ve kalan kullanışlı ömür kestirimi gibi konularda geliştirilen algoritmaların veya makine öğrenmesi modellerinin çıktılarının sahada faaliyet gösteren operator için takibi görsel araçlar üzerinden olmaktadır. Benzer şekilde takım liderleri veya yöneticiler de indikatörleri gerçek zamanlı veya herhangi bir an izleyebilmek yönünde motivasyona sahiptir. Veri görselleştirmeyle ilgili iş sahasından ve veriyi analiz eden gruplar tarafından kaynaklanan beklentiler operasyonel gösterge ve iş zekası araçları olmak üzere iki ana başlık üzerinden karşılanabilmektedir. Veri görselleştirme hem dinlenen veriyi hem de hareket halindeki veriyi muhatap alabilir.

6.1 Operasyonel Gösterge Paneli Araçları

Özellikle nesnelerin interneti alanındaki gelişmelerle birlikte telemetri verilerinin izlenebilmesi veya bir takım alarm sistemlerinin geliştirilmesi ve takibi gibi konular operasyonel gösterge panellerinin kullanılmasını gerektiren alanlara örnek olarak verilebilir. Operasyonel gösterge aracı olarak kullanılabilecek açık kaynak kodlu Grafana [107] aracı bu ihtiyaçları karşılamakta kullanılabilir. Grafana, tarayıcı üzerinden erişim sağlanan ve duyarlı yapısıyla gösterge panellerine hem mobil hem de bilgisayar ortamlarından ulaşabileceğiniz bir yapıdadır. Özellikle üretim yapan tesislerde sahadaki çıktılar çoğunlukla ya çeşitli tablet ekranları ya da sahaya yerleşik konumlanmış bilgisayar ekranları üzerinden paylaşılmaktadır. Grafana sahip olduğu özellikler sayesinde hem mevcut alışkanlıklara uyum sağlayabilir hem de özel veya genel bulut ortamlarında çalışan algoritmaların çıktılarının sahada yerleşik bir bilgisayar olmadan sunulmasına imkan tanır. İçerisinde birçok veri tabanı veya veri ambarı aracına entegratörler barındırır. Statik olarak hazırlanmış raporlarla, çeşitli UPA'lardan veri çeken raporları bir arada sunabileceğiniz bir formdadır. Grafana, büyük veri mimarisindeki çeşitli sistemlerden metrikler toparlamayı ve onları

sorgulamayı optimize eden açık kaynak kodlu Prometheus aracı gibi izleme araçlarıyla da entegre olabilmektedir.

6.2 İş Zekası Araçları

İş zekası, iş faaliyetlerinden elde edilen verileri, raporlama, veri madenciliği, süreç madenciliği, iş performans yönetimi veya tahmin analitiği gibi amaçlarla kullanışlı bilgiye dönüştüren teknolojilerin bir kümesi olarak ele alınabilir [108, 109].

Operasyonel gösterge araçları ve iş zekası araçları sorgu yürütme süresi, güncellik ve doğruluk gibi belirli yönlerden birbirrlerinden ayrışmaktadır Operasyonel gösterge araçları daha çok izleme odağındayken iş zekası karar almaya yardımcı olmak odağındadır. Bu bağlamda, saniyeler veya dakikalar mertebesindeki veriden ziyade günlük, haftalık, aylık veya yıllık verilerle daha çok ilgilenir [110]. Açık kaynak kodlu iş zekası araçlarına örnek olarak Apache Superset [111] verilebilir.

7. BÜYÜK VERİ MİMARİSİ PARADİGMALARI

7.1 Mantıksal Veri Ambarı

2012 yılında Gartner [112], Mantıksal Veri Ambarı konseptini tanıtarak talep odaklı, analitik uygulamalar için veri yönetimi yapabilen bir yapı olarak sunmuştur. Akan veri işleme süreçlerine de yer veren bu yapıda veri ambarı, veri gölü, veri sanallaştırma gibi kavramları birbiri yerine kullanılan kavramlar olarak değil birbirini tamamlayıcı kavramlar olarak mimari tasarımın içinde nitelemiştir.

7.2 Veri Yapısı

Veri yapısı ifadesi ilk olarak 2015 yılında ortaya konulmuştur [113]. Veri yapısı, mantıksal veri ambarı konseptinden farklı olarak içinde barındırdığı bileşenleri veri ambarı, veri gölü gibi ayrı ayrı uygulayıcı paradigmalar veya mimari arketipler olarak nitelendirmeyip, veri teslimatı yapacak çok dilli kalıcılık fikrini [114] temel alan ilişkisel veri tabanları, çizge veri tabanları, dosya/damla deposu (Hadoop gibi) gibi veri saklama yaklaşımlarını barındıran tasarımdır. Veri yapısı yaklaşımı bu yönleriyle veriyi servis olarak sunma davranışını sergilemektedir.

7.3 Veri Ağı

Veri ağı, veri yapısı tasarımının güncel yorumu şeklinde değerlendirilebilir. Veriden değer üretmek ve "ürün olarak veri" bakış açısıyla şekillenmiş veri ağı tasarımı, merkezi monolitik veri gölü veya onun atası olarak nitelendirilen veri ambarı paradigmalarının barındırdığı arıza modlarına çözüm olarak, modern dağıtık bir mimari üzerinden yeni bir paradigmaya geçişi önermektedir [115].

8. NESNELERİN İNTERNETİ

Gömülü internet veya sinen hesaplama gibi ifadeler 1970'li yıllarda ortaya çıkmasına [116] rağmen, nesnelerin interneti terimi ilk olarak 1999 yılında ortaya atılmıştır [117]. Nesnelerin interneti sensör, iletişim, ağ ve bilgi işleme teknojilerine dayanan yeni bir bilgi iletişim teknolojisi verisyonu olarak ele alınabilir [117]. Makineden makineye iletişim, endüstri 4.0 ve akıllı sistemler gibi başlıklarda çok sık rastlayabileceğimiz bir kavram olarak karşımıza çıkmaktadır. Büyük veri ekseninde ele alındığında, her bir cihazın ve sensörün veri kaynağı olarak değerlendirilmesi doğal bir çıkarımdır. Sensörler genellikle ham veriler içerir ve çok yüksek örnekleme oranlarına sahip olabilirler. Özellikle endüstriyel yapay zeka uygulamaları yapmak hedeflendiğinde sensörlerden üretilen ham verilerin veri gölünde saklanabilmesi önem kazanmaktadır. Ayrıca çoğu sensör, üretimdeki mevcut durumu izlemek ve bir takım alarmlar üretebilmek hedefinde konumlandırıldığından sensör verilerinin akan veri olarak değerlendirildiği ve bir takım olay akışı senaryolarının üzerinden sahaya gerçek zamanlı veya gerçeğe yakın zamanlı geri bildirimlerin gerçekleştirildiği birçok kullanım senaryosu akla gelmektedir. Nesnelerin interneti yaklaşımında veri daha çok telemetri terimiyle beraber anılmaktadır. Telemetri veya uzölçüm, bir sistemin uzaktan izlenmesi veya kontrol edilmesi anlamına gelmektedir [118]. Tele ve metro kelimelerinde türemiş, sırasıyla uzak ve ölçmek manasındadır [119].

8.1 Mesaj Kuyruklama Telemetri İletimi Protokolü

Mesaj Kuyruklama Telemetri İletimi (MQTT), 1999 yılında petrol boru hatlarında bataryalı çalışan sistemlerde verimli çalışacak bir mesajlaşma protokolü oluşturmak amacıyla IBM için geliştirilmiştir. Yüksek gecikmeli, düşük bant genişliğine sahip, güvenilmez ağlarda bile verimli çalışabilen hafif bir protokoldür [120]. MQTT protokollü, IBM'in açık standartlarla ilgili çalışan kâr amacı gütmeyen oluşumlara MQTT'yi arz etmesi sonucu, 2014 yılında açık standart olarak sunuldu [120]. Bu gibi avantajlarından dolayı günümüzde nesnelerin interneti alanında birçok cihaz veya sensör dahili olarak MQTT protokolünü desteklemektedir. MQTT protokolü yayınla – abone ol örüntüsünü kullanır. MQTT mesajların teslim edilmesini, çeşitli servis kalite seviyeleri üzerinden garanti eder. MQTT'nin anahtar özelliklerinden biri olan

servis kalite seviyesi sayesinde istemciler, ağ güvenirliğine göre servis kalite seviyelerini seçebilirler [121].

Mesajların teslim edilmesindeki ölçütleri ifade eden, servis kalite seviyeleri 3 tiptedir:

- En fazla bir kez (0)
- En az bir kez (1)
- Tam olarak bir kez (2)

8.2 Mesaj Aracısı

MQTT mesaj aracısı, istemcilerin iletişim kurmasını sağlayan bir vasıtadır. Birden fazla sayıda istemci bir mesaj aracısına bağlanabilir. Bir istemci bir konuya mesaj yayınlayabilir veya bir konudan mesaj dinleyebilir. Bir istemci mesaj aracısına mesaj gönderdiğinde, bu mesaj içerisinde konu bilgisi bulunur. Aracı, konu üzerinden hangi istemcilerin o konuya abone olduğunu belirler ve mesajları ilgili abonelere gönderir.

8.2.1 Kafka aracısı vs MQTT aracısı

Nesnelerin interneti bağlamında Kafka aracısı ve MQTT aracısı benzer gibi gözükseler de temelde önemli farklılıklar barındırmaktadırlar. Kafka'da üreticiler, olayları veya kayıtları aracı sunucu üzerinden konulara iter, tüketicilerse bir Kafka konusundan kayıtları çeker. MQTT aracısıysa mesajları abonelere iter. Diğer bir ifadeyle Kafka aracısından mesajlar çekilirken, MQTT aracısında mesajlar itilir. Bu da MQTT'de istemciler üzerindeki yükü azaltırken Kafka'da aracı üzerindeki yükü azaltır.

8.3 Nesnelerin İnterneti Platformu

MQTT aracısı olarak görev yapacak bir sunucu veya konteyner konulandırmak sensör veya cihazlardan veri toplamak için önemli bir adım olsa da büyük veri platformlarıyla entegre olmak adına yeterli değildir. Ayrıca, istemci yazılımların geliştirilmesi, veri saklamaya yönelik veri tabanı yapılandırmaları, Kafka gibi akan veri araçlarıyla entegrasyon gibi başlıklar kullanıcı olacak biri açısından zorlayıcı noktalardan bazılarıdır. Nesnelerin interneti yaklaşımını büyük veri yaklaşımlarıyla beraber ele almak, özellikle endüstriyel ortamlar için elzemdir. İlave olarak, izlenebilirlik ve entegre olabilirlik açısından da operasyonel bir takım bileşenlere ihtiyaç vardır. Bu bağlamda MQTT dışındaki endüstriyel protokollerden (Modbus, OPC vb.) de veri

toplayabilecek, bu verileri de MQTT protokolü üzerinden sunabilecek, gerçek zamanlı izlemeye yönelik bir takım göstergelere sahip, gerektiğinde Kafka gibi araçlar üzerinden veri gölüne ham sensör verilerini ulaştırabilecek kullanışlı bir arayüze sahip bir servisler bütününe, yani bir nesnelerin interneti platformuna ihtiyaç vardır. Kafka ile MQTT bağlantısının kurulmasında Kafka'ya özgü seçenekler şu şekilde belirtilmiştir [122]:

- MQTT aracısına her iki yönde de entegre olabilen, Kafka bağlantı kaynak ve çıkış birleştirici.
- Bir MQTT aracısına ihtiyaç duymadan cihazlardan veri alan, Confluent MQTT vekili.
- HTTP protokolü tabanlı bir entegrasyon için, Confluent REST vekili.

Kafka yazılımı, "Kafka bağlantı çerçevesi" denilen, Kafka'nın başka sistemlerle konuşabilmesini sağlayan dahili bir bileşen içermektedir. Kafka bağlantı MQTT birleştiricisi, MQTT aracısından veri alma ve MQTT aracısına veri göndermekte kullanılan bir eklentidir. Birleştirici kullanılan senaryoda MQTT aracısı mevcuttur ve MQTT'ye özgü avantajların sürdürülmesini sağlar. MQTT vekili kullanılan senaryoda ise MQTT aracısı kullanılmaz ve cihazlardan itilen veri Kafka aracısına iletilir. Bu senaryoda, Kafka telemetri verilerinin kalıcılığından ve güvenilirliğinden sorumlu araçtır. Son senaryo olan REST vekili senaryosunda nesnelerin interneti uygulamaları Kafka kümesine HTTP protokolü üzerinden bir arayüz vasıtasıyla erişir.

Gerektiğinde ham sensör verilerini Kafka aracısı vasıtyasıyla akıtabilecek, ölçeklenebilir şekilde servis edilebilir, nesnelerin interneti platformu şeklinde ifade edebileceğimiz çeşitli açık kaynak kodlu araçlar mevcuttur. Thingsboard [23] bu yazılımlara örnek olarak verilebilir. Bu tez çalışması kapsamında platform bakış açısıyla nesnelerin interneti platformunun gölevi platformuyla birleştirilmesi veya büyük veri paradigmaları bakış açısıyla mantıksal veri ambarı paradigmasına nesnelerin interneti bileşeninin eklenmesi önerilmiştir. Hem mantıksal veri ambarı paradigmasında hem de gölevi platformunda makine öğrenmesi süreçlerine işaret edilmiştir. Nesnelerin interneti odağında büyük veri için tahmine dayalı analitik mimarisi şeklinde ifade edebileceğimiz bütünsel bir yapı önerebilmek adına bir sonraki başlıkta makine öğrenmesi operasyonları ve burada kullanılabilecek araçlar genel hatlarıyla ifade edilmiştir.

9. MAKİNE ÖĞRENMESİ OPERASYONLARI

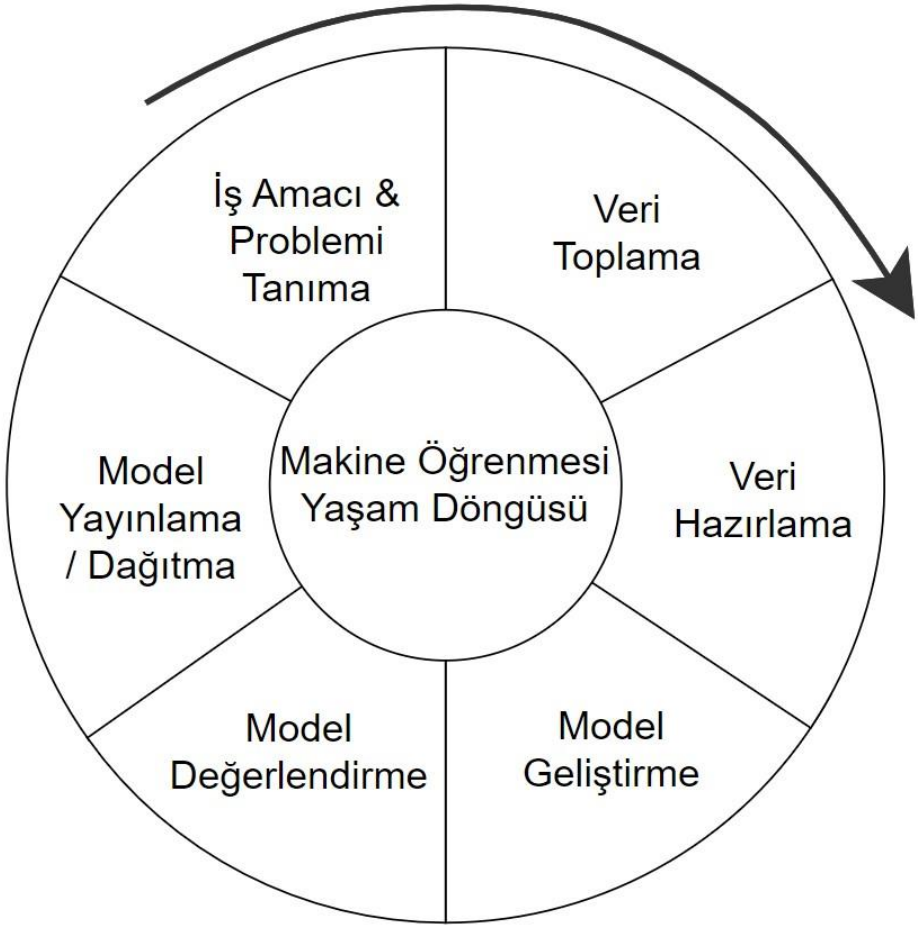
İnsan zekasının yaptığı işleri yapabilmeyi hedefleyen yapay zeka alanı büyük veri bağlamında çok sayıda araştırmacı tarafından çalışılan bir alandır. Veriden anlamlı örüntüleri öğrenip keşfedecek ve karar destek sistemlerini besleyecek algoritmaların geliştirilmesi yapay zekanın alt başlıklarından olan makine öğrenmesi alanının temel çalışma konularındandır. Nasıl ki yazılım geliştirme süreçlerinde geliştirme, test ve canlıya alma süreçleri gibi süreçleri bir takım hatlar üzerinden sürekli teslimat yapılarına dönüştürme operasyonları mevcutsa, benzer şekilde makine öğrenmesinde de modellerinin eğitilmesi, hiper-parametre ayarlamalarının yapılması ve devreye alınması gibi süreçler üzerinden sürekli öğrenme yapılarına yönelik operasyonlar mevcuttur. Gerek büyük veri mimari paradigmalarından olan mantıksal veri ambarı paradigmasında gerekse de gölevi gibi büyük veri platformu yapılarında makine öğrenmesi büyük veri süreçlerinde işaret edilmiş bir husus olarak karşımıza çıkmaktadır. Bu bağlamda, ölçeklenebilir şekilde makine öğrenmesi operasyonlarını yönetebilmek önem arz etmektedir. Günümüzde Kubernetes [123] gibi konteyner yönetim araçları ölçeklenebilir yapılar kurmayı ve yönetmeyi kolaylaştırmaktadır. Çok sayıda sunucudan oluşan bir Kubernetes kümesinde büyük veri araçlarını konteyner formda devreye almak mümkündür. Böylelikle sunucu kümesi üzerinde çalışan konteyner kümeleri şeklinde ifade edebileceğimiz Kafka kümesi, Spark kümesi gibi çoklu oluşumlar devreye alınabilir ve bu yazılımlar ölçeklenebilir, dağıtık şekilde hizmet verebilir, kendi kümelerindeki işçilere iş gönderebilirler.

Makine öğrenmesi özelinde ele alındığında çeşitli açık kaynak kodlu yazılımlar burada konumlandırılabilmektedir. Bellek-içi veri işleme araçlarından Spark gibi araçların barındırdığı makine öğrenmesi bileşenleri üzerinden veya Kubeflow [124] gibi Kubernetes üzerinde doğrudan makine öğrenmesi iş akışlarına özgü bir araç ile hem merkezi işlem birimi hem de grafik işlem birimi üzerinden model eğitmek mümkündür.. AWS Sagemaker, Dataiku gibi açık kaynak kodlu olmayan çeşitli araçlar da bu amaçla sıklıkla kullanılan araçlardandır [125].

9.1 Makine Öğrenmesi Yaşam Döngüsü

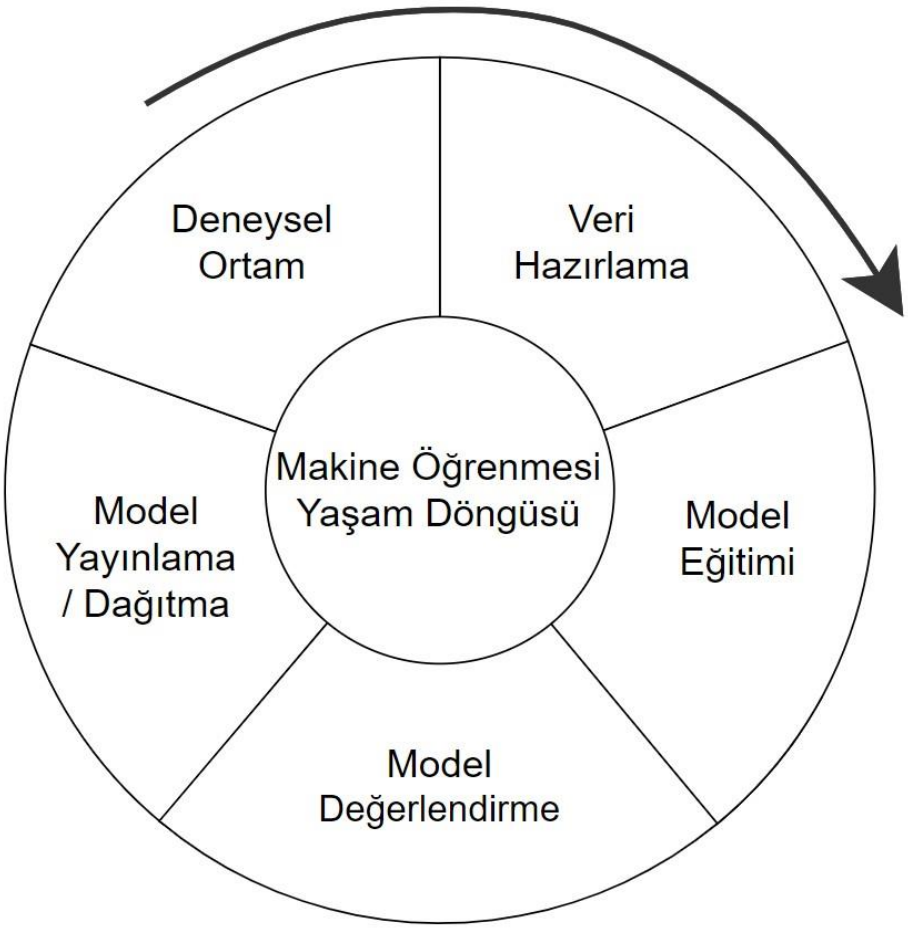
Makine öğrenmesi, farklı disiplinlerdenaraştırmacıların katkı sunduğu bir alan olması sebebiyle farklı algoritmaların çalışıldığı bir alandır. Bir makine öğrenmesi modelini

besleyecek veri kümesinin hazırlanmasından, eğitilen modelin servis vermesine kadar yapılabilecekler çok çeşitli başlıklar üzerinden ele alınabilir. Tez çalışması kapsamında bu başlıklar, veri hazırlama, model eğitimi, model değerlendirme, deneysel ortam ve modellerin servis vermesi şeklinde belirlenmiştir. Elbette, makine öğrenmesi süreçlerini veri analizine odaklı bir şekilde farklı başlıklarla ifade etmek de mümkündür. Şekil 9.1'de veri bilimci gözüyle genel anlamıyla bir makine öğrenmesi yaşam döngüsü örneklenmiştir.



Şekil 9.1 : Veri bilimci gözünden makine öğrenmesi yaşam döngüsü

Ancak, tez çalışması kapsamında bu başlıklar, büyük veri bileşenleriyle makine öğrenmesi operasyonlarının entegrasyonu ve ilişkisi bağlamında belirlenmiştir. Buradaki ilişkiler 10. başlıkta verilen Şekil 10.1 üzerinden daha net anlaşılabilir. Şekil 9.2'de büyük veri süreçlerine entegre makine öğrenmesi döngüsü örneklenmiştir.



Şekil 9.2 : Büyük veri entegre süreçlerde makine öğrenmesi yaşam döngüsü

9.1.1 Veri hazırlama

Veri hazırlama, makine öğrenmesi modelinin gerektirdiği özelliklere sahip bir veri kümesinin oluşturulmasında ilk adım olarak değerlendirilebilir. Ham veriler üzerinde yapılan düzenleme ve hazırlık işlemlerini kapsar. Gölevi platformunda veri hazırlama süreçlerine muhatap olan birim veri gölüdür. Örnek bir akış senaryosu üzerinden ele alacak olursak, akan veri araçları (Kafka) üzerinden akan veri, analitik motor (Spark) aracılığıyla veri gölüne yazılır. Böylelikle nesne deposu üzerinde konumlanmış veri gölü, ham verileri depolar. Bu verilerden bir model eğitimi söz konusu olduğunda, veri hazırlama süreçleri sonrasında eğitime hazır veri kümesi elde edilmiş olur.

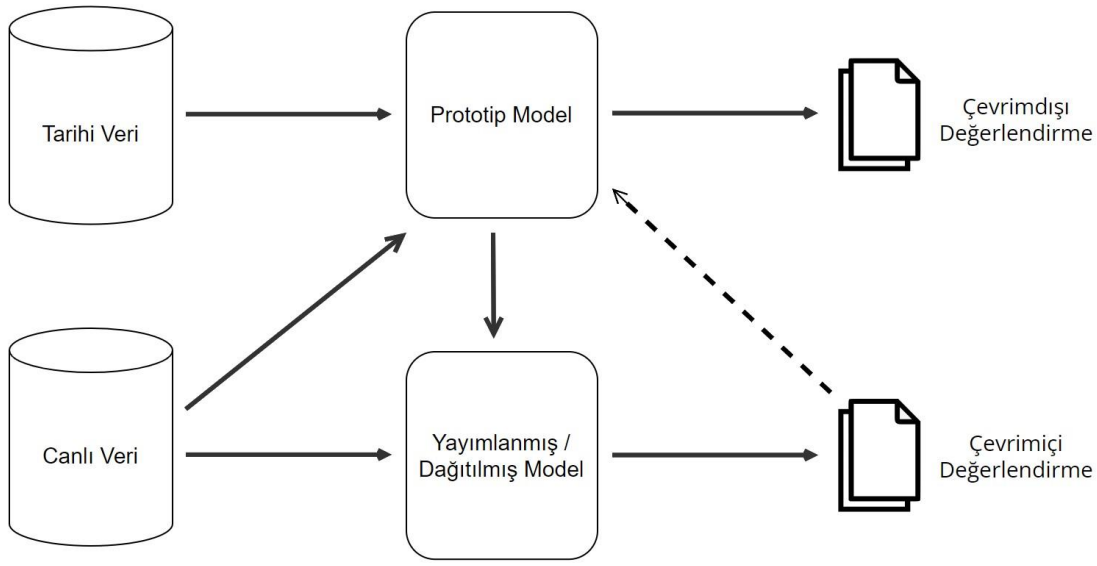
9.1.2 Model eğitimi

Makine öğrenmesi algoritmaları denetimli (danışmanlı), denetimsiz (danışmansız), yarı-denetimli (yarı danışmanlı), pekiştirmeli vb. başlıklar gibi çok sayıda başlıkta çalışılan algoritmalardır. Bu başlıklar veri kümesi, bağımlı-bağımsız değişken ilişkisi,

istatistiki varsayımlar gibi çeşitli yönlerden farklılıklar içermektedir. Veri bilimci, elindeki probleme ve veri kümesine göre en uygun başlığı seçer ve model tasarımını ortaya koyar. Sonrasında model, eğitim sürecine sokulur ve girdilerden keşfedilen örüntüler üzerinden model, hedeflenen çıktıyı optimal tahminlemeye uğraşır. Optimal tahminlemeyi sağlayabilmek için model değerlendirme (sınama) bileşeniyle etkileşimdedir. Değerlendirme metrikleri üzerinden hiperparametre düzenlemesi gibi model iyileştirme çalışmaları yapılır. Modelin örüntü keşfettiği bu süreç "model eğitimi" olarak ifade edilir. Veri bilimi açısından model eğitimi, elbetteki akademik bir altyapı gerektirmektedir. Ancak, bazı modellerin eğitilmesi özellikle büyük veri kullanımı durumunda yazılımsal veya donanımsal bir takım altyapı gereksinimlerini de doğurmaktadır. Bu bağlamda, kullanılacak programlama dilinden kod editörüne, merkezi işlem biriminden grafik işlem birimine kadar birçok kısıt ortaya çıkabilir. Kubeflow gibi araçlar bu kısıtlara uygun ortamları oluşturmaya yönelik enstrümanlar barındırarak Kubernetes kümesi üzerinde model eğitimine imkan verir.

9.1.3 Model değerlendirme

Model değerlendirme, prototip olarak ortaya koyulan modelin tarihi veriyle çevrimdışı değerlendirilmesinden başlayan, canlıya alınmış veya servis olarak yayımlanmış modelin canlı veriyle çevrimiçi değerlendirilmesine kadar süren bir iş akışını kapsamaktadır. Şekil 9.3'te örnek bir akış görülmektedir.



Şekil 9.3 : Model Değerlendirme süreci

Dolayısıyla, büyük veri bağlamında model, eğitme bileşenleriyle iletişimdeyken, modelin yazılım servisi olarak yayımlanması kısmında da nesne deposuna konumlandırılmış bir model deposu bileşeniyle etkileşimdedir. Hiperparametre düzenlemeleri sonucu eğitilmiş modeller nesne deposunda bulunan model deposunda saklanır. Burası veri depolamadığından veri gölü olarak ifade edilmemiştir. Görüldüğü üzere nesne deposunun ölçeklenebilir sistemlere ve büyük veri yapılarına etkileri sadece veri gölü bağlamında değildir. Makine öğrenmesi modellerinin saklandığı model deposu veya yayımlanabilecek uygulamalar için konteyner deposu olarak da kullanılabilmektedir. Model değerlendirme başlığında konumlanabilecek yazılım araçlarına örnek olarak Kubeflow ve TensorBoard [126] gibi araçlar gösterilebilir.

9.1.4 Deneysel ortam

Makine öğrenmesi hatlarında farklı konfigürasyonların denenebilmesi, bu denemelerin de mantıksal gruplar şeklinde organize edilebilmesi göz ardı edilemeyecek derecede önemlidir. Rutinde çalışan modellerin etkilenmemesi veya model eğitme ve değerlendirme süreçlerinde karışıklıklara yol açmamak için deney yapabileceğiniz bir ortamın olması gereklilik olarak ifade edilebilir. Örnek olarak bir deney, hiperparametre ayarlaması gibi bir süreci işleten bir çalışma ortamını karşılıyor olabilir [127]. Model tasarlama ve model oluşturma süreçlerimize katkı sunan, deney kurabildiğimiz bir çalışma ortamı şeklinde de ifade edilebilir. Kubeflow bu başlıkta da konumlanabilen bir araç olarak karşımıza çıkmaktadır.

9.1.5 Makine öğrenmesi modellerinin yazılım servisi olarak yayımlanması

Makine öğrenmesi modelleri de tıpkı uygulamalar veya UPA'larda olduğu gibi kişiler veya diğer servisler tarafından kullanılacağı için modellerin yayımlanması veya canlıya alınması gerekmektedir. Bu bağlamda model deposu, konteyner deposu ve sunucusuz platform değinilmesi gereken başlıklardır.

9.1.5.1 Model deposu

Eğitilmiş modellerin saklandığı yer şeklinde ifade edilebilecek model deposu nesne deposu üzerinde konumlanmaktadır.

9.1.5.2 Konteyner deposu

Konteyner deposu, konteyner formda çalışacak uygulamaların veya model servislerinin görüntülerinin nesne deposunda tutulduğu yer olarak tanımlanabilir. Bir yazılım aracı üzerinden ifade etmek gerekirse, Docker konteyner kayıt defteri aracının [128], konteyner saklama alanı olarak nesne deposu kullandığı bir kullanım durumu ifade edilmektedir.

9.1.5.3 Sunucusuz platform

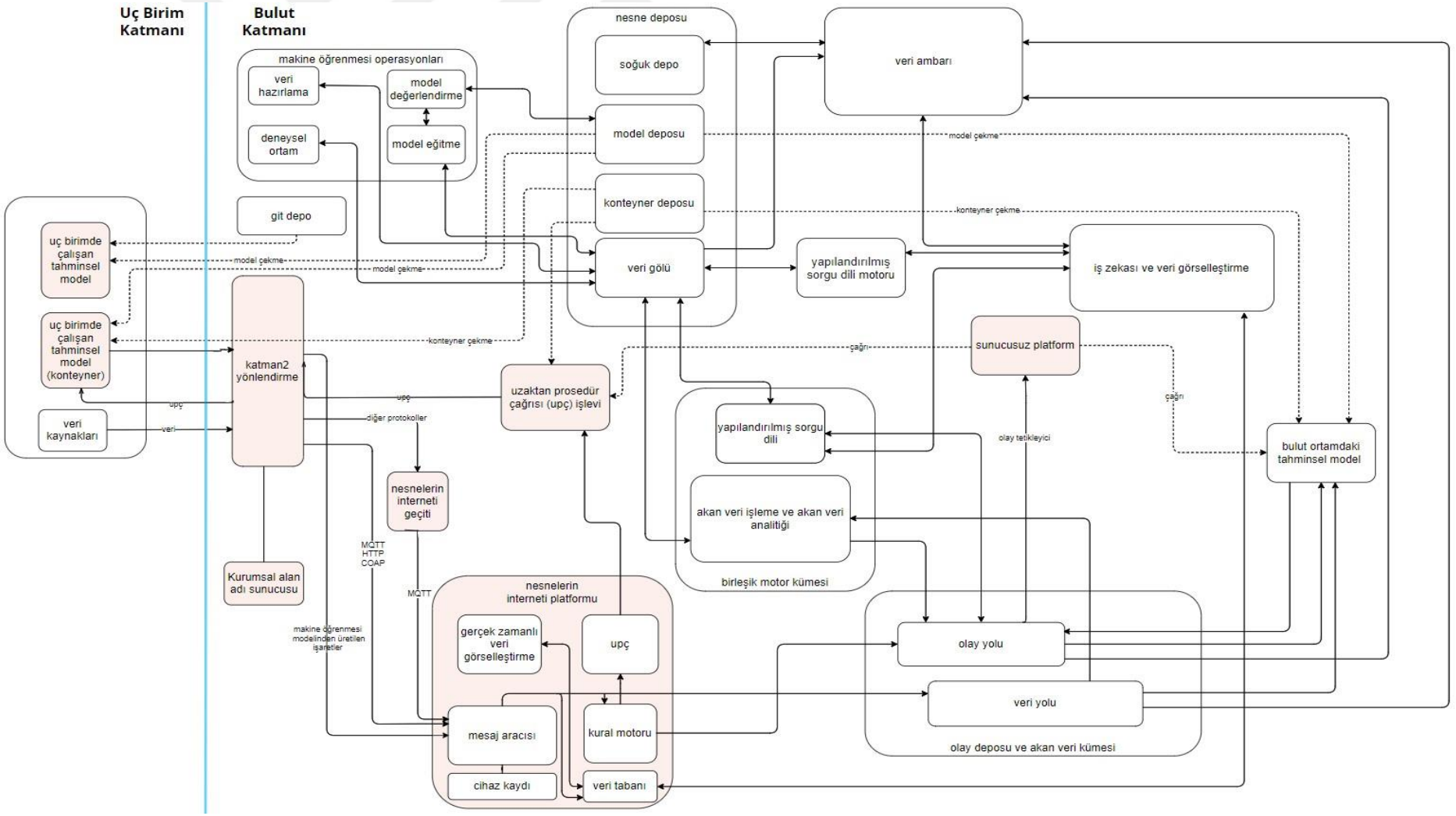
Yazılım projelerinin yetiştirilememesini veya başarısız olmasını önlemek adına ortaya çıkmış çevik yazılım geliştirme ve proje yönetimi yaklaşımları geliştiricilerin verimli bir proje süreci geçirmesini hedeflemektedir. Sunucu ortamlarının idaresi, çoğu zaman yazılım geliştiricileri aşan zorlayıcı durumlar içerebilir. Günümüzde sunucusuz platformlar sayesinde geliştiriciler, kodlarını sunucu ve altyapı yönetimi yapmaksızın çalıştırabilmektedirler. Buna benzer bir süreç, veri bilimi tabanlı uygulamalarda da kendine yer edinmiştir. Örneğin, Nuclio gibi açık kaynak kodlu sunucusuz platform kullanılarak makine öğrenmesi modelinin tahmin için kullanacağı, işlev sunucusuz bir hesaplama birimi olarak çalışabilir [129]. Sunucusuz platform, işlevlere erişim üzerinden "kullandığın kadar öde" gibi iş yapış yaklaşımına da imkan sağlayabilir.

10. NESNELERİN İNTERNETİ ODAĞINDA BÜYÜK VERİ İÇİN TAHMİNE DAYALI ANALİTİK MİMARİSİ

Nitel unsurlar dikkate alındığında, bu çalışma bulut ve büyük veri teknolojilerinin unsurlarının sahip olduğu fonksiyonel kalıplara, nesnelerin interneti unsurlarını da dahil eden bir yapıdadır. Nesnelerin interneti teknolojilerine ait bileşenler bulut katmanına ilave uç birim katmanıyla da etkileşimdedir. Söz konusu bileşenlerin sahip olduğu fonksiyonel yapılar, aşağıdaki kavramlarla örneklenebilir.

- Mesaj aracısı.
- Cihaz kaydı.
- Kural motoru.
- Gerçek zamanlı veri görselleştirme.
- Nesnelerin interneti geçidi.
- Uzaktan prosedür çağrısı işlevi.

Belirtilen kavramlar "sunucusuz platform" gibi bulut teknolojileriyle de etkileşimde olabilen süreçler barındırmaktadır. Buraya kadar anlatılan kavramlar ve tanıtılan araçlar üzerinden tez çalışması kapsamında ortaya konulan nesnelerin interneti odağında büyük veri için tahmine dayalı analitik mimarisi Şekil 10.1 üzerinden aktarılmıştır.



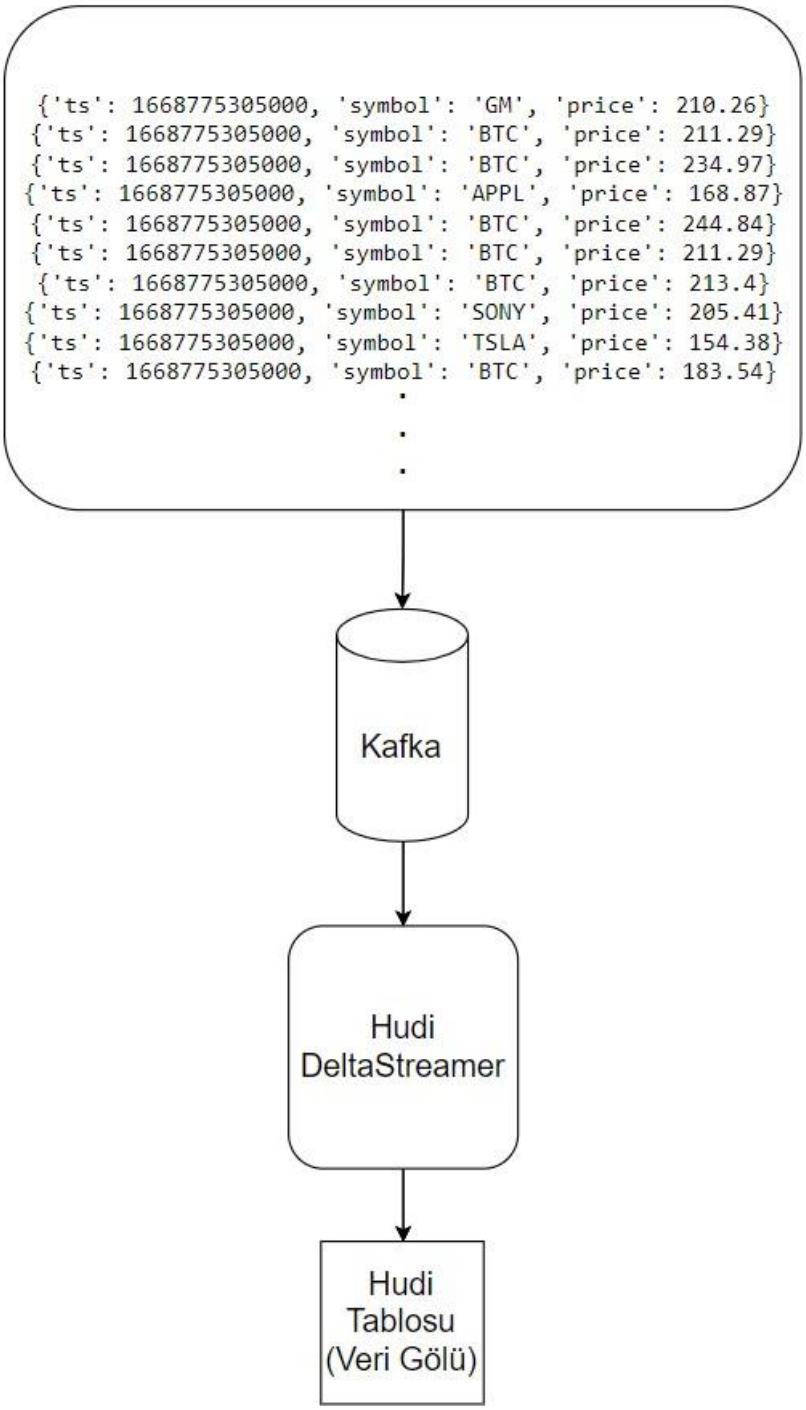
Şekil 10.1 : Nesnelerin interneti odağında büyük veri için tahmine dayalı analitik mimarisi

Endüstriyel bir üretim ortamındaki kullanım örneğiyle mimarideki bileşenleri ele almak daha aydınlatıcı olabilir. Örneğin, kestirimci bakım hedefinde, kalan kullanışlı ömür kestirimi yapacak bir modelin eğitilmesi hedeflenmiş olsun. Burada, nesnelerin interneti platformu (Thingsboard) üzerinden toplanan ham sensör verisi, akan veri aracısına (Kafka) gönderilecektir. Akan veri aracısına tüketici olarak bağlanan analitik motor (Spark) bileşeni olan Spark Structured Streaming veya Hudi'nin bir bileşeni olan HoodieDeltaStreamer üzerinden veri, nesne deposunda konumlanan veri gölüne (Ceph) veri tablo formatları (Hudi) veya veri dosya formatları (Parquet, Avro) şeklinde kaydedilecektir. Veri gölünde saklanan veri, bir makine öğrenmesi platformu kullanılarak (Kubeflow) eğitime hazır hale getirilecek ve aynı platform üzerinde model eğitimi gerçekleştirilecektir. Eğitilen model, nesne deposundaki model deposunda saklanacak ve gerektiğinde makine öğrenmesi operasyonlarında kullanılabilecektir. Bu model, Kubernetes üzerinde konteyner formda çalışacak bir yazılım olarak yayımlanabilir ve böylelikle bakıma kalan zamanı çıktı olarak sunan bir servis şeklinde, operasyonel gösterge panelleri (Grafana) veya veri tabanlarıyla iletişimde olabilir. Ayrıca ham sensör verisini saklayan veri gölü, iş zekası araçlarına (Superset) entegre olduğundan, veriye sorgu atılarak bir takım hesaplamalar yapmak ve indikatörler oluşturmak mümkün olacaktır.

10.1 Performans Testi

Hudi veri tablo formatı, Gölevi ve veri gölü tasarımında nesne deposunu konfigüre eden, veri dosya formatlarını kapsülleyen ve meta veri yönetimine imkan veren yapısıyla bu tez çalışması kapsamında önerilen en önemli araçlardan biridir. Ayrıca, Hive aracının Metastore bileşeniyle entegre olabilmesi sayesinde Hudi formatta saklanan veriye Superset gibi iş zekası araçları kullanarak erişmek mümkün hale gelmektedir.

Performans gözlemi yapabilmek için tez çalışması kapsamında önerilen araçların kullanıldığı bir senaryo kurgulanmıştır. Akan veri aracısı Kafka'ya saniyede 1812 adet kayıt akıtan üretici bir kod parçacığı tarafından, 3700 saniye (1 saat 1 dakika 40 saniye) boyunca veri akıtılmıştır. Veri, her bir kayıtta zaman bilgisi, sembol ve fiyat değerleri taşıyan faydalı yük şeklinde iletilmiştir. Şekil 10.2'de temsili bir akış görülmektedir.



Şekil 10.2 : Performans testi için kurgulanan süreç

Apache Hudi'nin bileşenlerinden olan DeltaStreamer bileşeni ile Kafka'dan alınan veri, nesne deposuna kaydedilmiştir. Kaydetme işlemi aynı üretici kod parçacığıyla aynı kayıt sayısında gerçekleştirilmiş, Hudi tablo tipleri olan "Copy On Write" ve "Merge On Read" tipleri için tekrarlanmıştır. Bir başka ifadeyle, her iki Hudi tablo tipinde aynı koşullarda veri saklama işlemi tekrarlanmıştır. Böylelikle, her iki Hudi

tablo tipiyle ilgili, toplam tablo boyutu, bir sembol için tabloya toplam yazım süresi ve bir sembol için Parquet dosyalarının ortalama boyutu gibi nicelikler toplanmıştır. İlgili nicelikler Çizelge 10.1'de görülmektedir. Akan veri, Kafka üzerinde de depolanmıştır. Böylelikle, Hudi tablolarında ve Kafka üzerinde saklandığında oluşan toplam boyut bilgisi paylaşılmıştır. İlgili sonuçlar Çizelge 10.2'de görülmektedir. Her bir kayıt için bölüm ve kaydırma değerleri incelendiğinde veri kaybı veya mükerrer tüketim olmadığı görülmüştür.

Çizelge 10.1 : Hudi tablo tipleri performans gözlemleri.

Özellikler	Copy On Write	Merge On Read
Toplam Boyut	34,62 MB	34,35 MB
Sembol Toplam Boyut	6,22 MB	6,22 MB
Sembol Toplam Yazma Süresi	60 dakika 36 saniye	60 dakika 17 saniye
Sembol Parquet Dosyalarının Ortalama Boyutu	490,12 KB	489,95 KB

Çizelge 10.2 : Hudi tablolarında ve Kafka üzerinde saklamakla ilgili performans gözlemleri.

Özellikler	Copy On Write	Merge On Read	Kafka Hacim Alanı
Toplam Boyut	34,62 MB	34,35 MB	617 MB

11. SONUÇLAR VE TARTIŞMA

Sunulan mimari, platform bakış açısıyla nesnelerin interneti platformunun gölevi platformuyla birleştirilmesi veya büyük veri paradigmaları bakış açısıyla mantıksal veri ambarı paradigmasına nesnelerin interneti bileşenin eklenmesini önermektedir. Bu yönüyle özellikle endüstrideki dijital dönüşüm çalışmalarına ışık tutan bir çalışmadır.

Performans testi neticesinde elde edilen bulgular ışığında, akan veriyi Kafka'nın saklama alanında saklamak, Hudi tablosu şeklinde veri gölünde saklamaya kıyasla yaklaşık 18 kat fazla hacim tüketmeye neden olmaktadır. Bu açıdan bakıldığında, tez çalışması kapsamında önerildiği üzere, büyük veri saklama alanı olarak nesne deposu ve Hudi Tablolarının kullanıldığı veri gölü yapısının verimlilik artışı ve depolama maliyetlerinde iyileştirme sağladığı söylenebilir. Yaklaşık 7 milyon sembol kullanılarak yapılan ölçümler sonucunda Hudi tablo tipleri kıyaslandığında, veri yazma verimlilikleri iki tablo tipi için oldukça benzerdir. Bunun sebebi, veri üzerinde bir güncelleme işleminin yapılmamasıdır. "Copy On Write" seçeneğinde, veriyi depolamak için Parquet dosya formatı kullanır. Veri güncellemesi yapıldığında, birleştirme dosyaları yazım esnasında eşitlenir, yalnızca dosya sürümü değiştirilerek yeniden yazılır. "Merge On Read" seçeneğinde ise Parquet ve Avro dosya formatları kullanılır. Veri güncellemesi yapıldığında, yeni veri bir "delta" dosyasına yazılır ve sonrasında eşzamanlı veya eşzamansız olarak kolon depolama dosyasının (Parquet) yeni bir sürümünde birleştirilir. Elde edilen bulgulara ilave olarak, literatürdeki Hudi tablo tiplerinde okuma performansına yönelik çalışmalar incelendiğinde [130] "Copy on Write" yaklaşımının sorgu gecikmesi daha düşüktür. Güncelleme işlemi yapılmadığı sürece "Copy on Write" tablo tipini tercih etmek daha avantajlı gözükmektedir.

Tez çalışması nitelik bakımından birçok bileşeni çeşitli birimler şeklinde birbirinden ayırıp, farklı sorunlara karşılık çözümler öneren kavramları bir araya getirmiştir. Her ne kadar büyük veri, nesnelerin interneti ve makine öğrenmesi kavramları, kırılımlar ve etkileşimler bakımından etraflıca ele alınmış olsa da organizasyonları, kurum kültürlerini ve kurumsal süreçleri tasarlamak bakımından yeterli değildir [131]. İnsan-Süreç-Teknoloji metodolojisinin [131, 132] önerdiği şekliyle organizasyonel verimlilik konusunda başarı, ancak işi yapan insanlarla, işi verimli kılan süreçlerle,

insanlara işlerini yapmakta yardımcı olacak teknolojilerle ve bunların arasındaki dengeyle mümkündür. Bu bağlamda, bu değerleri vurgulayan büyük veri mimari paradigmalarından "veri ağı" yapısı gelecekteki çalışmalarda da etraflıca irdelenecektir. Ürün olarak veri yaklaşımının tasarımsal dokunuşlarının da büyük veri, nesnelerin interneti ve makine öğrenmesi bağlamında ele alınması gerekliliği aşikardır.

KAYNAKLAR

[1] **Sagiroglu, S., & Sinanc, D.** (2013, May). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS) (pp. 42-47). IEEE.

[2] **Günther, W. A., Mehrizi, M. H. R., Huysman, M., & Feldberg, F.** (2017). Debating big data: A literature review on realizing value from big data. The Journal of Strategic Information Systems, 26(3), 191-209.

[3] **Tsai, C. W., Lai, C. F., Chao, H. C., & Vasilakos, A. V.** (2015). Big data analytics: a survey. Journal of Big data, 2(1), 1-32.

[4] **Wang, J., Xu, C., Zhang, J., & Zhong, R.** (2021). Big data analytics for intelligent manufacturing systems: A review. Journal of Manufacturing Systems.

[5] **Holst, A.** (2021). Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. Statista, June.

[6] **Coda, F. A., Santos Filho, D. J., Junqueira, F., & Miyagi, P. E.** (2020, July). Big Data Acquisition Architecture: An Industry 4.0 Approach. In Doctoral Conference on Computing, Electrical and Industrial Systems (pp. 222-229). Springer, Cham.

[7] **Hasbullah, H., Bareduan, S.A. and Hasibuan, S.,** (2021). Developing I4. 0 Readiness Index for Factory Operation in Indonesia to Enhance INDI 4.0. International Journal on Advanced Science, Engineering and Information Technology, 11(4), p.1668.

[8] **Priebe, T., Neumaier, S., & Markus, S.** (2021, December). Finding Your Way Through the Jungle of Big Data Architectures. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 5994-5996). IEEE.

[9] **Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M.** (2021, January). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR.

[10] **Feick, M., Kleer, N., & Kohn, M.** (2018). Fundamentals of real-time data processing architectures lambda and kappa. SKILL 2018-Studierendenkonferenz Informatik.

[11] **Url-1** <*https://parquet.apache.org*>, erişim tarihi 10.09.2022

[12] **Ghemawat, S., Gobioff, H., & Leung, S. T.** (2003, October). The Google file system. In Proceedings of the nineteenth ACM symposium on Operating systems principles (pp. 29-43).

[13] **Dean, J., & Ghemawat, S.** (2004). MapReduce: Simplified data processing on large clusters.

[14] **Url-2** <*https://www.ovaledge.com/story-data-lake*>, erişim tarihi 10.09.2022

[15] **Url-3** <*https://hadoop.apache.org*>, erişim tarihi 10.09.2022

[16] **Url-4** < *https://aws.amazon.com/s3*>, erişim tarihi 10.09.2022

[17] **Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., ... & Zaharia, M.** (2020). Delta lake: high-performance ACID table storage over cloud object stores. Proceedings of the VLDB Endowment, 13(12), 3411-3424.

[18] **Url-5** < *https://hive.apache.org*>, erişim tarihi 10.09.2022

[19] **Url-6** < *https://iceberg.apache.org*>, erişim tarihi 10.09.2022

[20] **Url-7** < *https://hudi.apache.org*>, erişim tarihi 10.09.2022

[21] **Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., ... & Zaharia, M.** (2015, May). Spark sql: Relational data processing in spark. In Proceedings of the 2015 ACM SIGMOD international conference on management of data (pp. 1383-1394).

[22] **Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M.** (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.

[23] **Ismail, A. A., Hamza, H. S., & Kotb, A. M.** (2018, December). Performance evaluation of open source IoT platforms. In 2018 IEEE global conference on internet of things (GCIoT) (pp. 1-5). IEEE.

[24] **Nakhuva, B. and Champaneria, T.,** 2015. Study of various internet of things platforms. International Journal of Computer Science & Engineering Survey, 6(6), pp.61-74.

[25] **Url-8** <*https://kafka.apache.org*>, erişim tarihi 10.09.2022

[26] **Codd, E.F.,** 1970. A relational model of data for large shared data banks. Communications of the ACM, 13(6), pp.377-387.

[27] **Chen, P.P.S.,** 1976. The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1), pp.9-36.

[28] **Berg, K.L., Seymour, T. and Goel, R.**, 2013. History of databases. International Journal of Management & Information Systems (IJMIS), 17(1), pp.29-36.

[29] **Shoshani, A.** (1997, May). OLAP and statistical databases: Similarities and differences. In Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (pp. 185-196).

[30] **Li, C., & Wang, X. S.** (1996, November). A data model for supporting on-line analytical processing. In Proceedings of the fifth international conference on Information and knowledge management (pp. 81-88).

[31] **Florescu, D., & Fourny, G.** (2013). JSONiq: The history of a query language. IEEE internet computing, 17(5), 86-90.

[32] **NoSQL–Vikipedi.** (t.y.). erişim tarihi 10.09.2022. https://en.wikipedia.org/wiki/NoSQL

[33] **Strozzi, C.** (t.y.). Shell-Ware Utilities, erişim tarihi 10.09.2022, http://www.strozzi.it/shared/swu/

[34] **Strozzi, C.** (2015). NoSQL Relational Database Management System: Home Page, erişim tarihi 10.09.2022, http://www.strozzi.it/cgi-

bin/CSA/tw7/I/en_US/NoSQL/Home%20Page

[35] **Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A.** (2015, June). A comparative study: MongoDB vs. MySQL. In 2015 13th International Conference on Engineering of Modern Electric Systems (EMES) (pp. 1-6). IEEE.

[36] **Eric E.** (2009) NoSQL: What's in a name?, erişim tarihi 10.09.2022, http://blog.sym-link.com/posts/2009/30/nosql_whats_in_a_name

[37] **Crockford, D.** (2006). The application/json media type for javascript object notation (json) (No. rfc4627).

[38] **Bassett, L.** (2015). Introduction to JavaScript object notation: a to-the-point guide to JSON. " O'Reilly Media, Inc.".

[39] **Url-9** <*https://www.postgresql.org*/>, erişim tarihi 10.09.2022

[40] **Url-10** <*https://mariadb.org*/>, erişim tarihi 10.09.2022

[41] **Url-11** <*https://clickhouse.com*/>, erişim tarihi 10.09.2022

[42] **Url-12** <*https://cassandra.apache.org*/>, erişim tarihi 10.09.2022

[43] **Url-13** <*https://hbase.apache.org*/>, erişim tarihi 10.09.2022

[44] **Url-14** <*https://www.mongodb.com*/>, erişim tarihi 10.09.2022

[45] **Url-15** <*https://neo4j.com*/>, erişim tarihi 10.09.2022

[46] **Gardner, S.R.**, 1998. Building the data warehouse. Communications of the ACM, 41(9), pp.52-60.

[47] **Url-16** <*https://druid.apache.org*/>, erişim tarihi 10.09.2022

[48] **Url-17** <*https://cloud.google.com/bigquery*>, erişim tarihi 10.09.2022

[49] **Url-18** <*https://aws.amazon.com/redshift*/>, erişim tarihi 10.09.2022

[50] **Url-19** <*https://www.sap.com/products/technology-platform/hana/what-is-sap-hana.html*>, erişim tarihi 10.09.2022

[51] **Martin F.** (2015) DataLake, erişim tarihi 10.09.2022, https://martinfowler.com/bliki/DataLake.html

[52] **Martin F.** (2014) BoundedContext, erişim tarihi 10.09.2022, https://martinfowler.com/bliki/BoundedContext.html

[53] **Walker, C., & Alrehamy, H.** (2015, August). Personal data lake with data gravity pull. In 2015 IEEE Fifth International Conference on Big Data and Cloud Computing (pp. 160-167). IEEE.

[54] **Srivastava K.** (2014). Four Common Mistakes that Makes for Toxic Data Lakes, erişim tarihi 10.09.2022, http://www.forbes.com/sites/ciocentral/2014/11/25/four-commonmistakes-that-make-for-toxic-data-lakes/

[55] **Pasha F.** (2022) Why We Need Hive Metastore, erişim tarihi 10.09.2022, https://blog.jetbrains.com/big-data-tools/2022/07/01/why-we-need-hive-metastore/

[56] **Url-20** <*https://docs.ceph.com/en/quincy/*>, erişim tarihi 10.09.2022

[57] **Ceph(yazılım)–Vikipedi.** (t.y.). erişim tarihi 10.09.2022. https://en.wikipedia.org/wiki/Ceph_(software)

[58] **Url-21** *<https://www.ambedded.com.tw/en/technology/tech_object-storage_s3.html#:~:text=Ceph%20Object%20Storage%20supports%20two,of%20the%20OpenStack%20Swift%20API>*, erişim tarihi 10.09.2022

[59] **Url-22** *<https://www.netapp.com/data-storage/storagegrid/what-is-object-storage/>*, erişim tarihi 10.09.2022

[60] **Jacop G.** (2012) Advantages of using an object storage system, erişim tarihi 10.09.2022, https://www.techtarget.com/searchstorage/tip/Advantages-of-using-an-object-storage-system>, erişim tarihi 10.09.2022

[61] **Url-23** *<https://www.worldstream.com/en/news/block-storage-vs-object-storage-understanding-the-differences>*, erişim tarihi 10.09.2022

[62] **Mike R.** (2018) Serverless Architectures, erişim tarihi 10.09.2022, https://martinfowler.com/articles/serverless.html

[63] **Arda Ç.** (2017) Serverless Nedir?, erişim tarihi 10.09.2022, https://devnot.com/2017/serverless-nedir/

[64] **Javier R.** (2020) Big Data File Formats Explained, erişim tarihi 10.09.2022, https://towardsdatascience.com/big-data-file-formats-explained-dfaabe9e8b33#:~:text=How%20you%20store%20the%20data,Buffers%2C%20Parquet%2C%20and%20ORC

[65] **Url-24** *<https://www.databricks.com/dataaisummit/session/data-lakehouse-and-data-mesh-two-sides-same-coin>*, erişim tarihi 10.09.2022

[66] **Url-25** *<https://technologytransfer.it/events-mec/data-mesh-data-fabric-unravelling-digital-information-systems/>*, erişim tarihi 10.09.2022

[67] **Url-26** *<https://lakefs.io/data-lakes/>*, erişim tarihi 10.09.2022

[68] **Belov, V., & Nikulchev, E.** (2021). Analysis of Big Data Storage Tools for Data Lakes based on Apache Hadoop Platform. International Journal of Advanced Computer Science and Applications, 12(8).

[69] **Apache ORC (Veri Dosya Formatı) – Vikipedi**. (t.y.). erişim tarihi 10.09.2022, https://en.wikipedia.org/wiki/Apache_ORC

[70] **Url-27** *<https://avro.apache.org/>*, erişim tarihi 10.09.2022

[71] **Url-28** *<https://wiki.openstreetmap.org/wiki/PBF_Format>*, erişim tarihi 10.09.2022

[72] **Jay K.** (2015). Why Avro for Kafka Data. erişim tarihi 10.09.2022. https://www.confluent.io/blog/avro-kafka-data

[73] **Url-29** *<https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/azure-data-lake-enhanced-metadata>*, erişim tarihi 10.09.2022

[74] **Url-30** *<https://spark.apache.org/sql/>*, erişim tarihi 10.09.2022

[75] **Url-31** *<https://prestodb.io/>*, erişim tarihi 10.09.2022

[76] **Simon S.** (2022). Data Lake / Lakehouse Guide: Powered by Data Lake Table Formats (Delta Lake, Iceberg, Hudi). erişim tarihi 10.09.2022 https://airbyte.com/blog/data-lake-lakehouse-guide-powered-by-table-formats-delta-lake-iceberg-hudi

[77] **Url-32** <*https://github.com/apache/hudi*>, erişim tarihi 10.09.2022

[78] **Url-33** <*https://delta.io/*>, erişim tarihi 10.09.2022

[79] **Url-34** <*https://iceberg.apache.org/*>, erişim tarihi 10.09.2022

[80] **Shefali A.** (2019). Qubole Open-Sources Multi-Engine Support for Updates and Deletes in Data Lakes. erişim tarihi 10.09.2022 https://www.qubole.com/blog/qubole-open-sources-multi-engine-support-for-updates-and-deletes-in-data-lakes

[81] **Kyle W.** (2022). Apache Hudi vs Delta Lake vs Apache Iceerg – Lakehouse Feature Comparison. erişim tarihi 10.09.2022 https://www.onehouse.ai/blog/apache-hudi-vs-delta-lake-vs-apache-iceberg-lakehouse-feature-comparison

[82] **Langhi, S., Tommasini, R., & Della Valle, E.** (2020, December). Extending Kafka Streams for Complex Event Recognition. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 2190-2197). IEEE.

[83] **Url-35** <*https://developer.confluent.io/patterns/event/event/*>, erişim tarihi 10.09.2022

[84] **Url-36** <*https://www.reactivemanifesto.org/*>, erişim tarihi 10.09.2022

[85] **Gorton, I. and Klein, J.**, 2014. Distribution, data, deployment: Software architecture convergence in big data systems. IEEE Software, 32(3), pp.78-85.

[86] **Laigner, R., Kalinowski, M., Diniz, P., Barros, L., Cassino, C., Lemos, M., ... & Zhou, Y.** (2020, August). From a monolithic big data system to a microservices event-driven architecture. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 213-220). IEEE.

[87] **Richards, M.** (2015). Software architecture patterns (Vol. 4, p. 1005). 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated.

[88] **Url-37** <*https://www.ibm.com/cloud/learn/microservices*>, erişim tarihi 10.09.2022

[89] **Martin F.** (2005). Event Sourcing. erişim tarihi 10.09.2022 https://martinfowler.com/eaaDev/EventSourcing.html

[90] **Pablo I.** (2021). Event driven architectures vs event sourcing patterns. erişim tarihi 10.09.2022. https://pablo-iorio.medium.com/event-driven-architectures-vs-event-sourcing-patterns-23d328289bf9

[91] **Martin F.** (2017). What do you mean by "Event-Driven"?. erişim tarihi 10.09.2022. https://martinfowler.com/articles/201701-event-driven.html

[92] **Martin K.** (t.y.). Making Sense of Stream Processing by Martin Kleppmann. erişim tarihi 10.09.2022 https://www.oreilly.com/library/view/making-sense-of/9781492042563/ch01.html

[93] **Url-38** <*https://kafka.apache.org*/>, erişim tarihi 10.09.2022

[94] **Url-39** <*https://www.confluent.io*/>, erişim tarihi 10.09.2022

[95] **Url-40** <*https://developer.confluent.io/patterns*/>, erişim tarihi 10.09.2022

[96] **Url-41** <*https://zookeeper.apache.org*/>, erişim tarihi 10.09.2022

[97] **Url-42** <*https://jaceklaskowski.gitbooks.io/apache-kafka/content/kafka-brokers.html*>, erişim tarihi 07.10.2022

[98] **Marz, N.** (2011). How to beat the CAP theorem. Thoughts from the Red Planet.

[99] **Iman S.** (2018). A brief introduction to two data processing architectures – Lambda and Kappa for Big Data. erişim tarihi 10.09.2022 https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb

[100] **Jagadish K. Eesha K & Thiyagarajan A.** (2022). erişim tarihi 10.09.2022. https://aws.amazon.com/tr/blogs/big-data/build-a-big-data-lambda-architecture-for-batch-and-real-time-analytics-using-amazon-redshift

[101] **Alexsandro S.** (2021). erişim tarihi 10.09.2022. https://dev.to/apssouza22/lambda-architecture-how-to-build-a-big-data-pipeline-11bl

[102] **Kreps, J.** (2014). Questioning the lambda architecture. Online article, July, 205, 18-34.

[103] **Kai W.** (2021). Kappa Architecture is Mainstream Replacing Lambda. erişim tarihi 10.09.2022. https://www.kai-waehner.de/blog/2021/09/23/real-time-kappa-architecture-mainstream-replacing-batch-lambda/

[104] **Zhelev, S., & Rozeva, A.** (2017, December). Big data processing in the cloud-Challenges and platforms. In AIP Conference Proceedings (Vol. 1910, No. 1, p. 060013). AIP Publishing LLC.

[105] **Url-43** <*https://luminousmen.com/post/modern-big-data-architectures-lambda-kappa*>, erişim tarihi 10.09.2022

[106] **Miguel S.** (2022). Apache Spark: Transformations and Lazy Evaluation. erişim tarihi 10.09.2022 https://blog.damavis.com/en/apache-spark-transformations-and-lazy-evaluation/

[107] **Url-44** <*https://grafana.com*/>, erişim tarihi 10.09.2022

[108] **İş Zekası – Vikipedi.** (t.y.). erişim tarihi 10.09.2022. https://tr.wikipedia.org/wiki/%C4%B0%C5%9F_zekas%C4%B1

[109] **Url-45** <*https://www.gtech.com.tr/is-zekasi-nedir-ozellikleri-nelerdir*/>, erişim tarihi 10.09.2022

[110] **Charity M.** (2022). Ask Miss O11y: Observability vs BI Tools & Data warehouses. erişim tarihi 10.09.2022. https://www.honeycomb.io/blog/ask-miss-o11y-observability-vs-bi-

tools-data-warehouses

[111] **Url-46** <*https://superset.apache.org*/>, erişim tarihi 10.09.2022

[112] **Beyer, M. A., & Edjlali, R.** (2012). Understanding the logical data warehouse: the emerging practice. Retrieved from Gartner database (ID: G00234996).

[113] **Zaidi, E., Thoo, E., De Simoni, G., & Beyer, M.** (2019). Data Fabrics Add Augmented Intelligence to Modernize Your Data Integration. With Eric Thoo. Gartner Grou, 17.

[114] **Martin F.** (2011). PolyglotPersistence. erişim tarihi 10.09.2022. https://martinfowler.com/bliki/PolyglotPersistence.html

[115] **Zhamak D.** (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. erişim tarihi 10.09.2022. https://martinfowler.com/articles/data-monolith-to-mesh.html

[116] **Knud L.** (2014). Why the Internet of Things is called Internet of Things: Definition, history, disambiguation. erişim tarihi 10.09.2022. https://iot-analytics.com/internet-of-things-definition/

[117] **Li, S., Xu, L.D. and Zhao, S.**, 2015. The internet of things: a survey. Information systems frontiers, 17(2), pp.243-259.

[118] **Telemetri – Vikipedi.** (t.y.). , erişim tarihi 10.09.2022. https://tr.wikipedia.org/wiki/Telemetri

[119] **Telemetry – Vikipedi.** (t.y.). , erişim tarihi 10.09.2022. https://en.wikipedia.org/wiki/Telemetry

[120] **Cosette C.** (2021). Everything About MQTT Protocol – Message Queue Telemetry Transport. erişim tarihi 10.09.2022. https://www.cometchat.com/blog/everything-about-mqtt-protocol-message-queue-telemetry-transport

[121] **Url-47** <*https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels*/>, erişim tarihi 10.09.2022

[122] **Kai W.** (2019). Internet of Things (IoT) and Event Streaming at Scale with Apache Kafka and MQTT. erişim tarihi 10.09.2022. https://www.confluent.io/blog/iot-with-kafka-connect-mqtt-and-rest-proxy

[123] **Url-48** <*https://kubernetes.io*/>, erişim tarihi 10.09.2022

[124] **Url-49** <*https://www.kubeflow.org*/>, erişim tarihi 10.09.2022

[125] **Url-50** <*https://valohai.com/mlops-platforms-compared*/>, erişim tarihi 10.09.2022

[126] **Url-51** <*https://www.tensorflow.org/tensorboard*>, erişim tarihi 10.09.2022

[127] **Url-52** <*https://www.kubeflow.org/docs/components/katib/experiment*/>, erişim tarihi 10.09.2022

[128] **Url-53** <*https://docs.docker.com/registry*/>, erişim tarihi 10.09.2022

[129] **Url-54** <*https://nuclio.io/*>, erişim tarihi 10.09.2022

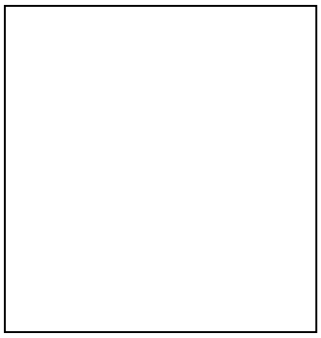
[130] **Url-55** <*https://programming.vip/docs/actual-write-to-hudi-using-spark-*

streaming.html>, erişim tarihi 10.09.2022

[131] **Kyle B.** (2022). DATA MESH AND THE RISE OF DATA LAKEHOUSE. erişim tarihi 10.09.2022. https://atrium.ai/resources/data-mesh-and-the-rise-of-data-lakehouse/

[132] **Aditya K.** (2022). People, Process, Technology: The PT Framework, Explained. erişim tarihi 10.09.2022. https://www.plutora.com/blog/people-process-technology-ppt-framework-explained

ÖZGEÇMİŞ



Ad-Soyad : Oğuzhan YÜCE

Doğum Tarihi ve Yeri :

E-posta :

ÖĞRENİM DURUMU:

- **Lisans :** 2018, Ankara Yıldırım Beyazıt Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Elektrik-Elektronik Mühendisliği (İng.)

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2018 – CITS Bilişim Hizmetleri – Ar-Ge Yapay Zeka Uzmanı

TEZDEN TÜRETİLEN ESERLER, SUNUMLAR VE PATENTLER:

- İsen, O. A. & Yüce, O. (2021). Scratch Detection on a Vehicle Safety Part Using Vibration Analysis. 26th International Sheet Metal Working Technology Exhibition Hanover, Germany
- Yüce, O., Aslan, U., Hanilçi, C., Korkmaz, E., İsen, O. A., & Cantez, E. (2020). Fault Diagnosis: Spectral Analysis of the Vibration Signals in Transfer Press. *Academic Perspective Procedia*, *3*(1), 1-9.

DİĞER ESERLER, SUNUMLAR VE PATENTLER:

- Comparison of Texture Features Based Classifiers for Detecting Cracks and Splits on Metal Automotive Parts. (2019) International Symposium on Intelligent Manufacturing and Service Systems, IMSS'19.